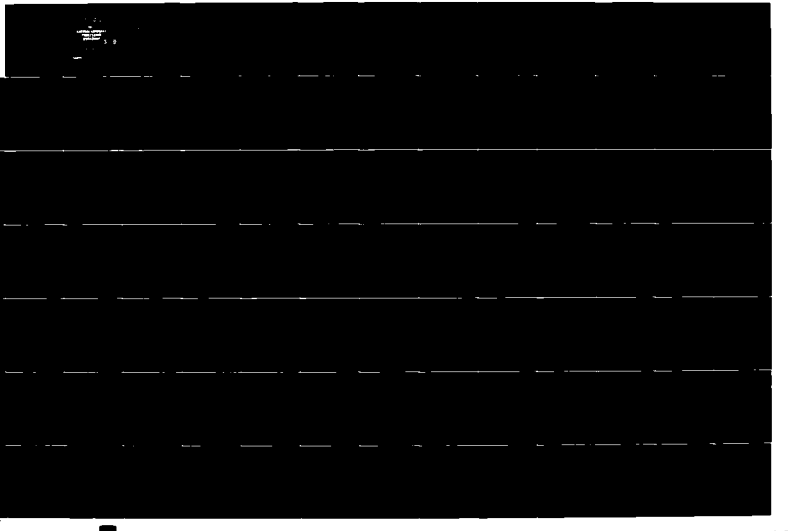
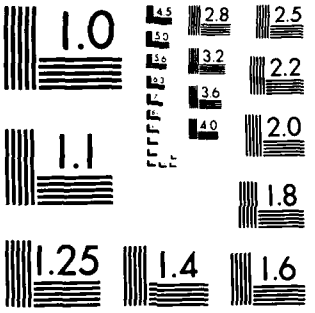
AD-A172 436 PROCEEDINGS OF THE STRATEGIC COMPUTING NATURAL LANGUAGE WORKSHOP HELD IN..(U) UNIVERSITY OF SOUTHERN CALIFORNIA MARINA DEL REY INFORMATION S.. N K SONDHEIMER MAY 86 1/3

UNCLASSIFIED ISI/SR-86-172 MDA903-81-C-0335 F/G 9/2 NL





*ISI Special Report*
*ISI/SR-86-172*
*May 1986*

(10)



Proceedings of

# DARPA's 1986 STRATEGIC COMPUTING

# NATURAL LANGUAGE PROCESSING WORKSHOP





## MAY 1986

***Sponsored By***

DARPA

***Held At***

USC INFORMATION SCIENCES INSTITUTE
4676 Admiralty Way, Suite 1001
Marina del Rey, California 90292
(213) 822-1511

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

AD-A172436

# REPORT DOCUMENTATION PAGE

| | |
|---|---|
| 1a. REPORT SECURITY CLASSIFICATION<br>Unclassified | 1b. RESTRICTIVE MARKINGS |
| 2a. SECURITY CLASSIFICATION AUTHORITY | 3. DISTRIBUTION/AVAILABILITY OF REPORT<br>This document is approved for public release; distribution is unlimited. |
| 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE | |
| 4. PERFORMING ORGANIZATION REPORT NUMBER(S)<br>ISI/SR-86-172 | 5. MONITORING ORGANIZATION REPORT NUMBER(S) |
| 6a. NAME OF PERFORMING ORGANIZATION<br>USC/Information Sciences Institute<br>6b. OFFICE SYMBOL (If applicable) | 7a. NAME OF MONITORING ORGANIZATION |
| 6c. ADDRESS (City, State, and ZIP Code)<br>4676 Admiralty Way<br>Marina del Rey, CA 90292 | 7b. ADDRESS (City, State, and ZIP Code) |
| 8a. NAME OF FUNDING/SPONSORING ORGANIZATION<br>Advanced Research Projects Agency<br>8b. OFFICE SYMBOL (If applicable) | 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER<br>MDA903 81 C 0335 |
| 8c. ADDRESS (City, State, and ZIP Code)<br>1400 Wilson Blvd.<br>Arlington, VA 22209 | 10. SOURCE OF FUNDING NUMBERS<br>PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO. |

11. TITLE (Include Security Classification)
Proceedings of the Strategic Computing Natural Language Workshop (Unclassified)

12. PERSONAL AUTHOR(S)
Sondheimer, Norman K., editor

| 13a. TYPE OF REPORT | 13b. TIME COVERED | 14. DATE OF REPORT (Year, Month, Day) | 15. PAGE COUNT |
|---|---|---|---|
| Special Report | FROM _____ TO _____ | 1986, May | 294 |

16. SUPPLEMENTARY NOTATION
Proceedings of a workshop held at Marina del Rey, Calif., May 1-2, 1986.

| 17 COSATI CODES | | | 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) |
|---|---|---|---|
| FIELD | GROUP | SUB-GROUP | artificial intelligence, natural language generation, natural language processing, natural language understanding, text generation, text processing, text understanding |
| 09 | 02 | | |
| | | | |

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This document contains the reviews and selected technical papers for the Natural Language Processing Program, sponsored by the Information Sciences & Technology Offices of the Defense Advanced Research Projects Agency, which were presented at a workshop conducted on 1-2 May 1986, in Marina del Rey, California.

| 20. DISTRIBUTION/AVAILABILITY OF ABSTRACT | 21. ABSTRACT SECURITY CLASSIFICATION |
|---|---|
| ☒ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS | Unclassified |
| 22a. NAME OF RESPONSIBLE INDIVIDUAL<br>Victor Brown/Sheila Coyazo | 22b. TELEPHONE (Include Area Code)<br>(213) 822-1511<br>22c. OFFICE SYMBOL |

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

# Strategic Computing
# Natural Language Workshop

**Proceedings of a Workshop**
**Held at**
**Marina del Rey, California**
**May 1-2, 1986**

**Sponsored by the**
**Defense Advanced Research Projects Agency**

**University of Southern California**
**Information Sciences Institute**
**Report Number ISI/SR-86-172**
**August, 1986**
**Norman K. Sondheimer**
**Workshop Organizer**

This report was supported by
the Defense Advanced Research
Projects Agency Under DARPA
Contract No. MDA903 81 C 0335

# TABLE OF CONTENTS - -

**FORWARD** . . . . . *i*
LTC. Robert Simpson DARPA/ISTO

**SECTION 1: OVERVIEW OF RESEARCH EFFORTS**

Research and Development in Natural Language Processing at BBN Laboratories in the Strategic Computing Program . . . . . *1*
R. Weischedel, R. Scha, E. Walker, D. Ayuso, A. Haas, E. Hinrichs, R. Ingria, L. Ramshaw, V. Shaked, D. Stallard

PROTEUS and PUNDIT: Research in Text Understanding . . . . . *11*
R. Grishman, L. Hirschman

Overview of the TACITUS Project . . . . . *19*
J. Hobbs

The Counselor Project at the University of Massachusetts . . . . . *26*
D. McDonald, J. Pustejovsky

Research in Natural Language Processing . . . . . *30*
A. Joshi, T. Finin, D. Miller, L. Shastri, B. Webber

Text Generation for Strategic Computing . . . . . *37*
W. Mann, N. Sondheimer, R. Albano, S. Cumming, T. Galloway
C. Matthiessen, B. Nebel, L. Poulton, G. Vamos, R. Whitney

**SECTION 2: RESEARCH CONTRIBUTIONS - Bolt, Beranek, and Newman, Inc.**

Out of the Laboratory: A Case Study of the IRUS Natural Language Interface . . . . . *44*
R. Weischedel, E. Walker, D. Ayuso, J. de Bruin, K. Koile, L. Ramshaw, V. Shaked

A Terminological Simplification Transformation for Natural Language Question-Answering Systems . . . . . *62*
D. Stallard

**SECTION 3: RESEARCH CONTRIBUTIONS - New York University/SDC**

Model-based Analysis of Messages about Equipment . . . . . *73*
R. Grishman, T. Ksiezyk, N. Than Nhan
New York University

An Equipment Model and its Role in the Interpretation of Nominal Compounds . . . . . *81*
T. Ksiezyk, R. Grishman
New York University

Recovering Implicit Information . . . . . *96*
M. Palmer, D. Dahl, R. Schiffman, L. Hirschman, M. Linebarger, J. Dowding
SDC

Focusing and Reference Resolution in PUNDIT . . . . . 114
D. Dahl
SDC

**SECTION 4: RESEARCH CONTRIBUTIONS - SRI International**

Commonsense Metaphysics and Lexical Semantics . . . . . 127
J. Hobbs, W. Croft, T. Davies, D. Edwards, K. Laws

**SECTION 5: RESEARCH CONTRIBUTIONS - University of Massachusetts**

Multi-Level Description Directed Generation . . . . . 137
D. McDonald

TAG's as a Grammatical Formalism Generation . . . . . 146
D. McDonald, J. Pustejovsky

Hypotheticals as Heuristic Device . . . . . 165
E. Rissland, K. Ashley

**SECTION 6: RESEARCH CONTRIBUTIONS - University of Pennsylvania**

Living Up To Expectations: Computing Expert Responses . . . . . 179
A. Joshi, B. Webber, R. Weischedel

The Role of Perspective In Responding to Property Misconceptions . . . . . 190
K. McCoy

Adapting MUMBLE: Experience with Natural Language Generation . . . . . 200
R. Rubinoff

Some Computational Properties of Tree Adapting Grammars . . . . . 212
K. Vijay-Shankar, A. Joshi

GUMS: A General User Modeling System . . . . . 224
T. Finin, D. Drager

**SECTION 7: RESEARCH CONTRIBUTIONS - University of Southern California**

A Logical-Form and Knowledge-Base Design for Natural Language Generation . . . . . 231
N. Sondheimer, B. Nebel

The Lexicon in Text Generation . . . . . 242
S. Cumming

Assertions from Discourse Structure . . . . . 257
W. Mann, S. Thompson

## FORWARD

*LTC. Robert Simpson, DARPA/ISTO*

# Natural Language Technology Base Contracts within DARPA's Strategic Computing Program

LTC. Robert Simpson, DARPA/ISTO

May 1,1986

The overall objective of the Strategic Computing Program (SC) of the Defense Advanced Research Projects Agency (DARPA) is to develop and demonstrate a new generation of machine intelligence technology which can form the basis for more capable military systems in the future and also maintain a position of world leadership for the US in computer technology. Begun in 1983, SC represents a focused research strategy for accelerating the evolution of new technology and its rapid prototyping in realistic military contexts. The more specific top level goals supporting this single broad objective are to produce technology that will:

1. enable the operation of military systems under critical constraints such as time, information overload, etc.,

2. enable the management of forces/resources under constraints of information overload, geographic distribution, cost of operation, etc., and

3. facilitate the design, manufacture, and maintenance of defense systems within time, performance, quality, reliability, and cost constraints.

Even though capabilities for man-machine interaction will ultimately form an important component of systems in all of these areas, the second of those goals has been selected as the initial area to include emphasis on decision-making aids, including natural language processing.

Subgoals of these top level goals include:

1. To strengthen/develop areas of science and technology that enables the building of computer systems needed to attain the top level goals.

2. The technologies identified are:

   - Artificial Intelligence,
   - Software development and Machine Architectures,
   - Micro-electronics, and related infrastructure.

3. To build demonstration systems in specific military areas that:

   - Provide focus for technology development,
   - Provide means for exercising technology in real environments,
   - Facilitate manpower training,
   - Facilitate development of industrial capability, and
   - Facilitate technology transfer to the military.

There are four very ambitious demonstration prototypes being developed within the SC Program. They are:

1. the Pilot's Associate which will aid the pilot in route planning, aerial target prioritization, evasion of missile threats, and aircraft emergency safety procedures during flight;

2. the Autonomous Land Vehicle (ALV) which integrates in a major robotic testbed the technologies for dynamic image understanding, knowledge-based route planning with replanning during execution, hosted on new advanced parallel architectures;

3. two battle management projects one for the for the Army, which is just getting started, called the AirLand Battle Management program (ALBM) which will use knowledge-based systems technology to assist in the generation and evaluation of tactical options and plans at the Corps level; and

4. the other more established program for the Navy is the Fleet Command Center Battle Management Program (FCCBMP) at Pearl Harbor. The FCCBMP is employing knowledge-based systems and natural language technology in a evolutionary testbed situated in an operational command center to demonstrate and evaluate intelligent decision-aids which can assist in the evaluation of fleet readiness and explore alternatives during contingencies. It is within this context that the natural language contractors are currently demonstrating the potential of natural language technology.

Competitive awards were made to seven contractors in 1984. Four (BBN Laboratories, Inc., University of Southern California Information Sciences Institute (USC-ISI), the University of Pennsylvania, and the University of Massachusetts) are involved in research and development in natural language interfaces; three others (New York University (NYU), Systems Development Corporation (SDC), and SRI International) are involved in research and development in text processing.

The work in natural language for strategic computing, which includes no work currently directed to speech recognition, focuses on producing and demonstrating two "new generation systems." One for natural language interfaces and another for processing free form text from military messages. One of the natural language new generation systems is a state-of-the-art interface being jointly designed and implemented by BBN and USC-ISI. The other is a highly accurate natural language text understanding system which is being constructed by the university/industry team of NYU and SDC. In each case, they will serve the purpose of supporting the integration of specific research efforts produced by participating component technology contractors. The design of the new generation systems will be developed in concert with the needs of other research contractors and the resulting implementation will be furnished to them for use as a framework to support their own research efforts.

This document is the proceedings of a workshop held to review the ongoing research. The first section of the document contains summary reports from most of the participating groups. The second section contains selected technical papers from the research groups.

The workshop was held May 1 & 2, 1986, at USC-ISI, Marina del Rey, California. Presentations were also made by research groups focusing on speech understanding (Carnegie-Mellon University and BBN Laboratories Inc.) and expert systems technology (Teknowledge, Inc., and Ohio State University). Also in attendance were representatives from a variety of organizations within the Department of Defense.

**DARPA/ISTO would like to extend its thanks to USC-ISI for hosting the workshop and preparing this proceedings.**

# SECTION 1: OVERVIEW OF RESEARCH EFFORTS

*Bolt, Beranek, and Newman, Inc.*
*New York University*
*SRI International*
*University of Massachusetts*
*University of Pennsylvania*
*University of Southern California*

Research and Development in Natural Language Processing
at BBN Laboratories
in the Strategic Computing Program

BBN Laboratories, Inc.
Cambridge, MA 02238

STAFF: Ralph Weischedel (Principal Investigator), Remko Scha, Edward Walker, Damaris Ayuso, Andrew Haas, Erhard Hinrichs, Robert Ingria, Lance Ramshaw, Varda Shaked, David Stallard

## 1 Background

BBN's responsibility is to conduct research and development in natural language interface technology. This responsibility has three aspects:

- to demonstrate state-of-the-art technology in a Strategic Computing application, collecting data regarding the effectiveness of the demonstrated heuristics,
- to conduct research in natural language interface technology, as itemized in the description of JANUS later in this note, and
- to integrate technology from other natural language interface contractors, including USC/Information Sciences Institute, the University of Pennsylvania, and the University of Massachusetts.

Of the three initial applications described in the overview, the Fleet Command Center Battle Management Program (FCCBMP) has been the application providing the domain in which our work is being carried out. The FCCBMP encompasses the development of expert system capabilities at the Pacific Fleet Command Center in Hawaii, and the development of an integrated natural language interface to these new capabilities as well as to the existing data bases and graphic display facilities. BBN is developing a series of increasingly sophisticated natural language understanding systems which will serve as an integrated interface to several facilities at the Pacific Fleet Command Center: the Integrated Data Base (IDB), which contains information about ships, their readiness states, their capabilities, etc.; the Operations Support Group Prototype (OSGP), a graphics system which can display locations and itineraries of ships on maps; and the Force Requirements Expert System (FRESH) which is being built by Texas Instruments.

The target users for this application are naval officers involved in decision

making at the Pacific Fleet Command Center; these are executives whose effort is better spent on navy problems and decision making than on the details of which software system offers a given information capability, how a problem should be divided to make use of the various systems, or how to synthesize the results from several sources into the desired answer. Currently they do not access the data base or OSGP application programs themselves; instead, on a round-the-clock basis, two operators act as intermediaries between the Navy staff and the computers. The utility of a natural language interface in such an environment is clear.

The starting point for development of the natural language interface system at the Pacific Fleet Command Center was the IRUS system, which has been under development at BBN for a number of years. A new version of this system, IRUS-86, has been installed in the FCCBM' testbed area at the Pacific Fleet Command Center for demonstration. Further ' .c research on the problems of natural language interfacing is continuin; and the results of this and future research will be incorporated into a nex' generation natural language interface system called JANUS, to be delivered to the Pacific Fleet Command Center at a later date. JANUS will share most of its domain-dependent data with IRUS-86, and it will share other modules as well; IRUS-86 will therefore be able to evolve gradually into the final version of JANUS.

## 2 IRUS-86: The Initial Test Bed System

The architecture of IRUS [Bates 83] is a cascade consisting of a sequence of translation modules.

- An ATN parser which produces a syntactic tree.
- A semantic interpreter which produces a formula of the meaning representation language MRL.
- A postprocessor for resolving anaphora and ellipsis.
- A translation module which produces a formula of the relational data base language ERL ("Extended Relational Language").
- A translation module which produces a sequence of commands for the underlying data base access system.

IRUS-86, the version of IRUS which is now installed at the Pacific Fleet Command Center, is a version of IRUS which is extended in several ways. Two of these extensions are especially worth mentioning:

- IRUS-86 uses the NIKL system [Moser 83] to represent its domain model, i.e., the relationships between the predicates and relations of the meaning representation language MRL. The NIKL domain model supports the system's treatment of semantic anomaly, anaphora, and nominal compounds.

- IRUS-86 contains a new module which exploits this NIKL domain model to simplify MRL expressions; this makes it possible to translate complex MRL-expressions into ERL constants, thus allowing for significant divergences between the input English and the structure of the underlying data base [Stallard 86].

In addition to accessing the NIKL domain model, the parser, semantic interpreter and MRL-to-ERL translator access other knowledge sources which contain domain-dependent information:

- the lexicon,
- the semantic interpretation rules for individual concepts,
- the MRL-to-ERL mapping rules for individual MRL constants, which introduce the details of underlying system structure, such as file and field names.

To port IRUS to the navy domain, the relevant domain-dependent data had to be supplied to the system. This task is being accomplished by personnel at the Naval Ocean Systems Center (NOSC). In August, 1985, BBN provided NOSC with an initial prototype system containing small example sets of lexical entries, semantic interpretation rules, and MRL-to-ERL rules; using acquisition tools provided by BBN, NOSC personnel have been entering the rest of the data.

IRUS-86 was delivered to the FRESH developers at Texas Instruments in January 1986, was installed in a test bed area of the Pacific Fleet Command Center in April 1986, and will be demonstrated in June 1986. Currently, the lexicon and the domain-dependent rules of the system only cover a relatively small part of the OSGP capabilities and the files and attributes of the Integrated Data Base. Once enough data have been entered so that the system covers a sufficiently large part of the data base, it will be tried out in actual use by Navy personnel. This will enable us to gather data about the way the system performs in a real environment, and to fine-tune the system in various respects. For instance, IRUS-86 makes use of shallow heuristic methods to address some aspects of natural language understanding such as anaphora and ellipsis for which general solutions are still research issues. The FCCBMP application provides a test bed in which such heuristic methods can be evaluated, and enhancements to them developed and tested, as part of the evolutionary technological growth intended to continue throughout the Natural Language Technology effort of the Strategic Computing Program.

## 3 Functional Goals for JANUS

The IRUS-86 system excels by its clean, modular structure, its broad syntactic/semantic coverage, its sophisticated domain model, and its systematic treatment of discrepancies between the English lexicon and the data base structure. We thus expect that it will demonstrate considerable utility as an interface component in the FCCBMP application. Nevertheless, IRUS-86 shares with other current systems several limitations which should be overcome if natural language interfaces are to become truly "natural". In developing JANUS, the successor of IRUS-86, we shall attempt to overcome some of those limitations. The areas of increased functionality we are considering are: semantics and knowledge representation, ill-formedness, discourse, cooperativeness, multiple underlying systems, and knowledge acquisition.

### 3.1 Semantics and Knowledge Representation

IRUS-86, like most other current systems, represents sentence meanings as formulas of a logical language which is a slight extension of first-order logic. As a consequence, many important phenomena in English have no equivalent in the meaning representation language, and cannot be dealt with correctly, e.g., modalities, propositional attitudes, generics, collective quantification, and context-dependence. Thus, one foregoes one of the most important potential assets of a natural language interface: the capacity of expressing complex semantic structures in a succinct and comfortable way.

In JANUS, we will therefore adopt a new meaning representation language which combines features from PHILIQA1's enriched lambda-calculus [Scha 76] with ideas underlying Montague's Intensional Logic [Montague 70], and possibly a distributed quote-operator [Haas 86]. It will have sufficient expressive power to incorporate a version of Carlson's treatment of generics [Carlson 79], a version of Scha's treatment of quantification [Scha 81], Montague's treatment of modality, and various possible approaches to propositional attitudes and context-dependence.

In adopting a higher order logic as proposed, one confronts problems of formula simplification and the need to apply meaning postulates to reduce the semantic representation of an input sentence to an expression appropriate to the underlying system, e.g., a relational algebra expression in the case that the underlying system is a data base. To do this, we will investigate the limited inference mechanisms of KL-TWO [Moser 83, Vilain 85], following up on our previous work [Stallard 86]. The advantage of these inference mechanisms is their tractability; discovering their power and limitations in this complex problem domain should be an interesting result.

### 3.2 Discourse

The meaning of a sentence depends in many ways on the context which has been set up by the preceding discourse. IRUS and other systems, however, currently ignore most of these dependencies, and employ a rather shallow model of discourse structure. To allow the user to exploit the full expressive potential of a natural language interaction, the system must track topics, reference times, possible antecedents for anaphora, etc.; it must be able to recognize the constituent units of a discourse and the subordination or coordination relations obtaining between them. A substantial amount of work has been done already on several of these issues, much of it by BBN researchers [Sidner 85, Hinrichs 81, Polanyi 84, Grosz 86]. Research in this area continues under a separate DARPA-funded contract. We expect to be able to integrate some of the results of that research in the JANUS system.

### 3.3 Ill-formedness

A natural interface system should be forgiving of a user's deviations from its expectations, be they misspellings, typographical errors, unknown words, poor syntax, incorrect presuppositions, fragmentary forms, or violated selection restrictions. Empirical studies show that as much as 25% of the input to data base query systems is ill-formed.

IRUS currently handles some classes of ill-formedness by using a combination of shallow heuristics and user interaction. It can correct for typographical misspellings, for omitted determiners or prepositions, and for some ungrammaticalities, like determiner-noun and subject-verb disagreement. The JANUS system will employ a more general approach to ill-formedness that will handle a larger class of ungrammatical constructions and a larger class of word selection problems, and that will also explore correcting several types of semantic ill-formedness.

These capabilites have major implications for the control of the understanding process, since considering such possibilities can exponentially expand the search space. Maintaining control will require care in integrating the ill-formedness capability into the rest of the system, and also making maximal use of the guidance that can be derived from a model of the discourse and user's goals to constrain the search.

### 3.4 Cooperativeness

A truly helpful system should not react to the literal meaning of a sentence, but to its perceived intent. If in the context of a given application it is possible to characterize the goals that a user may be expected to be pursuing through his interaction with the system, the system should try to infer from the user-input what the underlying goal could be. A system can do this by accessing a goal-subgoal

hierarchy which links the speech acts expressed by individual utterances to the global goals that the user may have. This strategy has been applied successfully to rather small domains [Allen 83, Sidner 85]. We wish to investigate whether it carries over to the FCCBMP applications.

### 3.5 Modelling the Capabilities of Multiple System

The way in which IRUS-86 decides whether an input sentence translates into an IDB query or an OSGP command may be refined. There is a need for work on what kind of knowledge would be necessary to interface smoothly and intelligently to multiple underlying systems. A reasoning component is needed that can determine which underlying system or systems can best fulfill a user's request. Such a reasoning component would have to combine a model of the capabilites of the underlying systems with a model of the user goals and current intentions in the discourse context in order to choose the correct system(s). Such a model would also be useful for providing supporting information to the user.

### 3.6 Knowledge Acquisition

Further research is also called for to expand the power of the knowledge acquisition tools that are used in adding to the lexicon, the set of case frame rules, the model of domain predicates, and the set of transformation rules between the Meaning Representation Language and the languages of the underlying systems. The acquisition tools available in IRUS, unlike those in some other systems, are not tied to the specific fields and relations in the underlying database. The acquisition tools should work on the higher level of the domain model, since that provides a more general and transportable result. The knowledge acquisition facilities for JANUS will also need to be redesigned to support and to make maximal use of the power of the new meaning representation language based on intensional logic.

## 4 New Underlying Technologies

### 4.1 Coping with Ambiguity

The new functionalities we described in the previous section, and the techniques we intend to use to achieve them, raise an issue which has important consequences for the design of JANUS: we will be faced with an explosion in the number of interpretations that the system will have to process; every sentence will be manifold ambiguous. One source of this phenomenon is the improvement of the semantic coverage and the broadening of the discourse context. Distinctions and ambiguities which so far were ignored will be dealt with: for instance, different interpretation and

scopes of quantifiers will be considered, and different antecedents for pronouns. Even more serious is the processing of ill-formed sentences, which may require trying out all partial interpretations to see which one can be extended to a complete interpretation after relaxing one or more constraints.

To cut down on the processing of spurious interpretations, it is very important that interpretations of sentences and their constituents be tested for plausibility at an early stage. Different techniques must probably be used in conjunction:

- Simplification transformations may show that an interpretation is absurd, by reducing it to TRUE or FALSE or the empty set.
- The discourse context and the model of the user's goals impose constraints on expected sentences.

### 4.2 Parallel Parsing

Since some of the techniques that we intend to use to fight the ambiguity explosion are themselves rather computation-intensive, it is clearly unavoidable that the improved system functionality that we aim for will lead to a considerable increase in the amount of processing required. To avoid a serious decrease of the new system's response times, we will therefore move it to a suitable parallel machine such as BBN's Butterfly or Monarch, running a parallel Common Lisp. This in itself has rather serious consequences for the software design. It means that from the outset we will keep parallelizability of the software in mind.

We have begun to address this issue in the area of syntax. A new declarative grammar is being written, which will ultimately have a coverage of English larger than the current RUS grammar; the grammar is written in a side-effect-free formalism (a context-free grammar with variables) so that different parsing algorithms may be explored which are easily parallelizable. The first such algorithm was implemented in May 1986 on BBN's Butterfly.

## 5 Contributions from Other Sites

### 5.1 ISI/UMass: Generation

We should not expect that JANUS will always be able to assess correctly which interpretation of a sentence is the intended one. In light of such situations, it is very important that the system can give a paraphrase of the input to the user, which shows the system's interpretation. This may be done either explicitly or as part of the answer. To be able to develop such capabilities, work on Natural Language Generation is needed. At USC/ISI a project directed by William Mann and Norman

Sondheimer is underway to develop the generation system PENMAN, using the NIGEL systemic grammar. PENMAN will be integrated to become the generation component of JANUS. PENMAN itself consists of several subcomponents. Some of these, specifically the "text planning" component, will be developed through joint work between USC/ISI and David McDonald at the University of Massachusetts, based on the latter's experience with the MUMBLE system.

### 5.2 UPenn: Cooperation and Clarification

Under the direction of Aravind Joshi and Bonnie Webber at the University of Pennsylvania, several focussed studies have been carried out to investigate various aspects of cooperative system behaviour and clarification interactions. (For more detail, see their paper in this issue.) As part of the Strategic Computing Natural Lanauge effort, UPenn will eventually develop this into a module which can be integrated into JANUS to further enhance its capabilities.

## References


[Allen 83] Allen, J.F.
Recognizing Intentions from Natural Language Utterances.
In M. Brady and R.C. Berwick (editors), *Computational Models of Discourse*, pages 107–166. Massachusetts Institute Technology Press, 1983.

[Bates 83] Bates, M. and Bobrow, R.J.
A Transportable Natural Language Interface for Information Retrieval.
In *Proceedings of the 6th Annual International ACM SIGIR Conference*. ACM Special Interest Group on Information Retrieval and American Society for Information Science, Washington, D.C., June, 1983.

[Carlson 79] Carlson, G.
*Reference to Kinds in English*.
Garland Press, New York, 1979.

[Grosz 86] Grosz, B.J. and Sidner, C.L.
The Structures of Discourse Structure.
In L. Polanyi (editor), *Discourse Structure*. Ablex Publishers, Norwood, NJ, 1986.

[Haas 86] Haas, A.R.
A Syntactic Theory of Belief and Action.
*Artificial Intelligence* , 1986.
Forthcoming.

[Hinrichs 81] Hinrichs, E.
Temporale Anaphora im Englischen.
1981.
Unpublished ms., University of Tuebingen.

[Montague 70] Montague, R.
Pragmatics and Intensional Logic.
*Synthese* 22:68–94, 1970.

[Moser 83] Moser, M.G.
An Overview of NIKL, the New Implementation of KL–ONE.
In Sidner, C. L., et al. (editors), *Research in Knowledge Representation for Natural Language Understanding – Annual Report, 1 September 1982 – 31 August 1983*, pages 7–26. BBN Laboratories Report No. 5421, 1983.

[Polanyi 84] Polanyi, L. and Scha, R.
A Syntactic Approach to Discourse Semantics.
In *Proceedings of Int'l. Conference on Computational Linguistics*. Stanford University, Stanford, CA, 1984.

[Scha 76] Scha, R.J.H.
Semantic Types in PHLIQA1.
In *Proceedings of the 6th International Conference on Computational Linguistics*. 1976.

[Scha 81] Scha, R.J.H.
Distributive, Collective and Cumulative Quantification.
*Formal Methods in the Study of Language, Part 2.*
Mathematisch Centrum, Amsterdam, 1981, pages 483–512.
Reprinted in: J.A.G. Groenendijk, T.M.V. Janssen and M.B.J. Stokhof (editors). *Truth, Interpretation and Information.* GRASS 3. Dordrecht, Foris, 1984.

[Sidner 85] Sidner, C.L.
Plan parsing for intended response recognition in discourse.
*Computational Intelligence* 1(1):1–10, February, 1985.

[Stallard 86] Stallard, D.G.
A Terminological Simplification Transformation for Natural Language Question-Answering Systems.
In *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, June, 1986.

[Vilain 85] Vilain, M.
The Restricted Language Architecture of a Hybrid Representation System.
In *Proceedings of IJCAI85*, pages 547–551. International Joint Conferences on Artificial Intelligence, Inc., Morgan Kaufmann Publishers, Inc., Los Angeles, CA, August, 1985.

# PROTEUS and PUNDIT: RESEARCH IN TEXT UNDERSTANDING

at
the Department of Computer Science, New York University
and
System Development Corporation -- A Burroughs Company

prepared by
Ralph Grishman
(New York University)
and
Lynette Hirschman
(System Development Corporation)

## 1. Introduction

We are engaged in the development of systems capable of analyzing short narrative messages dealing with a limited domain and extracting the information contained in the narrative. These systems are initially being applied to messages describing equipment failure. This work is a joint effort of New York University and the System Development Corp. for the Strategic Computing Program. Our aim is to create a system reliable enough for use in an operational environment. This is a formidable task, both because the texts are unedited (and so contain various errors) and because the complexity of any real domain precludes us from assembling a "complete" collection of the relationships and domain knowledge relevant to understanding texts in the domain.

A number of laboratory prototypes have been developed for the analysis of short narratives. None of the systems we know about, however, is reliable enough for use in an operational environment (the possible exceptions are expectation-driven systems, which simply ignore anything deviating from these built-in expectations). Typical success rates reported are that 75-80% of sentences are correctly analyzed, and that many erroneous analyses pass the system undetected; this is not acceptable for most applications. We see the central task of the work to be described below as the construction of a substantially more reliable system for narrative analysis.

Our basic approach to increasing reliability will be to bring to bear on the analysis task as many different types of constraints as possible. These include constraints related to syntax, semantics, domain knowledge, and discourse structure. In order to be able to capture the detailed knowledge about the domain that is needed for correct message analysis, we are initially limiting ourselves to messages about one particular piece of equipment (the "starting air compressor"); if we are successful in this narrow domain, we intend to apply the system to a broader domain.

The risk with having a rich set of constraints is that many of the sentences will violate one constraint or another. These violations may arise from problems in the messages or in the knowledge base. On the one hand, the messages frequently contain typographical or grammatical errors (in addition to the systematic use of fragments, which can be accounted for by our grammar). On the other hand, it is unlikely that we will be able to build a "complete" model of domain knowledge; gaps in the knowledge base will lead to constraint violations for some sentences. To cope with these violations, we intend to develop a "forgiving" or flexible analyzer which will find a best analysis (one violating the fewest constraints) if no "perfect" analysis is possible. One aspect of this is the use of syntactic and semantic information on an equal footing in assembling an analysis, so that

neither a syntactic nor a semantic error would, by itself, block an analysis.

## 2. Application

This work is work is a component of the Fleet Command Center Battle Management Program (FCCBMP), which is part of the Strategic Computing Program. The FCCBMP has two natural language components: one for interactive natural language access, the other for message processing. The interactive component -- which is to provide access to a data base and multiple expert systems -- is being integrated by Bolt Beranek and Newman. The message processing component is being integrated as a joint effort of New York University and the System Development Corporation.

Much of the information received by the Fleet Command Center is in the form of messages. Some of these messages have a substantial natural language component. Consequently, natural language analysis is required if the information in these messages is to be recorded in a data base in a form usable by other programs. The specific class of messages which we are studying are CASREPs, which are reports of equipment failures on board ships. These messages contain a brief narrative, typically 3 to 10 sentences in length, describing the symptoms, diagnosis, and possibly the attempts at repair of the failure. A typical narrative is shown in Figure 1. The problems we face in analyzing these messages are similar to those in analyzing short messages and reports in other technical domains, and we therefore expect that the solutions we develop will be widely applicable.

## 3. Project organization

This work is a joint research effort of New York University and the System Development Corporation. NYU has principal responsibility for development of the domain knowledge base; SDC has principal responsibility for development of the flexible parser and for the domain-independent discourse components. The division of the other tasks is noted in the detailed component descriptions below. We will also be integrating work on the knowledge base being done by SRI, which is a component technology developer for the FCCBMP natural language work.

The work by NYU is being done in LISP (primarily in COMMON LISP), as is most of the Strategic Computing research. SDC is doing its development in PROLOG because Prolog provides a powerful framework for writing grammars; it also provides the inference engine necessary for knowledge structuring and reasoning about the discourse structures in text processing. This division will permit us to make some valuable comparisons between the LISP and PROLOG development environments, and between the resulting systems.

The system being developed in LISP by NYU is called PROTEUS (PROtotype TExt Understanding System) (Grishman *et al.*, submitted for publication); the SDC system is called PUNDIT (Prolog UNDerstander of Integrated Text) (Palmer *et al.* 1986). Notwithstanding the difference in implementation languages, we have tried to maintain a high level of compatibility between the two systems. We use essentially the same grammar and have agreed on common representations for the output of the syntactic analyzer (the regularized syntactic structure) and the output of the semantic analyzer. This commonality makes is possible assign primary responsibility for the design of a component to one group, and then to take the design developed for one system and port it to the other in a straightforward way.

We are currently developing baseline systems which incorporate substantial domain knowledge but use a traditional sequential processing organization. When these systems are complete, we will begin experimenting with flexible parsing algorithms. The systems currently being developed (Figure 2) process input in the following stages: lexical look-up, parsing, syntactic regularization, semantic analysis, integration with the domain knowledge

representation, and discourse analysis. These components, and other tasks which are part of our research program, are described individually below.

## 4. System Components

### 4.1. Lexicon (SDC + NYU)

The lexicon consists of a modified version of the lexicon of the NYU Linguistic String Project, with words classified as to part of speech and subcategorized for various grammatical properties (e.g., verbs and adjectives are subclassified for their complement types).

### 4.2. Lexical acquisition (SDC)

The message vocabulary is large and will grow steadily as the system is modified to handle a wider range of equipment; several measures are planned to manage the growth of the lexicon. An interactive lexical entry program has been developed to facilitate adding words to the dictionary. Special constructions such as dates, times, and part numbers are processed using a small definite clause grammar defining special shapes. Future plans include addition of a component to use morphological analysis and selectional patterns to aid in classification of new lexical items.

### 4.3. Syntax analysis (NYU + SDC)

#### 4.3.1. Grammar

The syntactic component uses a grammar of BNF definitions with associated restrictions that enforce context-sensitive constraints on the parse. This grammar is generally modelled after that developed by the NYU Linguistic String Project (Sager 1981). The grammar has been expanded to cover the fragmentary constructions and complex noun phrases characteristic of the Navy message domain. A wide range of conjunction types is parsed by a set of conjunction rules which are automatically generated by metarules (Hirschman, in press). To serve as an interface between the syntactic and semantic components, an additional set of rules produces a normalized intermediate representation of the syntax.

#### 4.3.2. *Top-Down Parsers*

Two top-down parsers have been implemented using the common grammar just described. In each case, the analyzer applies the BNF definitions and their associated constraints to produce explicit surface structure parses of the input; the analyzer also invokes the regularization rules which produce the normalized intermediate representation.

In the NYU (LISP-based) system the basic algo ithm is a chart parser, which provides goal- directed analysis along with the recording (for possible re-use) of all intermediate goals tried. The context sensitive constraints are expressed in a version of Restriction Language (Sager 1975) which is compiled into LISP. The SDC (PROLOG-based) system uses a top-down left-to-right Prolog implementation of a version of the restriction grammar (Hirschman and Puder 1986).

### 4.4. Flexible Analyzer (SDC)

A major research focus for SDC during the first two years will be to produce a flexible analyzer that integrates application of syntactic and semantic constraints. The flexible analyzer will focus more quickly on the correct analysis and will have recovery strategies to prevent syntactic analysis from becoming a bottleneck for subsequent processing.

### 4.5. Semantic Analysis

The task of the semantic analyzer is to transform the regularized syntactic analysis into a semantic representation. This representation provides unique identifiers for specific equipment components mentioned in the text. It consists of predicates describing states and events involving the equipment, and higher-order predicates capturing the syntactically-expressed time and causal relations. Roughly speaking, the clauses from the syntactic analysis map into states and events, while the noun phrases map into particular objects (there are several exceptions, including nominalizations, e.g., "loss of pressure", and adjectives of state, such as "broken valve"). Accordingly, the semantic analysis is divided into two major parts, clause semantics and noun phrase semantics. In addition to these two main parts, a time analysis component captures the time information which can be extracted from the input.

#### 4.5.1. Clause semantics (SDC)

Semantic analysis of clauses is performed by Inference Driven Semantic Analysis (Palmer 1985), which analyzes verbs into their component meanings and fills their semantic roles, producing a semantic representation in predicate form. This representation includes information normally found in a case-frame representation, but is more detailed. The task of filling in the semantic roles is used to integrate the noun phrase analysis (described in the next section) with the clausal semantic analysis. In particular, the selection restriction information on the roles can be used to reject inappropriate referents for noun phrases.

The semantics also provides a filtering function, by checking selectional constraints on verbs and their arguments. The selectional constraints draw on domain knowledge for type and component information, as well as for information about possible relationships between objects in the domain. This function is currently used to accept or reject a completed parse. The goal for the flexible analyzer is to apply selectional filtering compositionally to partial syntactic analyses to rule out seman cally unacceptable phrases as soon as they are generated in the parse.

#### 4.5.2. Noun phrase semantics (SDC + NYU)

A noun phrase resolution component determines the reference of noun phrases, drawing on two sources: a detailed equipment model, and cumulative information regarding referents in previous sentences. SDC has concentrated on the role of prior discourse, and has developed a procedure which handles a wide variety of noun phrase types, including pronouns and missing noun phrases, using a focusing algorithm based on surface syntactic structure (Dahl, submitted for publication). NYU, as part of its work on the domain model, has developed a procedure which can identify a component in the model from any of the noun phrases which can name that component (Ksiezyk and Grishman, submitted for publication). After further development, these procedures will be integrated into a comprehensive noun phrase semantic analyzer.

#### 4.5.3. Time analysis (SDC)

SDC has started to develop a module to process time information. Sources of time information include verb tense, adverbial time expressions, prepositional phrases, co-ordinate and subordinate conjunctions. These are all mapped into a small set of predicates expressing a partial time ordering among the states and events in the message.

### 4.6. Domain model (NYU)

The domain model captures the detailed information about the general class of equipment, and about the specific pieces of equipment involved in the messages; this

information needed in order to fully understand the messages. The model integrates part/whole information, type/instance links, and functional information about the various components (Ksiezyk and Grishman, submitted for publication).

The knowledge base performs several functions: it provides the domain-specific constraints needed for the semantics to select the correct arguments for a predicate, so that modifiers are correctly attached to noun phrases. It enables noun phrase semantics to identify the correct referent for a phrase. It provides the prototype information structures which are instantiated in order to record the information in a particular message. It provides the information on equipment structure and function which is used by the discourse rules in establishing probable causal links between the sentences. And finally, associated with the components in the knowledge base are procedures for graphically displaying the status of the equipment as the message is interpreted.

These functions are performed by a large network of frames implemented using the Symbolics Zetalisp flavors system.

### 4.7. Discourse analysis (NYU)

The semantic analyzer generates separate semantic representations for the individual sentences of the message. For many applications it is important to establish the (normally implicit) intersentential relationships between the sentences. This is performed by a set of inference rules which (using the domain model) identify plausible causal and enabling relationships among the sentences. These relationships, once established, can serve to resolve some semantic ambiguities. They can also supplement the time information extracted during semantic analysis and thus clarify temporal relations among the sentences.

### 4.8. Diagnostics (NYU)

The diagnostic procedures are intended to localize the cause of failure of the analysis and provide meaningful feedback when some domain-specific constraint has been violated. We are initially concentrating on violations of local (selectional) constraints, and have built a small component for diagnosing such violations and suggesting acceptable sentence forms; later work will study more global discourse constraints.

## REFERENCES


Dahl, Deborah A. (submitted for publication). Focusing and Reference Resolution in PUNDIT.

Grishman, Ralph, Tomasz Ksiezyk, and Ngo Thanh Nhan. (submitted for publication). Model-based Analysis of Messages about Equipment.

Hirschman, Lynette and Karl Puder (1986). Restriction Grammar: A Prolog Implementation, in *Logic Programming and its Applications*, ed. D.H.D. Warren and M. VanCaneghem, pp. 244-261, Ablex Publishing Co., Norwood, N.J.

Hirschman, Lynette. (in press). "Conjunction in Meta-Restriction Grammar." *Journal of Logic Programming*.

Ksiezyk, Tomasz, and Ralph Grishman. (submitted for publication). An Equipment Model and its Role in the Interpretation of Nominal Compounds.

Palmer, Martha S. (1985) Driving Semantics for a Limited Domain. Ph.D. thesis. University of Edinburgh.

Palmer, Martha, Deborah Dahl, Rebecca Schiffman, Lynette Hirschman, Marcia Linebarger, and John Dowding. (1986) Recovering Implicit Information. To appear in *Proc. 24th Annl. Meeting Assn. Computational Linguistics*.

Sager, Naomi and Ralph Grishman (1975). The Restriction Language for Computer Grammars of Natural Language. *Comm. of the ACM*, vol. 18, pp. 390-400.

Sager, Naomi (1981). *Natural Language Information Processing: A Computer Grammar of English and its Applications*. Addison-Wesley, Reading, MA.

# A Sample CASREP

## about a SAC (Starting Air Compressor)

DURING NORMAL START CYCLE OF 1A GAS TURBINE, APPROX 90 SEC AFTER CLUTCH ENGAGEMENT, LOW LUBE OIL AND FAIL TO ENGAGE ALARM WERE RECEIVED ON THE ACC. (ALL CONDITIONS WERE NORMAL INITIALLY). SAC WAS REMOVED AND METAL CHUNKS FOUND IN OIL PAN. LUBE OIL PUMP WAS REMOVED AND WAS FOUND TO BE SEIZED. DRIVEN GEAR WAS SHEARED ON PUMP SHAFT.

Figure 1

# PROTEUS/PUNDIT SYSTEM STRUCTURE

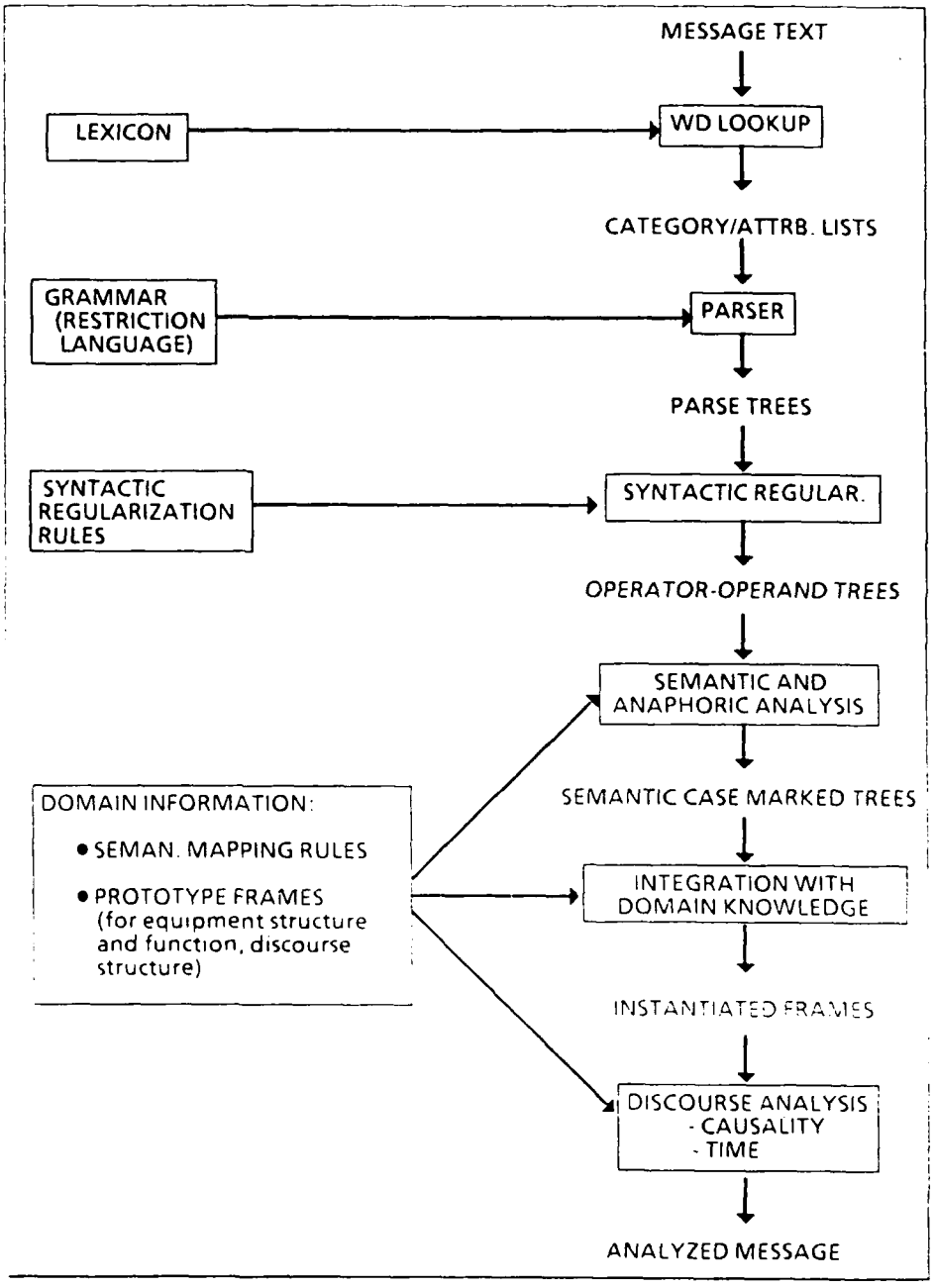



Figure 2

# Overview of the TACITUS Project

Jerry R. Hobbs
Artificial Intelligence Center
SRI International

## 1 Aims of the Project

The specific aim of the TACITUS project is to develop interpretation processes for handling casualty reports (casreps), which are messages in free-flowing text about breakdowns of machinery.[1] These interpretation processes will be an essential component, and indeed the principal component, of systems for automatic message routing and systems for the automatic extraction of information from messages for entry into a data base or an expert system. In the latter application, for example, it is desirable to be able to recognize conditions in the message that instantiate conditions in the antecedents of the expert system's rules, so that the expert system can reason on the basis of more up-to-date and more specific information.

More broadly, our aim is to develop general procedures, together with the underlying theory, for using commonsense and technical knowledge in the interpretation of written discourse. This effort divides into five subareas: (1) syntax and semantic translation; (2) commonsense knowledge; (3) domain knowledge; (4) deduction; (5) "local" pragmatics. Our approach in each of these areas is discussed in turn.

## 2 Syntax and Semantic Translation

Syntactic analysis and semantic translation in the TACITUS project are being done by the DIALOGIC system. DIALOGIC has perhaps as extensive a coverage of English syntax as any system in existence, it produces

[1]The TACITUS project is funded by the Defense Advanced Research Projects Agency under Office of Naval Research contract N00014-85-C-0013, as part of the Strategic Computing program.

a logical form in first-order predicate calculus, and it was used as the syntactic component of the TEAM system. The principal addition we have made to the system during the TACITUS project has been a menu-based component for rapid vocabulary acquisition, that allows us to acquire several hundred lexical items in an afternoon's work. We are now modifying DIALOGIC to produce neutral representations instead of multiple readings for the most common types of syntactic ambiguities, including prepositional phrase attachment ambiguities and very compound noun ambiguities.

# 3 Commonsense Knowledge

Our aim in this phase of the project is to encode large amounts of commonsense knowledge in first-order predicate calculus in a way that can be used for knowledge-based processing of natural language discourse. Our approach is to define rich core theories of various domains, explicating their basic ontologies and structure, and then to define, or at least to characterize, various English words in terms of predicates provided by these core theories. So far, we have alternated between working from the inside out, from explications of the core theories to characterizations of the words, and from the outside in, from the words to the core theories. Thus, we first proceeded from the outside in by examining the concept of "wear", as in "worn bearings", seeking to define "wear", and then to define the concepts we defined "wear" in terms of, pushing the process back to basic concepts in the domains of space, materials, and force, among others. We then proceeded from the inside out, trying to flesh out the core theories of these domains, as well as the domains of scalar notions, time, measure, orientation, shape, and functionality. Then to test the adequacy of these theories, we began working from the outside in again, spending some time defining, or characterizing, the words related to these domains that occurred in our target set of casreps. We are now working from the inside out again, going over the core theories and the definitions with a fine-tooth comb, checking manually for consistency and adequacy and proving simple consequences of the axioms on the KADS theorem-prover. This work is described in an enclosed publication [1].

# 4 Domain Knowledge

In all of our work we are seeking general solutions that can be used in a wide variety of applications. This may seem impossible for domain knowledge. In our particular case, we must express facts about the starting air compressor of a ship. It would appear difficult to employ this knowledge in any other application. However, our approach makes most of our work even in this area relevant to many other domains. We are specifying a number of "abstract machines" or "abstract systems", in levels, of which the particular device we must model is an instantiation. We define, for example, a "closed producer-consumer system". We then define a "closed clean fluid producer-consumer system" as a closed producer-consumer system with certain additional properties, and at one more level of specificity, we define a "pressurized lube-oil system". The specific lube-oil system of the starting air compressor, with all its idiosyncratic features, is then an instantiation of the last of these. In this way, when we have to model other devices, we can do so by defining them to be the most specific applicable abstract machine that has been defined previously, thereby obviating much of the work of specification. An electrical circuit, for example, is also a closed producer-consumer system.

# 5 Deduction

The deduction component of the TACITUS system is the KLAUS Automated Deduction System (KADS), developed as part of the KLAUS project for research on the interactive acquisition and use of knowledge through natural language. Its principal inference operation is nonclausal resolution, with possible resolution operations encoded in a connection graph. The nonclausal representation eliminates redundancy introduced by translating formulas to clause form, and improves readability as well. Special control connectives can be used to restrict use of the formulas to either forward chaining or backward chaining. Evaluation functions determine the sequence of inference operations in KADS. At each step, KADS resolves on the highest-rated link. The resolvent is then evaluated for retention and links to the new formula are evaluated for retention and priority. KADS supports the incorporation of theories for more efficient deduction, including deduction by demodulation, associative and commutative unification, many-sorted unification, and theory resolution. The last of these has been used for efficient deduction using a sort hierarchy. Its efficient methods for performing some reasoning about sorts and equality and the facility for or-

dering searches by means of an evaluation function make it particularly well suited for the kinds of deductive processing required in a knowledge-based natural language system.

# 6 Local Pragmatics

We have begun to formulate a general approach to several problems that lie at the boundary between semantics and pragmatics. These are problems that arise in single sentences, even though one may have to look beyond the single sentence to solve them. The problems are metonymy, reference, the interpretation of compound nominals, and lexical and syntactic ambiguity. All of these may be called problems in "local pragmatics". Solving them constitutes at least part of what the interpretation of a text is. We take it that interpretation is a matter of reasoning about what is possible, and therefore rests fundamentally on deductive operations. We have formulated very abstract characterizations of the solutions to the local pragmatics problems in terms of what can be deduced from a knowledge base of commonsense and domain knowledge. In particular, we have devised a general algorithm for building an expression from the logical form of a sentence, such that a constructive proof of the expression from the knowledge base will constitute an interpretation of the sentence. This can be illustrated with the sentence from the casreps

Disengaged compressor after lube oil alarm.

To resolve the reference of "alarm" one must prove constructively the expression

$$(\exists x)alarm(x)$$

To resolve the implicit relation between the two nouns in the compound nominal "lube oil alarm" (where "lube oil" is taken as a multiword), one must prove constructively from the knowledge base the existence of some possible relation, which we may call $nn$, between the entities referred to by the nouns:

$$(\exists x, y)alarm(x) \wedge lube\text{-}oil(y) \wedge nn(y, x)$$

A metonymy occurs in the sentence in that "after" requires its object to be an event, whereas the explicit object is a device. To resolve a metonymy that occurs when a predicate is applied to an explicit argument that fails to

satisfy the constraints imposed by the predicate on its argument, one must prove constructively the possible existence of an entity that is related to the explicit argument and satisfies the constraints imposed by the predicate. Thus, the logical form of the sentence is modified to

$$\ldots \wedge after(d,e) \wedge q(e,x) \wedge alarm(x) \wedge \ldots$$

and the expression to be proved constructively is

$$(\exists e)event(e) \wedge q(e,x) \wedge alarm(x) \wedge \ldots$$

In the most general approach, $nn$ and $q$ are predicate variables. In less ambitious approaches, they can be predicate constants, as illustrated below.

These are very abstract and insufficiently constrained formulations of solutions to the local pragmatics problems. Our further research in this area has probed in four directions.

(1) We have been examining various previous approaches to these problems in linguistics and computational linguistics, in order to reinterpret them into our framework. For example, an approach that says the implicit relation in a compound nominal must be one of a specified set of relations, such as "part-of", can be captured by treating "nn" as a predicate constant and by including in the knowledge base axioms like

$$(\forall x,y)part\text{-}of(y,x) \supset nn(x,y)$$

In this fashion, we have been able to characterize succinctly the most common methods used for solving these problems in previous natural language systems, such as the methods used in the TEAM system.

(2) We have been investigating constraints on the most general formulations of the problems. There are general constraints, such as the Minimality Principle, which states that one should favor the minimal solution in the sense that the fewest new entities and relations must be hypothesized. For example, the argument-relation pattern in compound nominals, as in "lube oil pressure", can be seen as satisfying the Minimality Principle, since the implicit relation is simply the one already given by the head noun. In addition, we are looking for constraints that are specific to given problems. For example, whereas whole-part compound nominals, like "regulator valve", are quite common, part-whole compound nominals seem to be quite rare. This is probably because of a principle that says that noun modifiers should further restrict the possible reference of the noun phrase, and parts are common to too many wholes to perform that function.

(3) A knowledge base contains two kinds of knowledge, "type" knowledge about what kinds of situations are possible, and "token" knowledge about what the actual situation is. We are trying to determine which of these kinds of knowledge are required for each of the pragmatics problems. For example, reference requires both type and token knowledge, whereas most if not all instances of metonymy seem to require only type knowledge.

(4) At the most abstract level, interpretation requires the constructive proof of a single logical expression consisting of many conjuncts. The deduction component can attempt to prove these conjuncts in a variety of orders. We have been investigating some of these possible orders. For example, one plausible candidate is that one should work from the inside out, trying first to solve the reference problems of arguments of predications before attempting to solve the compound nominal and metonymy problems presented by those predications. In our framework, this is an issue of where subgoals for the deduction component should be placed on an agenda.

# 7 Implementation

In our implementation of the TACITUS system, we are beginning with the minimal approach and building up slowly. As we implement the local pragmatics operations, we are using a knowledge base containing only the axioms that are needed for the test examples. Thus, it grows slowly as we try out more and more texts. As we gain greater confidence in the pragmatics operations, we will move more and more of the axioms from our commonsense and domain knowledge bases into the system's knowledge base. Our initial versions of the pragmatics operations are, for the most part, fairly standard techniques recast into our abstract framework. When the knowledge base has reached a significant size, we will begin experimenting with more general solutions and with various constraints on those general solutions.

# 8 Future Plans

In addition to pursuing our research in each of the areas described above, we will institute two new efforts next year. First of all, we will begin to extend our work in pragmatics to the recognition of discourse structure. This problem is illustrated by the following text:

> Air regulating valve failed.
> Gas turbine engine wouldn't turn over.

Valve parts corroded.

The temporal structure of this text is 3-1-2; first the valve parts corroded, and this caused the valve to fail, which caused the engine to not turn over. To recognize this structure, one must reason about causal relationships in the model of the device, and in addition one must recognize patterns of explanation and consequence in the text.

The second new effort will be to build tools for domain knowledge acquisition. These will be based on the abstract machines in terms of which we are presently encoding our domain knowledge. Thus, the system should be able to allow the user to choose one of a set of abstract machines and then to augment it with various parts, properties and relations.

## Researchers

The following researchers are participating in the TACITUS project: John Bear, William Croft, Todd Davies, Douglas Edwards, Jerry Hobbs, Kenneth Laws, Paul Martin, Fernando Pereira, Raymond Perrault, Stuart Shieber, Mark Stickel, and Mabry Tyson.

## Publication

1. Hobbs, Jerry R., William Croft, Todd Davies, Douglas Edwards, and Kenneth Laws, "Commonsense Metaphysics and Lexical Semantics", *Proceedings, 24th Annual Meeting of the Association for Computational Linguistics*, New York, June 1986., pp. 231-240.

# The Counselor Project at the University of Massachusetts

David D. McDonald & James D. Pustejovsky
Department of Computer and Information Science
University of Massachusetts,
Amherst, Massachusetts 01003

Participants in the Counselor Project, Fall 1984 through Summer 1986:

Principal Investigators: Edwina L. Rissland, David D. McDonald, Wendy G. Lehnert

Research Associates: Beverly Woolf, James D. Pustejovsky

Graduate Students: Marie M. Vaughan, Brian Stucky, Penelope Sibun, Seth Rosenberg, Kelly Murray, Kevin Gallagher, JoAnn M. Brooks, John Brolio, Sabine Bergler, Kevin D. Ashley, Scott D. Anderson

## Introduction

The COUNSELOR PROJECT began in the fall of 1984 with the goal of exploring basic problems in discourse structure and text processing within an integrated interface to a strong expert system. The program that we have developed, COUNSELOR, integrates separately developed components for natural language generation (MUMBLE see [7], [8], [9]), parsing (PLUM [5]), and case-based legal reasoning (HYPO [1], [2]). It adds a newly developed component, CICERO ([10]), positioned between the two text processors and the expert system; CICERO is responsible for managing textual inferences ("reading between the lines") by using common sense models of legal events. COUNSELOR can provide advise to an attorney about how to argue cases involving violations of trade secret law in the computer field. The attorney presents the facts of their case to the system, which may ask questions to elicit other facts that it knows to be relevant. The system then suggests lines of argument that the attorney might use, drawing on its library of litigated cases to find ones with analogous dimensions.

At its present state of development, COUNSELOR can handle simple variations on a single scenerio, exemplified by the following dialog:

| | |
|---|---|
| User: | **I represent a client named HackInc, who wants to sue SwipeInc and Leroy Soleil for misappropriating trade secrets in connection with software developed by my client. HackInc markets the software, known as Autotell, a program to automate some of a bank teller's functions, to the banking industry.** |
| Counselor: | **Did Soleil work for HackInc.?** |
| User: | **Yes, he was a key employee on the Autotell project.** |
| Counselor: | **Did he later work for SwipeInc.?** |
| User: | **Yes.** |
| Counselor: | **You can argue that there is an implied agreement arising out of Soleil's employment with HackInc. that he not disclose any trade secret information to which he gained access by virtue of his employment.** |

## Motivations

*Consequential results in natural language research will only come from working* with a strong underlying program whose communicative needs will challenge the capabilities of state of the art of language interfaces. As a group, we are not interested in building yet another question answering system: our goal is to understand the structure of discourse. We believe that an effective place to begin is with task specific, mixed initiative dialog where the particiants' goals cannot be satisfied by single utterances.

Working with a legal reasoning system like Kevin Ashley and Edwina Rissland's HYPO provides particular challenges to natural language research:

(1) Legal text is structurally complex. The need to avoid ambiguity leads to deeply *embedded clauses and heavy noun phrases.*

(2) As both the user and the system have a thorough knowledge of the law, they communicate vastly more information in conversations about legal arguments than ever appears in their literal utterances.

(3) HYPO's role as an advisory system creates a natural motivation to communicate through language.

(4) Legal cases are large, complex objects that can be viewed from many alternative perspectives. The purpose for which a case is being described strongly influences which of its attributes are salient and how that information should be structured as a text.

## Component Parts

We began the project with three partially developed components, HYPO, MUMBLE, and PLUM, each designed with independent motivations. An initial tension was whether to convert aspects of these programs that did not seem apt in their new setting, or alternatively to interpose new components between them to smooth out the differences. We concluded that the motivations underlying each component were strong enough that we should not change them just because they were now working together.

**HYPO** reasons with cases and hypotheticals. Actually litigated legal cases are encoded and indexed by "dimensions", which capture the utility of a case for making a particular kind of argument. When evaluating new cases, HYPO first analyzes them in terms of the dimensions they involve. Relevant cases are then retrieved to guide the reasoning. The system may ask pertinent questions about facts now found to be relevant. When the analysis is complete, HYPO describes the arguments available to the user, and responses and counter responses that may follow.

**MUMBLE**, the linguistic component for generation, is responsible for realizing conceptual specifications as grammatical text cohesive with the discourse which proceeds it. MUMBLE works within a description directed framework. Its input specification is a description of the message the underlying program wants to

communicate. This description is executed incrementally, producing an intermediate linguistic representation which defines the text's grammatical relations and imposes constraints on further realization. This surface structure description is concurrently executed, producing the actual text.

PLUM is a conceptual analyzer which has been given a well defined schematic structure so that it can be easily extended. It parses by doing prediction and completion over semantic concepts implied by the words rather than over syntactic categories. As in other conceptual analyzers, no explicit surface structure is recovered. PLUM's output is the set of completed frames.

CICERO is a new component, a discourse and inference manager between the language components and the expert system. From the understanding side, CICERO must integrate the clause by clause output of the parser into the larger discourse context, recognizing, for example, when noun phrases refer to the same object. In interpreting these small, lexically derived frames, CICERO draws on its own representation of events which bridges the gap between the way such information is expressed in language and the way it is organized for expert legal reasoning. For generation, CICERO is responsible for planning the message that is given to the generator. In particular, it determines what information should be included and what may be omitted as inferable, and it selects pivotal lexical items with appropriate perspective and rhetorical force.

## Future Directions

While the accomplishments of the individual components of Counselor are interesting in their own right, the greatest effect of the project has been to provide a workbench for studying the problems of language in an integrated context. Perennial problems in anaphora, lexical semantics, aspect, etc. become more tractable in an integrated system where there is a discourse context and intensional motivation. There are also semantic generalizations between the level at which the text processors operate and the level of the expert system which are more easily captured when parsing and generation can be studied in unison. On a larger scale, an explicit discourse manager, a requisite for more complex dialogs, can only be developed once an integrated system exists.

## References


[1] Ashley, Kevin D. (1986) "Modelling Legal Argument: Reasoning with cases and hypotheticals -- a thesis proposal", Technical Report 10, The Counselor Project, Department of Computer and Information Science, University of Massachusetts at Amherst.

[2] Ashley, Kevin D. and Edwina L. Rissland (1985) "Toward Modelling Legal Argument", Proceedings of the 2nd International Congress LOGICA, INFORMATICA, DIRITTO, Instituto Per La Documentazione Giuridica, Florence, Italy.

[3] Brooks, JoAnn M. (1985) "Themis: A Discourse Manager",unpublished Master's thesis, Department of Computer and Information Science, University of Massachusetts at Amherst.

[4] Gallagher, Kevin (1986) "The Design and Implementation of CICERO", unpublished Master's thesis, Department of Computer and Information Science, University of Massachusetts at Amherst.

[5] Lehnert, Wendy G. and Seth Rosenberg (1985) "The PLUM User's Manual" Technical Report 1, The Counselor Project, Department of Computer and Information Science, University of Massachusetts at Amherst.

[6] McDonald, David D. (1986) "Natural Language Generation: Complexities and Techniques", to appear in Nirenburg (ed.) *Theoretical and Methodological Issues in Machine Translation*, Cambridge University Press.

[7] McDonald, David D. and James Pustejovsky (1985) "Description Directed Natural Langauge Generation", Proceedings of IJCAI-85, pp. 799-805.

[8] McDonald, David D. and James Pustejovsky (1985) "TAGs as a Grammatical Formalism for Generation", Proceedings of the 23rd Meeting of the Association for Computational Linguistics, pp. 94-103.

[9] McDonald, David D. and James Pustejovsky (1985) "SAMSON: A Computational Theory of Prose Style for Natural Language Generation", Proceedings of the 1985 meeting of the European Association for Computational Linguistics.

[10] Pustejovsky, James (1986) "An Integrated Theory Discourse Analysis", Technical Report 11, The Counselor Project, Department of Computer and Information Science, University of Massachusetts at Amherst.

[11] Rissland Edwina L., Edward Valcarce, and Kevin Ashley (1984) "Explaining and Arguing with Examples", Proceedings of AAAI-84.

[12] Vaughan, Marie M. and David D. McDonald (1986) "A Model of Revision in Natural Language Generation", Proceedings of the 24th Meeting of the Association for Computational Linguistics.

# RESEARCH IN NATURAL LANGUAGE PROCESSING

**University of Pennsylvania**
**Department of Computer and Information Science**

This a brief report summarizing our work to date, our intermediate and long term goals, and a summary of some of our publications.

| | |
|---|---|
| FACULTY | Aravind Joshi, Tim Finin, Dale Miller, Lokendra Shastri, and Bonnie Webber |
| STUDENTS | Brant Cheikes, John Dowding, Amy Felty, Ellen Hays, Robert Kass, Ron Katriel, Sitaram Lanka, Megan Moser, Gopalan Nadathur, MaryAngela Papalaskaris, Martha Pollack, Robert Rubinoff, Yves Schabes, Ethel Schuster, Sunil Shende, Jill Smudski, Vijayshankar, David Weir, Blair Whitaker |
| FACILITIES | LINC (Langauge, Information, and Computation) laboratory, which consists of a dedicated VAX 11/785, 10 Symbolics Lisp machines, 7 HP 68020 based AI workstations, a SUN workstation, several Macintoshes, and a laser printer. These machines are networked together and to other research facilities in the department. |

## MAJOR THRUST

Natural language interfaces providing support for many different communicative functions.

- Providing definitions of concepts
- Recognizing and correcting user misconceptions
- Providing explanations
- Offering to provide information later, when known
- Verifying and demonstrating understanding
- Exploiting and enriching the context of natural language discourse between user and system.

## WORK-TO-DATE

- Integration of RUS-TEXT-MUMBLE (RTM) - This effort involves integrating three natural language system components (BBN's RUS parser-interpreter, McKeown's TEXT system (developed at Penn), and McDonald's MUMBLE system (received from U. Mass in January 1985). This integration of three independently developed systems has required suostantial effort. The version of RTM (to be completed in May 1986) [1] accepts a limited number of English language requests for definitions of, descriptions of, or comparisions between terms in the ONR database used by Kathy McKeown in her development of TEXT; [2] formulates appropriate reponses using TEXT and outputs those responses in English using MUMBLE; and [3] runs on a SYMBOLICS Lisp machine. This work has been done by Moser, Whitaker and Rubinoff.
- Initial work on incorporating a sense of *relevance* in monitor offers. Mays' dissertation work on monitor offers was limited to issues of *competancy*. This work is being done by Cheikes and Schabes.
- Completion of McCoy's dissertation work on correcting certain types of object-related misconceptions and implementation of a system called ROMPER which generates such corrections. (MUMBLE is used as the tactical generation component of this system as well.)
- Completion of Hirschberg's dissertation work on scalar implicatures and their use in constructing non-misleading responses.
- Completion of Pollack's dissertation work on plan inference in which user's and system's beliefs about actions and plans is decoupled.
- Continuation of work on integrating scalar-implicature-based reasoning within a general framework of circumscription-based non-monotonic reasoning.
- Development of methods for converting proofs in a system akin to first-order resolution into natural deduction (ND) proofs, which are then reorganized into cohesive paragraphs using Chester's 1976 algorithm.

- Development of methods of converting modal resolution proofs into modal ND proofs and higher-order resolution proofs into higher-order ND proofs.
- Initial development of domain-independant tools for expressing and reasoning about user models - in particular, for defining hierachies of stereotypical users, representing individual users, and drawing inferences about them using a default logic.
- Continuation of basic research on local coherence of discourse using the notions of centering and syntax, semantics, and parsing of tree adjoining grammars.

## FUTURE PLANS

Having gained the experience of integrating three natural language systems and carrying out some of the basic research as described in the previous section, we have now developed the plan described below, which summarizes the near term and long term goals.

### Near Term Goals

We have three tangible goals for the next year:

- Completing the RTM demonstration system (using the existing domain and knowledge representation) and producing a videotape which explains and demonstrates it.
- Developing TEXT into a more modular tool for defining and comparing terms, on the order of RUS and MUMBLE. This will eliminate its tie to a particular knowledge representation and increase its portability.
- Acquiring familiarity with the PENMAN approach to NL generation through acting as a beta-test site for NIGEL.

### Long Term Goals

#### Support for NL Definitions - Enriched Knowledge Representation

In our original proposal, we stated our intention of employing a richer knowledge representation as the basis for our work on text generation, especially for constructing definitions. Our original idea was to make use of BBN's NIKL system. In the past year though, we have become aware of some of NIKL's limitations, which essentially make it non-optimal, even as a next step, for our text generation work. On the other hand, we have identified several features with which a NIKL-like language could be enriched to make it more suitable for our work:

- associating non-definitional information with concepts in a way that maintains the underlying structure of that information, without interferring with NIKL's automatic classification mechanism.
- associating "evidential" information with concepts, especially frequency information - how often the concept is known to display particular features.
- allowing for what appears to be conflicting information coming down through inheritance - e.g., information that is contrary to expectations grounded in an alternative perspective on a concept
- allowing mutual definition of concepts - each being defined with reference to the others in a set
- incorporating notions of time and change - allowing the defining properties and evidential properties of concepts to include how they change over time
- allowing assertions about usual relations between properties of subtypes

Work on an enriched knowledge representation that includes all these features in a well-motivated way will take several years. However one that includes at least the first three of them can probably be developed over the next two years, with work on employing it in text generation beginning after the first six months to a year after the start of that work.

#### Support of NL Definitions - Use of Discourse and User Models

The TEXT system, as it is currently structured, will produce the same definition for a concept (or comparison between two concepts) whenever it is asked. It does not take into account what the user may have already found out about the concept, or what it is implicitly being contrasted with (e.g., some other concept the user has recent'y asked about), or what the user's goal is in making the request. Hence, other directions in which we would like to take this definitional/clarificational capability is to increase its sensitivity to (1) the discourse history, to avoid repetition and possibly to take advantage of the additional clarity brought by contrasting a new term to one explained before; (2) the user's level of expertise, to avoid either stating the obvious or going more deeply into a concept than the user can understand; and (3) the user's goals, to focus on those aspects of the concept being defined (or concepts being compared) which are significant to the current task. (The latter is related to the notion of "perspective" used in Kathy McCoy's recent thesis here.) For both these apects of user modelling (in contrast with the first point, which can be developed using the current discourse alone), we will draw on the other work being done here on domain-independent user-modelling mechanisms. This proposed work must be done in a domain in which tasks can be characterized and recognized. Thus we plan to do this initially in investment advising domain that we have started to develop. Work on

incorporating and using discourse history will involved about a one-year effort, once the knowledge base is built. Work on incorporating and using a model of a user's expertise and goals will take more time, on the order of two to three years.

## Explanations

Again in our original proposal, we proposed work on constructing natural language explanations - more specifically, on ways to loosen the current tight coupling between the form of the system's proof of some statement to the form of its explanation of why the statement is true. This coupling has kept systems which should be able to explain their reasoning from employing stronger proof methods which do not have a natural, understandable form of presentation to their human users.

Our immediate goals involve:

- developing a demonstration system which responds to NL queries posed to RUS by doing an efficient first-order resolution-based proof, transforming that proof into an ND proof, organizing that proof according to an improved version of the Chester algorithm, and then producing an English version of the text using MUMBLE or NIGEL.
- abstracting from the three separate sets of proof conversion methods (noted under WORK-TO-DATE) into general methods of transforming any resolution-style proof in any logic into its corresponding ND proof.
- determining whether existing methods of organizing first-order ND proofs into paragraphs are applicable to ND proofs in these stronger logics or whether more must be done to produce high-quality, cohesive, understandable text.

Our long-term goals remain as stated in our original proposal - the production of explanations sensitive to users' beliefs, expertise, desired level of detail and expectations. In this long-term research, we see taking expertise and desired level of detail into account in determining how much of the ND proof is made explicit. Of more interest is how users' beliefs and expectations should affect the explanations. Work on scientific explanation has shown that central to the explanation of what is the case is a set of alternative situations which are not the case. One explains what is in contrast to what is not. However, this requires additional work, to prove of each of the alternatives (which may be given explicitly by the user - "why this and not that?" - or inferred from the system's model of the user's expectations) that it is not true. Our planned approach involves guiding the (failing) proof of each alternative against the successful proof. The point is that although there may be many failing proofs of each alternative, the most relevant of these in the current situation is the one which is analogous - up to the point of failure - to the original successful proof not only should this technique provide relevant information, but is should also be efficient in reducing the search space. We expect this work to take on the order of two to three years, provided we have enough resources to pursue it in parallel with our more near-term goals.

## Natural Language Parsing and Generation

While continuing to use the RUS system, we will continue our work on tree adjoining grammar (TAG) both from the parsing and generation points of view. TAGs lead to some attractive approaches to parallelizing parsing and also seem to provide natural planning units for generation. This work will be integrated with our future work on parsing and generation. Our first language generator (used by TEXT) was one based on Kay's Functional Unification Grammar. While theoretically elegant, it was unacceptably slow (in its straightforward implementation), leading us last year to import the MUMBLE generator from McDonald at University of Massachusetts and adapt it to work with TEXT. Using MUMBLE has produced a 60-fold speed-up in generation time. However, adapting MUMBLE to work with TEXT and, independently, with two other systems has made us aware of MUMBLE's limitations, primarily its lack of knowledge of words or grammar. Essentially, MUMBLE's knowledge is limited to how to realize particular message units (i.e., to choose an acceptable one from an *a priori* specified set of choices), given constraints already imposed by message units that have already been realized. The large amount of work that must be invested in building a MUMBLE lexicon and the lack of inter-application portability of anything but the control structure comes from this fact - that one has to completely specify each set of choices beforehand for each message unit and *the sets are completely application specific*. We propose to work on the development of a new architecture, including our work on tags, that avoids these limitations by having more knowledge of syntax and words and hence is more portable between applications. The time frame for this project is approximately three years.

## Anaphora Resolution

The RUS parser/interpreter we received from BBN uses a limited method of resolving definite pronouns and noun phrases that is only a bit more advanced than the one originally developed for BBN's LUNAR system back in 1971. Since then, there have been major theoretical advances in our understanding of discourse anaphora (in the works of Grosz (at SRI), Joshi, Sidner (at BBN), Webber, and Weinstein), but these theoretical advances have not yet found their way into natural language understanding systems. We feel strongly qualified to undertake this work, having two of the major participants (Joshi and Webber) here at Penn already, and want to do so. For us, it is both of research interest and of practical importance, since it can mean a major improvement in system's understanding abilities. We will also integrate our work on tags with this effort as it relates to parsing and generation. This work will also complement additional work being done here on a theoretical and computational account of anaphoric reference to actions and events. We see this work as taking about two to two and a half years.

## User Modeling

The need for systems to model the knowledge and beliefs of their users has already been pointed out. We plan to address a number of issues which underly the succesful development and encorporation of explicit user models. Our current domain-independent user-modelling system, GUMS, provides mechanisms for defining hierachies of stereotypical users, representing individual users, and drawing inferences about them using a rich default logic. We will continue to develop this system as a tool which will support the user modeling needs of various applications. We also plan to study the problem of how new knowledge of individual users can be derived from their regular interaction - that is, how relevent information about users can be inferred from their queries and responses. In other situations it may become necessary for the system to explicitly pose a few crucial questions to the user to determine what he or she does and does not know.

## System Integration

Finally, we plan to begin work on system integration. In recent years, we have identified many types of behavior that interfaces to database systems and expert systems should demonstrate. Beginning with Kaplan's work on recognizing and responding to existential presupposition failures in his COOP system, we have developed and produced several modules, each demonstrating another type of desired behavior. These include the ability to recognize and respond to type failures, the ability to respond to object-related misconceptions, the ability to calculate and offer competant database monitors, the ability to use scalar implicatures to convey additional information, and the ability to respond to a class of "inappropriate" queries, and various paraphrase abilities.

Following the publication of Kaplan's thesis, the features of his COOP system were soon incorporated into several database interfaces (both natural slanguage and formal query language). This gave the resulting systems the ability to give two types of responses: either a direct answer, if there was one, or a statement concerning the absense of individuals satisfying some description in the given query. Now we plan to tackle the more significant problem related to this:

Given a system that is able to call upon a variety of response strategies, how does it decide what to do in a given circumstance? This is the issue we plan to explore by investigating the integration of multiple communicative behaviors. Given a system with several different types of useful behaviors, which can be combined in various ways, can one efficiently and effectively coordinate a response that is better (i.e., more useful, more helpful and more understandable) than simply a (direct) answer. While we speculate that it will be the case that identifying what one might consider the *best* response might take complex reasoning about the user's goals, level of expertise and need-to-know with respect to what the answer (if any) actually is, we also plan to look at how, with more limited resources, we can still improve system behavior.

This aspect of our future plans is the most long term, involving both the actual component integration itself (in which, in many cases, it is only the basic ideas that can be carried over, where the component must be re-programmed entirely to fit into the integrated system) and the development of that part of the total system that reasons about what kind of response(s) to give. The time frame here is approximately four years.

## Architecture

We plan to investigate parallel and connectionist architectures and algorithms for realizing our systems, especially those for knowledge representation, reasoning, explanations, and integrated parsing and generation.

# Abstracts of Recent Technical Reports

***INTERACTIVE CLASSIFICATION A Technique for the Aquisition and Maintenance of Knowledge Bases*, Tim Finin and David Silverman, MS-CIS-84-17.**

The practical application of frame-based knowledge-based systems, such as in expert systems, requires the maintenance of potentially very large amounts of declarative knowledge stored in their knowledge bases (KBs). As a KB grows in size and complexity, it becomes more difficult to maintain and extend. Even someone who is familiar with the representation and the contents of the existing KB may introduce inconsistencies and errors whenever an addition or modification is made.

This paper describes an approach to this problem based on a tool called an interactive classifier. An interactive classifier uses the contents of the existing KB and knowledge about its representation to assist the person who is maintaining the KB in describing new KB objects. The interactive classifier will identify the appropriate taxonomic location for the newly described object and add it to the KB. The new object is allowed to be a generalization of existing KB oojects, enabling the system to learn more about existing objects. The ideas have been tested in a system call KuBIC, for Knowledge Base Interactive Classifier, and are being extended to a more complete knowledge representation language.

***Correcting Object-Related Misconceptions: How Should The System Respond?*, Kathleen F. McCoy, MS-CIS-84-18.**

This paper describes a computational method for correcting users' misconceptions concerning the objects modeled by a computer system. The method involves classifying object-related misconceptions according to the knowledge-base feature involved in the incorrect information. For each resulting class sub-types are identified, according to the structure of the knowledge base, which indicate what information may be supporting the misconception and, therefore, what information to include in the response. Such a characterization, along with a model of what the user knows, enables the system to reason in a domain-independent way about how best to correct the user.

*Default Reasoning in Interaction*, **Aravind Joshi, Bonnie Webber, and Ralph Weischedel, MS-CIS-84-58**

Nonmonotonic reasoning is usually studied in the context of a logical system in its own right or as reasoning done by an agent, in which the agent reasons about the world from partial information and hence may draw conclusions unsupported by traditional logic. The main point of departure here is looking at nonmonotonic reasoning in the context of interacting with another agent. This information is partial, in that the other agent neither will not can make everything explicit. Knowing this, the agent may attempt to derive more from the interaction than what has been made explicit, by reasoning by default about what has been made explicit (often by contrast with what he assumes would have been made explicit, were something else the case). Thus there can be rules for default reasoning that are operative in the interactive situation ("interactional defaults") that are not operative with only a single agent.

*Preventing False Inferences*, **Aravind Joshi, Bonnie Webber, and Ralph M. Weischedel, MS-CIS-84-59**

In cooperative man-machine interaction, it is taken as necessary that a system truthfully and informatively respond to a user's question. It is not, however, sufficient. In particular, if the system has reason to believe that its planned response might lead the user to draw an inference that it knows to be false, then it must block it by modifying or adding to its response. The problem is that a system neither can nor should explore all conclusions a user might possibly draw: its reasoning must be constrained in some systematic and well motivated way.

*Living Up To Expectations: Computing Expert Responses* **Aravind Joshi, Bonnie Webber, and Ralph Weischedel, MS-CIS-84-60**

In cooperative man-machine interaction, it is necessary ***but not sufficient*** for a sustem to respond truthfully and informatively to a user's question. In particular, if the system has reason to believe that its planned response might mislead the user, then it must block that conclusion by modifying its response. This paper focusses on identifying and avoiding potentially misleading responses by acknowledging types of "informing behavior" usually expected of an expert. We attempt to give a formal account of several types of assertations that should be included in response to questions concerning the achievement of some goal (in addition to the simple answer), lest the questioner otherwise be misled.

*A Modal Temporal Logic for Reasoning About Changing Databases with Applications to Natural Language Question Answering*, Eric Mays, Aravind Joshi, Bonnie Webber, MS-CIS-85-01.

A database which models a changing world must evolve in correspondence to the world. Previous work on natural language question answering systems for databases has largely ignored the issues which arise when the database is viewed as a dynamic (rather than a static) object. We investigate the question answering behaviors that become possible with the ability to represent and reason about the possible evolution of a database. These behaviors include offering to monitor for a possible future state of the database as an indirect response to a query, and directly answering questions about prior and future possibility. We apply a propositional modal temporal logic that captures possibility and temporality to represent and reason about dynamic databases, and present a sound axiomatization and proof and proof procedure.

*Explaining Concepts in Expert Systems: The CLEAR System*, **Robert Rubinoff, MS-CIS-85-06, LINC LAB 02**

Existing expert systems provide limited explanatory ability. They can explain the specific reasoning the system uses, but if the user is confused about the concepts and terms the system is using, no help is available. The CLEAR system allows users to ask for explanations of specific concepts. The system generates the explanations by examining the rule base, selecting rules that are relevant to the concept asked about. These rules are then turned into English by various simple translation schemes and presented to the user, providing an explanation of how the concept is used by the system.

*The Linguistic Relevance of Tree Adjoining Grammars*, **Anthony S. Kroch, Department of Linguistics, and Aravind K. Joshi, Department of Computer and Information Science, MS-CIS-85-16, LINC LAB 03**

In this paper the linguistic significance of the Tree Adjoining Grammar (TAG) has been investigated. An important property of TAG is that it defines a constrained theory of syntactic embedding, one requiring that embedded structures be composed out of elementary structures in a fixed way, and one which forces co-occurence relations between elements that are separated in surface constituent structures to be stated broadly as constraints on elementary trees in which those elements are copresent. The extra generative power of TAG beyond context-free grammar emerges as a corollary of factoring recursion and co-occurence relations. The linguistic details specifically discussed are raising constructions, passive, and WH-movements.

*A Computational Logic Approach to Syntax and Semantics* **Dale A. Miller and Gopalan Nadathur, MS-CIS-85-17**

It is well known that higher-order logics are very expensive, and for this reason have been used to represent many problems in mathematics and theoretical computer science. In the latter domain, higher-order logics are often used to describe the semantics of first-order logics, natural languages, or programs, since the formalization of such semantics needs a recourse to quantification over the domain of functions and sets. In these settings, higher-order logic has generally been limited to a descriptive role. Once the formalization is made little has been made of it computationally, largely because there is abundant evidence that theorem proving in higher-order logics is very difficult. In this paper we shall look at a sublogic of a particular higher-order logic that is derived from Church's Theory of Types, and examine its representational power and its computational tractability. This sublogic can also be described as Horn clauses logic extended with quantifications over function variables and λ-contraction. We shall present a sound and complete theorem prover for this logic, which uses higher-order unification and may be described as an extension of a unification procedure for the typed λ-calculus. There are at least three ways in which this logic is different from the first-order logic that it generalizes. First it possesses function variables which can be instantiated with λ-terms and evaluated through λ-contractions. This provides the logic with a new source of computation. Second, since λ-terms do not have most general unifiers, the process of finding appropriate unifiers must branch, and hence involves real search. This facet provides a new source of nondeterminism in specifying computations. Finally, this logic can directly encode first-order logic in its term stucture and can manipulate such terms in logically meaningful ways. We illustrate this with examples taken from knowledge representation and natural language parsing.

*The Role of Perspective In Responding to Property Misconceptions*, **Kathleen F. McCoy, MS-CIS-85-31, May 1985**

In order to adequately respond to misconceptions involving an object's properties, we must have a context-sensitive method for determining object similarity. Such a method is introduced here. Some of the necessary contextual information is captured by a new notion of ***object perspective*** It is shown how object perspective can be used to account for different responses to a given

misconception in different contexts.

***Some Computational Properties of Tree Adjoining Grammars*, Vijayshenkar and Joshi, MS-CIS-85-07**

Tree Adjoining Grammar (TAG) is a formalism for natural language grammars. Some of the basic notions of TAG's were introduced in [Joshi,Levy, and Takahashi 1975] and by [Joshi, 1983]. A detailed investigation of the linguistic relevance of TAG's has been carried out in [Kroch and Joshi,1985]. In this paper, we will describe some new results for TAG's, especially in the following areas: (1) parsing complexity of TAG's, (2) some closure results for TAG's, and (3) the relationship to Head grammars.

***Grammar, Phrase Structure*, Aravind K. Joshi, MS-CIS-85-45**

Phrase-structure trees (phrase-markers) provide structural descriptions for sentences. Phrase-structure trees can be generated by phrase-structure grammars. Phrase-structure trees can be shown to be appropriate to characterize structural descriptions for sentences, including those aspects which are usually characterized by transformational grammars, by making certain amendations to CFG's, without increasing their power, or by generating them from elementary trees (phrase-markers) by a suitable rule of composition, increasing the power only mildly beyond that of CFG's. Structural descriptions provided by phrase-structure trees are used explicitly or implicitly in natural language processing systems.

***Question, Answer and Responses: Interacting with Knowledge Base Systems*, Bonnie Lynn Webber, MS-CIS-85-50, LINC LAB 04**

The purpose of this chapter is to examine the character of information-seeking interactions between a user and a knowledge base system (KBS). In doing so, I advocate that a clear distinction be made between an answer to a question and a response. The chapter characterizes questions, answers, and responses, the role they play in effective information interchanges, and what is involved in facilitating such interactions between user and KBS.

***A Theory Of Scalar Implicature*, Julia Bell Hirschberg, MS-CIS-85-56.**

***The Relationship Between Tree Adjoining Grammars And Head Grammars*, K. Vijay-Shanker, David J. Weir and Aravind K. Joshi, MS-CIS-86-01, LINC LAB 06**

Tree Adjoining Grammars (TAG) and Head Grammars (HG) were introduced to capture certain structural properties of natural languages. These formalisms, which were developed independently, appear to be quite different notationally. In this paper we discuss the formal relationship between the class of languages generated by TAG's (TAL) and the class of languages generated by HG's (HL). In particular, we show that HL's are included in TAL's and that TAG's are equivalent to a modification of HG's called Modified Head Grammars (MHG's). The inclusion of MHL in HL, and thus the equivalence of HG's and TAG's, in the most general case remains to be established. We show that this relationship is very close both linguistically and formally, the difference hinging on the status of heads of empty strings and whether one deals with heads directly or with the left and right wrapping positions around the head.

***Natural Language Interactions With Artificial Experts*, Tim Finin, Aravind K. Joshi and Bonnie Lynn Webber, MS-CIS-86-16, LINC LAB 08).**

The aim of this paper is to justify why Natural Language (NL) interaction, of a very rich functionality, is critical to the effective use of Expert Systems and to describe what is needed and what has been done to support such interaction. Interactive functions discussed here include defining terms, paraphrasing, correcting misconceptions, avoiding misconceptions and modifying questions.

***Higher-Order Logic Programming*, Dale A. Miller and Gopalan Nadathur, MS-CIS-86-17**

In this paper we consider the problem of extending Prolog to include predicate and function variables and typed $\lambda$-terms. For this purpose, we use a higher-order logic to describe a generalization to first-order Horn clauses. We show that this extension possesses certain desirable computational properties. Specifically, we show that the familiar operational and least fixpoint semantics can be given to these clauses. A language, $\lambda$Prolong that is based on this generalization is then presented, and several examples of its use are provided. We also discuss an interpreter for this language in which new sources of branching and backtracking must be accommodated. An experimental interpreter has been constructed for the language, and all the examples in this paper have been tested using it.

***Some Uses of Higher-Order Logic in Computational Linguistics*, Dale A. Miller and Gopalan Nadathur, MS-CIS-86-31, LINC LAB 08**

Consideration of the question of meaning in the framework of linguistics often requires and allusion to sets and other higher-order notions. The traditional approach to representing and reasoning about meaning in a computational setting has been to use knowledge representation systems that are either based on first-order logic or that use mechanisms whose formal justifications are to be provided after the fact. In this paper we shall consider the use of a higher-order logic for this task. We first present a version of definite clauses (positive Horn clauses) that is based on this logic. Predicate and function variables may occur in such clauses the terms in the language are the typed $\lambda$-terms. Such term structures have a richness that may be exploited in representing meanings. We also describe a higher-order logic programming language, called $\lambda$Prolog, which represents programs as higher-order definite clauses and interprets them using a depth-first interpreter. A virtue of this language is that it is possible to write programs in it that integrate syntactic and semantic analyses into one computational paradigm. This is to be contrasted with the more common practice of using two entirely different computation paradigms, such as DCGs or ATNs for parsing and frames or semantic nets for semantic processing. We illustrate such and integration in this language by considering a simple example, and we claim that its use makes the task of providing formal justifications for the computations specified much more direct.

***Some Aspects Of Default Reasoning In Interactive Discourse*, Aravind K. Joshi, Bonnie L. Webber and Ralph M. Weischedel, MS-CIS-86-27 (revised version of MS-CIS-84-58)**

In cooperative interaction, it is taken as necessary that a system truthfully and informatively respond to a user's question. It is not, however, sufficient. In particular, if the system has reason to believe that its planned response might lead the user to draw an inference that it knows to be false, then it must block it by modifying or adding to its response. In this paper we investigate several aspects of such reasoning in interactive discourse.

***Adapting MUMBLE: Experience with Natural Language Generation*, Robert Rubinoff, MS-CIS-86-32, LINC LAB 09)**

This paper describes the construction of a MUMBLE-based [McDonald 83b tactical component for the TEXT text generation system [McKeown 85]. This new component, which produces fluent English sentences from the sequence of structured message units output from TEXT's strategic component, has produced a 60-fold speed-up in sentence production. Adapting MUMBLE required work on each of the three parts of the MUMBLE framework: the interpreter, the grammar, and the dictionary. It also provided some insight into the generation process and the consequences of MUMBLE's commitment to a deterministic model.

***GUMS$_1$ : A General User Modeling System*, Tim Finin and David Drager, MS-CIS-86-35**

This paper describes a general architecture of a domain independent system for building and maintaining *long term models* of *individual users*. The user modeling system is intended to provide a well defined set of services for an *application system* which is interacting with various users and has a need to build and maintain models of them. As the application system interacts with a user, it can acquire knowledge of him and pass that knowledge on to the user model maintenance system for incorporation. We describe a prototype *general user modeling system* which we have implemented in Prolog. This system satisfies some of the desirable characteristics we discuss.

***Breaking the Primitive Concept Barrier*, Robert Kass, Ron Katriel, and Tim Finin, MS-CIS-86-36**

Building and maintaining a large knowledge base of general information requires a knowledge representation system with precise semantics and an easy knowledge acquisition procedure. Systems such as KL-ONE meet these criteria by using a classifier to install new concepts into a taxonomic structure. These systems use a formal notion of a definition for concepts. Unfortunately, many concepts do not seem to have such precise definitions, and end up represented as primitive concepts. Primitive concepts form a barrier to classification, forcing the user to manually classify a new concept with respect to all primitive concepts in the knowledge base.

We propose an extension to KL-ONE which retains its soundness and greatly reduces the burden on the user during knowledge acquisition. This extension consists of adding an explicit definitional component to concepts and relaxing the strictness of concept definitions themselves. The relaxed definition reduces the number primitive concepts in a knowledge base, enables the classifier to handle concepts that do not have complete definitions and enhances the usefulness of an interactive classifier.

# Text Generation for Strategic Computing

USC/Information Sciences Institute
Marina del Rey, CA 90292

Project Leaders: William Mann & Norman Sondheimer

Project Staff: Robert Albano, Susanna Cumming, Thomas Galloway, Christian Matthiessen, Bernhard Nebel, Lynn Poulton, George Vamos, Richard Whitney

## 1 Objectives

The US military is an information-rich, computer intensive organization. It needs to have easy, understandable access to a wide variety of information. Currently, information is often in obscure computer notations that are only understood after extensive training and practice. Although easy discourse between users and machines is an important objective for any situation, this issue is particularly critical in regards to automated decision aids such as expert system based battle management systems that have to carry on a dialog with a force commander. A commander can not afford to miss important information, nor is it reasonable to expect force commanders to undergo highly specialized training to understand obscure computer dialects which differ from machine to machine.

The great deal of work that has been done in the area of natural language understanding is starting to pay off with the delivery of functional interfaces that interact naturally with the user. Comparatively little, however, has been done in the area of natural language generation. Currently, there is no effective technology for expressing complex computer notations in ordinary English. If there were computer-based military information could be made more accessible and understandable in a manner less subject to personnel changes.

The Text Generation for Strategic Computing project is creating and demonstrating new technology to provide an English-in, English-out interface to computer data to be embodied in a system called **Janus**. ISI is developing the English-out (text generation) portion of this overall system, a module named **Penman**. It's initial capability will be demonstrated on a naval database, but most of the techniques are more general and will be able to be reapplied to other military problems. The end result will be an exciting new capability for the military that produces answers to queries and commands in the form of text that is understandable to any user who understands English.

This project was put in place at the beginning of FY85 in order to develop the first natural language generation capability robust and capable enough to be used in DARPA's Strategic Computing Program. It is intended that this interface be coupled to the battle management system being developed under DARPA's Fleet Command Center Battle Management Program(FCCBMP). In the first 1.5 years of the effort, we were able to demonstrate the first generation system to produce English from output demands in formal mathematical logic using a broad coverage grammar and an artificial intelligence knowledge base We have developed the basic technology that will permit the realization of a full text generation system. In the next phase of the work, we will extend text generation from sentences to paragraphs, increase the power of the grammar, dictionary, and planner, expand the knowledge base to cover more fully the battle management problem, optimize

and increase the robustness of the software, and tune the resulting system to the Navy problem. Finally, we will deliver our generation system to BBN, Inc, who will combine it with an understanding system for delivery to DARPA and subsequent integration with FCCBMP.

This report details our approach, Penman's current status, and our plans for future work.

## 2 Our Approach

Both understanding and generation components are necessary in an effective information system interface. It is also necessary that the technologies used by these two components be absolutely consistent. If they are not, the embarrassing situation of the interface failing to understand a piece of text which it has just output can occur, causing a lack of confidence in the system by users.

In order to alleviate this problem, we are developing Penman in close cooperation with a natural language understanding project at Bolt, Beranek, and Newman Inc. (BBN). This cooperation extends to joint development of knowledge representation software and lexicon design, and work to insure compatibility between the RUS understanding grammar and the Nigel generation grammar.

The understanding/generation compatibility effort can be considered the first of our goals for the Penman system. Another is the ability to generate substantial amounts of English text, up to the multiparagraph level. A useful interface should not be restricted to short replies or comments.

A third goal is to make Penman as portable and domain independent as possible. There are many knowledge domains available for interface systems, and few specialists to create the interfaces. Thus we are emphasizing domain independent capabilities over domain dependent ones.

Finally, there are many linguistically and computationally significant issues that must be addressed if high quality generation is to be achieved. For example, we are working on improved methods for knowledge representation, vocabulary development and for using semantics and discourse context to guide the generation of text.

## 3 Accomplishments

By the time of this report, the project had created and delivered a Master Lexicon facility, called ML, for vocabulary acquisition and use [Cumming 86a, Cumming 86b, Cumming 86c]. ML is unusual in that it is compatible with the two radically different grammars of English in Janus; the Nigel generation grammar and the Rus understanding grammar. The system features a multi-window bit mapped display which is manipulatable through the keyboard and a mouse. Online documentation is provided for all lexical choices. ML is built to allow for the extension of the Janus lexical system. It is also possible to use ML with other natural language processing systems by replacement of a single data structure.

In addition, work has been done on bringing the Nigel and RUS grammars into compatibility.

Our general goal is the development of natural language generation capabilities. Our vehicle for these capabilities will be a reusable module designed to meet all of a system's needs for generated sentences. The generation module must have an input notation in which demands for expression are

represented. This notation should be of **general** applicability. For example, a good notation ought to be generally useful in a reasoning system. Also, the notation should have a **well-defined** semantics and the generator has to have some way of **interpreting** the demands. This interpretation has to be **efficient.**

In our research, we have chosen to use **formal logic** as a demand language. **Network knowledge-bases** are used to define the domain of discourse in order to help the generator interpret the logical forms. And a **restricted, hybrid knowledge representation** is utilized to analyze demands for expression using the knowledge base [Sondheimer 86]. We have:

1. Developed a demand language, Penman Logical Form (PLF), based on first order logic [USC/ISI 85],

2. Structured a NIKL (New Implementation of KL-ONE) network [Kaczmarek 86] to reflect conceptual distinctions observed by functional systemic linguists.

3. Developed a method for translation of demands for expression into a propositional logic database.

4. Employed KL-TWO [Vilain 85] to analyze the translated demands, and

5. Used the results of the analyses to provide directions to the Nigel English sentence generation system [Mann & Matthiessen 83].

To first order logic, PLF adds restricted quantification, i.e., the ability to restrict the set quantified over. In addition, we allow for equality and some related quantifiers and operators, such as the quantifier for "there exists exactly one ..." ($\exists!$), the operator for "the one thing that ..." ($\iota$). We permit the formation and manipulation of sets, including a predicate for set membership (ELEMENT-OF). And we have some quantifiers and operators based on Habel's $\eta$ operator [Habel 82].

Our system's knowledge has been organized in a new way in order to facilitate English expression. Abstract concepts corresponding to the major conceptual categories of English have been defined in NIKL (a KL-ONE dialect) and used to organize the conceptual hierarchy of the domain. By defining concepts such as process, event, quality, relationship and object in this way, fluent English expression is facilitated.

KL-TWO is a hybrid knowledge representation system that uses NIKL's formal semantics. KL-TWO links another reasoner, PENNI to NIKL. For our purposes, PENNI can be viewed as restricted to reasoning using propositional logic.

We translate a logical form into an equivalent KL-TWO structure. All predications appearing in the logical form are put into the PENNI database as assertions. A separate tree is created which reflects the variable scoping. Separate scopes are kept for the range restriction of a quantification and its predication.

The Nigel grammar and generator realizes the functional systemic framework at the level of sentence generation. Within this framework, language is viewed as offering a set of grammatical choices to its speakers. Speakers make their choices based on the information they wish to convey and the discourse context they find themselves in. Nigel captures the first of these notions by

organizing minimal sets of choices into **systems**. The grammar is actually just a collection of these systems. The factors the speaker considers in evaluating his communicative goal are shown by questions called **inquiries** inside of the **chooser** that is associated with each system. A choice alternative in a system is chosen according to the responses to one or more of these inquiries. It is these inquiries which we have implemented.

Our implementation of Nigel's inquiries using the connection and scope structures with the NIKL upper structure is fairly straightforward. Since the logical forms reflecting the world view are in the highest level of the NIKL model, the information decomposition inquiries use these structures to do search and retrieval. With all of the predicates in the domain specializing concepts in the functional systemic level of the NIKL model, information characterization inquiries that consider aspects of the connection structure can test for the truth of appropriate PENNI propositions. The inquiries that relate to information presented in the quantification structure of the logical form will search the scope structure. Finally, to supply lexical entries, we associate lexical entries with NIKL concepts as attached data and use the retrieval methods of PENNI and NIKL to retrieve the appropriate terms.

Although we have done some generation using the BBN naval knowledge base, our most extensive experience comes from the Consul knowledge domain of computer mail and calendars [Kaczmarek, Mark, and Sondheimer 83]. Using the developments above, translation from first-order logic to English organized by NIKL has been demonstrated for a wide variety of sentence types.

Table 3 shows a series of independent sentences that Penman is now able to generate. We have arranged and labeled them to show how they might enter into computer database dialogue. Paraphrases in <brackets> show corresponding sentences of the type that will be generated from Naval domain knowledge.

Besides the technology described above, work has been done on aspects of discourse modeling. This included a study of object description and an initial noun phrase planner, which space does not allow us to detail.

In order to produce the texts that are needed in a user assistance facility, work is under way to apply Rhetorical Structure Theory to text planning. A procedure has been developed which can design structures for paragraph-length texts given information about the text goal and communication situation.

## 4 Future Effort

In the next phase of our effort, Penman will be delivered to BBN, integrated into Janus, and demonstrated in an extensive series of laboratory exercises. This version of Penman will support coordinated English input and output, paraphrasing, and user assistance, as well as constructive rhetorical structure theory to create descriptive texts [Mann 84].

For a machine to create fluent English texts, several problems of text organization and presentation must be solved. We will design, implement and demonstrate two kinds of new text planning processes. One will be a planner for gross organization of text based on rhetorical structure theory, and the other a low level planner for sentence configurations. In addition, we will examine the expert system explanation functionality necessary within FCCBMP and will design a text planner to meet those needs.

**Table 1:** Sentences from Penman

1. User: What is the read status of the message which was sent by Jones? <What is the readiness of the ship deployed by CINCPACFLT?>
2. System: Message 40 is unread. <The Nimitz is C4.>
3. User: Are any messages which were sent by Jones in the tape file? <Are any ships deployed by CINCPACFLT in the Atlantic fleet?>
4. User: Are any messages which were sent by Jones in the old ISI tape file which Smith has? <Are any ships deployed by CINCPACFLT in the new battle group which Spivak has?>
5. System: Yes. <Yes.>
6. System: Do you want me to display the read status of the messages? <Do you want me to display the readiness of the ships?>
7. User: No. <No.>
8. System: There are messages which were sent by Jones. <There are ships which were deployed by CINCPACFLT.>
9. System: However, the tape is in use. <However, the battle group is en route.>
10. User: Show me Message 40. <Show me the JFK.>
11. User: Did Smith send any messages? <Did the JFK send any CASREPs?>
12. System: Message 45 was sent by Smith. <CASREP 45 was sent by the JFK.>
13. User: Did Smith schedule his meeting with me in New York? <Did the JFK schedule its port visit in San Diego?>
14. System: Yes. <Yes.>
15. User: What does the window show? <What does the map show?>
16. System: The window shows a message which was sent by Jones. <The map shows a sail plan which was sent by CINCPAC>
17. System: The window shows a request by Jones. <The map shows a request which was sent by CINCPACFLT.>
18. User: Did I create the window? <Did I create the map?>
19. System: The window was created when you asked me to display Message 50. <The map was created when you asked me to display Sail Plan 33.>
20. System: You deleted the message, then you asked me to display it. <You deleted Sail Plan 33, then you asked me to display it.>

We also plan to develop enhancements to the basic interface to Penman. As part of Janus, BBN plans to develop methods for translating English inputs into a higher order predicate calculus form. In addition, the logical form will be manipulated inside the BBN understanding component to reflect database organization, expressions in this form are expected to be much more complex than the direct expressions of the corresponding English, requiring multiple sentence expression. Penman will be extended so that it can express these complex forms in English.

Following successful completion of initial laboratory testing, Penman will be delivered to NOSC for knowledge base development and testing. This Penman will have a basic text generation capability with only limited capability for responding to the knowledge of the user and the state of the user-

machine interaction. In subsequent years, these basic capabilities will be extensively expanded to provide a much more useful interface, with particular development in the areas of user assistance, ability of the system to understand the user's knowledge and expectations, and needs for expert system explanation and user dialogue, as described below.

Development of extensions is required in the area of user assistance functionality. Text generation can assist the user by explaining system actions and objects, and by clarifying its interpretation of incoming English. These capabilities must be integrated into the Janus interface in a natural, easy-to-use form. We expect to demonstrate increased user assistance functionality as time goes on.

Development of extensions is also required in the area of responsiveness to users' knowledge and expectations. This includes making the system aware of the ongoing topics and issues, what the user has already been told, and what he can be expected to know without being told. This in turn involves more extensive use of inference. It also requires implementing a notion of the state of the dialogue -- what things are in attention, what interactions are in process and what has been interrupted and suspended.

Penman is scheduled for use in an expert-system explanation context whose details are yet to be determined. We will study what functionality is required in FCCBMP. The study will lead to a design of code for this use of Penman. Implementation and demonstration of full functionality are expected following the user assistance work.

# References


[Cumming 86a] Susanna Cumming, *Design of a Master Lexicon*, USC/Information Sciences Institute, Marina del Rey,CA, Technical Report ISI/RR-85-163, February 1986.

[Cumming 86b] Susanna Cumming, Robert Albano, *A Guide to Lexical Acquisition in the JANUS System*, USC/Information Sciences Institute, Marina del Rey,CA, Technical Report ISI/RR-85-162, February 1986.

[Cumming 86c] Susanna Cumming, *The Lexicon in Text Generation*, USC/Information Sciences Institute, Marina del Rey,CA. Technical Report ISI/RR-86-168, 1986. Presented at The Workshop on Automating the Lexicon, Pisa, Italy, May, 1986.

[Habel 82] Christopher Habel, "Referential nets with attributes," in Horecky (ed.), *Proc. COLING-82*, North-Holland, Amsterdam, 1982.

[Kaczmarek 86] T. Kaczmarek, R. Bates, G. Robins, "Recent Developments in NIKL," in *AAAI-86, Proceedings of the National Conference on Artificial Intelligence*, AAAI, Philadelphia, PA, August 1986.

[Kaczmarek, Mark, and Sondheimer 83] T. Kaczmarek, W. Mark, and N. Sondheimer, "The Consul/CUE Interface: An Integrated Interactive Environment," in *Proceedings of CHI '83 Human Factors in Computing Systems*, pp. 98-102, ACM, December 1983.

[Mann 84] Mann, W., *Discourse Structures for Text Generation*, USC/Information Sciences Institute, Marina del Rey, CA, Technical Report RR-84-127, February 1984.

[Mann & Matthiessen 83] William C. Mann & Christian M.I.M. Matthiessen, *Nigel: A Systemic Grammar for Text Generation*, USC/Information Sciences Institute, Technical Report ISI/RR-83-105, Feb 1983.

[Sondheimer 86] Norman Sondheimer, Bernhard Nebel, "A Logical-Form and Knowledge-Base Design for Natural Language Generation," in *AAAI-86, Proceedings of the National Conference on Artificial Intelligence*, AAAI, August 1986.

[USC/ISI 85] *Penman's Logical Language*, USC/Information Sciences Institute, Marina del Rey, CA, 1985.

[Vilain 85] M. Vilain, "The Restricted Language Architecture of a Hybrid Representation System," in *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pp. 547-551, Los Angeles, CA, August 1985.

# SECTION 2: RESEARCH CONTRIBUTIONS

*Bolt, Beranek, and Newman, Inc.*

# Out of the Laboratory:
# A Case Study with the IRUS Natural Language Interface[1]

by

Ralph M. Weischedel, Edward Walker, Damaris Ayuso, Jos de Bruin,
Kimberle Koile, Lance Ramshaw, Varda Shaked

BBN Laboratories Inc.
10 Moulton St.
Cambridge, MA 02238

Abstract

As part of DARPA's Strategic Computing Program, we have moved a large natural language system out of the laboratory. This involved:

- Delivery of knowledge acquisition software to the Naval Ocean Systems Center (NOSC) to build linguistic knowledge bases, such as dictionary entries and case frames,
- Demonstration of the natural language interface in a naval decision-making setting, and
- Delivery of the interface software to Texas Instruments, which has integrated it into the total software package of the Strategic Computing Fleet Command Center Battle Management Program (FCCBMP).

The resulting natural language interface will be delivered to the Pacific Fleet Command Center in Hawaii.

This paper is an overview of this effort in technology transfer, indicating the technology features that have made this possible and reflecting upon what the experience illustrates regarding transportability, technology status, and delivery of natural language processing outside of a laboratory setting. The paper will be most valuable to those engaged in applying state-of-the-art techniques to deliver natural language interfaces and to those interested in developing the next generation of complete natural language interfaces.

---

[1]The work presented here was supported under DARPA contract #N00014-85-C-0016. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or of the United States Government.

# 1 Introduction

DARPA's Strategic Computing Program in the application area of Navy Battle Management has provided us several challenges and opportunities in natural language processing research and development. At the beginning of the effort, a set of domain-independent software components, developed through fundamental research efforts dating back as much as seven years, existed. The IRUS software [1] consists of two subsystems: one for linguistic processing and one for adding specifics of the back end. The first subsystem is linguistic in nature, while the second subsystem is not. Linguistic processing includes morphological, syntactic, semantic, and discourse analysis to generate a formula in logic corresponding to the meaning of an English input. The linguistic subsystem is application-independent and also independent of data base interfaces. (This is achieved by factoring all application specifics into the back end processor or into knowledge bases such as dictionary entries and case frame rules, that are domain-specific.) The non-linguistic components convert the logical form to the code necessary for a given underlying system, such as a relational data base.

The IRUS system, or its components, had been used extensively in the laboratory, not just at BBN, but also in research projects at USC/Information Sciences Institute, the University of Delaware, GTE Research, and General Motors Research. However, it had not been exercised thoroughly outside of a research environment.

Our goals in participating in the Strategic Computing Program are manifold:

- To test the collection of state-of-the-art heuristics for natural language processing with a user community trying to solve their problems on a daily basis.
- To test the heuristics on a broad, extensive domain.
- To incorporate research ideas (which are often developed in relative isolation in the laboratory) into a complete system so that effective evaluation and refinement can occur.
- To continue the feedback loop of incorporating new research ideas, testing them in a complete system with real users, evaluating the results, and refining the research accordingly on a repeated basis for several years.

There are several accomplishments in the first year and a half of this work. First, the IRUS software has been delivered to the Naval Ocean Systems Center (NOSC) so that their team may encode the dictionary information, case frame rules, and transformation rules for generating queries appropriate for the underlying systems. The NOSC staff involves a linguist plus individuals trained in computer science, but

does not involve experts in natural language processing nor in artificial intelligence. Second, the natural language interface software has been delivered to Texas Instruments (TI), which has integrated it into the Force Requirements Expert System (FRESH). Demonstrations of the natural language interface are being given at several conferences this year as well as to the navy personnel at the Pacific Fleet Command Center. Testing and evaluation of IRUS, both its software and the knowledge bases defined by NOSC for the FCCBMP, will be carried out in the spring of 1986, by the Navy Personnel Research and Development Center.

In this section and section two we present evidence that this is one of the most ambitious applications and tests of natural language processing ever attempted. Section two provides more background regarding the technical challenges inherent in the application environment and in the goals of the Strategic Computing Program. Section three describes what was changed in each system component to support the technology transfer. Section four presents and illustrates the principles that have been underscored in moving this substantial AI system from the laboratory to use; while some principles may appear like common sense, reporting on all the experience should be valuable to future efforts. Section five briefly discusses possible future directions, while section six states our conclusions.

# 2 Background Constraints and Goals

The following sections summarize several constraints and goals which have made this not only a demanding challenge for natural language processing but also an ambitious demonstration of the fruit of AI research.

### 2.1 Multiple Underlying Systems

The decision support environment of the Fleet Command Center Battle Management Program (FCCBMP) involves a suite of decision-making tools. A substantial data base is at the core of those tools and includes roughly 40 relations and 250 fields. In addition, application programs for drawing and displaying maps, various calculations and additional decision support capabilities are provided in the Operations Support Group Prototype (OSGP). In a parallel part of the Strategic Computing Program, two expert systems are being provided: the Force Requirements Expert System (FRESH) and the Capabilities Assessment Expert System (CASES). TI is building the FRESH expert system; the contract for the CASES expert system has not been awarded as of the writing of this paper.

The target users are navy commanders involved in decision making at the Pacific

Fleet Command Center; these are top-level executives whose energy is best spent on navy problems and decision making rather than on the details of which of four underlying systems offers a given information capability, on how to divide a problem into the various information capabilities required and how to synthesize the results into the desired answer. Currently they do not access the data base or OSGP application programs themselves; rather, on a round-the-clock basis, two operators are available as intermediates between commander and computer. Consequently, the need for a natural language interface (NLI) is paramount.

### 2.2 The Need For Transportability

There are three ways that transportability has been absolutely required for the natural language interface. First, since we had no experience previously with this application domain, and since the schedule for demonstrations and delivery was highly ambitious, only the application-independent software could be brought to bear on the problem initially; therefore, transportability across application domains was required. Second, the underlying systems have been and will continue to be evolving. For instance, the data base structure is being modified both to support additional information needs for the new expert systems and to provide shorter response time in service of human requests and expert system requests to the data base.

Third, the target output of the natural language interface is subject to change. For instance, the capabilities of FRESH are being developed in parallel with the natural language interface and the CASES expert system has not been started as of this date. Interestingly enough, the target language for the data base could change as well. For instance, there is the possibility of replacing the ORACLE data base management system with a data base machine, in which case the target language would change though the application and data base structure remained constant during the period of installing the data base machine.

### 2.3 Technology Testbed

The project has two goals which at first seem to conflict. First, the software must be hardened enough to be an aid in the daily operations of the Fleet Command Center. Second, the delivered systems are to be a testbed for research results; feedback from use of the systems is to provide a solid empirical base for suggesting new areas of research and refinement of existing research.

As a consequence, software engineering demands placed upon the AI software are quite rigorous. The architecture of the software must support high quality, well worked out, non-toy systems. The software must also support substantial evolution in

the heuristics and methods employed as natural language processing provides new research ideas that can be incorporated.

# 3 Adequacy of the Components

In this section we present a brief analysis of the adequacy of the various components in the system, given that the software had not been built with this domain in mind (but had been built with transportability in mind) and given that one of the goals of the effort is to provide a flexible technological base allowing evolution of the techniques and heuristics employed.

## 3.1 Knowledge Representation

At the start of the project, the underlying knowledge representation consisted of a hierarchy of concepts (unary predicates), a list of functions on instances of those concepts, and a list of n-ary predicates. The knowledge representation served several purposes:

- To identify the predicate symbols and function symbols that could be used in the first order logic representing the meaning of sentences,
- To validate selection restrictions (case frame constraints) during the parsing process.

Early on we concluded that greater inference capabilities were required. We wanted to be able to:

- State and reason about knowledge of binary relationships. For instance, every vessel has an arbitrary number of overall readiness ratings associated with it, corresponding to the history of its readiness.
- Represent events and states of affairs flexibly. There may be a variable number of arguments expressed in the input for a given event. For instance, *Admiral Foley deployed the Eisenhower yesterday* or *Admiral Foley deployed the Eisenhower C3.*[2] Also, we needed to be able to count occurrences of events or states of affairs over history, as in *How many times was the the Eisenhower C3 in the last 12 months?* Consequently, we have chosen to represent events and states of affairs as entities, which participate in a number of binary relationships, for instance, specifying the agent, time, location, etc. of the event.

Therefore, the initial ad hoc knowledge representation formalism was replaced with a more general framework, NIKL [10], the new implementation of KL-ONE. This met the needs stated above, and also provided inference mechanisms [15] which could serve as

[2] *C3* is an overall readiness rating.

a partial consistency checker on the axioms for the navy domain. Of course, there are other ways to achieve the goals above. However, NIKL was available, and this would be its first use in a technology transfer effort, providing us the opportunity to further explore the power and limitations of limited inference systems.

In NIKL, one can state the classes of entities, the binary relations between entities (including functional relationships), subclass relationships, and subsumption relations among binary relations. It is now used to support:

- The validation of selection restrictions during the parsing process,
- Proposal of possible case frame constraints and possible predicates by the semantic knowledge acquisition component,
- Proposal of the meaning of vague relationships, such as "have", and
- The mapping from first-order logic to relational data base queries.

Once the more powerful knowledge representation and inference mechanisms [15] were available to IRUS, we began using them in unanticipated ways, for instance, the last three in the list above.

### 3.2 The Lexicon and Grammar

The current grammar (RUS) [2] and lexicon are based on the ATN formalism [23]. Though RUS was designed to be a general grammar of dialogue and was clearly among a handful of implemented grammars having the broadest coverage of English, the question was how much modification would be needed for the Navy domain, which was totally new to us.

Very few changes were needed to the software that supports the lexicon and morphological analysis. Those that were required centered around special military forms, such as allowing *06Mar86* as a date and *0600z* as a time. Special symbols and codes such as those are bound to arise in many applications, no matter how transportable the software is.

Very few modifications to the grammar had to be made; those that have been made thus far correspond to special forms and have required very little effort to add. Examples include military (and European) versions of dates, such as *6 March 1986*. This is not to claim that everything a navy user types will be parsed; fully general treatments for conjunction, gapping, and ellipsis, are still research issues for us, as for everyone else. Rather, the experience testifies to the fact that domain-independent grammars can be written for natural language interfaces and that modification of them for a new application can be very small. Sager [12] has reported

that few rules of the Linguistic String Parser need to be changed when it is moved to a new application.

The current system handles several classes of ill-formed input, including typographical errors that result in an unknown word; omitted words such as determiners and prepositions; various grammatical errors such as subject verb disagreement and determiner noun disagreement; case errors in using pronouns; and elliptical inputs. The strategy is that of [21].

### 3.3 Semantic Interpretation

Though the software for the semantic interpreter did not depend on domain specifics, the limitations of the initial knowledge representation formalism and of the class of linguistic expressions for which it could compute a semantic representation meant that the semantic interpreter had to be substantially changed. First, the semantic interpreter was modified to take advantage of the stronger knowledge representation formalism and inference available in NIKL. For instance, the interpreter must compute the semantic representation for descriptions of events and states of affairs. It now finds the interpretation of *X has Y* by looking for a relation in the knowledge representation between X and Y.

Second, the semantic interpreter has been changed to correspond more and more to general linguistic analysis. One strength of the initial version of the semantic interpreter [1] was its ability to handle idiomatic expressions, such as *blue forces*. *Blue forces* refers to U.S. forces, as opposed to forces that are blue (in color). The semantic interpreter has been generalized now so that it is much easier to capture the general meaning of *blue* as a predicate, as well as allowing for specification of idiomatic expressions, such as *blue forces*.

A major focus in the next year will be continuing modification of the semantic interpreter so that we have a fully compositional semantics and an intensional logic, rather than a first order logic as the meaning representation of a given sentence. The compositional semantics will still allow, of course, for idiomatic expressions. The enhanced semantic interpreter will be applicable to a much broader class of English expressions, while still being domain-independent and driven by domain-specific case frame rules.

The semantic interpreter does not allow for semantic ill-formedness at present; removing this restriction is a high priority research area.

### 3.4 Discourse Phenomena

Since discourse analysis is the least understood area in natural language processing, the discourse processing component in the system is limited. The system handles anaphora based on the class of the entity required by the selection restrictions upon the anaphor. A benefit of the change in representation making events and states of affairs entities is that the simple heuristic above allows the anaphor in each of the following sequences to be correctly understood.

- *The Eisenhower was deployed C2. When did that occur?*
- *The Eisenhower had been C3. When was that?*

Elliptical inputs that are noun phrases or prepositional phrases are handled as follows: If the class of the entity inherent in the elliptical input is consistent with a class in the previous input, the semantic representation of the new entity is substituted for the semantic representation in the previous input. If not, the ellipsis is interpreted as a request to display the appropriate information.

Far more sophisticated discourse processing is a high priority not only for our project but for natural language work altogether.

### 3.5 Introducing Back end Specifics

The result of linguistic processing in IRUS is a formula in logic. Another component translates the logical expression representing the meaning of an input into an expression in an abstract relational algebra. Simple optimization of the resulting expression is performed in the same component. The initial version of that component (MRLtoERL) [17] used local transformations to translate the n-ary predicates of the logic into the appropriate sequence of projections, joins, etc. on files and fields of the data base.

A straightforward, syntax-directed code generator translates the abstract relational expression into the query language required by the underlying data base management system. Code generators have been built for System 1022, the Britton-Lee Data Base Machine, and ORACLE. An experienced person needs only two to three weeks to create the code generator.

With the move to NIKL and the representation of events and states of affairs as concepts participating in binary relations, the context-free translation of predicates to expressions in relational algebra was no longer adequate. However, the limited inference mechanism [15] of NIKL formed a basis for a simplifier [18] as a preprocess

to the MRLtoERL component so that the translation from logic to relational algebra could still be done using only local transformations. Furthermore, the simplifier enabled general translation of linguistic expressions whose data base structure bears little resemblance to the conceptual structure of the English query [18]. We believe the simplification techniques can be generalized further to support the simplification of a subclass of expressions in the intensional logic to be generated by the planned semantic interpreter [19].

Introduction of back end specifics for the OSGP application package and the FRESH expert system is handled by an ad hoc translator from logic to target code at present.

### 3.6 Linguistic Knowledge Acquisition

IRUS's four knowledge bases are:

- The lexicon, which states syntactic and morphological information,
- The taxonomy of case frame rules,
- The model of predicates in the domain, stated in NIKL, and
- The transformation rules for mapping predicates in the logic into projections, joins, etc. of fields in the data base.

The first two of these are linguistic knowledge bases; sophisticated acquisition tools are available to aid the system builder, though not necessarily trained in AI, to build the necessary linguistic knowledge about the vocabulary.

Powerful knowledge acquisition tools for building these domain-specific constraints could greatly ease the process of bringing up a natural language interface for a new application and consequently for broadening the applicability of NLI technology. Perhaps the most powerful demonstration of acquisition tools to date has been TEAM [6]. Based on the fields and files of a given data base, TEAM's acquisition tools lead the individual through a sequence of questions to acquire the specific linguistic and domain knowledge needed to understand a broad subset of language for querying the data base. However, since those heuristics are in large part specific to the task of accessing data bases, that technology could not be directly applied to the FCCBMP application, which encompasses a relational data base, an application package including both map drawing and calculation, and expert systems.

Knowledge acquisition tools for IRUS, developed under earlier DARPA-funded work at BBN, were not specific to data base applications and therefore could be applied in the FCCBMP. Even if applicability of the TEAM heuristics were not a problem, there

are theoretical and technical difficulties in translating English requests into data base queries [9] which would argue for a more general approach such as ours. As Scha [13, 14] has argued, these difficulties, as well as the issues of transportability and generality, suggest keeping linguistic knowledge rather independent of assumptions about the back end.

IRACQ, the semantic acquisition tool made available to NOSC for specifying case frames and their associated translations, is quite powerful. The initial version [11] allowed one to specify the case frame for a new word sense by giving an example of a phrase using that word sense. For instance, if *the admiral*, *a vessel*, and *C2* are known to the system, then one can define a new case frame for *deploy* by giving a phrase such as *the admiral deployed a vessel C2*. The system suggests generalizations of the arguments specified in the example using the NIKL knowledge base, so that the inferred case frame is the most general that the user authorizes. For example, generalizations of admiral are commanding officer, person, and physical object; generalizations of vessel are unit, platform, and physical object; generalizations of C2 are rating and code. Furthermore, based on the introduction of the more general knowledge representation system NIKL, IRACQ is being extended to propose the binary relations that might be part of the translation of the new word. Of course, if the relations and concepts needed are not already present in the domain predicate model, the user can define new concepts and relations in the NIKL hierarchy as well.

The availability of such knowledge acquisition tools has made it possible for NOSC representatives, rather than AI experts, to define the naval language expected as input. We have found that even with the tool described above, reasonable linguistic sophistication is very helpful in defining the case frames. In fact, an individual with a master's degree in linguistics is defining the case frames at NOSC. More sophisticated tools, which do not presuppose only one kind of back end, are one of the most important research topics for natural language interfaces. These would combine the strengths of the linguistic knowledge acquisition tools of both IRUS and TEAM.

# 4 Principles Underscored

In the course of the effort, a number of principles have been underscored. Many of these once stated may appear to be common sense; however, we hope that illustrating them from our experience will prove helpful.

### 4.1 The Necessity For General Solutions

The availability of domain-independent software driven by domain-dependent, declarative knowledge bases was of paramount importance because of the following:

- The application was not only broad (three underlying systems) but also evolving (with a fourth system to be added).
- Great habitability is necessary for delivery to the Pacific Fleet Command Center.
- The time frame for demonstration was relatively short compared to the scope of the underlying systems to be covered.

Furthermore, it is critical that the knowledge bases state a linguistic or domain fact once and that the domain-independent software be able to use that one fact in all predictable linguistic variations. The reasons are obvious: the efficiency in building the knowledge bases, the consistency of stating a fact only once, and the habitability of the resulting system which can understand things no matter what form they are expressed in.[3]

The IRUS system attains the goal mentioned above relatively well; a linguistic or application constraint is stated once in the knowledge base but applied in all possible ways in the language processing. This is particularly true because of the substantial grammar [2, 3] and to a lesser extent due to the semantic interpreter. Recognition of this fact is part of the reason that substantial changes, as mentioned in section three, are planned in the semantic interpreter to make the linguistic facts that drive it even more general.

---

[3] An interesting anecdote that arose in early discussions in the planning of this project centered around the tight deadlines and the breadth of the application area. Since it was clear that one could not cover all three underlying systems in every area for which they could provide information, the question arose whether to focus on a substantial subpart of the application domain initially or to sacrifice linguistic coverage to gain in coverage of the underlying systems. Because the information needs of the various navy personnel differed widely, and because the scope of needs seemed impossible to predict, navy personnel initially suggested that coverage of all possible information stored in the underlying systems was of such importance that sacrifices regarding the language understood could be made even if there were only one way that a given piece of information could be accessed. The interesting thing however is that as demonstrations were given, the first things people request following the demonstration is to try various rephrasings of the requests in the demonstration, thereby in behavior indicating how important not being restricted to special forms is.

### 4.2 The Necessity of Heuristic Solutions

In the previous section we have argued for the need of general purpose solutions to problems in NLI. Clearly this cannot be taken to an extreme; otherwise one would not have an NLI in the foreseeable future, since there are well-known outstanding problems for which there is no general, comprehensive solution on the horizon. Consequently, heuristic, state-of-the-art solutions are being demonstrated for problems such as ambiguity, vagueness, discourse context, ill-formed input, definite reference, quantifier scope, conjunction, and ellipsis. Though laboratory use of the system embodying that set of heuristics is quite promising, we expect that placing the system in the hands of individuals trying to solve their day-to-day problems will produce interesting corpora of dialogues that cannot be handled by one or more of those heuristics. Careful study of those corpora will tell us not only the effectiveness of state-of-the-art solutions but will also suggest new directions of research.

### 4.3 The Necessity of Extra-linguistic Elements in a Natural Language Interface

Having only a natural language processor is not sufficient to provide a truly natural interface. Four elements seem highly valuable for typed input: editing, a readily accessible history of the session, human factors elements in the presentation, and a minimum of key strokes. Editing should include more than deleting the last character of the string and deleting the whole string. We are currently relying on Emacs, which is readily available on Symbolics workstations. However, that is also unattractive because of the arcane nature of the link between the myriad control key commands of Emacs and the actual textual tasks the user needs to perform.

IRUS's on-line history of the session provides reviewing earlier results, editing the text of earlier requests to create new ones, and generating a standard protocol for routine operations that occur on a regular basis. Our user community anticipates a need for both routine sequences of questions as would be useful in preparing daily or weekly reports, and ad hoc queries, e.g., when crises arise.

Issues in presentation are important as well. No matter what the underlying application is, IRUS lets it produce output on the complete bitmap screen. A popup input window and an optional popup history window can be moved to any part of the screen so that all parts of the underlying system's output may be visible.

Certain operations occur so frequently that one would like to have them available on the screen at all times in menus to minimize memory load and key strokes. Examples are clearing a window and aborting a request.

A future capability that would be quite attractive is pointing to individual data items, classes of data items, field headings, or locations on maps, causing the appropriate linguistic description of that entity to be made available as part of the natural language input. While this is possible in the future, providing such a capability is not currently funded.

Speech input as a mode of communication would also be highly desirable, even if extremely limited initially. As a consequence, the next generation of natural language understanding systems in the FCCBMP will include modifications specifically to provide an infrastructure which could at a later date support speech input.

# 5 Future Possibilities

In addition to the enhancements we have mentioned earlier regarding the semantic interpreter, linguistic knowledge acquisition tools, and discourse processing, there are three substantial areas of research and development possible. First, research in ill-formed input is necessary in order to allow for additional grammatical problems in the input and for relaxation of semantic constraints, e.g., to allow for figures of speech. The problem with an ill-formed input is that there is no interpretation which satisfies all linguistic constraints. Therefore, the very constraints that limit search must be relaxed, thereby opening Pandora's Box in terms of the number of alternatives in the search space. Not only IRUS, but apparently all systems that process any ill-formed input attain the success they do by considering very few kinds of ill-formed input and by assuming that semantic constraints can never be violated.[4] Consequently, determining what the user meant in an ill-formed input is a substantial problem requiring research.

Second, we propose exploring parallel architectures to add functional capability. Run time performance of IRUS on a Symbolics machine is quite acceptable. Typical inputs are fully processed to give the target language input to the underlying system within a few seconds; naturally, the relational data base and underlying expert systems are not expected to be able to perform at comparable speeds. There are three areas where functional performance could be improved by parallelism.

1. The current system ranks the partial parses using both semantic and syntactic information, and it explores those partial parses based on following up the most promising one first. The technique is relatively effective, but clearly not infallible. Finding all interpretations and then

[4] Early work on allowing semantic relaxation is reported in [5, 21, 22].

ranking them based not only on local syntactic and semantic tests but also on global semantic, pragmatic, and discourse information is critical to improving the identification of what the user intended.

2. A second area related to the first, is greater coverage of ill-formed input. As mentioned earlier, ill-formedness requires relaxing the rules that constrain search; therefore the search space grows dramatically in processing an ill-formed input.

3. Real-time, large vocabulary, large branching factor, continuous speech recognition is beyond the state of the art, and requires highly parallel machines to support speech signal processing. While this is highly desirable, it is not part of our current effort.

Within the next two years we intend to replace the ATN grammar with a declarative, side-effect free grammar and a parallel parsing algorithm, following work reported in [16].

Third, our evolving system is being interfaced to the Penman generation component from USC/Information Sciences Institute (USC/ISI) [8]. Penman is based upon systemic linguistics. The ultimate goal of the effort with USC/ISI is twofold: to have systems that can understand whatever they generate and to achieve this by having common knowledge sources for the lexicon, for the NIKL model of domain predicates, and for discourse information.

# 6 Conclusions

Though the project will be ongoing for several years yet, there are several preliminary conclusions from the first year and a half of effort, given the constraints and goals mentioned in section two.

1. Providing language coverage for this broad application with multiple underlying systems has not been a problem. However, since determining what system(s) must be accessed for a given input is a research problem that has been little addressed, only simple linguistic clues are used in the current version. The problem in general involves not only reasoning about the capabilities of the underlying systems [7] but also significant linguistic issues. For instance, if one says *Show me the carriers whose condition code changed in the last 24 hours*, either a list (from the data base) or a map (from OSGP) is appropriate. If one says *Show me a display of the carriers whose condition code changed in the last 24 hours*, only OSGP is appropriate. The linguistic cue is *display*. Furthermore, some contexts favor one underlying system over the other, requiring the system to maintain a dialogue context model, including the user's inferred goals in the dialogue, in order to integrate cues from dialogue context with the linguistic cues.

2. The architecture has supported transportability well. For instance, this new application required only minor changes to the grammar and morphological analyzer. As FRESH has been further defined and as the data base structure has evolved, only small local changes have been required to the content of the knowledge bases. Should a data base machine replace the

current data base management system in Hawaii, only two to three person weeks should be needed to generate the new target language. However, more sophisticated linguistic knowledge acquisition tools not dependent on the type of the underlying application system are a critical goal for NLI both for far greater applicability of the technology and for far broader availability of NLIs.

3. The success of this effort as a technology testbed depends on evaluation after installation at the Pacific Fleet Command Center and on the success of the architecture to support substantial enhancements, such as the planned semantic interpreter based on compositional semantics and the planned parallel parser. However, it already has supported massive changes well, such as the change in underlying knowledge representation when NIKL was introduced.

   The potential of the testbed is great because it offers empirical research of a realistic kind unfortunately largely lacking heretofore; the placement of TQA in the hands of users to solve their daily problems for a year [4] is a notable exception. The results of research on heuristics for definite reference; semantic ambiguity; ellipsis; syntactically or semantically ill-formed input; and inference from world knowledge and context, to name a few studied in isolation, must be tested in a complete system. The opportunity in the FCCBMP will help to determine the effectiveness of such heuristics in a large diverse application domain where combinatorial issues cannot be ignored. Collecting corpora in an experiment can be highly instructive, as shown in [20]. However, corpus collection using people solving their own problems provides an uncommon degree of realism and legitimacy to the empirical process.

# References


[1] Bates, M., Stallard, D., and Moser, M.
The IRUS Transportable Natural Language Database Interface.
*Expert Database Systems.*
Cummings Publishing Company, Menlo Park, CA, 1985.

[2] Bobrow, R.J.
The RUS System.
In B.L. Webber, R. Bobrow (editors), *Research in Natural Language Understanding.*Bolt, Beranek and Newman, Inc., Cambridge, MA, 1978.
BBN Technical Report 3878.

[3] Bobrow, R. and Bates, M.
The RUS Parser Control Structure.
In *Research in Knowledge Representation for Natural Language Understanding, Annual Report.*Bolt Beranek and Newman Inc., 1982.
BBN Report No. 5188.

[4] Damerau, F.J.
Operating Statistics for the Transformational Question Answering System.
*American Journal of Computational Linguistics* 7(1):30–42, 1981.

[5] Fass, D. and Wilks, Y.
Preference Semantics, Ill–Formedness, and Metaphor.
*American Journal of Computational Linguistics* 9(3–4):178–187, 1983.

[6] Grosz, B., Appelt, D. E., Martin, P., and Pereira, F.
*TEAM: An Experiment in the Design of Transportable Natural Language Interfaces.*
Technical Report 356, SRI International, 1985.
To appear in Artificial Intelligence.

[7] Kaczmarek, T., Mark, W., and Sondheimer, N.
The Consul/CUE Interface: An Integrated Interactive Environment.
In *Proceedings of CHI '83 Human Factors in Computing Systems*, pages 98–102. ACM, December, 1983.

[8] Mann, W.C. and Matthiessen, C.M.I.M.
Nigel: A Systemic Grammar for Text Generation.
*Systemic Perspectives on Discourses: Selected Theoretical Papers from the 9th International Systemic Workshop.*
Ablex, Norwood, NJ, forthcoming.

[9] Moore, R.C.
Natural Language Access to Databases – Theoretical/Technical Issues.
In *Proceedings of the 20th Annual Meeting of the Association for Computational Linguistics*, pages 44–45. Association for Computational Linguistics, June, 1982.

[10] Moser, M.G.
An Overview of NIKL, the New Implementation of KL–ONE.
In Sidner, C. L., et al. (editors), *Research in Knowledge Representation for Natural Language Understanding – Annual Report, 1 September 1982 – 31 August 1983*, pages 7–26.BBN Laboratories Report No. 5421, 1983.

[11] Moser, M.G.
Domain Dependent Semantic Acquisition.
In *The First Conference on Artificial Intelligence Applications*, pages 13–16. IEEE Computer Society, December, 1984.

[12] Sager, N.
The String Parser for Scientific Literature.
In R. Rustin (editor), *Natural Language Processing*, pages 61–88.Algorithmics Press, Inc., New York, NY, 1973.

[13] Scha, R.J.H.
English Words and Data Bases: How to Bridge the Gap.
In *Proceedings of the 20th Annual Meeting of the Association for Computational Linguistics*, pages 57–59. Association for Computational Linguistics, June, 1982.

[14] Scha, R.J.H.
*Logical Foundations for Question Answering.*
Technical Report, Eindhoven: Philips Research Labs, M.S. 12.331., 1983.

[15] Schmolze, J.G., Lipkis, T.A.
Classification in the KL–ONE Knowledge Representation System.
In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence.* 1983.

[16] Sridharan, N.S.
*Semi–Applicative Programming: Examples of Context Free Recognizers.*
Technical Report Report No. 6135, BBN Laboratories Inc., January, 1986.

[17] Stallard, D.
Data Modelling for Natural Language Access.
In *The First Conference on Artificial Intelligence Applications*, pages 19–24. IEEE Computer Society, December, 1984.

[18] Stallard, D.G.
A Terminological Simplification Transformation for Natural Language Question–Answering Systems.
In *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics.* Association for Computational Linguistics, July, 1986.
To appear.

[19] Stallard, D.G.
*Taxonomic Inference on Predicate Calculus Expressions.*
Technical Report, BBN Laboratories Inc., 1986.
In preparation.

[20] Thompson, B.H.
Linguistic Analysis of Natural Language Communication with Computers.
In *Proceedings of the Eighth International Conference on Computational Linguistics*, pages 190–201. International Committee on Computational Linguistics, October, 1980.

[21] Weischedel, R. M. and Sondheimer, N. K.
Meta–rules as a Basis for Processing Ill–Formed Input.
*American Journal of Computational Linguistics* 9(3–4):161–177, 1983.

[22] Weischedel, R.M. and Sondheimer, N.K.
*Relaxing Constraints in MIFIKL.*
Technical Report, USC/Information Sciences Institute, 1983.

[23] Woods, W.A.
Transition Network Grammars for Natural Language Analysis.
*CACM* 13(10):591–606, October, 1970.

# A Terminological Simplification Transformation for Natural Language Question-Answering Systems [1]

David G. Stallard
BBN Laboratories Inc.
10 Moulton St.
Cambridge, MA.
02238

## Abstract

A new method is presented for simplifying the logical expressions used to represent utterance meaning in a natural language system. This simplification method utilizes the encoded knowledge and the limited inference-making capability of a taxonomic knowledge representation system to reduce the constituent structure of logical expressions. The specific application is to the problem of mapping expressions of the meaning representation language to a database language capable of retrieving actual responses. Particular account is taken of the model-theoretic aspects of this problem.

## 1. Introduction

A common and useful strategy for constructing natural language interface systems is to divide the processing of an utterance into two major stages: the first mapping the utterance to a logical expression representing its "meaning" and the second producing from this logical expression the appropriate response. The second stage is not neccesarily trivial: the difficulty of its design is signifigantly affected by the complexity and generalness of the logical expressions it has to deal with. If this issue is not faced squarely, it may affect choices made elsewhere in the system. Indeed, a need to restrict the form of the meaning representation can be at odds with particular approaches towards producing it - as for example the "compositional" approach, which does not seek to control expression complexity by giving interpretations for whole phrasal patterns, but simply combines together the meaning of individual words in a manner appropriate to the syntax of the utterance. Such a conflict is certainly not desirable. we want to have freedom of linguistic action as well as to be able to obtain correct responses to utterances.

This paper treats in detail the particular manifestation of these issues for natural-language systems which serve as interfaces to a database: the problems that arise in a module which maps the meaning representation to a second logical language for

---

[1] The work presented here was supported under DARPA contract #N00014-85-C-0016. The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or of the United States Government. This paper was originally published in the *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics*, 10-13 June, 1986, Columbia University, New York. Requests for copies should be addressed to:

Dr.Donald E. Walker (ACL)
Bell Communications Research
435 South Street MRE 2A379
Morristown, NJ 07960, USA

expressing actual database queries. A module performing such a mapping is a component of such question-answering systems as TEAM [4], PHLIQA1 [7] and IRUS [.]. As an example of difficulties which may be encountered, consider the question "Was the patient's mother a diabetic?" whose logical representation must be mapped onto a particular boolean field which encodes for each patient whether or not this complex property is true. Any variation on this question which a compositional semantics might also handle, such as "Was diabetes a disease the patient's mother suffered from?", would result in a semantically equivalent but very different-looking logical expression; this different expression would also have to be mapped to this field. How to deal with these and many other possible variants, without making the mapping process excessively complex, is clearly a problem.

The solution which this paper presents is a new level of processing, intermediate between the other two: a novel simplification transformation which is performed on the result of semantic interpretation before the attempt is made to map it to the database. This simplification method relies on knowledge which is stored in a taxonomic knowledge representation system such as NIKL [5]. The principle behind the method is that an expression may be simplified by translating its subexpressions, where possible, into the language of NIKL, and classifying the result into the taxonomy to obtain a simpler equivalent for them. The result is to produce an equivalent but syntactically simpler expression in which fewer, but more specific, properties and relations appear. The benefit is that deductions from the expression may be more easily "read off"; in particular, the mapping becomes easier because the properties and relations appearing are more likely to be either those of the database or composable from them.

The body of the paper is divided into four sections. In the first, I will summarize some past treatments of the mapping between the meaning representation and the query language, and show the problems they fail to solve. The second section prepares the way by showing how to connect the taxonomic knowledge representation system to a logical language used for meaning representation. The third section presents the "recursive terminological simplification" algorithm itself. The last section describes the implementation status and suggests directions for interesting future work.

## 2. A Formal Treatment of the Mapping Problem

This section discusses some previous work on the problem of mapping between the logical language used for meaning representation and the logical language in which actual database queries are expressed. The difficulties which remain for these approaches will be pointed out.

A common organization for a database is in terms of tables with rows and columns. The standard formulation of these ideas is found in the relational model of Codd [3], in which the tables are characterized as relations over sets of atomic data values. The elements (rows) of a relation are called "tuples', while its individual argument places (columns) are termed its "attributes". Logical languages for the construction of queries, such as Codd's relational algebra, must make reference to the relations and attributes of the database.

The first issue to be faced in consideration of the mapping problem is what elements of the database to identify with the objects of discourse in the utterance - that is,

with the non-logical constants[2] in the meaning representation. In previous work [9] I have argued that these should not be the rows of the tables, as one might first think, but rather certain sets of the atomic attribute-values themselves. I presented an algorithm which converted expressions of a predicate calculus-based meaning representation language to the query language ERL, a relational algebra [3] extended with second-order operations. The translations of non-logical constants in the meaning representation were provided by fixed and local translation rules that were simply ERL expressions for computing the total extension of the constant in the database. The expressions so derived were then combined together in an appropriate way to yield an expression for computing the response for the entire meaning representation expression. If the algorithm encountered a non-logical constant for which no translation rule existed, the translation failed and the user was informed as to why the system could not answer his question.

By way of illustration, consider the following relational database, consisting of clinical history information about patients at a given hospital and of information about doctors working there:

```
PATIENTS(PATID,SEX,AGE,DISEASE,PHYS,DIAMOTHER)
DOCTORS(DOCID,NAME,SEX,SPECIALTY)
```

where "PHYS" is the ID of the treating physician, and "DIAMOTHER" is a boolean field indicating whether or not the patient's mother is diabetic. Here are the rules for the one-place predicate PATIENTS, the one-place predicate SPECIALTIES, and the two-place predicate TREATING-PHYSICIAN:

```
PATIENTS    => (PROJECT PATIENTS OVER PATID)

SPECIALTIES    => (PROJECT DOCTORS OVER SPECIALTY)

TREATING-PHYSICIAN    => (PROJECT (JOIN PATIENTS
                                        TO DOCTORS
                                        OVER PHYS DOCID)
                                  OVER PATID DOCID)
```

Note that while no table exists for physician SPECIALTIES, we can nonetheless give a rule for this predicate in way that is uniform with the rule given for the predicate PATIENTS.

One advantage of such local translation rules is their simplicity. Another advantage is that they enable us to treat database question-answering model-theoretically. The set-theoretic structure of the model is that which would be obtained by generating from the relations of the database the much larger set of "virtual" relations that are expressible as formulas of ERL. The interpretation function of the model is just the translation function itself. Note that it is a *partial* function because of the fact that some non-logical constants may not have translations. We speak therefore of the database constituting a "partially specified model" for the meaning representation language. Computation of a response to a user's request, instead of being characterizable only as a procedural operation, becomes interpretation in such a model.

A similar model-theoretic approach is advocated in the work on PHLIQA1 [8], in which a number of difficulties in writing local rules are identified and overcome. One class of techniques presented there allows for quite complex and general expressions

---

[2]This term, while a standard one in formal logic, may be confused with other uses of the word "constant". It simply refers to the function, predicate and ordinary constant symbols, such as "MOTHER" or "JOHN", whose denotations depend on the interpretation of the language, as opposed to fixed symbols like "FORALL","AND", "TRUE".

to result from local rule application, to which a post-translation simplification process is applied. Other special-purpose techniques are also presented, such as the creation of "proxies" to stand in for elements of a set for which only the cardinality is known.

A more difficult problem, for which these techniques do not provide a general treatment, arises when we want to get at information corresponding to a complex property whose component properties and relations are not themselves stored. For example, suppose the query "List patients whose mother was a diabetic", is represented by the meaning representation:

```
(display ↑(setof X:PATIENT
            (forall Y:PERSON (->(MOTHER X Y)
                                (DIABETIC Y)))))
```

The information to compute the answer may be found in the field DIAMOTHER above. It is very hard to see how we will use local rules to get to it, however, since nothing constructable from the database corresponds to the non-logical constants MOTHER and DIABETIC. The problem is that the database chooses to highlight the complex property DIAMOTHER while avoiding the cost of storing its constituent predicates MOTHER and DIABETIC - the conceptual units corresponding to the words of the utterance.

One way to get around these difficulties is of course to allow for a more general kind of transformation: a "global rule" which would match against a whole syntactic pattern like the univerally quantified sub-expression above. The disadvantage of this, as is pointed out in [8], is that the richness of both natural language and logic allows the same meaning to be expressed in many different ways, which a complete "global rule" would have to match. Strictly syntactic variation is possible: pieces of the pattern may be spread out over the expression, from which the pattern match would have to grab them. Equivalent formulations of the query may also use completely different terms. For example, the user might have employed the equivalent phrase "female parent" in place of the word "mother", presumably causing the semantic interpretation to yield a logical form with the different predicates "PARENT" and "FEMALE". This would not match the pattern. It becomes clear that the "pattern-matching" to be performed here is not the literal kind, and that it involves unspecified and arbitrary amounts of inference.

The alternative approach presented by this paper takes explicit account of the fact that certain properties and relations, like "DIAMOTHER", can be regarded as built up from others. In the next section we will show how the properties and relations whose extensions the database stores can be axiomatized in terms of the ones that are more basic in the application domain. This prepares the way for the simplification transformation itself, which will rely on a limited and sound form of inference to reverse the axiomatization and transform the meaning representation, where possible, to an expression that uses only these database properties and relations. In this way, the local rule paradigm can be substantially restored.

# 3. Knowledge Representation and Question-Answering

The purpose of this section of the paper is to present a way of connecting the meaning representation language to a taxonomic knowledge representation system in such a way that the inference-making capability of the latter is available and useful for the problems this paper addresses. Our approach may be constrasted with that of others, e.g. TEAM in which such a taxonomy is used mainly for simple inheritance and attachment duties.

The knowledge representation system used in this work is NIKL [5]. Since NIKL has been described rather fully in the references, I will give only a brief summary here.

NIKL is a taxonomic frame-like system with two basic data structures: concepts and

roles. Concepts are just classes of entities, for which roles function somewhat as attributes. At any given concept we can restrict a role to be filled by some other concept, or place a restriction on the number of individual "fillers" of the role there. A role has one concept as its "domain" and another as its "range": the role is a relation between the sets these two concepts denote. Concepts are arranged in a hierarchy of sub-concepts and superconcepts; roles are similarly arranged. Both concepts and roles may associated with names. In logical terms, a concept may be identified as the one-place predicate with its name, and a role as the two-place predicates with its name.

I will now give the meaning postulates for a term-forming algebra, similar to the one described in [2] in which one can write down the sort of NIKL expressions I will need. Expressions in this language are combinable to yield a complex concept or role as their value.

```
(CONJ C1 — CN)  ≡ (lambda (X) (and (C1 X) — (Cn X)))

(VALUERESTRICT R C) ≡ (lambda (X) (forall Y (-> (R X Y)
                                                (C Y)))

(NUMBERRESTRICT R 1 NIL) ≡ (lambda (X) (exists Y (R X Y)))

(VRDIFF R C) ≡ (lambda (X Y) (and (R X Y) (C Y)))

(DOMAINDIFF R C) ≡ (lambda (X Y) (and (R X Y) (C X)))
```

The key feature of NIKL which we will make use of is its classifier, which computes subsumption and equivalence relations between concepts, and a limited form of this among roles. Subsumption is sound, and thus indicates entailment between terms:

```
(SUBSUMES C1 C2)  ->  (forall X (-> (C2 X) (C1 X)))
```

If the classifier algorithm is complete, the reverse is also true, and entailment indicates subsumption. Intuitively, this means that classified concepts are pushed down as far in the hierarchy as they can go.

Also associated with the NIKL system, though not a part of the core language definition, is a symbol table which associates atomic names with the roles or concepts they denote, and concepts and roles with the names denoting them. If a concept or role does not have a name, the symbol table is able to create and install one for it when demanded.

**The domain model**

In order to be able to use NIKL in the analysis of expressions in the meaning representation language, we make the following stipulations for any use of the language in a given domain. First, any one-place predicate must name a concept, and any two-place predicate name a role. Second, any constant, unless a number or a string, must name an "individual" concept - a particular kind of NIKL concept that is defined to have at most one member. N-ary functions are treated as a N+1 - ary predicates. A predicate of N arguments, where N is greater than 2, is reified as a concept with N roles. This set of concepts and roles, together with the logical relationships between them, we call the "domain model".

Note that all we have done is to stipulate an one-to-one correspondence between two sets of things - the concepts and roles in the domain model and the non-logical constants of the meaning representation language. If we wish to include a new non-logical constant in the language we must enter the corresponding concept or role in the domain model. Similarly, the NIKL system's creating a new concept or role, and creation of a name in the symbol table to stand for it, furnishes us with a new non-logical constant.

**Axiomatization of the database in terms of the domain model**

The translation rules presented earlier effectively seek to axiomatize the properties and relations of the domain model in terms of those of the database. This is not the only way to bridge the gap. One might also try the reverse: to axiomatize the properties and relations of the database in terms of those of the domain model. Consider the DIAMOTHER field of our sample database. We can write this in NIKL as the concept PATIENT-WITH-DIABETIC-MOTHER using terms already present in the domain model:

```
(CONJ PATIENT
      (VALUERESTRICT MOTHER
                     DIABETIC))
```

If we wanted to axiomatize the relation implied by the SEX attribute of the PATIENTS table in our database, we could readily do so by defining the role PATIENT-SEX in terms of the domain model relation SEX:

```
(DOMAINDIFF SEX
            PATIENT)
```

These two defined terms can actually be entered into the model, and be treated just like any others there. For example, they can now appear as predicate letters in meaning representations. Moreover, to the associated data structure we can attach a translation rule, just as we have been doing with the original domain model elements. Thus, will attach to the concept PATIENT-WITH-DIABETIC-MOTHER the rule:

```
(PROJECT (SELECT FROM PATIENTS WHERE (EQ DIAMOTHER "YES"))
         OVER PATID)
```

The next section will illustrate how we map from expressions using "original" domain model elements to the ones we create for axiomatizing the database, using the NIKL system and its classifier.

# 4. Recursive Terminological Simplification

We now present the actual simplification method. It is composed of two separate transformations which are applied one after the other. The first, the "contraction phase", seeks to contract complicated subexpressions (particularly nested quantifications) to simpler one-place predications, and to further restrict the "sorts" of remaining bound variables on the basis of the one-place predicates so found. The second part of the transformation, the "role-tightening" phase, replaces general relations in the expression with more specific relations which are lower in the NIKL hierarchy. These more specific relations are obtained from the more general by considering the sorts of the variables upon which a given relational predication is made.

**The contraction phase**

The contraction phase is an algorithm with three steps, which occur sequentially upon application to any expression of the meaning representation. First, the contraction phase applies itself recursively to each non-constant subexpression of the expression. Second, depending upon the syntactic category of the expression, one of the "pre-simplification" transformations is applied to place it in a normalized form. Third and finally, one of the actual simplification transformations is used to convert the expression to one of a simpler syntactic category.

Before working through the example, I will lay out the transformations in detail. In what follows, X and X1,X2 -- Xn are variables in the meaning representation language. The symbol "<rest>" denotes a (possibly empty) sequence of formulae. The expression

"(FORMULA X)" denotes a formula of the meaning representation language in which the variable X (and perhaps others) appears freely. The symbol "<quant>" is to be understood as being replaced by either the operator SETOF or the quantifier EXISTS.

First, the normalization transformations, which simply re-arrange the constituents of the expressions to a more convienent form without changing its syntactic category:

```
(1)  (and (P1 X1) (P2 X1) — (PN X1)
          (Q1 X2) (Q2 X2) — (QN X2)
          <rest>)

        ==>  (and (P' X1) (Q' X2) <rest>)

       where P' := (CONJ P1 P2 — PN)
       and   Q' := (CONJ Q1 Q2 — QN)

(2)  (<quant> X:S (and (P X) <rest>)  ==>

       (<quant> X:S' (and <rest>))
       where S' := (CONJ S P)

(3) (<quant> X:S (P X))  ==>

       (<quant> X:S')
       where S' := (CONJ S P)

(4) (forall X:S (-> (and (P X) <rest>)
                    (FORMULA X))   ==>

       (forall X:S' (-> (and <rest>)
                        (FORMULA X)))
```

In (2) and (4) above, the conjunction or implication, respectively, are collapsed out if the sequence <rest> is empty.

Now the actual simplification transformations, which seek to reduce a complex sub-expression to a one-place predication.

```
(5) (forall X2:S (-> (R X1 X2) (P X2)))
       ==>    (P' X1)
      where P' := (VALUERESTRICT (VRDIFF R S) P)

(6) (exists X2:S (R X1 X2))  ==>  (P' X1)

      where P' := (VALUERESTRICT R S)
      and R must be a functional role

(7) (exists X2:S (R X1 X2))   ==> (P' X1)

     where P' := (NUMBERRESTRICT (VRDIFF R S) 1 NIL)

(8) (and (P X))  ==> (P X)

(9) (R X C) ==> (P X)

      where P := (VALUERESTRICT R C)
      and R is functional, C an individual concept
```

Now, let us suppose that the exercise at the end of the last section has been carried out, and that the concept PATIENT-WITH-DIABETIC-MOTHER has been created and given the appropriate translation rule. To return to the query "List patients whose mother was a diabetic", we recall that it has the meaning representation:

```
(DISPLAY 1(SETOF X:PATIENTS
                 (FORALL Y:PERSON
```

```
(-> (MOTHER X Y)
    (DIABETIC Y)))))
```

Upon application to the SETOF expression, the algorithm first applies itself to the inner FORALL. The syntactic patterns of none of the pre-simplification transformations (2) - (4) are satisfied, so transformation (5) is applied right way to produce the NIKL concept:

```
(VALUERESTRICT (VRDIFF MOTHER PERSON)
               DIABETIC)
```

This is given to the NIKL classifier, which compares it to other concepts already in the hierarchy. Since MOTHER has PERSON as its range already, (VRDIFF MOTHER PERSON) is just MOTHER again. The classifier thus computes that the concept specified above is a subconcept of PERSON - a PERSON such that his MOTHER was a DIABETIC. If this is not found to be equivalent to any pre-existing concept, the system assigns the concept a new name which no other concept has, say PERSON-1. The outcome of the simplification of the whole FORALL is then just the much simpler expression:

```
(PERSON-1 X)
```

The recursive simplification of the arguments to the SETOF is now completed, and the resulting expression is:

```
(DISPLAY †(SETOF X:PATIENT
                (PERSON-1 X)))
```

Transformations can now be applied to the SETOF expression itself. The pre-simplification transformation (3) is found to apply, and a concept expressed by:

```
(CONJ PATIENT PERSON-1)
```

*is given to the classifier, which recognizes* it as equivalent to the already existing concept PATIENT-WITH-DIABETIC-MOTHER. Since any concept can serve as a sort, the final simplification is:

```
(DISPLAY †(SETOF X:PATIENT-WITH-DIABETIC-MOTHER))
```

This is the very concept for which we have a rule, so the ERL translation is:

```
(PRINT FROM (SELECT FROM PATIENT
                    WHERE (EQ DIAMOTHER "YES"))
      PATID)
```

Suppose now that the semantic interpretation system assigned a different logical expression to represent the query "List patients whose mother was a diabetic", in which the embedded quantification is existential instead of universal. This might actually be more in line with the number of the embedded noun. The meaning representation would now be:

```
(display †(setof X:PATIENT
                 (exists Y:PERSON (and (MOTHER X Y)
                                       (DIABETIC Y)))
```

The recursive application of the algorithm proceeds as before. Now, however, the pre-simplification transformation (2) may be applied to yield:

```
(exists Y:DIABETIC (MOTHER X Y))
```

since a DIABETIC is already a PERSON. Transformation (6) can be applied if MOTHER is a "functional" role - mapping each and every person to exactly one mother. This can be checked by asking the NIKL system if a number restriction has been attached at the domain of the role, PERSON, specifying that it have both a minimum and a maximum of one. If the author of the domain model has provided this reasonable and perfectly true fact about motherhood, (6) can proceed to yield:

```
(PATIENT-WITH-DIABETIC-MOTHER X)
```

as in the preceding example.

**The role tightening phase**

This phase is quite simple. After the contraction phase has been run on the whole expression, a number of variables have had their sorts changed to tighter ones. This transformation sweeps through an expression and changes the roles in the expression on that basis. Thus:

```
(10) (R X Y)  ==>  (R' X Y)

     where S1 is the sort of X
     and S2 is the sort of Y
     and R' := (DOMAINDIFF (VRDIFF R S2)
                           S1)
```

One can see that a use of the relation SEX, where the sort of the first argument is known to be DOCTOR, can readily be converted to a use the relation DOCTOR-SEX.

**Back conversion: going in the reverse direction**

There will be times when the simplification transformation will "overshoot", creating and using new predicate letters which have not been seen before by classifying new data structures into the model to correspond to them. The use of such a new predicate letter can then be treated exactly as would its equivalent lambda-definition, which we can readily obtain by consulting the NIKL model. For example, a query about the sexes of leukemia victims may after simplification result in a rather strange role being created and entered into the hierarchy:

```
PATIENT-SEX-1 := (DOMAINDIFF PATIENT-SEX LEUKEMIA-PATIENT)
```

This role is a direct descendant of PATIENT-SEX, its name is system generated. By the meaning-postulate of DOMAINDIFF given in section 3 above, it can be rewritten as the following lambda-abstract:

```
(lambda (X Y) (and (PATIENT-SEX X Y)
                   (LEUKEMIA-PATIENT X)))
```

For PATIENT-SEX we of course have a translation rule as discussed in section 2. A rule for LEUKEMIA-PATIENT can be imagined as involving the DISEASE field of the PATIENTS table. At this point we can simply call the translation algorithm recursively, and it will come up with a translation:

```
(PROJECT (SELECT FROM PATIENTS
                 WHERE (EQ DISEASE "LEUK"))
         OVER PATID SEX)
```

This supplies us with the needed rule. As a bonus, we can avoid having to recompute it later by simply attaching it to the role in the normal way. The similar computation of rules for complex concepts and roles which are already in the domain comes for free.

# 5. Conclusions, Implementation Status and Further Work

As of this writing, we have incorporated NIKL into the implementation of our natural language question-answering system, IRUS. NIKL is used to represent the knowledge in a Navy battle-management domain. The simplification transformation described in this paper has been implemented in this combined system, and the axiomatization of the database as described above is being added to the domain model. At that point, the

methodology will be tested as a solution to the difficulties now being experienced by those trying to write the translation rules for the complex database and domain of the Fleet Command Center Battle Management Program of DARPA's Strategic Computing Program.

I have presented a limited inference method on predicate calculus expressions, whose intent is to place them in a canonical form that makes other inferences easier to make. Metaphorically, it can be regarded as "sinking" the expression lower in a certain logical space. The goal is to push it down to the "level" of the database predicates, or below. We cannot guarantee that we will always place the expression as low as it could possibly go – that problem is undecidable. But we can go a good distance, and this by itself is very useful for restoring the tractability of the mapping transformation and other sorts of deductive operations [10].

Somewhat similar simplifications are performed in the work on ARGON [6], but for a different purpose. There the database is assumed to be a full, rather than a partially specified, model and simplifications are performed only to gain an increase in efficiency. The distinguishing feature of the present work is its operation on an expression in a logical language for English meaning representation, rather than for restricted queries. A database, given the purposes for which it is designed, cannot constitute a full model for such a language. Thus, the terminological simplification is needed to reduce the logical expression, when possible, to an expression in a "sub-language" of the first for which the database is a full model.

An important outcome of this work is the perspective it gives on knowledge representation systems like NIKL. It shows how workers in other fields, while maintaining other logical systems as their primary mode of representation, can use these systems in practical ways. Certainly NIKL and NIKL-like systems could never be used as full meaning representations – they don't have enough expressive power, and were never meant to. This does not mean we have to disregard them, however. The right perspective is to view them as attached inference engines to perform limited tasks having to do with their specialty – the relationships between the various properties and relations that make up a subject domain in the real world.

# Acknowledgements

First and foremost, I must thank Remko Scha, both for valuable and stimulating technical discussions as well as for patient editorial criticism. This paper has also benefited from the comments of Ralph Weischedel and Jos De Bruin. Beth Groundwater of SAIC was patient enough to use the software this work produced. I would like to thank them, and thank as well the other members of the IRUS project – Damaris Ayuso, Lance Ramshaw and Varda Shaked – for the many pleasant and productive interactions I have had with them.

# References


[1] Bates, Madeleine and Bobrow, Robert J.
A Transportable Natural Language Interface for Information Retrieval.
In *Proceedings of the 6th Annual International ACM SIGIR Conference*. ACM Special Interest Group on Information Retrieval and American Society for Information Science, Washington, D.C., June, 1983.

[2] Brachman, R.J., Fikes, R.E., and Levesque, H.J.
Krypton: A Functional Approach to Knowledge Representation.
*IEEE Computer, Special Issue on Knowledge Representation* , October, 1983.

[3] Codd,E.F.
A Relational Model of Data for Large Shared Data Banks.
*CACM* 13(6), June, 1970.

[4] Barbara Grosz, Douglas E. Appelt, Paul Martin, and Fernando Pereira.
*TEAM: An Experiment in the Design of Transportable Natural-Language Interfaces.*
Technical Report 356, SRI International, Menlo Park, CA, August, 1985.

[5] Moser, Margaret.
*An Overview of NIKL.*
Technical Report Section of BBN Report No. 5421, Bolt Beranek and Newman Inc., 1983.

[6] Patel-Schneider, P.F., H.J. Levesque, and R.J. Brachman.
ARGON: Knowledge Representation meets Information Retrieveal.
In *Proceedings of The First Conference on Artificial Intelligence Applications.* IEEE Computer Society, Denver, Colorado, December, 1984.

[7] W.J.H.J. Bronnenberg, H.C. Bunt, S.P.J. Landsbergen, R.J.H. Scha, W.J. Schoenmakers and E.P.C. van Utteren.
The Question Answering System PHLIQA1.
In L. Bolc (editor), *Natural Language Question Answering Systems.* Macmillan, 1980.

[8] Scha, Remko J.H.
English Words and Data Bases: How to Bridge the Gap.
In *20th Annual Meeting of the Association for Computational Linguistics, Toronto.* Association for Computational Linguistics, June, 1982.

[9] Stallard, David G.
Data Modeling for Natural Language Access.
In *Proceedings of the First IEEE Conference on Applied Artificial Intelligence, Denver, Colorado.* IEEE, December, 1984.

[10] Stallard, David G.
Taxonomic Inference on Predicate Calculus Expressions.
Submitted to AAAI April 1, 1986.

# SECTION 3: RESEARCH CONTRIBUTIONS

*New York University/SDC*

# Model-based Analysis of Messages about Equipment

Ralph Grishman, Tomasz Ksiezyk, and Ngo Thanh Nhan

Department of Computer Science
Courant Institute of Mathematical Sciences
New York University

**ABSTRACT**

The aim of PROTEUS -- a system for the analysis of short technical texts -- is to increase the reliability of the analysis process through the integration of syntactic and semantic constraints, domain knowledge, and knowledge of discourse structure. This system is initially being applied to the analysis of messages describing the failure, diagnosis, and repair of selected pieces of equipment. This has required us to develop a detailed model of the structure and function of the equipment involved. We focus in this paper on the nature of this model and the roles it plays in the syntactic and semantic analysis of the text.

## 1. Introduction

Considerable progress has been made in developing systems which understand short passages of technical text. Several prototypes have been developed, for such domains as patient medical records [Sager 1978], equipment failure reports [Marsh 1984], and intelligence messages [Montgomery 1983]. Except for very narrow domains such as weather reports, however, none of these systems seem to be robust enough for operational use. Typical success rates - where any are reported - are in the range of 70 to 80% of sentences correctly analyzed; substantially better rates are very hard to obtain, even with careful system tuning[1].

Our objective in developing PROTEUS (the PROtotype TExt Understanding System) is to see if this rate can be substantially improved for a domain of moderate complexity. In order to achieve this improvement, we must bring to bear on the language analysis task the various syntactic, semantic, and discourse constraints, along with a fairly detailed knowledge of the domain of discourse. Our system is initially being applied to equipment failure reports ("CASREPs") for selected equipment on board Navy ships (initially, the equipment in the starting air system); a sample message is shown in Figure 1. In this case, the domain knowledge is the knowledge of the structure and function of these pieces of equipment.

In this paper we first present an overview of the PROTEUS system. We then focus on the domain information: how it is represented, how it is integrated with the language processing, and how it serves to resolve ambiguities in the input text.

## 2. Prior work

New York University has been involved in the automated analysis and structuring of technical text for over a decade. Most of this work has been on medical records [Sager 1978, Hirschman 1982], but we have also been involved with the Naval Research Laboratory on a system for CASREP's [Marsh 1984]. These systems used domain-specific frame-like target structures, and employed selectional constraints to weed out bad parses, but did not incorporate detailed domain models. Our experience with these systems - in particular, the difficulty of obtaining success rates (% of sentences correctly analyzed) much above 75% - led us to our work on PROTEUS.

The use of detailed domain models in language processing systems is, of course, not new. Script-based systems, and some of the frame-based language analysis systems, have been motivated by a desire to incorporate detailed domain knowledge. The task we confront, however, differs in several regards from those of earlier systems. One is the matter of scale; our initial set of equipment - the starting air system for a gas turbine - includes several hundred separately nameable components (and many lesser components, such as bolts and

---

DURING NORMAL START CYCLE OF 1A GAS TURBINE, APPROX 90 SEC AFTER CLUTCH ENGAGEMENT, LOW LUBE OIL AND FAIL TO ENGAGE ALARM WERE RECEIVED ON THE ACC. (ALL CONDITIONS WERE NORMAL INITIALLY). SAC WAS REMOVED AND METAL CHUNKS FOUND IN OIL PAN. LUBE OIL PUMP WAS REMOVED AND WAS FOUND TO BE SEIZED. DRIVEN GEAR WAS SHEARED ON PUMP SHAFT.

Figure 1. A sample CASREP about a starting air compressor (SAC).

---

[1]Substantially better rates have been cited for strongly expectation-based parsers, which are considered successful if they locate all the expected items within an input text.

gear teeth, without specific names). While not raising any intrinsic difficulties, a domain of this size clearly provides a more rigorous test of our ability to acquire and organize domain knowledge than did many earlier "toy" domains.

Another unusual aspect is the *nature* of the domain information. Scripts, for example, encode essentially procedural information (how to perform complex actions). The information for our domain, in contrast, is primarily structural (part-whole relationships, interconnections, etc.) and to a lesser degree functional. This difference is reflected in differences in the way the information is used - in particular, in the analysis of noun phrases, as we shall see below. Our domain information bears greater resemblance to that used in some equipment simulation packages (e.g., STEAMER [Hollan 1984]) and diagnosis packages [Cantone 1983] than it does to that conventionally seen in natural language systems.

The domain knowledge plays a role in many phases of the language processing task: in the recovery of implicit operands and intersentential relations, in the analysis of noun-phrase reference, and in the determination of syntactic and semantic structure. In particular, we shall consider below its role in the processing of compound nominals, which appear frequently in such technical domains. There have been several prior studies of the processing of such compounds. The work both of Brachman [1978] and of McDonald and Hayes-Roth [McDonald 1978] emphasized the use of search procedures within semantic networks to identify the wide variety of implicit relations possible with compound nominals. We have also used network search techniques, although of a more directed sort. However, their work cited isolated examples from a variety of areas to show the generality of their approach, while we have been concerned with achieving detailed and thorough coverage within a narrower domain. Finin [1980, 1986] has sought to develop, within a sublanguage, general semantic categories for the relations and consituents involved in compounds. Although there are some similarities to our classification efforts, he also has aimed at providing a relatively broad and loose set of constraints. In contrast, the detailed knowledge in our equipment model -- provided for several purposes, of which noun phrase interpretation is only one -- make possible much tighter constraints in our system.

### 3. System overview

The PROTEUS system has three major components: a syntactic analyzer, a semantic analyzer, and a discourse analyzer. The syntactic analyzer parses the input and regularizes the clausal syntactic structure. The semantic analyzer converts this to a "logical form" specifying states and actions with reference to specific components of the equipment. The discourse component establishes temporal and causal links between these states and actions.

Initial implementations have been completed of the syntactic and semantic components, so that we are able to generate semantic representations of individual sentences. The discourse component is still under development, and so will not be discussed further here.

The syntactic analyzer uses an augmented-context-free grammar and an active chart parser. The grammar is generally based on linguistic string theory and the Linguistic String Project English Grammar [Sager 1981] and includes extensions to handle the various sentence fragment forms found in these messages [Marsh 1983]; it is written in a modified form of the Restriction Language used by the NYU Linguistic String Parser [Sager 1975]. Syntactic regularization maps the various forms of clauses (active, passive, relative, reduced relative, fragmentary) into a canonical form *(verb operand1, operand2...)* The regularization is performed by a set of interpretation rules which are associated with the individual productions and which build the regularized syntactic structure compositionally.[2]

[2] The parser and syntactic regularization procedures were developed by Jean Mark Gawron. The regularization procedures were modeled after those developed for a GPSG parser [Gawron 1982], although the generated structures are quite different.

The semantic analysis component consists of two parts: *clause semantics* and *noun phrase semantics*. The clause semantics maps a clause (a verb plus operands which include syntactic case labels) into a predicate with arguments representing a state or action. Each verb and operand belongs to one or more *semantic classes*. Clause semantics relys on a set of pattern-action rules to perform the translation, with one pattern for each valid combination of verb and operand classes. Noun phrase semantics maps a noun phrase into the identifier of the equipment component specified by that phrase. Noun phrase semantics depends heavily on the equipment model, and so will be discussed further in a later section.

(The division between the two parts of semantic analysis is not quite so neat as the foregoing would suggest. Some noun phrases are nominalizations representing states or actions; these are processed by clause semantics. In many noun phrases, some modifiers identify the object and the remainder describe its state. For example, in "broken hub ring gear", *hub* and *ring* identify the gear, *broken* describes its state. We return to this problem in our description of noun phrase semantics below.)

Our long-term objective is to dynamically schedule among the three analysis components (syntax, semantics, and discourse), as is done in some blackboard models. For program development, however, we have found it better to use a sequential organization (first syntax, then semantics, then discourse). In order to have syntactic choices influenced by semantics and discourse, and semantic choices influenced by discourse, each component may generate multiple analyses, some of which are rejected by later stages. Sometime these multiple analyses are transmitted explicitly, as a list of alternatives. More often, however, they are transmitted using a representation neutral with respect to particular features. The output of syntactic analysis is neutral with respect to quantifier scope. It is also neutral with respect to the distribution of modifiers in conjoined noun phrases (for example, in "filter change and adjustment of pressure regulator," whether *filter* modifies *adjustment* and *of pressure regulator* modifies *change*). Furthermore, it does not assign structure to prenominal adjectives and nouns (so for example, in the phrase "low lube oil pressure alarm" it does not decide whether *low* modifies *lube, oil, pressure,* or *alarm*).

This system development has been conducted in close cooperation with a group at the System Development Corp., Paoli, PA. Their system, PUNDIT [Palmer 1986], is written in PROLOG but has many points of commonality with PROTEUS in terms of overall structure, grammar, and semantic representation. They are involved in future development of several areas, including semantic representation, time analysis, and anaphora resolution, for both the PUNDIT and PROTEUS systems.

## 4. The equipment model

The equipment model currently serves three functions within our system:

*object identification*. The noun phrases in the message are matched against the model (by a procedure outlined in the next section) in order to identify the objects referenced in the message. This is important both for syntactic disambiguation and as a prelude to applying domain-specific inferences.

*identification of intersentential relations*. The identification of these relations (temporal, causal, and others) is important both for disambiguation (of adjuncts and anaphoric references, in particular) and for establishing the meaning of the message as a whole. Much of the information needed for this process - information on the structure of the equipment and the function of its components - is recorded in the equipment model.

*display of equipment structure and status*. In order to provide some feedback to indicate whether the text was correctly understood, our system displays a structural diagram of the equipment at several levels of detail. Objects mentioned in the text, and changes in

equipment status described in the message, can be shown on the display. The information for generating these displays (positions, shapes, etc.) is stored with the equipment model.

The messages refer to relatively low-level comp nents, such as individual gears within the air compressor. We therefore had to constuct a relatively detailed model of the equipment involved. Our model has been developed through a study of the Navy manuals for this equipment.

The model is basically organized as two hierarchies: a type-instance hierarchy and a part-whole hierarchy. The leaves of the part-whole hierarchy are called *basic parts;* the internal nodes (composite objects) are called *systems*. We record for each system the primary *medium* which it provides, conveys, or transforms; in our starting air system, the three media are compressed air, lubricating oil, and mechanical rotation. We have organized our part-whole hierarchy in part along functional lines (rather than purely on physical proximity), grouping together parts which are connected together and operate on the same medium.

Since some parts are identified by their physical location, we provide a *location* field in both *basic part* and *system* nodes. Both types of nodes also have a *function* field, which indicates the effect of this part on the media or other parts. Nodes of specific types may have additional fields; for example, some mechanical components have a *speed* field.

All of the fields just mentioned record permanent characteristics of the parts. In addition, each node has an *operational-status* field, which holds information about a part which is reported in a message.

The model contains a lot of information about equipment structure which is specific to a particular piece of equipment. Some information, however, is more general: for example, that gears have teeth,or that impellors have blades. It would be most uneconomic to have a separate instance of *tooth* for each gear in the model. Instead we create an instance of the teeth for a specific gear when it is referenced in the text. Such very-low-level objects, which are instantiated dynamically as needed, are called *components*.

The equipment model has been implemented using flavors on a Symbolics LISP Machine. Types of objects are represented by flavors; instances of objects are represented by instances of flavors. The part-whole hierarchy and other fields are stored in instance variables. The structure display is performed by procedures associated with the flavors. The equipment model, and its use in the system, are described in more detail in [Ksiezyk 1986].

## 5. Noun phrase analysis

The syntactic analysis component analyzes the clause structure and delimits the noun phrases, but does not assign any structure to the pre-nominal modifiers. The noun phrase analyzer within the semantic component therefore has a dual role: to determine the structure of the pre-nominal modifiers and to identify the instance in the equipment model named by the noun phrase (or the set of instances, if this phrase could be applied to any of several parts). (Although there are a limited number of instances, it is not possible to record a single name for each part and then interpret noun phrases by simply looking the name up in a table. A single part can be named in many different ways -- depending in part on prior context -- so a full-fledged interpretation procedure is required.)

The noun phrase is analyzed bottom-up using a set of reduction rules. Each reduction rule combines the head of a phrase with some of its modifiers to form a larger constituent. By reference to the model, each rule also determines the set of instances which can be named by the constituent; if the set is empty, the application of the rule is rejected. Reductions are performed repeatedly until the entire phrase is reduced to a single constituent. If no such reduction is possible, the syntactic analysis is rejected; in this way noun phrase semantics can weed out some incorrect syntactic analyses.

The applicable reductions are determined by the dictionary entries for the words in the noun phrase. Each word is assigned two properties, its *model class* and its *semantic class*. The model class indicates how the word can be related to some entity in the domain model. One value of model class is *instance*, specifying that the word names a set of instances in the model; this set is also included in the dictionary entry. Examples are "pump", "shaft", "gear", etc. Larger constituents built while analyzing the noun phrase are also considered to be of type *instance*. One reduction rule allows us to combine two instances:

instance + instance → instance

for example, "LO" + "PUMP" → "LO PUMP", "SAC" + ("LO PUMP") → "SAC LO PUMP". The set of model instances for the result consists of those instances of the second constituent which can be linked through some path in the model to some instance of the first constituent. The types of links traversed in the search are a function of the *semantic class* of the first constituent; for example, "SAC" has the semantic class *machinery*, so we search the part/whole links, the location links, and the from/to links (which tie together components of the same system).

There are several other model classes and corresponding reduction rules. The class *slot-filler* is used for words which are values of features of instances, but are not themselves instances (for example, "LUBE" in the phrase "LUBE OIL"). The class *slot-name* is used for words which correspond to feature names, such as "SPEED" in "HIGH SPEED ASSEMBLY". The class *component* is used for parts which (as explained in the previous section) are not instantiated in the permanent equipment model but can be instantiated dynamically as needed.

Modifiers describing the state of a part, such as "cracked" or "sheared", are handled differently. If noun phrase semantics gets the input "sheared ring gear" it will look for an instance of ring gear with the *operational-state* "sheared". Such an instance would be present if a *previous* sentence had mentioned that a gear was sheared. If such an instance is found, it is identified as the correct referent; noun phrase semantics has in effect done anaphora resolution. If no instance is found, noun phrase semantics returns the instances of "ring gear" and the left-over modifier "sheared". Clause semantics (which invokes noun phrase semantics) then treats this like a clause "ring gear was sheared"; later in the processing of this sentence, this will cause "sheared" to be assigned as the operational-state of ring gear.

A related technique can be used to handle some of the ambiguities in cojoined noun phrases. For example, in the sentence "INVESTIGATION REVEALED STRIPPED LO PUMP DRIVE AND HUB RING GEAR", syntax alone cannot determine which of the modifiers "STRIPPED", "LO", "PUMP", or "DRIVE" also modify "HUB RING GEAR". So syntax marks these as *possibly* applicable to "HUB RING GEAR" and passes the phrase to semantics. If semantics finds that some of these modifiers cannot be integrated into the noun phrase, they will be ignored, thus implicitly resolving the syntactic ambiguity.

## 6. Conclusion

We have described a new text-processing system, PROTEUS, for analyzing messages about equipment failure. We have focussed on its equipment model and the role of this model in the process of interpreting of noun phrases. This process is part of semantic analysis but also plays a role in syntactic analysis and discourse analysis.

In addition to the elaboration of the existing components, substantial work will be required in at least two areas before we can hope to obtain a robust text processing system. First, we are developing a discourse component to identify temporal and plausible causal links between sentences. This information is needed not only for some of the applications (e.g., message summarization) but also to resolve some of the syntactic and semantic ambiguities in the messages. Second, we will need to move from a pass/fail strategy for enforcing our constraints to a best-fit strategy. Because of imperfections in the input, and the

inevitable omissions in a model as complex as ours, we must expect that many messages will violate one or another constraint; by employing a rich set of constraints, however, and selecting the analysis which violates the fewest constraints, we beleive that we will be able to identify the intended reading for most sentences.

The initial motivation for the system has been the conversion of a stream of messages to a data base for subsequent querying, summarization, and trend analysis. However, the use of a detailed equipment model, similar to that employed in simulation and diagnostic systems, suggests that it may be equally useful as an interface for such systems. A diagnostic system, for example, would then be able to accept initial observations in the form of a brief textual summary rather than force the user to go through an elaborate questionnaire; this may be a substantial advantage for broad-coverage diagnostic systems, which must be able to accept a wide variety of different symptoms.

**Acknowledgement**

This research was supported in part by the Defense Advanced Research Projects Agency under contract N00014-85-K-0163 from the office of Naval Research, and by the National Science Foundation under grant DCR-85-01843.

## REFERENCES


[Brachman 1978] Brachman, R. A. A structured paradigm for representing knowledge. Tech. Rep. No. 3605, Bolt Beranek and Newman Inc., Cambridge, MA.

[Cantone 1983] Cantone, R., Pipitone, F., Lander, W. B., and Marrone, M. Model-based probabilistic reasoning for electronics troubleshooting. *Proc. Eighth Intl. Joint Conf. Artificial Intelligence*, Karlsruhe, West Germany.

[Finin 1980] Finin, T. The semantic interpretation of compound nominals. *Proc. First National Conf. on Artificial Intelligence*. Stanford, CA., Am. Assn. of Artificial Intelligence.

[Finin 1986] Finin, T. Nominal compounds in a limited context. In *Analyzing Language in Restricted Domains*, R. Grishman and R. Kittredge, Eds. Lawrence Erlbaum Assoc., Hillsdale, NJ.

[Gawron 1982] Gawron, J. M., King, J. J., Lamping, J., Loebner, E. E., Paulson, E. A., Pullum, G. K., Sag, I. A., and Wasow, T. A. Processing English with a generalized phrase structure grammar. *Proc. 20th Annual Meeting Assn. Computational Linguistics*, Toronto, Canada.

[Hirschman 1982] Hirschman, L., and Sager, N. Automatic information formatting of a medical sublanguage. In *Sublanguage: Studies of Language in Restricted Domains*, R. Kittredge and J. Lehrberger, Eds. Walter de Gruyter, Berlin.

[Hollan 1984] Hollan, J., Hutchins, E., and Weitzman, L. STEAMER: an interactive inspectable simulation-based training system. *AI Magazine*, Summer 1984, 15-27.

[Ksiezyk 1986] Ksiezyk, T. An equipment model and its role in noun phrase interpretation. Submitted to *ACM Conf. on Object Oriented Programming Systems, Languages, and Applications*, Portland, OR.

[Marsh 1983] Marsh, E. Utilizing domain-specific information for processing compact text. *Proc. Conf. Applied Natural Language Processing*, Santa Monica, CA.

[Marsh 1984] Marsh, E., Hamburger, H., and Grishman, R. A production rule system for message summarization. *Proc. 1984 National Conf. on Artificial Intelligence*, Austin, TX, Am. Assn. of Artificial Intellgience.

[McDonald 1978] McDonald, D., and Hayes-Roth, F. Inferential searches of knowledge networks as an approach to extensible language understanding systems. In *Pattern-directed inference systems*, Waterman and Hayes-Roth, Eds. Academic Press, New York.

[Montgomery 1983] Montgomery, C. Distinguishing fact from opinion and events from meta-events. *Proc. Conf. Applied Natural Language Processing*, Santa Monica, CA.

[Palmer 1986] Palmer, M., Dahl, D., Schiffman, R., Hirschman, L., Linebarger, M., and Dowding, J. Recovering implicit information. To appear in *Proc. 1986 Annl. Conf. Assn. Computational Linguistics*, New York, NY.

[Sager 1975] Sager, N., and Grishman, R. The restriction language for computer grammars of natural language. *Comm. Assn. Computing Machinery 18*, 390-400.

[Sager 1978] Sager, N. Natural language information formatting: the automatic conversion of texts to a structured data base. *Advances in Computers 17*, 89-162.

[Sager 1981] Sager, N. *Natural Language Information Processing*. Addison-Wesley, Reading, MA.

# An Equipment Model and its Role in the Interpretation of Nominal Compounds

Tomasz Ksiezyk and Ralph Grishman

*Department of Computer Science*
*Courant Institute of Mathematical Sciences*
*New York University*
251 Mercer St.
New York, New York 10012
(212) 460-7446, (212) 460-7492

## Abstract

For natural language understanding systems designed for domains including relatively complex equipment, it is not sufficient to use general knowledge about this equipment. We show problems which can be solved only if the system has access to a detailed equipment model. We discuss features of such models, in particular, their ability to simulate the equipment's behavior. As an illustration, we describe a simulation model for an air compressor. Finally, we demonstrate how to find referents in this model for nominal compounds.

## 1. Introduction

The work presented here is part of the PROTEUS (PROtotype TExt Understanding System) system currently under development at the Courant Institute of Mathematical Sciences, New York University.[1] The objective of our research is to understand short natural language texts about equipment. Our texts at present are CASualty REPorts (CASREPs) which describe failures of equipment installed on Navy ships. Our initial domain is the starting air system for propulsion gas turbines. A typical CASREP consists of several sentences, for example:

> Unable to maintain lube oil pressure to SAC [*Starting Air Compressor*]. Disengaged immediately after alarm. Metal particles in oil sample and strainer.

It is widely accepted among researchers that in order to achieve natural language understanding systems robust enough for practical application, it is necessary to provide them with a lot of common-sense and domain-specific knowledge. However, so far, there is no consensus as to what is the best way of choosing, organizing and using such knowledge.

The novelty of the approach presented here is that, besides general knowledge about equipment, we also use a quite extensive simulation model for the specific piece of equipment which the texts deal with. We found that for understanding purposes it is more appropriate to make the simulation qualitative rather than quantative. Thus, for example, we are not interested in the precise value of oil pressure, but only whether it is too low or too high. The model is built from instances of prototypes which contain the bulk of general knowledge. It exists in the system permanently. In this situation the analysis of a piece of text consists of two stages: (1) locating in the model the objects mentioned in text; (2) interpreting the text using both the specific information residing in the model and the general knowledge which is accessible from the model. There is no clear-cut distinction between these two stages (see discussion of the examples in the next section).

[1] An overview of the system is given in [Grishman 1986], submitted to the AAAI-86.

We see the following merits of having a simulation model:

(a) the model provides us with a reliable background against which we can check the correctness of the understanding process on several levels: finding referents of noun phrases, assigning semantic cases to verbs, establishing causal relationships between individual sentences of the text.

(b) the requirements of simulation help us to decide what kind of knowledge about the equipment should be included in the model, how it could best be organized and which inferences it should be possible to make. It appears that the information needed for simulation largely coincides with that necessary for language understanding.

(c) the ability to simulate the behavior of a piece of equipment provides a very nice verification method for the understanding process at the level of interaction with a user - it is relatively straightforward to build a dynamic graphical interface which allows the user to have a friendly insight in the way his input has been understoood by the system.

In the remainder of the paper we will show examples of problems which can be solved only if the system has access to some kind of a simulation model of the domain equipment. Having demonstrated the need for such a model, we will discuss the design decisions which we found important for our domain and which seem to apply generally for complex equipment. How these considerations influenced the model for the SAC may be seen in the next section. Then we present a method of finding referents in the model for nominal compounds describing SAC's components. Finally, we briefly describe our future work.

## 2. Need for a Model

In most natural language understanding systems the knowledge about the domain of discourse is organized in the form of prototypes for objects and actions, and for the relations between them which are relevant for the domain. The prototypes are repositories for knowledge about the instances they subsume. This knowledge is highly structured - there are many links through which apparently distant concepts may be connected. The text is processed on a sentence by sentence basis. Usually, each sentence is split into linguistic entities with syntactic and semantic information attached. This information is used to determine the prototype for each entity. Through these prototypes there is access to general information about the concepts invoked by the sentence. This information is often necessary for the adequate interpretation (i.e. understanding) of the sentence. To account for the fact that the understanding of an utterance depends sometimes on the context in which the utterance is set, it is necessary to maintain information about the discourse context. One way of organizing this information is by creating and storing instances of prototypes for entities from the text as they come under analysis. The combined information coming from the context and from the processed sentence is used to solve problems like anaphora resolution, connectivity, etc.

Assuming this approach, let's consider the following sentence (let it be the first sentence in the analyzed text):

Starting air regulating valve failed.

Having completed the syntactic and semantic analysis of the sentence, we would recognize *starting air regulating valve* as an example of the prototype *regulating valve*. We would then fetch its description and create an instance of a regulating valve. Next, using the general knowledge about valves (of which regulating valve is a more specific case), and the semantic information about *starting air*, we would modify the just created instance with the fact that the substance the valve regulates is starting air. From the syntactic analysis we would know that *starting air regulating valve* is the subject of verb *fail*. Using the prototype of the action *fail*, we would create its instance and possibly also would further modify the instance of the valve so that the fact about its operational state is recorded. These two instances would now constitute the discourse context so far. Now, suppose the message continues with the sentence:

Unable to consistently start nr 1b turbine.

The processing would be similar to what has been described above for the first sentence. We would create an instance of a gas turbine, would fill its proper name slot with *nr 1b* and finally use the instance as an argument in another instance recording the finding about start problems.

These two sentences come from an actual CASREP. In the starting air system (our initial domain) there are three different valves regulating starting air. Two questions might be posed in connection with this short, two-sentence text: (1) which of the three valves was meant in the first sentence? (2) could the failure of the valve mentioned in the first sentence be the cause of the trouble reported in the second sentence?

The general knowledge of equipment may tell us a lot about failures, such as: if a machinery element fails, then it is inoperative, or if an element is inoperative, then the element of which it is part is probably inoperative as well, etc. Unfortunately, such knowledge is not enough: there is no way to answer these two questions (not only for an artificial understanding system, but even for us, humans) without access to rather detailed knowledge about how various elements of the given piece of equipment are interconnected and how they work as an ensemble. In our case we could hypothesize (using general knowledge about text structures) that there is a causal relationship between the facts stated in the two sentences. To test this, we would have to consider each of the three valves in turn and check how its inoperative state could affect the starting of the specific (i.e. nr 1b) turbine. If one of the three valves, when inoperative, would make the turbine starting unreliable, then we could claim that this valve is the proper referent for the *starting air regulating valve* mentioned in the first sentence. This finding would let us also answer question (2) affirmatively.

The above example, as well as others of similar nature, demonstrate that in cases where the domain is very specialized and complicated (a typical situation for real-life equipment), language understanding systems should be provided not only with general knowledge about the equipment but also have access to its model.

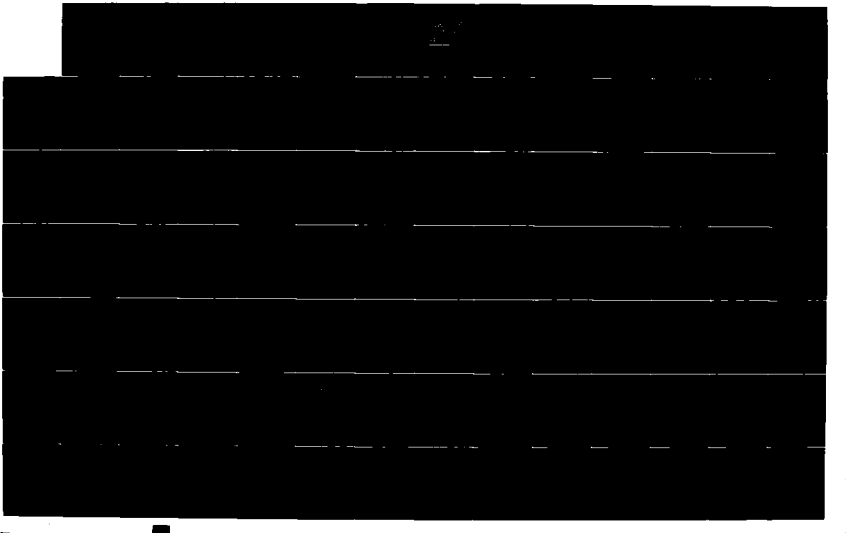
With an equipment model available, the processing of the two sentences would change: for the first sentence, instead of building a new description for the *starting air regulating valve*, we would rather try to find an object / objects in the model which could be described by this noun phrase. We would treat *nr 1b gas turbine* similarly. The semantics of *start* would be a kind of simulation procedure defined for the model. Now, let's consider problems (1) and (2) again. Viewing *nr 1b* as a proper name, we should easily find the object in the model which corresponds to the referred turbine. The analysis of *starting air regulating valve* would leave us with three pointers to the three objects in the model corresponding to the three starting air regulating valves in the equipment. In order to resolve this ambiguity we could make the following assumption, which seems very reasonable:

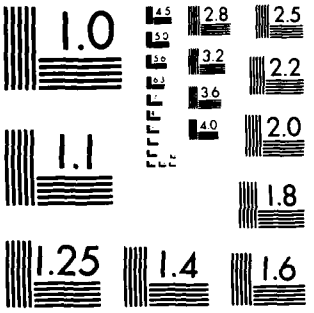
> Suppose first, that the valve's failure has indeed caused problems for the turbine. Now, if we confirm that at least one among the three valves, if inoperative, has this effect, then our assumption was correct and we found the right referent(s); if none of the three valves has any impact on the turbine, then our assumption was wrong: it answers question (2) negatively and leaves (1) still open.

Then we would proceed with the confirmation phase, considering each of the three candidates separately. We would temporarily set its operational state to INOPERATIVE, initiate the START procedure, and then check whether the functional state of the *nr 1b gas turbine* in the model has been set to RUNNING (for simplicity reasons let's assume that there is no *consistently* adverb in the second sentence). If for all three simulation experiments we wind up with the value RUNNING for the turbine, then we must conclude that there is no causal relationship between the sentences. Otherwise, we would claim to have found the right referent for the valve. Having unambiguously located the object referred to in the first sentence, we would modify its operational state accordingly.

AD-A172 436 PROCEEDINGS OF THE STRATEGIC COMPUTING NATURAL LANGUAGE WORKSHOP HELD IN..(U) UNIVERSITY OF SOUTHERN CALIFORNIA MARINA DEL REY INFORMATION S.. N K SONDHEIMER MAY 86 2/3

UNCLASSIFIED ISI/SR-86-172 MDA903-81-C-0335 F/G 9/2 NL





### 3. Characteristics of an Equipment Model

In the preceding section we tried to show that general knowledge about equipment is by itself not enough to solve some problems of understanding. The decision to provide PROTEUS with an equipment model confronted us with a new question. Where and how to draw a division line between the knowledge about equipment in general and a model of a specific piece of equipment? The ultimate objective of our research is to design PROTEUS in such a way that it may be adapted easily to new equipment. Clearly, the model has to be built anew each time we want to use PROTEUS for a new piece of equipment. The general knowledge, on the other hand, should undergo, in such cases, only a slight extension due to the new types of components in the new equipment. For example, moving from the starting air system to the main reduction gear, we would have to build a new model for the gear, but while doing this, we should be able to use many of the structures designed for modelling components which also occured in the starting air system, like bearings, lubrication system elements, etc. This can be achieved using prototypes and their instances: the model would be built of instances of prototypes. The prototypes would constitute part of the general knowledge data base. In the instances we would store only the information which is specific to the object described by the instance. For example, in case of a gearbox, the information about its function (i.e. speed change) should be stored in the prototype, and only the ratio of this change should reside in the instance of a specific gearbox. Also the information about how a specific gearbox is used in the domain equipment must be kept in the instance. Of course, the prototype-instance scheme ensures that all the general knowledge connected with the prototype is also accessible from instances of this prototype. We found the rich repertoire of programming tools constituting the flavor system in *Symbolics-Lisp* a very convenient vehicle for implementing this strategy.

On the level of prototypes we should apply the principle of generality as well. Hence, for example, we should consider the prototype of a regulating valve as a special case of a valve and have the knowledge characteristic for all possible types of valves connected with the valve prototype. This knowledge could then be propagated down in the hierarchy if necessary. Because the problems of structuring knowledge in the form of prototypes have been extensively investigated (research on frames, scripts, semantic nets, etc.), we won't elaborate on this here. We will comment on only one aspect of the hierarchy of prototypes. It seems to us that, for purposes of equipment modelling, this hierarchy should have the structure of a graph rather than of a tree: its nodes should be allowed to have more than just one immediate parent. We mentioned already that there are regulating valves in our equipment. These are valves whose function is to regulate the medium in some manner, usually changing one of its parameters, like pressure or temperature. We also have other valves whose function is different, for example relief or shut-off valves. Thus, is it conceivable to divide valves into classes according to their function. However, this is not the only dimension along which classification is possible. Valves may be also categorized according to their operating principle as electric, hydraulic or pneumatic valves. Now, the problem with a tree-like taxonomy is that we have to arrange the dimensions linearly: if we decide to consider the functional aspect first, we will have to repeat the division according to the operational aspect at each node of the functional level of the hierarchy tree. With the reversed order of dimensions the problem remains the same. It would be therefore much better to allow a node in the hierarchy to inherit properties from more than one immediately preceding node. The flavor system, with its mechanism allowing flavors to be mixed, proved to be very helpful here.

It's obvious that any real-life equipment deserving a natural language front end is big and complex. For example, the starting air system (our initial domain) consists of several hundred elements each of which may be referred to by its descriptive name and be mentioned in a casaulty report. A good measure of the system's complexity is the size of its description in the ship's manual: 28 pages of text, figures and tables. What is the best way of organizing

this vast amount of data into a managable model? Clearly, some simplification is unavoidable. How much? Let us address the former problem first. A salient feature of a piece of equipment is its task, i.e. what it should do. Generally speaking, all complex equipment may be viewed as processors of something - if this something is changed qualitatively into something else (e.g. fuel into rotary movement) we may speak of generators; if only some parameters of this something are changed (e.g. low-pressure air into high-pressure air) we may speak of transformers. Usually only part of the equipment's components are directly involved in this primary task. The rest are there to ensure that special conditions are created at certain points in the equipment. This observation provides us with an important structural hint: we can treat a piece of equipment as a functional system consisting of component systems among which one is responsible for the primary function (the equipment task) and the others fulfill auxiliary functions. If necessary, we may apply the same approach recursively to any of the lower level systems. Systems of this kind may be viewed as chains of components linked together in such a way that, at each node of the chain, the processed substance changes slightly, becoming thus more similar to its desired form at the end of the chain. Many of these components work properly only if special conditions are created. Hence the need for auxiliary systems. Another, more conventional way of structuring the model is in the form of a part/whole hierarchy. A natural question arises: where one should stop with these two types of refinements (in system/basic-part and part/whole hierarchies)? This is a more specific version of the question we posed above: how much to simplify? A possible answer is to refine the hierarchy far enough so that everything which potentially may be referred to in the reports would have a description in the model. This, however, seems impractical. Consider, for example, the following sentence:

Borescope investigation revealed a broken tooth on the hub ring gear.

Considering that there are several different gears in our starting air system and each of them has many teeth which are very much alike, it's obvious that creating a separate description for each of them wouldn't be reasonable. The same remark is true for balls in bearings or for connecting elements like screws, bolts or pins. On the other hand, information about the tooth conveyed in the above sentence cannot go unnoticed. The solution we accepted for such elements is not to include their descriptions in the model on a permanent basis but to keep the possibility open to create and to implant into the model their descriptions if such a need arises during the analysis. A rule of thumb for deciding whether a particular element deserves a permanent place in the model can be formulated in the form of the question: how much information specific to this element is necessary to solve understanding problems, like finding referents (see the section on nominal compounds) or making inferences? As an example of the latter, let's consider a specific gear. We would like to know, among other things, what is this gear's role and place in the modelled equipment so that, in case of its damage, we could determine the impact of this on the equipment. Information of this type can be deduced neither from the analyzed text nor from general knowledge about gears. It must be known in advance. Our way to achieve this is to keep the gear's description permanently in the model.

There are, however, elements like teeth which have so little relevant structure that they are always referred to as *tooth, teeth* together with the element nigher up in the part/whole hierarchy (let's call such an element a host). Thus, it is not necessary to maintain any specific information about them in the model. It is enough, if we are able to create their descriptions only when they occur in the text. All the possible information we will ever need to include into such descriptions will come from the text. The information relating such elements with other parts of the equipment will come from their hosts. For example, the impact of a tooth's damage on the equipment may be derived from the functional information connected with its host.

It is important to notice that there is nothing absolute in distinctions such as the one made above. It is conceivable to have a piece of equipment of a larger scale than the SAC, where elements like gears are not essential enough for us to be bothered with their shapes or

locations; if broken they probably would be referred to by giving the higher-level element of which they are part. In such cases we would rather treat gears like we treat teeth here.

It is desirable to be able to use the model on several levels of abstraction. For some purposes it is enough to treat, say, a speed increasing gearbox as a system for which we only know its outside behavior; in other cases, we would like to use information about its internal structure as well. It should, of course, be possible to deduce the external behavior of an object by analyzing its parts; however, it wouldn't be practical to go down to the level of basic components each time we need to know something about the behavior of the equipment on the intermediate level. Our approach of gradually refined levels of functional systems described above fulfills this desideratum. It seems inevitable that any division into levels will always be artificial and therefore, whatever structure of the model we could design, we always will find sentences which mention objects from different levels. Consider for example:

Believe the coupling from diesel to SAC lube oil pump to be sheared.

In our model for the starting air system the diesel and SAC are at the same level of abstraction. The lube oil pump is two levels below the SAC in the hierarchy. How we solve the problem of determining the referent for the above coupling is described in the section on nominal compounds (see below). Here we want only to point out that for any multi-level model, there must be mechanisms available for moving between abstraction levels flexibly.

In the preceding section we discussed two understanding problems. The solution we proposed there relied heavily on the ability to simulate certain actions and processes of the domain equipment. We have mentioned already in the introduction that it is sufficient to simulate equipment behavior qualitatively. It is clear that the solution to the simulation problem depends a lot on the structure of the model. Therefore, the simulation requirement should be one of the important design criteria for the model. Dividing the equipment into functional subsystems and modelling them as chains of components (comp. above) facilitates the simulation task considerably.

There is another aspect of natural language understanding systems whose satisfactory treatment depends a lot on an effective solution to the simulation problem. We may expect that in real-life cases, the output of such systems is either fed into some expert system or communicated to a human user. In both cases important decisions are presumably made, based on this output - otherwise, why to spend money for building them. It is therefore very important for such systems to provide users with means to check the quality of their understanding. In the case of equipment, one quick and user-friendly way of verifying the analysis is through graphics (we elaborate on this a little more in the section describing future work, below). Because equipment is very dynamic, most texts about them involve actions, events, procedures occuring in a certain time sequence. In order to show this graphically, it is necessary to simulate the essential aspects of this on the screen.

The simulation should be designed in such a way that its two independent applications in the system (i.e. text understanding and communication with users) wouldn't require two seperate simulation systems.

### 4. The Starting Air System Model

As mentioned above, the equipment we have chosen as our initial domain is the starting air system on Navy ships. Its function is to supply a ship's propulsion gas turbines with the high-pressure air necessary to start the turbines. The main part of the starting air system is its compressor (SAC - Starting Air Compressor). It is by far the most complicated element and therefore is prone to various kinds of damage and malfunction. Because of its importance, we started our efforts by building a model of the SAC. So far we have implemented parts of it on a *Symbolics Lisp* machine using *Zeta-Lisp*.

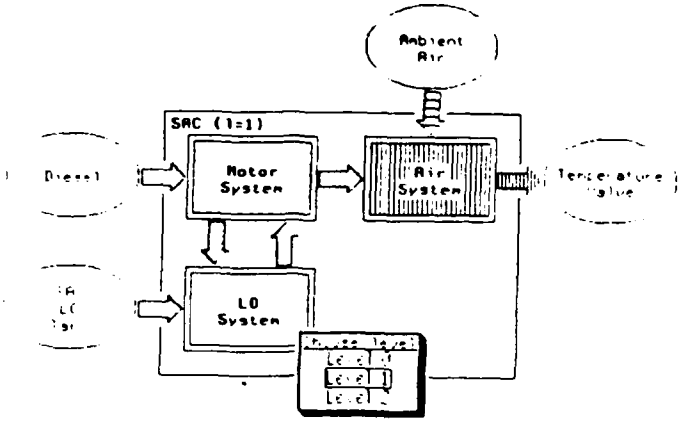



Figure 1. Division of the SAC into subsystems.

Following the guidelines for equipment models given in the preceding section, we divided the SAC into its three functional subsystems (comp. Fig. 1):

(a) Air System - this is the system partially responsible for the SAC's primary task: it takes ambient air, compresses it to the desired pressure and outputs the flow to a system of temperature and pressure regulating valves which precede the turbine starter;

(b) Motor System (auxiliary) - its function is to transmit mechanical rotation from the diesel motor to the compressor blade assembly and lubrication oil pump;

(c) Lubrication Oil (LO) System (auxiliary) - it distributes the oil throughout the SAC and supplies it under pressure to such elements as bearings and some couplings.

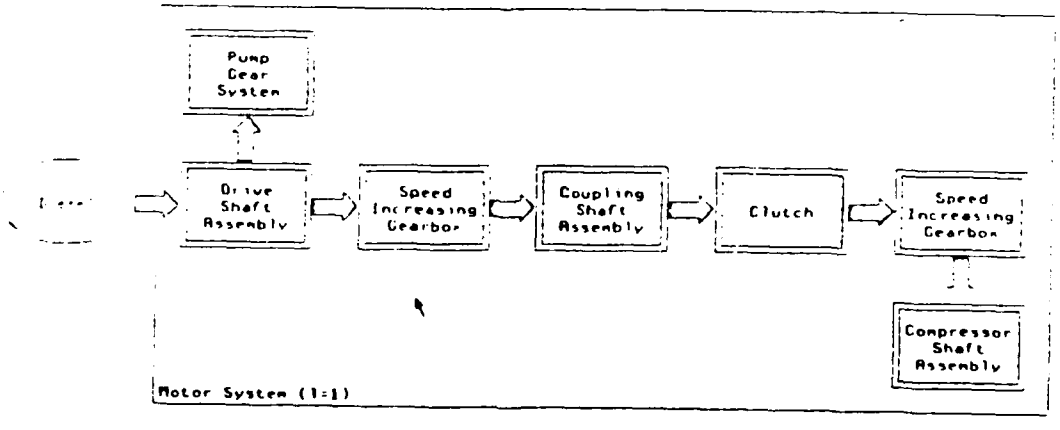



Figure 2. Division of the SAC Motor System into subsystems on level 1.

Each of these three systems may be split into further systems. For example, we view the Motor System as consisting of subsystems shown in Fig. 2. Each of these constituents is again a system consisting of more basic elements. So, for example, one of the two speed increasing gearboxes consists of a hub, a ring gear, an arrangement of three star gears, and a pinion mounted on a shaft.

Every system may be viewed on several levels of abstraction. For example, Fig. 2 shows level 1 of the Motor System. Fig. 3 and 4 show the same system on level 0 and level 2, respectively.

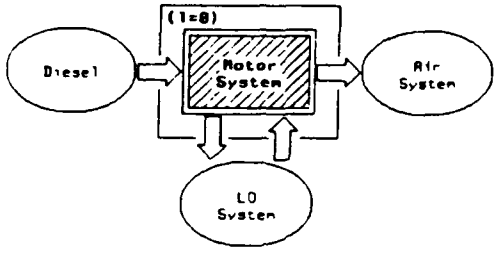



Figure 3. The SAC Motor System on level 0.

All the figures presented here are *Symbolics* screen images generated by PROTEUS from descriptions of the model's elements used for the understanding process. As a matter of fact, we have provided dynamic displays reflecting some of the simulation possibilities of the model. Consider, for example, Fig. 4. It is possible, using the mouse, to position the cursor on, say, the DIESEL ON switch and click on it causing the diesel to be turned on. The compressor starts to run: the small globes inside each of the square elements (from diesel shaft to the clutch) start to rotate in circles with different speeds depending on their place in the system (before or after the speed increasing gearbox); furthermore, all the elements which should be lubricated (those which have in- and outlets in the form of arrows) get oil influx (depicted as dots appearing inside the elements). This follows from the way the SAC operates: the Motor System transmits the rotary movement to the lube oil pump, which starts to work and to supply oil via the LO System (not shown here). Similarly, when we set the clutch to the IN position, the other elements (following the clutch in the chain) will start to rotate. Again, all this is achieved as a side effect of the simulation used for understanding purposes. We want to stress that the "movie" is not the point here. We have to know how the rotary movement propagates in the system, if we want to conduct tests like the one described in section 2, above. Such tests are the primary reason why we equipped our model with a simulation capability.

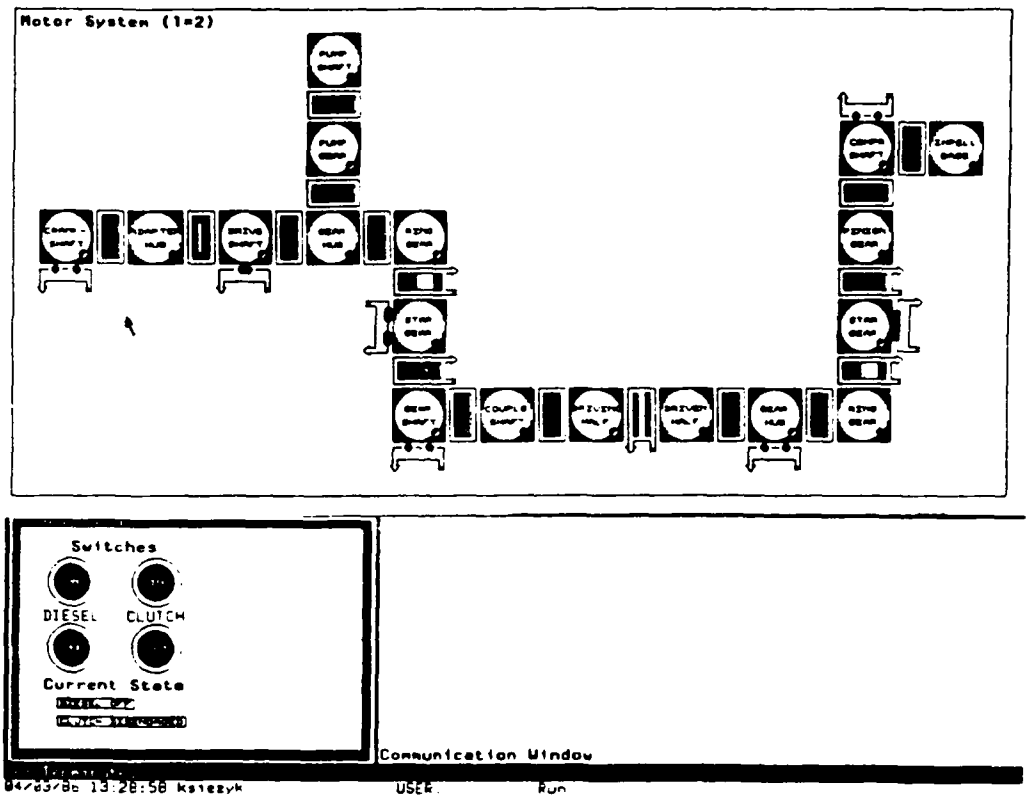



Figure 4. The SAC Motor System on level 2.

Let's turn now to the internal structure of our model for SAC. The structure of the model is based on the *Symbolics-Lisp* flavor system. The prototypes of elements of which the model is built are represented as flavors. The specific elements of the model are encoded as instances of their prototype flavors. The general knowledge about elements is stored in the prototype flavors and can be divided into two parts: (1) declarative knowledge expressed in the form of defaults and restrictions on instance variables; (2) procedural knowledge in the form of methods defined for the flavors. The flavor instances contain only declarative knowledge comprised of instance-variable -- value pairs (we will use more traditional names here: slot -- slot-filler). The prototype flavors are built as mixtures of component flavors, each of which captures a certain aspect(s) of the prototype. The component flavors, which form a graph-like hierarchy, may be viewed as sets of isolated features common to several different prototypes. The sophisticated inheritance mechanism of the flavor system, which works on the level of instance variables (slots) and on the level of methods, allows us to design this hierarchy of flavors in a consise manner. We illustrate these points below with a couple of examples.

Every element which is represented permanently in the model is an instance of a flavor which has the *%building-block* flavor as one of its components flavors (directly or indirectly through intermediate flavors). This reflects the observation that certain facts about model

elements will have to be recorded for any kind of element. For example, for every element we want to know its operational state (remember that the texts we are dealing with are about equipment failures) or the system of which it is a part. So, we define:

```
(defflavor %building-block
  (location operational-state part-of screen-location caption)
  ()
  (:settable-instance-variables :screen-location :operational-state)
  :gettable-instance-variables
  (:initable-instance-variables :function :location :part-of)
  (:default-init-plist :operational-state 'OK))
```

In the above definition the first element is the flavor's name, the second is a list of instance variables, the third is a list of component flavors (empty here), and the rest of the definition describes various aspects of instance variables, such as their defaults, how they can be initialized, accessed, etc. (we have omitted this part from flavor definitions given below).

The permanent elements in the model fall into two categories: systems and basic parts. *systems* are those ***building blocks*** which have structural information. They are chains of elements united by a working substance which they process (for example, the lube oil system). Systems are described at several levels of abstraction. The filler of the *structure* slot is a list of descriptions of the system on different levels - each element in this list specifies, among others, the start and end nodes of the chain of components on this level:

```
(defflavor %system
  (working-substance structure)
  (%building-block))
```

*basic parts* are those ***building blocks*** which are at the bottom of the part/whole hierarchy. The *components* slot is initially set to an empty list. It is provided as a destination for those equipment parts which were not included into the model *a priori* but have to be recorded if they occur in the analyzed text (see section 3 for our discussion on this issue).

```
(defflavor %basic-part
  (components)
  (%building-block))
```

Another very common flavor describes the aspects of a building block which capture its role as a component (a node) in some system. It is used as a mixin flavor for building blocks which are systems or basic parts. Its slots record how it is incorporated in the system (*from*, *to* slots) and what its function is with respect to the working substance (i.e. how the substance changes while passing this element). The filler of the *function* slot is a formula interpreted by a method defined for the prototype flavor of the element. This method accesses values of several slots of the instance to which it is applied, for example *input* or *operational-state*. The latter is important because of the potential of failure or damage of the element (see our discussion in section 2, above):

```
(defflavor %system-node
  (from to input output function)
  ())
```

Complex equipment is usually controlled from outside automatically or manually by service personnel. There are, therefore, elements whose operational modes may be changed. Examples of such elements in the SAC are the diesel and clutch. To account for such elements, we defined a flavor *%multiplexer*, which may be mixed with other component flavors to form a prototype. The filler of *switch-locations* is a list of all places from which switching is possible (in our case these are the local and remote control consoles). *switch-actions* specifies for each possible switch position a procedure which has to be run in case the element is set into this position.

```
(defflavor %multiplexer
  (switch-locations switch-actions actual-switch-position)
```

())

None of the above are prototype flavors. They are component flavors which we can use to define prototypes. Let us consider the prototype for diesel motors. It is a piece of equipment complicated enough to treat as a system. Diesels generate rotary movement which is then used to run other pieces of equipment. Thus they are parts of larger systems. Because they run only if a need arises, there must be ways to influence their operational modes. All these facts justify the following definition of a flavor which can serve as a prototype for diesel motors. The point of this definition is to mix together several component flavors corresponding to the just mentioned features.

```
(defflavor %diesel
  ()
  (%system %system-node %multiplexer))
```

Now we are ready to introduce the instance of a specific diesel motor which is part of the starting air system.

```
(setq @diesel-2 (make-instance '%diesel
  ':part-of
        '@ssdg-2
  ':working-substance
        '(ROTATION)
  ':caption
        '("Diesel")
  ':to
        '((ROTATION @sac-2 RIGHT))
  ':from
        '((OIL @container-2 LEFT)
          (AIR @container-1 UP))
  ':function
        '((ROTATION ((OIL . LOW) (AIR  LOW)) . (ROTATION . LOW)))
  ':structure
        '((0 . (DOWN . ((ROTATION OIL AIR) @diesel-2 (@diesel-2) (2 . 2))))
          (1 . (...)))))
```

This instance, its prototype, and the component flavors we showed, are in fact simplified versions of the structures we use in our model. We have included here only these parts which we considered helpful to convey the basic ideas of our prototype-instance scheme used for building the model.

The examples discussed so far demonstrate only the declarative aspect (i.e. the inheritance of instance variables) of the hierarchy we may build using flavors. We also define with each flavor a set of methods which, when combined, provide each instance with a lot of procedural knowledge. It is more difficult to show examples of this because methods are typically long procedures. Describing the *%system-node* flavor above, we mentioned one such method. Similarly, for *%multiplexer* we define a method which, using the data stored in instance's slots, simulates the switching action. Still another example of a method is a drawing procedure which we define for prototypes whose instances may be displayed on the screen. The flavor system supports object-oriented programming. This is reflected in the way methods are invoked - by sending messages (method names) to instances. This allow us to use identically named methods to invoke quite different procedures. For example, it's obvious from looking at the pictures that we use several different drawing procedures. However, we may use the same name, say, *:draw* for all of them. Suppose we have identified an element by locating its instance in the model and want to draw it. We don't have to bother about its prototype in order to know how to draw it - it's enough to send the *:draw* message to this instance. The right method will be chosen automatically. This situation is advantageous for the language understanding process as well. The first thing we do during clause analysis is to find referents in the model (i.e. instances) for linguistic entities occuring in the sentence. The semantics of the verb or predicate adjective is typically expressed in the

form of a method. The interpretation of the clause with respect to the model consists then in sending this method to one of the arguments. The part of PROTEUS which deals with the interpretation of clauses hasn't been implemented yet, so we won't go deeper into this subject here.

### 5. Finding Referents for Nominal Compounds

One notable feature of technical texts is the heavy use of nominal compounds. It seems that their average length is proportional to the complexity of the discourse domain. In the domain of the starting air system, examples like

stripped lube oil pump drive gear and hub ring gear,

are, by no means, seldom occurences.

The problem with nominal compounds is their ambiguity. The syntactic analysis is of almost no help here. Semantically they are also very difficult to deal with [Finin 1986]. The problem may be metaphorically described as a jigsaw puzzle: given several pieces (compound descriptions) put them together to build a sensible picture (nominal compound description). The task becomes somewhat easier in cases when we know that nominal compounds refer to objects existing in the system. In terms of our metaphor it translates into a hint: a set of pictures is given with the assumption that the solution is one of these pictures.

The above observation is the next argument for maintaining an equipment model. Not all nominal compounds fall into this category (a notable class here are verb nomalizations, like *borescope investigation*). However, most of them (especially the longest ones) refer to objects maintained in the model.

PROTEUS processes sentences sequentially (first syntax, then semantics, finally discourse). Both the syntactic and semantic analyzers have been implemented already. [Grishman 1986] describes the overall organization of PROTEUS in some detail. The syntactic component delimits the noun phrases, but does not assign any structure to the pre-nominal modifiers. The interpreter of nominal compounds takes as input an ordered list of words of which the nominal compound consists, and tries to achieve two goals: (1) to determine the structure of the pre-nominal modifiers; (2) to locate the instance(s) in the equipment model referred to by the nominal compound.

The parsing of the nominal compound proceeds bottom-up without backtracking. The words are analyzed from right to left. The parser maintains a **Parse Stack** where all possible partial parses are kept. The information about each partial parse (State Vector) consists of three lists: (1) the **Word List**: the unparsed part of the nominal compound; initially contains the whole compound; (2) the **Forest**: list of partial parse trees for the part of the compound which has been analyzed so far: initially empty; (3) the **List of Referents**: for each partial parse tree in the **Forest**, a list of the model instances which may be named by the words in that partial parse tree.

The condition for a successful parse is twofold: (1) the **Word List** is empty; (2) the **Forest** contains one tree (in such a case the **List of Referents** will, necessarily, also have one list of instances - they will be considered the referents of the compound nominal). The parser works as the following coroutine:

```
LOOP WHILE Parse Stack not empty
  State-Vect = Pop (Parse Stack).
  Word = next word from the Word List of State-Vect;
  Dict-Entry = dictionary entry for Word;
  FOR each reduction rule applicable to State-Vect and Dict-Entry
    Create New-State-Vect;
    IF (termination conditions fulfilled for New-State-Vect)
     THEN return (New-State-Vect)
     ELSE push (Parse Stack New-State-Vect)
```

Each word in the dictionary is assigned two properties: its model class (MOD-C) and its semantic class (SEM-C). We use five different model classes:

*Instance* - a word of this class names a set of instances in the model; this set is part of the dictionary entry (in Fig. 5 the word *pump* is an example; *(p1 p2 p3)* are instances of pumps which occur in the model),

*Slot-Filler* - a word of this class can carry information used as slot fillers in some instances; taken alone it doesn't name any model instance (in Fig. 5 the word *lube* is an example),

*Slot-Name* - a word of this class indicates how to interpret some other adajcent words in the compound; an example is *speed* - it tells how to treat *low* in the nominal compound *low speed gearbox*,

*Procedure* - each word of this class is assigned a procedure which, when called with arguments coming from other parts of the noun phrase, returns a referent(s); an example is *coupling*, as in *coupling from diesel to sac lube oil pump* - the coupling meant here is not a single coupling, but a whole sequence of them on the path between diesel and lube oil pump; this sequence has to be evaluated using the model,

*Component* - a word of this class names a set of objects in the domain equipment which are not permanently present in the model (for examples and discussion of this issue see section 3).

**DICTIONARY**

```
(lube    (MOD-C Slot-Filler)              (SEM-C Function))
(oil     (MOD-C Instance (o1 o2 o3))      (SEM-C Working-Substance))
(pump    (MOD-C Instance (p1 p2 p3))      (SEM-C Machinery))
(SAC     (MOD-C Instance (s1))            (SEM-C Machinery))
```

**SEM-C --> SLOT-NAME TABLE**

```
(Function              :function)
(Machinery             :part-of :components :location)
(Working-Substance     :working-substance)
```

**INSTANCES**

```
;;; SAC lube oil pump
(setq p3 (make-instance %pump
   ...
  ':part-of 'los2
  ':working-substance '(OIL . o3)))

;;; SAC lube oil
(setq o3 (make-instance %working-substance
   ...
  ':function 'LUBE))

;;; SAC lube oil system
(setq los2 (make-instance %system
   ...
  ':part-of 's1))

;;; SAC
(setq s1 (make-instance %system
   ...
```

Figure 5. Fragments of data used by the parser of nominal compounds.

The two most often used reduction rules are:

(1) *instance + instance --> instance*
(2) *slot-filler + instance --> instance*

In (1), the set of model instances for the result consists of those instances of the second constituent which can be linked through some path in the model to some instance of the first

constituent. In (2), the resulting instances are those instances of the second constituent which have a slot whose filler may be matched with the first constituent. The types of links traversed in the search (in the first case) or the checked slots (in the second case) are a function of the semantic class (SEM-C) of the first constituent. This function assigns to each semantic class a set of slot names (see **SEM-C --> SLOT-NAME TABLE** in Fig. 5).

Let us illustrate the way the interpreter works with an example. Fig. 6 shows the trace of parsing *SAC lube oil pump*. We enclosed State Vectors in square brackets; the lists delimited by curled brackets represent (from left to right): the **Word List**, the **Forest**, and the **List of Referents**. The words are represented by numbers; the names (*p1, p2, p3, o1*, ...) are model instances taken from the dictionary (comp. Fig. 5). We analyze the words from right to left. We start with *pump*. We remove it from the **Word List**, find its definition in the dictionary (Fig. 5), and applying a rule not shown above, create the new State Vector (Fig. 6, first vector above the compound). The next word is *oil*. Now, two reduction rules are applicable: the same one we used for *pump* - resulting in the left branch on Fig. 6 and rule (1) above. To apply rule (1), we first find in the dictionary that *oil* is of class *Instance* and names the instances *(o1 o2 o3)*. Next, we try to find out whether any of these instances may be linked to any of the *(p1 p2 p3)*. To do this we take the semantic class of *oil* from the dictionary (Fig. 5): *Working-Substance*. Then we check in the **SEM-C --> SLOT-NAME TABLE** (Fig. 5) which slot names we should consider - the only candidate in this case is *:working-substance*. Finally, we consider each of the instances *(p1 p2 p3)* and check the fillers of their *:working-substance* slots. In Fig. 5 we show only the instance *p3* (the instances *p1* and *p2* are similar). For *p3* we indeed find that it can be linked with one of the considered candidates (namely with *o3*) through the *:working-substance* link. Thus, we include *p3* into the resulting set. A similar analysis for *p1* and *p2* would result in including them into the resulting set as well. Hence, the State Vector in the right branch in Fig. 6 has *(3 4)* as a partial parse tree whose leaves, when combined into one constituent, refer to the *(p1 p2 p3)*. The analysis at the other points of the trace is similar.

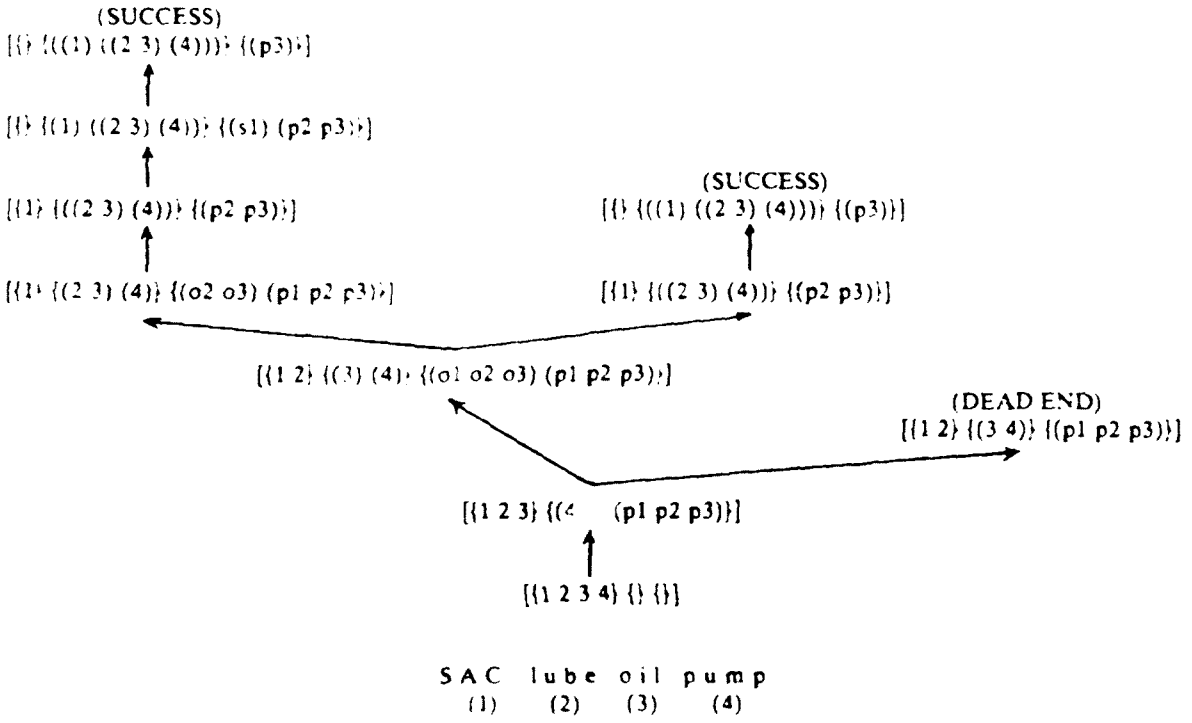



Figure 6. The parsing trace for the nominal compound *SAC lube oil pump*.

[Grishman 1986] discusses how to treat modifiers describing the state of a part, such as *cracked* or *sheared*, and also how to handle some ambiguities in conjoined noun phrases (for an example see the beginning of this section).

### 6. Future Work

The immediate next step in the development of our system is to extend the coverage of the interpreter of nominal compounds to full-fledged noun phrases (including relative clauses, prepositional phrases and conjunctions). Then we plan to work on the interpretation of clauses. It should be possible to define the semantics of most verbs from the domain as operations on the equipment model. Finally, to obtain a robust system, it will be necessary to develop components for finding temporal and causal links between sentences in the text. As is known from previous research (e.g. [Charniak 1977]), success in this area depends mainly on the quality of solutions to the knowledge representation and inference problems. As we indicated in section 2 of this paper, one of the possible approaches to inference mechanism involves the use of a simulation model.

The initial motivation for the system has been the conversion of a stream of messages to a data for subsequent querying, summarization, and trend analysis. However, the use of a detailed equipment model, similar to that employed in simulation systems (e.g. STEAMER [Hollan 1984]), suggests that it may be equally useful as an interface for such systems.

### Acknowledgement

This research was supported in part by the Defense Advanced Research Projects Agency under contract N00014-85-K-0163 from the Office of Naval Research.

### References

[Bobrow 1977] Bobrow, D. and Winograd, T. An overview of KRL - a knowledge representation language. *Cognitive Science*, 1977, 3-46

[Charniak 1977] Charniak, E. Inference and knowledge in language comprehension. In *Machine Intelligence 8*, D. Michie, Ed. American Elsevier, New York, 541-574

[Finin 1986] Finnin, T. Nominal compounds in a limited context. In *Analyzing Language in Restricted Domains*, R. Grishman and R. Kittredge, Eds. Lawrence Erlbaum Assoc., Hillsdale, NJ

[Grishman 1986] Grishman, R., Ksiezyk, T., and Nhan, N.T. Model-based analysis of messages about equipment. Submitted to the *AAAI-86*

[Hollan 1984] Hollan, J., Hutchins, E., and Weitzman, L. STEAMER: an interactive inspectable simulation-based training system. *AI Magazine*, Summer 1984, 15-27

Reference Guide to Symbolics-Lisp, *Symbolics*, Cambridge, MA, 1984

# RECOVERING IMPLICIT INFORMATION

Martha S. Palmer, Deborah A. Dahl, Rebecca J. Schiffman, Lynette Hirschman,
Marcia Linebarger, and John Dowding
Research and Development Division
SDC -- A Burroughs Company
P.O Box 517
Paoli, PA 19301 USA

## ABSTRACT

This paper describes the SDC PUNDIT, (Prolog UNDerstands Integrated Text), system for processing natural language messages.[1] PUNDIT, written in Prolog, is a highly modular system consisting of distinct syntactic, semantic and pragmatics components. Each component draws on one or more sets of data, including a lexicon, a broad-coverage grammar of English, semantic verb decompositions, rules mapping between syntactic and semantic constituents, and a domain model.

This paper discusses the communication between the syntactic, semantic and pragmatic modules that is necessary for making implicit linguistic information explicit. The key is letting syntax and semantics recognize missing linguistic entities as implicit entities, so that they can be labelled as such, and reference resolution can be directed to find specific referents for the entities. In this way the task of making implicit linguistic information explicit becomes a subset of the tasks performed by reference resolution. The success of this approach is dependent on marking missing syntactic constituents as elided and missing semantic roles as ESSENTIAL so that reference resolution can know when to look for referents.

[1] This work is supported in part by DARPA under contract N00014-85-C-0012, administered by the Office of Naval Research APPROVED FOR PUBLIC RELEASE, DISTRIBUTION UNLIMITED

## 1. Introduction

This paper describes the SDC PUNDIT[2] system for processing natural language messages. PUNDIT, written in Prolog, is a highly modular system consisting of distinct syntactic, semantic and pragmatics components. Each component draws on one or more sets of data, including a lexicon, a broad-coverage grammar of English, emantic verb decompositions, rules mapping between syntactic and semantic constituents, and a domain model. PUNDIT has been developed cooperatively with the NYU PROTEUS system (Prototype Text Understanding System), These systems are funded by DARPA as part of the work in natural language understanding for the Strategic Computing Battle Management Program. The PROTEUS/PUNDIT system will map Navy CASREP's (equipment casualty reports) into a database, which is accessed by an expert system to determine overall fleet readiness. PUNDIT has also been applied to the domain of computer maintenance reports, which is discussed here.

The paper focuses on the interaction between the syntactic, semantic and pragmatic modules that is required for the task of making implicit information explicit. We have isolated two types of implicit entities: syntactic entities which are missing syntactic constituents, and semantic entities which are unfilled semantic roles. Some missing entities are optional, and can be ignored. Syntax and semantics have to recognize the OBLIGATORY missing entities and then mark them so that reference resolution knows to find specific referents for those entities, thus making the implicit information explicit. Reference resolution uses two different methods for filling the different types of entities which are also used for general noun phrase reference problems. Implicit syntactic entities, ELIDED CONSTITUENTS, are treated like pronouns, and implicit semantic entities, ESSENTIAL ROLES are treated like definite noun phrases. The pragmatic module as currently implemented consists mainly of a reference resolution component, which is sufficient for the pragmatic issues described in this paper. We are in the process of adding a time module to handle time issues that have arisen during the analysis of the Navy CASREPS.

## 2. The Syntactic Component

The syntactic component has three parts: the grammar, a parsing mechanism to execute the grammar, and a lexicon. The grammar consists of context-free BNF definitions (currently numbering approximately 80) and associated restrictions (approximately 35). The restrictions enforce context-sensitive well-formedness constraints and, in some cases, apply optimization strategies to prevent unnecessary structure-building. Each of these three parts is described further below.

---

[2] Prolog UNDderstands Integrated Text

## 2.1. Grammar Coverage

The grammar covers declarative sentences, questions, and sentence fragments. The rules for fragments enable the grammar to parse the "telegraphic" style characteristic of message traffic, such as ***disk drive down,*** and ***has select lock.*** The present grammar parses sentence adjuncts, conjunction, relative clauses, complex complement structures, and a wide variety of nominal structures, including compound nouns, nominalized verbs and embedded clauses.

The syntax produces a detailed surface structure parse of each sentence (where "sentence" is understood to mean the string of words occurring between two periods, whether a full sentence or a fragment). This surface structure is converted into an "intermediate representation" which regularizes the syntactic parse. That is, it eliminates surface structure detail not required for the semantic tasks of enforcing selectional restrictions and developing the final representation of the information content of the sentence. An important part of regularization involves mapping fragment structures onto canonical verb-subject-object patterns, with missing elements flagged. For example, the **tvo** fragment consists of a **tensed verb + object** as in ***Replaced spindle motor.*** Regularization of this fragment, for example, maps the **tvo** syntactic structure into a verb+subject+object structure:

*verb(replace),subject(X),object(Y)*

As shown here, **verb** becomes instantiated with the surface verb, e.g., ***replace*** while the arguments of the **subject** and **object** terms are variables. The semantic information derived from the noun phrase object ***spindle motor*** becomes associated with **Y**. The absence of a surface subject constituent results in a lack of semantic information pertaining to **X**. This lack causes the semantic and pragmatic components to provide a semantic filler for the missing subject using general pragmatic principles and specific domain knowledge.

## 2.2. Parsing

The grammar uses the Restriction Grammar parsing framework [Hirschman1982, Hirschman1985], which is a logic grammar with facilities for writing and maintaining large grammars. Restriction Grammar is a descendent of Sager's string grammar [Sager1981]. It uses a top-down left-to-right parsing strategy, augmented by dynamic rule pruning for efficient parsing [Dowding1986]. In addition, it uses a meta-grammatical approach to generate definitions for a full range of co-ordinate conjunction structures [Hirschman1986].

## 2.3. Lexical Processing

The lexicon contains several thousand entries related to the particular subdomain of equipment maintenance. It is a modified version of the LSP lexicon with words classified as to part of speech and subcategorized in limited ways (e.g., verbs are subcategorized for their complement types). It also handles

multi-word idioms, dates, times and part numbers. The lexicon can be expanded by means of an interactive lexical entry program.

The lexical processor reduces morphological variants to a single root form which is stored with each entry. For example, the form ***has*** is transformed to the root form ***have*** in ***Has select lock.*** In addition, this facility is useful in handling abbreviations: the term ***awp*** is regularized to the multi-word expression ***waiting^for^part.*** This expression in turn is regularized to the root form ***wait^for^part*** which takes as a direct object a particular part or part number, as in ***is awp 2155-6147.***

Multi-word expressions, which are typical of jargon in specialized domains, are handled as single lexical items. This includes expressions such as ***disk drive*** or ***select lock***, whose meaning within a particular domain is often not readily computed from its component parts. Handling such frozen expressions as "idioms" reduces parse times and number of ambiguities.

Another feature of the lexical processing is the ease with which special forms (such as part numbers or dates) can be handled. A special "forms grammar", written as a definite clause grammar[Pereira1980] can parse part numbers, as in ***awaiting part 2155-6147***, or complex date and time expressions, as in ***disk drive up at 11/17-1236***. During parsing, the forms grammar performs a well-formedness check on these expressions and assigns them their appropriate lexical category.

## 3. Semantics

There are two separate components that perform semantic analysis, NOUN PHRASE SEMANTICS and CLAUSE SEMANTICS. They are each called after parsing the relevant syntactic structure to test semantic well-formedness while producing partial semantic representations. Clause semantics is based on Inference Driven Semantic Analysis [Palmer1985] which decomposes verbs into component meanings and fills their semantic roles with syntactic constituents. A KNOWLEDGE BASE, the formalization of each domain into logical terms, SEMANTIC PREDICATES, is essential for the effective application of Inference Driven Semantic Analysis, and for the final production of a text representation. The result of the semantic analysis is a set of PARTIALLY instantiated semantic predicates which is similar to a frame representation. To produce this representation, the semantic components share access to a knowledge base, the DOMAIN MODEL, that contains generic descriptions of the domain elements corresponding to the lexical entries. The model includes a detailed representation of the types of assemblies that these elements can occur in. The semantic components are designed to work independently of the particular model, and rely on an interface to ensure a well-defined interaction with the domain model. The domain model, noun phrase semantics and clause semantics are all explained in more detail in the following three subsections.

### 3.1. Domain Model

The domain currently being modelled by SDC is the Maintenance Report domain. The texts being analyzed are actual maintenance reports as they are called into the Burroughs Telephone Tracking System by the field engineers and typed in by the telephone operator. These reports give information about the customer who has the problem, specific symptoms of the problem, any actions take by the field engineer to try and correct the problem, and success or failure of such actions. The goal of the text analysis is to automatically generate a data base of maintenance information that can be used to correlate customers to problems, problem types to machines, and so on.

The first step in building a domain model for maintenance reports is to build a semantic net-like representation of the type of machine involved. The machine in the example text given below is the B4700. The possible parts of a B4700 and the associated properties of these parts can be represented by an **isa** hierarchy and a **haspart** hierarchy. These hierarchies are built using four basic predicates: **system,isa,hasprop, haspart**. For example the system itself is indicated by **system(b4700)**. The **isa** predicate associates TYPES with components, such as **isa(spindle^motor,motor)**. Properties are associated with components using the **hasprop** relationship, are are inherited by anything of the same type. The main components of the system: **cpu**, **power_supply**, **disk**, **printer**, **peripherals**, etc., are indicated by **haspart** relations, such as **haspart(b4700,cpu), haspart(b4700,power_supply), haspart(b4700,disk),**,etc. These parts are themselves divided into subparts which are also indicated by **haspart** relations, such as **haspart(power_supply, converter)**.

This method of representation results in a general description of a computer system. Specific machines represent INSTANCES of this general representation. When a particular report is being processed, **id** relations are created by noun phrase semantics to associate the specific computer parts being mentioned with the part descriptions from the general machine representation. So a particular B4700 would be indicated by predicates such as these: **id(b4700,system1), id(cpu,cpu1), id(power_supply,power_supply1),** etc.

### 3.2. Noun phrase semantics

Noun phrase semantics is called by the parser during the parse of a sentence, after each noun phrase has been parsed. It relies heavily on the domain model for both determining semantic well-formedness and building partial semantic representations of the noun phrases. For example, in the sentence, ***field engineer replaced disk drive at 11/2/0800***, the phrase ***disk drive at 11/2/0800*** is a syntactically acceptable noun phrase, (as in ***participants at the meeting***). However, it is not semantically acceptable in that ***at 11/20/800*** is intended to designate the time of the replacement, not a

property of the disk drive. Noun phrase semantics will inform the parser that the noun phrase is not semantically acceptable, and the parser can then look for another parse. In order for this capability to be fully utilized, however, an extensive set of domain-specific rules about semantic acceptability is required. At present we have only the minimal set used for the development of the basic mechanism. For example, in the case described here, ***at 11/2/0800*** is excluded as a modifier for ***disk drive*** by a rule that permits only the name of a location as the object of ***at*** in a prepositional phrase modifying a noun phrase.

The second function of noun phrase semantics is to create a semantic representation of the noun phrase, which will later be operated on by reference resolution. For example, the semantics for ***the bad disk drive*** would be represented by the following Prolog clauses.

```
[id(disk^drive,X),
 bad(X),
 def(X),        that is, X was referred to with a full, definite noun phrase,
 full_npe(X)]   rather than a pronoun or indefinite noun phrase.
```

### 3.3. Clause semantics

In order to produce the correct predicates and the correct instantiations, the verb is first decomposed into a semantic predicate representation appropriate for the domain. The arguments to the predicates constitute the SEMANTIC ROLES of the verb, which are similar to cases. There are domain specific criteria for selecting a range of semantic roles. In this domain the semantic roles include: **agent**,**instrument**,**theme**, **object1**,**object2**, **symptom** and **mod**. Semantic roles can be filled either by a syntactic constituent supplied by a mapping rule or by reference resolution, requiring close cooperation between semantics and reference resolution. Certain semantic roles are categorized as ESSENTIAL, so that pragmatics knows that they need to be filled if there is no syntactic constituent available. The default categorization is NON-ESSENTIAL, which does not require that the role be filled. Other semantic roles are categorized as NON-SPECIFIC or SPECIFIC depending on whether or not the verb requires a specific referent for that semantic role (see Section 4). The example given in Section 5 illustrates the use of both a non-specific semantic role and an essential semantic role. This section explains the decompositions of the verbs relevant to the example, and identifies the important semantic roles.

The decomposition of ***have*** is very domain specific.

```
have(time(Per)) <-
        symptom(object1(O1),symptom(S),time(Per))
```

It indicates that a particular **symptom** is associated with a particular **object**, as in "the disk drive has select lock." The **object1** semantic role

would be filled by the disk drive, the subject of the clause, and the **symptom** semantic role would be filled by ***select lock***, the object of the clause. The **time(Per)** is always passed around, and is occasionally filled by a time adjunct, as in ***the disk drive had select lock at 0800.***

In addition to the mapping rules that are used to associate syntactic constituents with semantic roles, there are selection restrictions associated with each semantic role. The selection restrictions for ***have*** test whether or not the filler of the **object1** role is allowed to have the type of symptom that fills the **symptom** role. For example, only disk drives have select locks.

### Mapping Rules

The decomposition of ***replace*** is also a very domain specific decomposition that indicates that an **agent** can use an **instrument** to exchange two **objects**.

```
replace(time(Per)) < -
   cause(agent(A),
      use(instrument(I),
         exchange(object1(O1),object2(O2),time(Per))))
```

The following mapping rule specifies that the **agent** can be indicated by the subject of the clause.

```
agent(A)  < - subject(A)  /  X
```

The mapping rules make use of intuitions about syntactic cues for indicating semantic roles first embodied in the notion of case [Fillmore1968, Palmer1981]. Some of these cues are quite general, while other cues are very verb-specific. The mapping rules can take advantage of generalities like "SUBJECT to AGENT" syntactic cues while still preserving context sensitivities. This is accomplished by making the application of the mapping rules "situation-specific" through the use of PREDICATE ENVIRONMENTS. The previous rule is quite general and can be applied to every **agent** semantic role in this domain. This is indicated by the X on the right hand side of the "/" which refers to the predicate environment of the **agent**, i.e., anything. Other rules, such as "WITH-PP to OBJECT2," are much less general, and can only apply under a set of specific circumstances. The predicate environments for an **object1** and **object2** are specified more explicitly. An **object1** can be the object of the sentence if it is contained in the semantic decomposition of a verb that includes an **agent** and belongs to the ***repair*** class of verbs. An **object2** can be indicated by a ***with*** prepositional phrase if it is contained in the semantic decomposition of a replace verb:

```
object1(Part1) < -  obj(Part1)/  cause(agent(A),Repair_event)

object2(Part2) < -
     pp(with,Part2) /
```

cause(agent(A),use(I,exchange(object1(O1),object2(Part2),T)))

**Selection Restrictions**

The selection restriction on an **agent** is that it must be a field engineer, and an **instrument** must be a tool. The selection restrictions on the two objects are more complicated, since they must be machine parts, have the same type, and yet also be distinct objects. In addition, the first object must already be associated with something else in a **haspart** relationship, in other words it must already be included in an existing assembly. The opposite must be true of the second object: it must not already be included in an assembly, so it must not be associated with anything else in a **haspart** relationship.

There is also a pragmatic restriction associated with both objects that has not been associated with any of the semantic roles mentioned previously. Both **object1** and **object2** are essential semantic roles. Whether or not they are mentioned explicitly in the sentence, they must be filled, preferably by an an entity that has already been mentioned, but if not that, then entities will be created to fill them [Palmer1983]. This is accomplished by making an explicit call to reference resolution to find referents for essential semantic roles, in the same way that reference resolution is called to find the referent of a noun phrase. This is not done for non-essential roles, such as the **agent** and the **instrument** in the same verb decomposition. If they are not mentioned they are simply left unfilled. The **instrument** is rarely mentioned, and the **agent** could easily be left out, as in ***The disk drive was replaced at 0800.***[3] In other domains, the **agent** might be classified as obligatory, and then it wold have to be filled in.

There is another semantic role that has an important pragmatic restriction on it in this example, the **object2** semantic role in ***wait^for^part (awp).***

idiomVerb(wait^for^part,time(Per)) < -
ordered(object1(O1),object2(O2),time(Per))

The semantics of ***wait^for^part*** indicates that a particular type of part has been ordered, and is expected to arrive. But it is not a specific entity that might have already been mentioned. It is a more abstract object, which is indicated by restricting it to being non-specific. This tells reference resolution that although a syntactic constituent, preferably the object, can and should fill this semantic role, and must be of type **machine-part**, that reference resolution should not try to find a specific referent for it (see Section 4).

The last verb representation that is needed for the example is the representation of ***be.***

be(time(Per)) < -

[3]Note that an elided subject is handled quite differently, as in ***replaced disk drive***. Then the missing subject is

attribute(theme(T),mod(M),time(Per))

In this domain ***be*** is used to associate predicate adjectives or nominals with an object, as in ***disk drive is up*** or ***spindle motor is bad.*** The representation merely indicates that a **mod**ifier is associated with an **theme** in an attribute relationship. Noun phrase semantics will eventually produce the same representation for ***the bad spindle motor,*** although it does not yet.

## 4. Reference Resolution

Reference resolution is the component which keeps track of references to entities in the discourse. It creates labels for entities when they are first directly referred to, or when their existence is implied by the text, and recognizes subsequent references to them. Reference resolution is called from clause semantics when clause semantics is ready to instantiate a semantic role. It is also called from pragmatic restrictions when they specify a referent whose existence is entailed by the meaning of a verb.

The system currently covers many cases of singular and plural noun phrases, pronouns, ***one***- anaphora, nominalizations, and non-specific noun phrases; reference resolution also handles adjectives, prepositional phrases and possessive pronouns modifying noun phrases. Noun phrases with and without determiners are accepted. Dates, part numbers, and proper names are handled as special cases. Not yet handled are compound nouns, quantified noun phrases, conjoined noun phrases, relative clauses, and possessive nouns.

The general reference resolution mechanism is described in detail in [Dahl1986]. In this paper the focus will be on the interaction between reference resolution and clause semantics. The next two sections will discuss how reference resolution is affected by the different types of semantic roles.

### 4.1. Obligatory Constituents and Essential Semantic Roles

A slot for a syntactically obligatory constituent such as the subject appears in the intermediate representation whether or not a subject is overtly present in the sentence. It is possible to have such a slot because the absence of a subject is a syntactic fact, and is recognized by the parser. Clause semantics calls reference resolution for such an implicit constituent in the same way that it calls reference resolution for explicit constituents. Reference resolution treats elided noun phrases exactly as it treats pronouns, that is by instantiating them to the first member of a list of potential pronominal referents, the **FocusList**.

assumed to fill the **agent** ro , and an appropriate referent is found by reference resolution

The general treatment of pronouns resembles that of[Sidner1979], although there are some important differences, which are discussed in detail in [Dahl1986]. The hypothesis that elided noun phrases can be treated in much the same way as pronouns is consistent with previous claims by [Gundel1980], and [Kameyama1985], that in languages which regularly allow zero-np's, the zero corresponds to the focus. If these claims are correct, it is not surprising that in a sublanguage that allows zero-np's, the zero should also correspond to the focus.

After control returns to clause semantics from reference resolution, semantics checks the selectional restrictions for that referent in that semantic role of that verb. If the selectional restrictions fail, backtracking into reference resolution occurs, and the next candidate on the FocusList is instantiated as the referent. This procedure continues until a referent satisfying the selectional restrictions is found. For example, in ***Disk drive is down. Has select lock***, the system instantiates the disk drive, which at this point is the first member of the **FocusList**, as the **object1** of ***have***:

```
[event39]
have(time(time1))
  symptom(object1([drive10]),
        symptom([lock17]),
        time(time1))
```

Essential roles might also not be expressed in the sentence, but their absence cannot be recognized by the parser, since they can be expressed by syntactically optional constituents. For example, in ***the field engineer replaced the motor.***, the new replacement motor is not mentioned, although in this domain it is classified as semantically essential. With verbs like ***replace***, the type of the replacement, ***motor***, in this case, is known because it has to be the same type as the replaced object. Reference resolution for these roles is called by pragmatic rules which apply when there is no overt syntactic constituent to fill a semantic role. Reference resolution treats these referents as if they were full noun phrases without determiners. That is, it searches through the context for a previously mentioned entity of the appropriate type, and if it doesn't find one, it creates a new discourse entity. The motivation for treating these as full noun phrases is simply that there is no reason to expect them to be in focus, as there is for elided noun phrases.

### 4.2. Noun Phrases in Non-Specific Contexts

Indefinite noun phrases in contexts like ***the field engineer ordered a disk drive*** are generally associated with two readings. In the specific reading the disk drive ordered is a particular disk drive, say, the one sitting on a certain shelf in the warehouse. In the non-specific reading, which is more likely in this

sentence, no particular disk drive is meant; any disk drive of the appropriate type will do. Handling noun phrases in these contexts requires careful integration of the interaction between semantics and reference resolution, because semantics knows about the verbs that create non-specific contexts, and reference resolution knows what to do with noun phrases in these contexts. For these verbs a constraint is associated with the semantics rule for the semantic role **object2** which states that the filler for the **object2** must be non-specific.[4] This constraint is passed to reference resolution, which represents a non-specific noun phrase as having a variable in the place of the pointer, for example, **id(motor,X)**.

Non-specific semantic roles can be illustrated using the **object2** semantic role in ***wait^for^part (awp)***. The part that is being ***awaited*** is non-specific, i.e., can be any part of the appropriate type. This tells reference resolution not to find a specific referent, so the referent argument of the **id** relationship is left as an uninstantiated variable. The analysis of ***fe is awp spindle motor*** would fill the **object1** semantic role with **fe1** from **id(fe,fe1),** and the **object2** semantic role with **X** from **id(spindle^motor,X),** as in **ordered(object1(fe1),object2(X)).** If the spindle motor is referred to later on in a relationship where it must become specific, then reference resolution can instantiate the variable with an appropriate referent such as **spindle^motor3** (See Section 5.6).

## 5. Sample Text: A sentence-by-sentence analysis

The sample text given below is a slightly emended version of a maintenance report. The parenthetical phrases have been inserted. The following summary of an interactive session with PUNDIT illustrates the mechanisms by which the syntactic, semantic and pragmatic components interact to produce a representation of the text.

1. disk drive (was) down (at) 11/16-2305.
2. (has) select lock.
3. spindle motor is bad.
4. (is) awp spindle motor.
5. (disk drive was) up (at) 11/17-1236.
6. replaced spindle motor.

### 5.1. Sentence 1: Disk drive was down at 11/16-2305.

As explained in Section 3.2 above, the noun phrase ***disk drive*** leads to the creation of an **id** *of the form:* **id(disk^drive,[drive1])** Because dates and names generally refer to unique entities rather than to exemplars of a general type, their **ids** do not contain a type argument: **date([11/16-**

[4] The specific reading is not available at present, since it is considered to be unlikely to occur in this domain

**1100]),name([paoli]).**

The interpretation of the first sentence of the report depends on the semantic rules for the predicate ***be***. The rules for this predicate specify three semantic roles, an **theme** to whom or which is attributed a **modifier**, and the **time.** After a mapping rule in the semantic component of the system instantiates the **theme** semantic role with the sentence subject, ***disk drive,*** the reference resolution component attempts to identify this referent. Because ***disk drive*** is in the first sentence of the discourse, no prior references to this entity can be found. Further, this entity is not presupposed by any prior linguistic expressions. However, in the maintenance domain, when a disk drive is referred to it can be assumed to be part of a B3700 computer system. As the system tries to resolve the reference of the noun phrase ***disk drive*** by looking for previously mentioned disk drives, it finds that the mention of a disk drive presupposes the existence of a system. Since no system has been referred to, a pointer to a system is created at the same time that a pointer to the disk drive is created.

Both entities are now available for future reference. In like fashion, the propositional content of a complete sentence is also made available for future reference. The entities corresponding to propositions are given event labels; thus **event1** is the pointer to the first proposition. The newly created disk drive, system and event entities now appear in the discourse information in the form of a list ong with the date.

**id(event,[event1])**
**id(disk^drive,[drive1])**
**date([11/16-2305])**
**id(system,[system1])**

Note however, that only those entities which have been explicitly mentioned appear in the **FocusList**:

**FocusList: [[event1],[drive1],[11/16-2305]]**

The propositional entity appears at the head of the focus list followed by the entities mentioned in full noun phrases.[5]

In addition to the representation of the new event, the pragmatic information about the developing discourse now includes information about part-whole relationships, namely that **drive1** is a part which is contained in **system1**.

**Part-Whole Relationships:**
**haspart([system1],[drive1])**

The complete representation of **event1**, appearing in the event list in the form shown below, indicates that at the time given in the prepositional phrase ***at 11/16-2305*** there is a state of affairs denoted as **event1** in which a particular

[5] The order in which full noun phrase mentions are added to the **FocusList** depends on their syntactic function and linear order. For full noun phrases, direct object mentions precede subject mentions followed by all other mentions given in the order in which they occur in the sentence. See [Dahl1986], for details

disk drive, i.e., **drive1**, can be described as ***down***.

**[event1]**
**be(time([11/16-2305]))**
**attribute(theme([drive1]),**
**mod(down),time([11/16-2305]))**

### 5.2. Sentence 2: Has select lock.

The second sentence of the input text is a sentence fragment and is recognized as such by the parser. Currently, the only type of fragment which can be parsed can have a missing subject but must have a complete verb phrase. Before semantic analysis, the output of the parse contains, among other things, the following constituent list: **[subj([X]),obj([Y])]**. That is, the syntactic component represents the arguments of the verb as variables. The fact that there was no overt subject can be recognized by the absence of semantic information associated with **X**, as discussed in Section 3.2. The semantics for the maintenance domain sublanguage specifies that the thematic role instantiated by the direct object of the verb ***to have*** must be *a symptom of the entity* referred to by the subject. Reference resolution treats an empty subject much like a pronominal reference, that is, it proposes the first element in the **FocusList** as a possible referent. The first proposed referent, **event1** is rejected by the semantic selectional constraints associated with the verb ***have***, which, for this domain, require the role mapped onto the subject to be classified as a machine part and the role mapped onto the direct object to be classified as a symptom. Since the next item in the **FocusList**, **drive1**, is a machine part, it passes the selectional constraint and becomes matched with the empty subject of ***has select lock***. Since no select lock has been mentioned previously, the system creates one. For the sentence as a whole then, two entities are newly created: the select lock (**[lock1]**) and the new propositional event (**[event2]**): **id(event,[event2]), id(select^lock,[lock1]).** The following representation is added to the event list, and the **FocusList** and **Ids** are updated appropriately.[6]

**[event2]**
**have(time(time1))**
**symptom(object1([drive1]),**
**symptom([lock1]),time(time1))**

### 5.3. Sentence 3: Motor is bad.

In the third sentence of the sample text, a new entity is mentioned, ***motor***. Like ***disk drive*** from sentence 1, ***motor*** is a dependent entity. However, the entity it presupposes is not a computer system, but rather, a disk drive. The

[6] This version only deals with explicit mentions of time, so for this sentence the time argument is filled in with a gensym that stands for an unknown time period. The current version of PUNDIT uses verb tense and verb semantics

newly mentioned motor becomes associated with the previously mentioned disk drive.

After processing this sentence, the new entity **motor3** is added to the **FocusList** along with the new proposition **event3**. Now the discourse information about part-whole relationships contains information about both dependent entities, namely that **motor1** is a part of **drive1** and that **drive1** is a part of **system1**.

```
haspart([drive1],[motor1])
haspart([system1],[drive1])
```

### 5.4. Sentence 4: is awp spindle motor.

***Awp*** is an abbreviation for an idiom specific to this domain, ***awaiting part.*** It has two semantic roles, one of which maps to the sentence subject. The second maps to the direct object, which in this case is the non-specific spindle motor as explained in Section 4.2. The selectional restriction that the first semantic role of ***awp*** be an engineer causes the reference resolution component to create a new engineer entity because no engineer has been mentioned previously. After processing this sentence, the list of available entities has been incremented by three:

```
id(event,[event4])
id(part,[_2317])
id(field^engineer,[engineer1])
```

The new event is represented as follows:

```
[event4]
idiomVerb(wait^for^part,time(time2))
wait(object1([engineer1]),
    object2([_2317]),time(time2))
```

**5.5. Sentence 5: disk drive was up at 11/17-0800** In the emended version of sentence 5 the ***disk drive*** is presumed to be the same drive referred to previously, that is, **drive1.** The semantic analysis of sentence 5 is very similar to that of sentence 1. As shown in the following event representation, the predicate expressed by the modifier ***up*** is attributed to the theme **drive1** at the specified time.

```
[event5]
 be(time([11/17-1236]))
 attribute(theme([drive1]),
 mod(up),time([11/17-1236]))
```

to derive implicit time arguments

### 5.6. Sentence 6: Replaced motor.

The sixth sentence is another fragment consisting of a verb phrase with no subject. As before, reference resolution tries to find a referent in the current **FocusList** which is a semantically acceptable subject given the thematic structure of the verb and the domain-specific selectional restrictions associated with them. The thematic structure of the verb ***replace*** includes an **agent** role to be mapped onto the sentence subject. The only **agent** in the maintenance domain is a field engineer. Reference resolution finds the previously mentioned engineer created for ***awp spindle motor,*** **[engineer1].** It does not find an **instrument,** and since this is not an essential role, this is not a problem. It simply fills it in with another gensym that stands for an unknown filler, **unknown1**.

When looking for the referent of a spindle motor to fill the **object1** role, it first finds the non-specific spindle motor also mentioned in the ***awp spindle motor*** sentence, and a specific referent is found for it. However, this fails the selection restrictions, since although it is a machine part, it is not already associated with an assembly, so backtracking occurs and the referent instantiation is undone. The next spindle motor on the **FocusList** is the one from ***spindle motor is bad,*** (**[motor1]**). This does pass the selection restrictions since it participates in a **haspart** relationship.

The last semantic role to be filled is the **object2** role. Now there is a restriction saying this role must be filled by a machine part of the same type as **object1,** which is not already included in an assembly, viz., the non-specific spindle motor. Reference resolution finds a new referent for it, which automatically instantiates the variable in the **id** term as well. The representation can be decomposed further into the two semantic predicates **missing** and **included**, which indicate the current status of the parts with respect to any existing assemblies. The **haspart** relationships are updated, with the old **haspart** relationship for **[motor1]** being removed, and a new **haspart** relationship for **[motor3]** being added. The final representation of the text will be passed through a filter so that it can be suitably modified for inclusion in a database.

```
[event6]
 replace(time(time3))
 cause(agent([engineer1]),
    use(instrument(unknown1),
        exchange(object1([motor1]),
                 object2([motor2]),
                 time(time3))))
 included(object2([motor2]),time(time3))
 missing(object1([motor1]),time(time3))
```

**Part-Whole Relationships:**

```
haspart([drive1],[motor3])
haspart([system1],[drive1])
```

## 6. Conclusion

This paper has discussed the communication between syntactic, semantic and pragmatic modules that is necessary for making implicit linguistic information explicit. The key is letting syntax and semantics recognize missing linguistic entities as implicit entities, so that they can be marked as such, and reference resolution can be directed to find specific referents for the entities. Implicit entities may be either empty syntactic constituents in sentence fragments or unfilled semantic roles associated with domain-specific verb decompositions. In this way the task of making implicit information explicit becomes a subset of the tasks performed by reference resolution. The success of this approach is dependent on the use of syntactic and semantic categorizations such as ELLIDED and ESSENTIAL which are meaningful to reference resolution, and which can guide reference resolution's decision making process.

ACKNOWLEDGEMENTS

We would like to thank Bonnie Webber for her very helpful suggestions on exemplifying semantics/pragmatics cooperation.

## REFERENCES

[Dahl1986]
Deborah A. Dahl, Focusing and Reference Resolution in PUNDIT, submitted for publication, 1986.

[Dowding1986]
John Dowding and Lynette Hirschman, Dynamic Translation for Rule Pruning in Restriction Grammar, submitted to AAAI-86, Philadelphia, 1986.

[Fillmore1968]
C. J. Fillmore, The Case for Case. In ***Universals in Linguistic Theory***, E. Bach and R. T. Harms (ed.), Holt, Rinehart, and Winston, New York, 1968.

[Gundel1980]
Jeanette K. Gundel, Zero-NP Anaphora in Russian. ***Chicago Linguistic Society Parasession on Pronouns and Anaphora***, 1980.

[Hirschman1982]
L. Hirschman and K. Puder, Restriction Grammar in Prolog. In ***Proc. of the First International Logic Programming Conference***, M. Van Caneghem (ed.), Association pour la Diffusion et le Developpement de Prolog, Marseilles, 1982, pp. 85-90.

[Hirschman1985]
L. Hirschman and K. Puder, Restriction Grammar: A Prolog Implementation. In ***Logic Programming and its Applications***, D.H.D. Warren and M. VanCaneghem (ed.), 1985.

[Hirschman1986]
L. Hirschman, Conjunction in Meta-Restriction Grammar. ***J. of Logic Programming***, 1986.

[Kameyama1985]
Megumi Kameyama, Zero Anaphora: The Case of Japanese, Ph.D. thesis, Stanford University, 1985.

[Palmer1983]
M. Palmer, Inference Driven Semantic Analysis. in ***Proceedings of the National Conference on Artificial Intelligence (AAAI-83)***, Washington, D.C., 1983.

[Palmer1981]
Martha S. Palmer, A Case for Rule Driven Semantic Processing. ***Proc. of the 19th ACL Conference***, June, 1981.

[Palmer1985]
Martha S. Palmer, Driving Semantics for a Limited Domain, Ph.D. thesis, University of Edinburgh, 1985.

[Pereira1980]
F. C. N. Pereira and D. H. D. Warren, Definite Clause Grammars for Language Analysis -- A Survey of the Formalism and a Comparison with Augmented Transition Networks. ***Artificial Intelligence* 13**, 1980, pp. 231-278.

[Sager1981]
N. Sager, ***Natural Language Information Processing: A Computer Grammar of English and Its Applications.*** Addison-Wesley, Reading, Mass., 1981.

[Sidner1979]
Candace Lee Sidner, Towards a Computational Theory of Definite Anaphora Comprehension in English Discourse, MIT-AI TR-537, Cambridge, MA, 1979.

# Focusing and Reference Resolution in PUNDIT

Deborah A. Dahl

Research and Development Division
SDC -- A Burroughs Company
PO Box 517
Paoli, PA 19301

Science Track

Natural Language

## ABSTRACT

This paper describes the use of focusing in the PUNDIT text processing system.[1] Focusing, as discussed by [Sidner1979] (as well as the closely related concept of centering, as discussed by [Grosz1983] ), provides a powerful tool for pronoun resolution. However, its range of application is actually much more general, in that it can be used for several problems in reference resolution. Specifically, in the PUNDIT system, focusing is used for ***one***-anaphora, elided noun phrases, and certain types of definite and indefinite noun phrases, in addition to its use for pronouns. Another important feature in the PUNDIT reference resolution system is that the focusing algorithm is based on syntactic constituents, rather than on thematic roles, as in Sidner's system. This feature is based on considerations arising from the extension of focusing to cover ***one***-anaphora. These considerations make syntactic focusing a more accurate predictor of the interpretation of ***one***-anaphoric noun phrases without decreasing the accuracy for definite pronouns.

[1] This work is supported in part by DARPA under contract N00014-85-C-0012, administered by the Office of Naval Research

# 1. Background

## 1.1. Focusing

Linguistically reduced forms, such as pronouns, are typically used in texts to refer to the entity or entities with which the text is most centrally concerned.[2] Thus, keeping track of this entity, (the ***topic***, of [Gundel1974], the ***focus*** of [Sidner1979], and the ***backward-looking center*** of [Grosz1983, Kameyama1985] ) is clearly of value in the interpretation of pronouns. However, while 'pronoun resolution' is generally presented as a problem in computational linguistics to which focusing can provide an answer (See for example, the discussion in [Hirst1981]), it is useful to consider focusing as a problem in its own right. By looking at focusing from this perspective, it can be seen that its applications are more general than simply finding referents for pronouns. Focusing can in fact play a role in the interpretation of several different types of noun phrases. In support of this position, I will show how focus is used in the PUNDIT (Prolog UNDerstander of Integrated Text) text processing system to interpret a variety of forms of anaphoric reference; in particular, pronouns, elided noun phrases, ***one***-anaphora, and context-dependent full noun phrase references.

A second position advocated in this paper is that surface syntactic form can provide an accurate guide to determining what entities are in focus. Unlike previous focusing algorithms, such as that of [Sidner1979], which used thematic roles (for example, ***theme, agent, instrument*** as described in [Gruber1976] ), the algorithm used in this system relies on surface syntactic structure to determine which entities are expected to be in focus. The extension of the focusing mechanism to handle ***one***-anaphora has provided the major motivation for the choice of syntactic focusing.

The focusing mechanism in this system consists of two parts--a **FocusList**, which is a list of entities in the order in which they are to be considered as foci, and a focusing algorithm, which orders the **FocusList**. The implementation is discussed in detail in Section 5.

## 1.2. Overview of the PUNDIT System

I will begin with a brief overview of the PUNDIT system, currently under development at SDC. PUNDIT is written in Quintus Prolog 1.5. It is designed to integrate syntax, semantics, and discourse knowledge in text processing for limited domains. The system is implemented as a set of distinct interacting components which communicate with each other in clearly specified and restricted ways.

The syntactic component, Restriction Grammar,[Hirschman1982, Hirschman1985], performs a top-down parse by interpreting a set of context-free BNF definitions and enforcing context-sensitive restrictions associated with the BNF definitions. The grammar is generally modelled after that developed by the NYU Linguistic String Project [Sager1981]. Restrictions which enforce context-sensitive constraints on the parse are associated with the bnf rules

[2] I am grateful for the helpful comments of Lynette Hirschman, Marcia Linebarger, Martha Palmer, and Rebecca Schiffman on this paper John Dowding and Bonnie Webber also provided useful comments and suggestions on an earlier version

Some semantic filtering of the parse is done at the noun phrase level. That is, after a noun phrase is parsed, it is passed to the noun phrase semantics component, which determines if there is an acceptable semantics associated with that parse. If the noun phrase is acceptable, the semantics component constructs a semantic representation. If the noun phrase is not semantically acceptable, another parse is sought.

At the conclusion of parsing, the sentence-level semantic interpreter is called. This interpreter is based on Palmer's Inference Driven Semantic Analysis system, [Palmer1985], which analyzes verbs into their component meanings and fills their thematic roles. In the process of filling a thematic role the semantic analyzer calls reference resolution for a specific syntactic constituent in order to find a referent to fill the role. Reference resolution instantiates the referent, and adds to the discourse representation any information inferred during reference resolution.

Domain-specific information is available for both the noun phrase and clause level semantic components through the knowledge base. The domain currently being modelled by SDC is that of computer maintenance reports. Currently the knowledge base is implemented as a semantic net containing a part-whole hierarchy and an ***isa*** hierarchy of the components and entities in the application domain.

Following the semantic analysis, a discourse component is called which updates the discourse representation to include the information from the current sentence and which runs the focusing algorithm.

## 2. Uses of Focusing

Focusing is used in four places in PUNDIT -- for definite pronouns, for elided noun phrases, for ***one***-anaphora, and for implicit associates.

As stated above, reference resolution is called by the semantic interpreter when it is in the process of filling a thematic role. Reference resolution proposes a referent for the constituent associated with that role. For example, if the verb is ***replace*** and the semantic interpreter is filling the role of **agent,** reference resolution would be called for the surface syntactic subject. After a proposed referent is chosen for the subject, any specific selectional restrictions on the agent of ***replace*** (such as the constraint that the agent has to be a human being) are checked. If the proposed referent fails selection, backtracking into reference resolution occurs and another referent is selected. Cooperation between reference resolution and the semantic interpreter is discussed in detail in [Palmer1986]. The semantic interpreter itself is discussed in [Palmer1985].

### 2.1. Pronouns and Elided Noun Phrases

Pronoun resolution is done by instantiating the referent of the pronoun to the first member of the **FocusList** unless the instantiation would violate syntactic constraints on coreferentiality.[3] (As noted above, if the proposed referent fails selection,

[3] At the moment, the syntactic constraints on coreferentiality used by the system are very simple. If the direct object is reflexive it must be instantiated to the same referent as the subject. Otherwise it must be a different referent. Obviously, as the system is extended to cover sentences with more complex structures, a more sophisticated treatment of syntactic constraints on

backtracking occurs, and another referent is chosen.)

The reference resolution situation in the maintenance texts however, is complicated by the fact that there are very few overt pronouns. Rather, in contexts where a noun phrase would be expected, there is often elision, or a zero-np as in ***Won't power up*** and ***Has not failed since Hill's arrival.*** Zeroes are handled exactly as if they were pronouns. The hypothesis that elided noun phrases can be treated in the same way as pronouns is consistent with previous claims in [Gundel1980] and [Kameyama1985] that in languages such as Russian and Japanese, which regularly allow zero-np's, the zero corresponds to the focus. If these claims are correct, it is not surprising that in a sublanguage like that found in the maintenance texts, which also allows zero-np's, the zero should correspond to the focus.

Another kind of pronoun (or zero) also occurs in the maintenance texts, which is not associated with the local focus, but is concerned with global aspects of the text. For example, the field engineer is a default agent in the maintenance domain, as in ***Thinks problem is in head select area.*** This is handled by defining ***default elided referents*** for the domain. The referent is instantiated to one of these if no suitable candidate can be found in the **FocusList**.

### 2.2. Implicit Associates

Focusing is also used in the processing of certain full noun phrases, both definite and indefinite, which involve ***implicit associates.*** The term implicit associates refers to the relationship between ***a disk drive*** and ***the motor*** in examples like ***The field engineer installed a disk drive. The motor failed.*** It is natural for a human reader to infer that the motor is part of the disk drive. In order to capture this intuition, it is necessary for the system to relate the motor to the disk drive of which it is part. Relationships of this kind have been extensively discussed in the literature on definite reference. For example, implicit associates correspond to ***inferrable*** entities described by [Prince1981], the ***associated use definites*** of [Hawkins1978], and the ***associated*** type of implicit backwards specification discussed by [Sidner1979]. Sidner suggests that implicit associates should be found among the entities in focus. Thus, when the system encounters a definite noun phrase mentioned for the first time, it sequentially examines each member of the **FocusList** to determine if it is a possible associate of the current noun phrase. The specific association relationships (such as part-whole, object-property, and so on) are defined in the knowledge base.

This mechanism is also used in the processing of certain indefinite noun phrases. In every domain, it is claimed, there are certain types of entities which can be classified as ***dependent.*** By this is meant an entity which is not typically mentioned on its own, but which is referred to in connection with another entity, on which it is dependent. In the maintenance domain, for example, parts such as keyboards, motors, and printed circuit boards are dependent, since when they are mentioned, they are normally mentioned as being part of something else, such as a console, disk drive, or

coindexing using some of the insights of [Reinhart1976], and [Chomsky1981] will be required.

printer.[4] In an example like *The system is down. The field engineer replaced a bad printed circuit board,* it seems clear that a relationship between the printed circuit board and the system should be represented. Upon encountering a reference to a dependent entity like the printed circuit board, the system looks through the **FocusList** to determine if any previously mentioned entities can be associated with a printed circuit board, and if so, the relationship is made explicit. If no associate has been mentioned, the entity will be associated with a default defined in the knowledge base. For example, in the maintenance domain, parts are defined as dependent entities, and in the absence of an explicitly mentioned associate, they are represented as associated with the system.

### 2.3. One-Anaphora

PUNDIT extends focusing to the analysis of *one*-anaphora following [Dahl1984], which claims that focus is central to the interpretation of *one*-anaphora. Specifically, the referent of a *one*-anaphoric noun phrase (e.g., *the blue one, some large ones*) is claimed to be a member or members of a set which is the focus of the current clause. For example, in *Installed two disk drives. One failed,* the set of two disk drives is assumed to be the focus of *One failed*, and the disk drive that failed is a member of that set. This analysis can be contrasted with that of [Halliday1976], which treats *one*-anaphora as a surface syntactic phenomenon, completely distinct from reference. It is more consistent with the theoretical discussions of [1976], and [Webber1983]. [5] These analyses advocate a discourse-pragmatic treatment for both *one*-anaphora and definite pronouns. The main computational advantage of treating *one*-anaphora as a discourse problem is that, since definite pronouns are treated this way, little modification is needed to the basic anaphora mechanism to allow it to handle *one*-anaphora. In contrast, an implementation following the account of Halliday and Hasan would be much more complex and specific to *one*-anaphora.

The process of reference resolution for *one*-anaphora occurs in two stages. The first stage is resolution of the anaphor, *one*, and this is the stage that involves focusing. When the system processes the head noun *one*, it instantiates it with the *category* of the first set in the **FocusList** (*disk drive* in this example).[6] In other words, the referent of the noun phrase must be a member of the previously mentioned set of disk drives. The second stage of reference resolution for *one*-anaphora assigns a specific disk drive as the referent of the entire noun phrase, using the same procedures that would be used for a full noun phrase, *a disk drive*.

The extension of the system to *one*-anaphora provides the clearest motivation for the choice of a syntactic focus in PUNDIT. Before I discuss the kinds of examples

[4] There are exceptions to this generalization. For example, in a sentence like *field engineer ordered motor*, the motor on order is not part of anything else (yet). In PUNDIT, these cases are assumed to depend on the verb meaning. In this example, the object of *ordered* is categorized as *non-specific*, and reference resolution is not called. See [Palmer1986] for details.

[5] Although not Webber's analysis in [Webber1978], which advocates an approach similar to Halliday and Hasan's.

[6] Currently the only sets in the **FocusList** are those which were explictly mentioned in the text. However, as pointed out by [Dahl1982.], and [Webber1983, Dahl1984], other sets besides those explictly mentioned are available for anaphoric reference. These have not yet been added to the system.

which support this approach, I will briefly describe the relevant part of the focusing algorithm based on thematic roles which is proposed by[Sidner1979]. After each sentence, the focusing algorithms order the elements in the sentence in the order in which they are to be considered as potential foci in the next sentence. Sidner's ordering and that of PUNDIT are compared in Figure 1.

The idea that surface syntax is important in focusing comes from a suggestion by [Erteschik-Shir1979], that every sentence has a ***dominant*** syntactic constituent, which provides a default topic for the following utterance[7]. Intuitively, the dominant constituent can be thought of as the one to which the hearer's attention is primarily drawn. Operationally the dominance of a constituent is tested by seeing if a referent with that constituent as the antecedent can be cooperatively referred to with an unstressed pronoun in the following sentence.

The feature of ***one***-anaphora which motivates the syntactic algorithm is that the availability of certain noun phrases as antecedents for ***one***-anaphora is strongly affected by surface word order variations which change syntactic relations, but which do not affect thematic roles. If thematic roles are crucial for focusing, then this pattern would not be observed.

Consider the following examples:

(1) A: I'd like to plug in this lamp, but the bookcases are blocking the electrical outlets.

B: Well, can we move one?

(2) A: I'd like to plug in this lamp, but the electrical outlets are blocked by the bookcases.

| **Sidner** | **PUNDIT** |
|---|---|
| **Theme** | **Sentence** |
| **Other thematic roles** | **Direct Object** |
| **Agent** | **Subject** |
| **Verb Phrase** | **Objects of Prepositional Phrases** |

**Figure 1: Comparison of Potential Focus Ordering in Sidner's System and PUNDIT**

[7] As discussed in [Dahl1984] there are problems with Erteschik-Shir's definition of dominance and slightly different definition is proposed. However the details of this reformulation do not concern us here.

B: Well, can we move one?

In (1), most informants report an initial impression that B is talking about moving the electrical outlets. This does not happen for (2). This indicates that the expected focus following (1) A is the outlets, while it is the bookcases in (1) B. However, in each case, the thematic roles are the same, so an algorithm based on thematic roles would predict no difference between (1) and (2).

Similar examples using definite pronouns do not seem to exhibit the same effect. In (3) and (4), ***they*** seems to be ambiguous, until world knowledge is brought in. Thus, in order to handle definite pronouns alone, either algorithm would be adequate.

(3) A: I'd like to plug in this lamp, but bookcases are blocking the electrical outlets.

B: Well, can we move them?

(4) A: I'd like to plug in this lamp, but the electrical outlets are blocked by the bookcases.

B: Well, can we move them?

(5) and (6) illustrate another example with ***one***-anaphora. In (5) but not in (6), the initial interpretation seems to be that a bug has lost its leaves. As in (1) and (2), however, the thematic roles are the same, so a thematic-role-based algorithm would predict no difference between the sentences.

(5) The plants are swarming with the bugs. One's already lost all its leaves.

(6) The bugs are swarming over the plants. One's already lost all its leaves.

In addition to theoretical considerations, there are a number of obvious practical advantages to defining focus on constituents rather than on thematic roles. For example, constituents can often be found more reliably than thematic roles. In addition, thematic roles have to be defined individually for each verb.[8] Since thematic roles for verbs can vary across domains, defining focus on syntax makes it less domain dependent, and hence more portable. While in principle focus based on thematic roles does not have to be domain-dependent, a general algorithm based on thematic roles would have to rely on a a general, domain-neutral specification of all possible thematic roles and their behavior in focusing. Until such a specification exists, a thematic-role based focusing algorithm must be redefined for each new domain as the domain requires the definition of new thematic roles, and because of this, will continue to be less portable than an approach based on syntax.

[8] Of course, some generalizations can be made about how arguments map to thematic roles. For example, the basic definition of the thematic role ***theme*** is that, for a verb of motion, the theme is the argument that moves. More generally, the theme is the argument that is most affected by the action of the verb, and its typical syntactic manifestation is as a direct object of a transitive verb, or the subject of an intransitive verb. However, even if these generalizations are accurate, they are no more than guidelines for finding the themes of verbs. The verbs still have to be classified individually.

## 3. Implementation

### 3.1. The FocusList and CurrentContext

The data structures that retain information from sentence to sentence in the PUNDIT system are the **FocusList** and the **CurrentContext**. The **FocusList** is a list of all the discourse entities which are eligible to be considered as foci, listed in the order in which they are to be considered. For example, after a sentence like *The field engineer replaced the disk drive,* the following **FocusList** would be created.

**[[event1],[drive1],[engineer1]]**

The members of the **FocusList** are unique identifiers that have been assigned to the three discourse entities -- the disk drive, the field engineer, and the event. The **CurrentContext** contains the information that has been conveyed by the discourse so far. After the example above, the **CurrentContext** would contain three types of information:

(1) *Discourse id's*, which represent classifications of entities. For example, **id(field^engineer,[engineer1])** means that **[engineer1]** is a a field engineer.[9]

(2) Facts about part-whole relationships (**hasparts**). In the example in Figure 2, notice that the lack of a representation of time results in both drives being part of the system, which they are, but not at the same time. Work to remedy this problem is in progress.

(3) Representations of the events in the discourse. For example, if the event is that of a disk drive having been replaced, the representation consists of a unique identifier (**[event1]**), the surface verb (**replace(time(_))**), and the decomposition of the verb with its (known) arguments instantiated[10]. The thematic roles involved are **object1,** the replaced disk drive, **object2,** the replacement disk drive, **time** and **instrument** which are uninstantiated, and **agent,** the field engineer. (See[Palmer1986], for details of this representation). Figure 2 illustrates how the **CurrentContext** looks after the discourse-initial sentence, ***The field engineer replaced the disk drive.***

### 3.2. The Focusing Algorithm

The focusing algorithm used in this system resembles that of [Sidner1979], although it does not use the actor focus and uses surface syntax rather than thematic roles, as discussed above. The focusing algorithm is illustrated in Figure 3. Removing candidates from the **FocusList** when they are no longer eligible to be the referents of pronouns is not currently done in this system. The conditions determining this have not been fully investigated, and since the texts involved are short, few problems are created in practice. This problem will be addressed by future research.

---

[9] **field^engineer** is an example of the representation used in PUNDIT for an idiom.

[10] **_8176** is an uninstantiated variable representing the time of the replacement. It appears in several places, such as **included(object2([drive2]),time(_8176))**, and **missing(object1([drive1]),time(_8176))**.

```
id(field^engineer,[engineer1]),

id(disk^drive,[drive1]),

id(system,[system1]),

id(disk^drive,[drive2]),

id(event,[event1]),


haspart([system1],[drive1]),

haspart([system1],[drive2])]


event([event1],
       replace(time( 8176)),
       [included(object2([drive2]),time( 8176)),
        missing(object1([drive1]),time( 8176)),
        use(instrument( 8405),
           exchange(object1([drive1]),object2([drive2]),time( 8176))),
        cause(agent([engineer1]),
             use(instrument( 8405),
                exchange(object1([drive1]),object2([drive2]),time( 8176))))])
```

Figure 2: CurrentContext after *The field engineer replaced the disk drive.*

(1) **First Sentence of a Discourse:**

**Establish expected foci for the next sentence (order FocusList): the order reflects how likely that constituent is to become the focus of the following sentence.**

**Sentence**
**Direct Object**
**Subject**
**Objects of Prepositional Phrases**

**(2) Subsequent Sentences (update FocusList):**

**If there is a pronoun in the current sentence, move the focus to the referent of the pronoun. If there is no pronoun, retain the focus from the previous sentence. Order the other elements in the sentence as in (1).**

**Figure 3: The Focusing Algorithm**

## 4. Summary

Several interesting research issues are raised by this work. For example, what is the source of the focusing algorithm? Is it derivable from theoretical considerations about how language is processed by human beings, or is it simply an empirical observation about conventions used in particular languages to bring discourse entities into prominence? Evidence bearing on this issue would be to what extent the focusing mechanism carries over to other, non-related languages. Kameyama's work on Japanese suggests that there are some similarities across languages. To the extent that such similarities exist, it would suggest that the algorithm is derivable from other theoretical considerations, and is not simply a reflection of linguistic conventions.

This paper has described the reference resolution component of PUNDIT, a large text understanding system in Prolog. A focusing algorithm based on surface syntactic constituents is used in the processing of several different types of reduced reference: definite pronouns, *one*-anaphora, elided noun phrases, and implicit associates. This generality points out the usefulness of treating focusing as a problem in itself rather than simply as a tool for pronoun resolution.

## REFERENCES

[1976]
Jorge Hankamer and Ivan Sag, Deep and Surface Anaphora. ***Linguistic Inquiry*** **7**(3), 1976, pp. 391-428.

[Chomsky1981]
Noam Chomsky, ***Lectures on Government and Binding.*** Foris Publications, Dordrecht, 1981.

[Dahl1982.]
Deborah A. Dahl, Discourse Structure and one-anaphora in English, presented at the 57th Annual Meeting of the Linguistic Society of America, San Diego, 1982..

[Dahl1984]
Deborah A. Dahl, The Structure and Function of One-Anaphora in English, PhD Thesis; (also published by Indiana University Linguistics Club, 1985), University of Minnesota, 1984.

[Erteschik-Shir1979]
Nomi Erteschik-Shir and Shalom Lappin, Dominance and the Functional Explanation of Island Phenomena. ***Theoretical Linguistics***, 1979, pp. 41-86.

[Grosz1983]
Barbara Grosz, Aravind K. Joshi, and Scott Weinstein, Providing a Unified Account of Definite Noun Phrases in Discourse. ***Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics***, 1983, pp. 44-50.

[Gruber1976]
Jeffery Gruber, ***Lexical Structure in Syntax and Semantics.*** North Holland, New York, 1976.

[Gundel1974]
Jeanette K. Gundel, Role of Topic and Comment in Linguistic Theory, Ph.D. thesis, University of Texas at Austin, 1974.

[Gundel1980]
Jeanette K. Gundel, Zero-NP Anaphora in Russian. ***Chicago Linguistic Society Parasession on Pronouns and Anaphora***, 1980.

[Halliday1976]
Michael A. K. Halliday and Ruqaiya Hasan, ***Cohesion in English.*** Longman, London, 1976.

[Hawkins1978]
John A. Hawkins, ***Definiteness and Indefiniteness.*** Humanities Press, Atlantic Highlands, New Jersey, 1978.

[Hirschman1982]
L. Hirschman and K. Puder, Restriction Grammar in Prolog. In ***Proc. of the First International Logic Programming Conference***, M. Van Caneghem (ed.), Association pour la Diffusion et le Developpement de Prolog, Marseilles, 1982, pp. 85-90.

[Hirschman1985]
L. Hirschman and K. Puder, Restriction Grammar: A Prolog Implementation. In ***Logic Programming and its Applications***, D.H.D. Warren and M. VanCaneghem (ed.), 1985.

[Hirst1981]
Graeme Hirst, ***Anaphora in Natural Language Understanding.*** Springer-Verlag, New York, 1981.

[Kameyama1985]
Megumi Kameyama, Zero Anaphora: The Case of Japanese, Ph.D. thesis, Stanford University, 1985.

[Palmer1985]
Martha S. Palmer, Driving Semantics for a Limited Domain, Ph.D. thesis, University of Edinburgh, 1985.

[Palmer1986]
Martha S. Palmer, Deborah A. Dahl, Rebecca J. Schiffman, Lynette Hirschman, Marcia Linebarger, and John Dowding, Recovering Implicit Information, to be presented at the 24th Annual Meeting of the Association for Computational Linguistics, Columbia University, New York, August 1986.

[Prince1981]
Ellen F. Prince, Toward a Taxonomy of Given-New Information. In ***Radical Pragmatics***, Peter Cole (ed.), Academic Press, New York, 1981.

[Reinhart1976]
Tanya Reinhart, The Syntactic Domain of Anaphora, Ph.D. thesis, Massachusetts Institute of Technology, 1976.

[Sager1981]
N. Sager, ***Natural Language Information Processing: A Computer Grammar of English and Its Applications.*** Addison-Wesley, Reading, Mass., 1981.

[Sidner1979]
Candace Lee Sidner, Towards a Computational Theory of Definite Anaphora Comprehension in English Discourse, MIT-AI TR-537, Cambridge, MA, 1979.

[Webber1978]
Bonnie Lynn Webber, ***A Formal Approach to Discourse Anaphora.*** Garland, New York, 1978.

[Webber1983]
Bonnie Lynn Webber, So What Can We Talk About Now?. In *Computational Models of Discourse*, Michael Brady and Robert C. Berwick (ed.), 1983.

# SECTION 4: RESEARCH CONTRIBUTIONS

*SRI International*

# COMMONSENSE METAPHYSICS AND LEXICAL SEMANTICS

Jerry R. Hobbs, William Croft, Todd Davies,
Douglas Edwards, and Kenneth Laws

Artificial Intelligence Center
SRI International

## 1 Introduction

In the TACITUS project for using commonsense knowledge in the understanding of texts about mechanical devices and their failures, we have been developing various commonsense theories that are needed to mediate between the way we talk about the behavior of such devices and causal models of their operation. Of central importance in this effort is the axiomatization of what might be called "commonsense metaphysics". This includes a number of areas that figure in virtually every domain of discourse, such as scalar notions, granularity, time, space, material, physical objects, causality, functionality, force, and shape. Our approach to lexical semantics is then to construct core theories of each of these areas, and then to define, or at least characterize, a large number of lexical items in terms provided by the core theories. In the TACITUS system, processes for solving pragmatics problems posed by a text will use the knowledge base consisting of these theories in conjunction with the logical forms of the sentences in the text to produce an interpretation. In this paper we do not stress these interpretation processes; this is another, important aspect of the TACITUS project, and it will be described in subsequent papers.

This work represents a convergence of research in lexical semantics in linguistics and efforts in AI to encode commonsense knowledge. Lexical semanticists over the years have developed formalisms of increasing adequacy for encoding word meaning, progressing from simple sets of features (Katz and Fodor, 1963) to notations for predicate-argument structure (Lakoff, 1972; Miller and Johnson-Laird, 1976), but the early attempts still limited access to world knowledge and assumed only very restricted sorts of processing. Workers in computational linguistics introduced inference (Rieger, 1974; Schank, 1975) and other complex cognitive processes (Herskovits, 1982) into our understanding of the role of word meaning. Recently, linguists have given greater attention to the cognitive processes that would operate on their representations (e.g., Talmy, 1983; Croft, 1986). Independently, in AI an effort arose to encode large amounts of commonsense knowledge (Hayes, 1979; Hobbs and Moore, 1985; Hobbs et al. 1985). The research reported here represents a convergence of these various developments. By developing core theories of several fundamental phenomena and defining lexical items within these theories, using the full power of predicate calculus, we are able to cope with complexities of word meaning that have hitherto escaped lexical semanticists, within a framework that gives full scope to the planning and reasoning processes that manipulate representations of word meaning.

In constructing the core theories we are attempting to adhere to several methodological principles.

1. One should aim for characterization of concepts, rather than definition. One cannot generally expect to find necessary and sufficient conditions for a concept. The most we can hope for is to find a number of necessary conditions and a number of sufficient conditions. This amounts to saying that a great many predicates are primitive, but primitives that are highly interrelated with the rest of the knowledge base.

2. One should determine the minimal structure necessary for a concept to make sense. In efforts to axiomatize some area, there are two positions one may take, exemplified by set theory and by group theory. In axiomatizing set theory, one attempts to capture exactly some concept one has strong intuitions about. If the axiomatization turns out to have unexpected models, this exposes an inadequacy. In group theory, by contrast, one characterizes an abstract class of structures. If there turn out to be unexpected models, this is a serendipitous discovery of a new phenomenon that we can reason about using an old theory. The pervasive character of metaphor in natural language discourse shows that our commonsense theories of the world ought to be much more like group theory than set theory. By seeking minimal structures in axiomatizing concepts, we optimize the possibilities of using the theories in metaphorical and analogical contexts. This principle is illustrated below in the section on regions. One consequence of this principle is that our approach will seem more syntactic than semantic. We have concentrated more on

specifying axioms than on constructing models. Our view is that the chief role of models in our effort is for proving the consistency and independence of sets of axioms, and for showing their adequacy. As an example of the last point, many of the spatial and temporal theories we construct are intended at least to have Euclidean space or the real numbers as one model, and a subclass of graph-theoretical structures as other models.

3. A balance must be struck between attempting to cover all cases and aiming only for the prototypical cases. In general, we have tried to cover as many cases as possible with an elegant axiomatization, in line with the two previous principles, but where the formalization begins to look baroque, we assume that higher processes will suspend some inferences in the marginal cases. We assume that inferences will be drawn in a controlled fashion. Thus, every outré, highly context-dependent counterexample need not be accounted for, and to a certain extent, definitions can be geared specifically for a prototype.

4. Where competing ontologies suggest themselves in a domain, one should attempt to construct a theory that accommodates both. Rather than commit oneself to adopting one set of primitives rather than another, one should show how each set of primitives can be characterized in terms of the other. Generally, each of the ontologies is useful for different purposes, and it is convenient to be able to appeal to both. Our treatment of time illustrates this.

5. The theories one constructs should be richer in axioms than in theorems. In mathematics, one expects to state half a dozen axioms and prove dozens of theorems from them. In encoding commonsense knowledge it seems to be just the opposite. The theorems we seek to prove on the basis of these axioms are theorems about specific situations which are to be interpreted, in particular, theorems about a text that the system is attempting to understand.

6. One should avoid falling into "black holes". There are a few "mysterious" concepts which crop up repeatedly in the formalization of commonsense metaphysics. Among these are "relevant" (that is, relevant to the task at hand) and "normative" (or conforming to some norm or pattern). To insist upon giving a satisfactory analysis of these before using them in analyzing other concepts is to cross the event horizon that separates lexical semantics from philosophy. On the other hand, our experience suggests that to avoid their use entirely is crippling; the lexical semantics of a wide variety of other terms depends upon them. Instead, we have decided to leave them minimally analyzed for the moment and use them without scruple in the analysis of other commonsense concepts. This approach will allow us to accumulate many examples of the use of these mysterious concepts, and in the end, contribute to their successful analysis. The use of these concepts appears below in the discussions of the words "immediately", "sample", and "operate".

We chose as an initial target problem to encode the commonsense knowledge that underlies the concept of "wear", as in a part of a device wearing out. Our aim was to define "wear" in terms of predicates characterized elsewhere in the knowledge base and to infer consequences of wear. For something to wear, we decided, is for it to lose imperceptible bits of material from its surface due to abrasive action over time. One goal,which we have not yet achieved, is to be able to prove as a theorem that since the shape of a part of a mechanical device is often functional and since loss of material can result in a change of shape, wear of a part of a device can result in the failure of the device as a whole. In addition, as we have proceeded, we have characterized a number of words found in a set of target texts, as it has become possible.

We are encoding the knowledge as axioms in what is for the most part a first-order logic, described in Hobbs (1985a), although quantification over predicates is sometimes convenient. In the formalism there is a nominalization operator " ' " for reifying events and conditions, as expressed in the following axiom schema:

$$(\forall x)p(x) \equiv (\exists e)p'(e,x) \wedge Exist(e)$$

That is, $p$ is true of $x$ if and only if there is a condition $e$ of $p$ being true of $x$ and $e$ exists in the real world.

In our implementation so far, we have been proving simple theorems from our axioms using the CG5 theorem-prover developed by Mark Stickel (1982), but we are only now beginning to use the knowledge base in text processing.

## 2 Requirements on Arguments of Predicates

There is a notational convention used below that deserves some explanation. It has frequently been noted that relational words in natural language can take only certain types of words as their arguments. These are usually described as selectional constraints. The same is true of predicates in our knowledge base. They are expressed below by rules of the form

$$p(x,y) : r(x,y)$$

This means that for $p$ even to make sense applied to $x$ and $y$, it must be the case that $r$ is true of $x$ and $y$. The logical import of this rule is that wherever there is an axiom of the form

$$(\forall x,y)p(x,y) \supset q(x,y)$$

this is really to be read as

$$(\forall x,y)p(x,y) \wedge r(x,y) \supset q(x,y)$$

The checking of selectional constraints, therefore, falls out as a by-product of other logical operations: the constraint $r(x, y)$ must be verified if anything else is to be proven from $p(x, y)$.

The simplest example of such an $r(x, y)$ is a conjunction of sort constraints $r_1(x) \wedge r_2(y)$. Our approach is a generalization of this, because much more complex requirements can be placed on the arguments. Consider, for example, the verb "range". If $x$ ranges from $y$ to $z$, there must be a scale $s$ that includes $y$ and $z$, and $x$ must be a set of entities that are located at various places on the scale. This can be represented as follows:

$$\begin{aligned} &range(x, y, z) : (\exists s)scale(s) \wedge y \in s \\ &\quad \wedge z \in s \wedge set(x) \\ &\quad \wedge (\forall u)[u \in x \supset (\exists v)v \in s \wedge at(u, v)] \end{aligned}$$

# 3 The Knowledge Base

## 3.1 Sets and Granularity

At the foundation of the knowledge base is an axiomatization of set theory. It follows the standard Zermelo-Frankel approach, except that there is no Axiom of Infinity.

Since so many concepts used in discourse are grain-dependent, a theory of granularity is also fundamental (see Hobbs 1985b). A grain is defined in terms of an indistinguishability relation, which is reflexive and symmetric, but not necessarily transitive. One grain can be a *refinement* of another with the obvious definition. The most refined grain is the identity grain, i.e., the one in which every two distinct elements are distinguishable. One possible relationship between two grains, one of which is a refinement of the other, is what we call an "Archimedean relation", after the Archimedean property of real numbers. Intuitively, if enough events occur that are imperceptible at the coarser grain $g_2$ but perceptible at the finer grain $g_1$, then the aggregate will eventually be perceptible at the coarser grain. This is an important property in phenomena subject to the Heap Paradox. Wear, for instance, eventually has significant consequences.

## 3.2 Scales

A great many of the most common words in English have scales as their subject matter. This includes many prepositions, the most common adverbs, comparatives, and many abstract verbs. When spatial vocabulary is used metaphorically, it is generally the scalar aspect of space that carries over to the target domain. A scale is defined as a set of elements, together with a partial ordering and a granularity (or an indistinguishability relation). The partial ordering and the indistinguishability relation are consistent with each other:

$$(\forall x, y, z)x < y \wedge y \sim z \supset x < z \vee x \sim z$$

It is useful to have an adjacency relation between points on a scale, and there are a number of ways we could introduce it. We could simply take it to be primitive; in a scale having a distance function, we could define two points to be adjacent when the distance between them is less than some $\epsilon$; finally, we could define adjacency in terms of the grain-size:

$$\begin{aligned} &(\forall x, y, s)adj(x, y, s) \equiv \\ &\quad (\exists z)z \sim x \wedge z \sim y \wedge \neg[x \sim y], \end{aligned}$$

Two important possible properties of scales are connectedness and denseness. We can say that two elements of a scale are connected by a chain of *adj* relations:

$$\begin{aligned} &(\forall x, y, s)connected(x, y, s) \equiv \\ &\quad adj(x, y, s) \vee \\ &\quad (\exists z)adj(x, z, s) \wedge connected(z, y, s) \end{aligned}$$

A scale is connected (*sconnected*) if all pairs of elements are connected. A scale is dense if between any two points there is a third point, until the two points are so close together that the grain-size won't let us tell what the situation is. Cranking up the magnification could well resolve the continuous space into a discrete set, as objects into atoms.

$$\begin{aligned} &(\forall s)dense(s) \equiv \\ &\quad (\forall x, y, <)x \in s \wedge y \in s \wedge order(<, s) \wedge x < y \\ &\quad\quad \supset (\exists z)(x < z \wedge z < y) \\ &\quad\quad\quad \vee (\exists z)(x \sim z \wedge z \sim y) \end{aligned}$$

This captures the commonsense notion of continuity.

A subscale of a scale has as its elements a subset of the elements of the scale and has as its partial ordering and its grain the partial ordering and the grain of the scale.

$$\begin{aligned} &(\forall s_1, <, \sim)order(<, s_1) \wedge grain(\sim, s_1) \\ &\quad \supset (\forall s_2)[subscale(s_2, s_1) \\ &\quad\quad \equiv subset(s_2, s_1) \wedge order(<, s_2) \wedge grain(\sim, s_2)] \end{aligned}$$

An interval can be defined as a connected subscale:

$$\begin{aligned} &(\forall i)interval(i) \equiv (\exists s)scale(s) \\ &\quad \wedge subscale(i, s) \wedge sconnected(i) \end{aligned}$$

The relations between time intervals that Allen and Kautz (1985) have defined can be defined in a straightforward manner in the approach presented here, applied to intervals in general.

A concept closely related to scales is that of a "cycle". This is a system which has a natural ordering locally but contains a loop globally. Examples include the color wheel, clock times, and geographical locations ordered by "east of". We have axiomatized cycles in terms of a ternary *between* relation, whose axioms parallel the axioms for a partial ordering.

The figure-ground relationship is of fundamental importance in language. We encode this with the primitive predicate *at*. The minimal structure that seems to be necessary for something to be a ground is that of a scale; hence, this is a selectional constraint on the arguments of *at*.

$$at(x,y) : (\exists s)y \in s \wedge scale(s)$$

At this point, we are already in a position to define some fairly complex words. As an illustration, we give the example of "range" as in "$x$ ranges from $y$ to $z$":

$$\begin{aligned}&(\forall x,y,z)range(x,y,z) \equiv\\ &\quad(\exists s,s_1,u_1,u_2)scale(s) \wedge subscale(s_1,s)\\ &\quad\wedge bottom(y,s_1) \wedge top(z,s_1)\\ &\quad\wedge u_1 \in x \wedge at(u_1,y)\\ &\quad\wedge u_2 \in x \wedge at(u_2,z)\\ &\quad\wedge(\forall u)[u \in x \supset (\exists v)v \in s_1 \wedge at(u,v)]\end{aligned}$$

A very important scale is the linearly ordered scale of numbers. We do not plan to reason axiomatically about numbers, but it is useful in natural language processing to have encoded a few facts about numbers. For example, a set has a cardinality which is an element of the number scale.

Verticality is a concept that would be most properly analyzed in the section on space, but it is a property that many other scales have acquired metaphorically, for whatever reason. The number scale is one of these. Even in the absence of an analysis of verticality, it is a useful property to have as a primitive in lexical semantics.

The word "high" is a vague term that asserts an entity is in the upper region of some scale. It requires that the scale be a *vertical* one, such as the number scale. The verticality requirement distinguishes "high" from the more general term "very"; we can say "very hard" but not "highly hard". The phrase "highly planar" sounds all right because the high register of "planar" suggests a quantifiable, scientific accuracy, whereas the low register of "flat" makes "highly flat" sound much worse.

The test of any definition is whether it allows one to draw the appropriate inferences. In our target texts, the phrase "high usage" occurs. Usage is a set of using events, and the verticality requirement on "high" forces us to coerce the phrase into "a high or large number of using events". Combining this with an axiom that says that the use of a mechanical device involves the likelihood of abrasive events, as defined below, and with the definition of "wear" in terms of abrasive events, we should be able to conclude the likelihood of wear.

## 3.3 Time: Two Ontologies

There are two possible ontologies for time. In the first, the one most acceptable to the mathematically minded, there is a time line, which is a scale having some topological structure. We can stipulate the time line to be linearly ordered (although it is not in approaches that build ignorance of relative times into the representation of time (e.g., Hobbs, 1974) nor in approaches using branching futures (e.g., McDermott, 1985)), and we can stipulate it to be dense (although it is not in the situation calculus). We take *before* to be the ordering on the time line:

$$\begin{aligned}&(\forall t_1,t_2)before(t_1,t_2) \equiv\\ &\quad(\exists T,<)Time\text{-}line(T) \wedge order(<,T)\\ &\quad\wedge t_1 \in T \wedge t_2 \in T \wedge t_1 < t_2\end{aligned}$$

We allow both instants and intervals of time. Most events occur at some instant or during some interval. In this approach, nearly every predicate takes a time argument.

In the second ontology, the one that seems to be more deeply rooted in language, the world consists of a large number of more or less independent processes, or histories, or sequences of events. There is a primitive relation *change* between conditions. Thus,

$$change(e_1,e_2) \wedge p'(e_1,x) \wedge q'(e_2,x)$$

says that there is a change from the condition $e_1$ of $p$ being true of $x$ to the condition $e_2$ of $q$ being true of $x$.

The time line in this ontology is then an artificial construct, a regular sequence of imagined abstract events—think of them as ticks of a clock in the National Bureau of Standards—to which other events can be related. The change ontology seems to correspond to the way we experience the world. We recognize relations of causality, change of state, and copresence among events and conditions. When events are not related in these ways, judgments of relative time must be mediated by copresence relations between the events and events on a clock and change of state relations on the clock.

The predicate *change* possesses a limited transitivity. There has been a change from Reagan being an actor to Reagan being President, even though he was governor in between. But we probably do not want to say there has been a change from Reagan being an actor to Margaret Thatcher being Prime Minister, even though the second comes after the first.

We can say that times, viewed in this ontology as events, always have a *change* relation between them.

$$(\forall t_1,t_2)before(t_1,t_2) \supset change(t_1,t_2)$$

The predicate *change* is related to *before* by the axiom

$$\begin{aligned}&(\forall e_1,e_2)change(e_1,e_2) \supset\\ &\quad(\exists t_1,t_2)at(e_1,t_1)\\ &\quad\wedge at(e_2,t_2) \wedge before(t_1,t_2)\end{aligned}$$

This does not allow us to derive change of state from temporal succession. For this, we need axioms of the form

$$\begin{aligned}&(\forall e_1,e_2,t_1,t_2,x)p'(e_1,x) \wedge at(e_1,t_1)\\ &\qquad\wedge q'(e_2,x) \wedge at(e_2,t_2) \wedge before(t_1,t_2)\\ &\quad\supset change(e_1,e_2)\end{aligned}$$

That is, if $x$ is $p$ at time $t_1$ and $q$ at a later time $t_2$, then there has been a change of state from one to the other. Time arguments in predications can be viewed as abbreviations:

$$(\forall x,t)p(x,t) \equiv (\exists e)p'(e,x) \wedge at(e,t)$$

The word "move", or the predicate *move*, (as in "$x$ moves from $y$ to $z$") can then be defined equivalently in terms of change

$$(\forall x, y, z)move(x, y, z) \equiv$$
$$(\exists e_1, e_2)change(e_1, e_2)$$
$$\wedge at'(e_1, x, y) \wedge at'(e_2, x, z)$$

or in terms of the time line

$$(\forall x, y, z)move(x, y, z) \equiv$$
$$(\exists t_1, t_2)at(x, y, t_1) \wedge at(x, z, t_2) \wedge before(t_1, t_2)$$

In English and apparently all other natural languages, both ontologies are represented in the lexicon. The time line ontology is found in clock and calendar terms, tense systems of verbs, and in the deictic temporal locatives such as "yesterday", "today", "tomorrow", "last night", and so on. The change ontology is exhibited in most verbs, and in temporal clausal connectives. The universal presence of both classes of lexical items and grammatical markers in natural languages requires a theory which can accommodate both ontologies, illustrating the importance of methodological principle 4.

Among temporal connectives, the word "while" presents interesting problems. In "$e_1$ while $e_2$", $e_2$ must be an event occurring over a time interval; $e_1$ must be an event and may occur either at a point or over an interval. One's first guess is that the point or interval for $e_1$ must be included in the interval for $e_2$. However, there are cases, such as

> It rained while I was in Philadelphia.

or

> The electricity should be off while the switch is being repaired.

which suggest the reading "$e_2$ is included in $e_1$". We came to the conclusion that one can infer no more than that $e_1$ and $e_2$ overlap, and any tighter constraints result from implicatures from background knowledge.

The word "immediately" also presents a number of problems. It requires its argument $e$ to be an ordering relation between two entities $x$ and $y$ on some scale $s$.

$$immediate(e) : (\exists x, y, s)less\text{-}than'(e, x, y, s)$$

It is not clear what the constraints on the scale are. Temporal and spatial scales are okay, as in "immediately after the alarm" and "immediately to the left", but the size scale isn't:

> * John is immediately larger than Bill.

Etymologically, it means that there are no intermediate entities between $x$ and $y$ on $s$. Thus,

$$(\forall e, x, y, s)immediate(e) \wedge less\text{-}than'(e, x, y, s)$$
$$\supset \neg(\exists z)less\text{-}than(x, z, s) \wedge less\text{-}than(z, y, s)$$

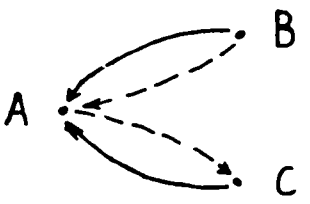



Figure 1: The simplest space.

However, this will only work if we restrict $z$ to be a *relevant* entity. For example, in the sentence

> We disengaged the compressor immediately after the alarm.

the implication is that no event that could damage the compressor occurred between the alarm and the disengagement, since the text is about equipment failure.

## 3.4 Spaces and Dimension: The Minimal Structure

The notion of dimension has been made precise in linear algebra. Since the concept of a region is used metaphorically as well as in the spatial sense, however, we were concerned to determine the *minimal* structure that a system requires for it to make sense to call it a space of more than one dimension. For a two-dimensional space, there must be a scale, or partial ordering, for each dimension. Moreover, the two scales must be independent, in that the order of elements on one scale can not be determined from their order on the other. Formally,

$$(\forall sp)space(sp) \equiv$$
$$(\exists s_1, s_2, <_1, <_2)scale_1(s_1, sp) \wedge scale_2(s_2, sp)$$
$$\wedge order(<_1, s_1) \wedge order(<_2, s_2)$$
$$\wedge (\exists x)(\exists y_1)(x <_1 y_1 \wedge x <_2 y_1)$$
$$\wedge (\exists y_2)(x <_1 y_2 \wedge y_2 <_2 x)$$

Note that this does not allow $<_2$ to be simply the reverse of $<_1$. An unsurprising consequence of this definition is that the minimal example of a two-dimensional space consists of three points (three points determine a plane), e.g., the points A, B, and C, where

$$A <_1 B,\ A <_1 C,\ C <_2 A,\ A <_2 B.$$

This is illustrated in Figure 1.

The dimensional scales are apparently found in all natural languages in relevant domains. The familiar three-dimensional space of common sense is defined by the three scale pairs "up-down", "front-back", and "left-right"; the two-dimensional plane of the commonsense conception of the earth's surface is represented by the two scale pairs "north-south" and "east-west".

The simplest, although not the only, way to define adjacency in the space is as adjacency on both scales:

$$(\forall x, y, sp)adj(x, y, sp) \equiv$$
$$(\exists s_1, s_2)scale_1(s_1, sp) \wedge scale_2(s_2, sp)$$
$$\wedge adj(x, y, s_1) \wedge adj(x, y, s_2)$$

A region is a subset of a space. The surface and interior of a region can be defined in terms of adjacency, in a manner paralleling the definition of a boundary in point-set topology. In the following, $s$ is the boundary or surface of a two- or three-dimensional region $r$ embedded in a space $sp$.

$$(\forall s, r)surface(s, r, sp) \equiv$$
$$(\forall x)x \in r \supset [x \in s \equiv$$
$$(Ey)(y \in sp \wedge \neg(y \in r) \wedge adj(x, y, sp))]$$

Finally, we can define the notion of "contact" in terms of points in different regions being adjacent.

$$(\forall r_1, r_2, sp)contact(r_1, r_2, sp) \equiv$$
$$disjoint(r_1, r_2) \wedge$$
$$(Ex, y)(x \in r_1 \wedge y \in r_2 \wedge adj(x, y, sp))$$

By picking the scales and defining adjacency right, we can talk about points of contact between communicational networks, systems of knowledge, and other metaphorical domains. By picking the scales to be the real line and defining adjacency in terms of $\epsilon$-neighborhoods, we get Euclidean space and can talk about contact between physical objects.

## 3.5 Material

Physical objects and materials must be distinguished, just as they are apparently distinguished in every natural language, by means of the count noun - mass noun distinction. A physical object is not a bit of material, but rather is comprised of a bit of material at any given time. Thus, rivers and human bodies are physical objects, even though their material constitution changes over time. This distinction also allows us to talk about an object losing material through wear and still being the same object.

We will say that an entity $b$ is a bit of material by means of the expression $material(b)$. Bits of material are characterized by both extension and cohesion. The primitive predication $occupies(b, r, t)$ encodes extension, saying that a bit of material $b$ occupies a region $r$ at time $t$. The topology of a bit of material is then parasitic on the topology of the region it occupies. A *part* $b_1$ of a bit of material $b$ is a bit of material whose occupied region is always a subregion of the region occupied by $b$. Point-like particles (*particle*) are defined in terms of points in the occupied region, disjoint bits (*disjointbit*) in terms of disjointness of regions, and contact between bits in terms of contact between their regions. We can then state as follows the Principle of Non-Joint-Occupancy that two bits of material cannot occupy the same place at the same time:

$$(\forall b_1, b_2)(disjointbit(b_1, b_2)$$
$$\supset (\forall x, y, b_3, b_4)interior(b_3, b_1)$$
$$\wedge interior(b_4, b_2) \wedge particle(x, b_3)$$
$$\wedge particle(y, b_4)$$
$$\supset \neg(Ez)(at(x, z) \wedge at(y, z))$$

At some future point in our work, this may emerge as a consequence of a richer theory of cohesion and force.

The cohesion of materials is also a primitive property, for we must distinguish between a bump on the surface of an object and a chip merely lying on the surface. Cohesion depends on a primitive relation *bond* between particles of material, paralleling the role of *adj* in regions. The relation *attached* is defined as the transitive closure of *bond*. A topology of cohesion is built up in a manner analogous to the topology of regions. In addition, we have encoded the relation that *bond* bears to motion, i.e. that bonded bits remain adjacent and that one moves when the other does, and the relation of bond to force, i.e. that there is a characteristic force that breaks a bond in a given material.

Different materials react in different ways to forces of various strengths. Materials subjected to force exhibit or fail to exhibit several invariance properties, proposed by Hager (1985). If the material is shape-invariant with respect to a particular force, its shape remains the same. If it is topologically invariant, particles that are adjacent remain adjacent. Shape invariance implies topological invariance. Subject to forces of a certain strength or degree $d_1$, a material ceases being shape-invariant. At a force of strength $d_2 \geq d_1$, it ceases being topologically invariant, and at a force of strength $d_3 \geq d_2$, it simply breaks. Metals exhibit the full range of possibilities, that is, $0 < d_1 < d_2 < d_3 < \infty$. For forces of strength $d < d_1$, the material is "hard"; for forces of strength $d$ where $d_1 < d < d_2$, it is "flexible"; for forces of strength $d$ where $d_2 < d < d_3$, it is "malleable". Words such as "ductile" and "elastic" can be defined in terms of this vocabulary, together with predicates about the geometry of the bit of material. Words such as "brittle" ($d_1 = d_2 = d_3$) and "fluid" ($d_1 = 0, d_3 = \infty$) can also be defined in these terms. While we should not expect to be able to *define* various material terms, like "metal" and "ceramic", we can certainly characterize many of their properties with this vocabulary.

Because of its invariance properties, material interacts with containment and motion. The word "clog" illustrates this. The predicate *clog* is a three-place relation: $x$ clogs $y$ against the flow of $z$. It is the obstruction by $x$ of $z$'s motion through $y$, but with the selectional restriction that $z$ must be something that can flow, such as a liquid, gas, or powder. If a rope is passing through a hole in a board, and a knot in the rope prevents it from going through, we do not say that the hole is clogged. On the other hand, there do not seem to be any selectional constraints on $x$. In particular, $x$ can be identical with $z$: glue, sand, or molasses can clog a passageway against its own flow. We

can speak of clogging where the obstruction of flow is not complete, but it must be thought of as "nearly" complete.

## 3.6 Other Domains

### 3.6.1 Causal Connection

Attachment within materials is one variety of causal connection. In general, if two entities $x$ and $y$ are causally connected with respect to some behavior $p$ of $x$, then whenever $p$ happens to $x$, there is some corresponding behavior $q$ that happens to $y$. In the case of attachment, $p$ and $q$ are both *move*. A particularly common variety of causal connection between two entities is one mediated by the motion of a third entity from one to the other. (This might be called a "vector boson" connection.) Photons mediating the connection between the sun and our eyes, rain drops connecting a state of the clouds with the wetness of our skin and clothes, a virus being transmitted from one person to another, and utterances passing between people are all examples of such causal connections. Barriers, openings, and penetration are all with respect to paths of causal connection.

### 3.6.2 Force

The concept of "force" is axiomatized, in a way consistent with Talmy's treatment (1985), in terms of the predications $force(a, b, d_1)$ and $resist(b, a, d_2)$—$a$ forces against $b$ with strength $d_1$ and $b$ resists $a$'s action with strength $d_2$. We can infer motion from facts about relative strength. This treatment can also be specialized to Newtonian force, where we have not merely movement, but acceleration. In addition, in spaces in which orientation is defined, forces can have an orientation, and a version of the Parallelogram of Forces Law can be encoded. Finally, force interacts with shape in ways characterized by words like "stretch", "compress", "bend", "twist", and "shear".

### 3.6.3 Systems and Functionality

An important concept is the notion of a "system", which is a set of entities, a set of their properties, and a set of relations among them. A common kind of system is one in which the entities are events and conditions and the relations are causal and enabling relations. A mechanical device can be described as such a system—in a sense, in terms of the plan it executes in its operation. The *function* of various parts and of conditions of those parts is then the role they play in this system, or plan.

The intransitive sense of "operate", as in

> The diesel was operating.

involves systems and functionality. If an entity $x$ operates, then there must be a larger system $s$ of which $x$ is a part. The entity $x$ itself is a system with parts. These parts undergo normative state changes, thereby causing $x$ to undergo normative state changes, thereby causing $x$ to produce an effect with a normative function in the larger system $s$. The concept of "normative" is discussed below.

### 3.6.4 Shape

We have been approaching the problem of characterizing shape from a number of different angles. The classical treatment of shape is via the notion of "similarity" in Euclidean geometry, and in Hilbert's formal reconstruction of Euclidean geometry (Hilbert, 1902) the key primitive concept seems to be that of "congruent angles". Therefore, we first sought to develop a theory of "orientation". The shape of an object can then be characterized in terms of changes in orientation of a tangent as one moves about on the surface of the object, as is done in vision research (e.g., Zahn and Roskies, 1972). In all of this, since "shape" can be used loosely and metaphorically, one question we are asking is whether some minimal, abstract structure can be found in which the notion of "shape" makes sense. Consider, for instance, a graph in which one scale is discrete, or even unordered. Accordingly, we have been examining a number of examples, asking when it seems right to say two structures have different shapes.

We have also examined the interactions of shape and functionality (cf. Davis, 1984). What seems to be crucial is how the shape of an obstacle constrains the motion of a substance or of an object of a particular shape (cf Shoham, 1985). Thus, a funnel concentrates the flow of a liquid, and similarly, a wedge concentrates force. A box pushed against a ridge in the floor will topple, and a wheel is a limiting case of continuous toppling.

## 3.7 Hitting, Abrasion, Wear, and Related Concepts

For $x$ to hit $y$ is for $x$ to move into contact with $y$ with some force.

The basic scenario for an abrasive event is that there is an impinging bit of material $m$ which hits an object $o$ and by doing so removes a pointlike bit of material $b_0$ from the surface of $o$:

$$\begin{aligned}
&abr\text{-}event'(e, m, o, b_0) : material(m) \\
&\quad \wedge topologically\text{-}invariant(o) \\
&(\forall e, m, o, b_0) abr\text{-}event'(e, m, o, b_0) \equiv \\
&\quad (\exists l, b, s, b_0, e_1, e_2, e_3) at(e, l) \\
&\quad \wedge consists\text{-}of(o, b, l) \wedge surface(s, b) \\
&\quad \wedge particle(b_0, s) \wedge change'(e, e_1, e_2) \\
&\quad \wedge attached'(e_1, b_0, b) \wedge not'(e_2, e_1) \\
&\quad \wedge cause(e_3, e) \wedge hit'(e_3, m, b_0)
\end{aligned}$$

After the abrasive event, the pointlike bit $b_0$ is no longer a part of the object $o$:

$$(\forall e, m, o, b_0, e_1, e_2, t_2)abr\text{-}event'(e, m, o, b_0)$$
$$\wedge change'(e, e_1, e_2) \wedge attached'(e_1, b_0, b)$$
$$\wedge not'(e_2, e_1) \wedge at(e_2, t_2)$$
$$\wedge consists\text{-}of(o, b_2, t_2)$$
$$\supset \neg part(b_0, b_2)$$

It is necessary to state this explicitly since objects and bits of material can be discontinuous.

An abrasion is a large number of abrasive events widely distributed through some nonpointlike region on the surface of an object:

$$(\forall e, m, o)abrade'(e, m, o) \equiv$$
$$(\exists be)[(\forall e_1)[e_1 \in e \supset$$
$$(\exists b_0)b_0 \in be \wedge abr\text{-}event'(e_1, m, o, b_0)]$$
$$\wedge (\forall b, s, t)[at(e, t)$$
$$\wedge consists\text{-}of(o, b, t) \wedge surface(s, b)$$
$$\supset (\exists r)subregion(r, s)$$
$$\wedge widely\text{-}distributed(be, r)]]$$

Wear can occur by means of a large collection of abrasive events distributed over time as well as space (so that there may be no time at which enough abrasive events occur to count as an abrasion). Thus, the link between wear and abrasion is via the common notion of abrasive events, not via a definition of wear in terms of abrasion.

$$(\forall e, m, o)wear'(e, x, o) \equiv$$
$$(\exists be)(\forall e_1)[e_1 \in e \supset$$
$$(\exists b_0)b_0 \in be) \wedge abr\text{-}event'(e_1, m, o, b_0)]$$
$$\wedge (\exists i)[interval(i) \wedge widely\text{-}distributed(e, i)]$$

The concept "widely distributed" concerns systems. If $x$ is distributed in $y$, then $y$ is a system and $x$ is a set of entities which are located at components of $y$. For the distribution to be wide, most of the elements of a partition of $y$ determined independently of the distribution must contain components which have elements of $x$ at them.

The word "wear" is one of a large class of other events involving cumulative, gradual loss of material – events described by words like "chip", "corrode", "file", "erode", "rub", "sand", "grind", "weather", "rust", "tarnish", "eat away", "rot", and "decay". All of these lexical items can now be defined as variations on the definition of "wear", since we have built up the axiomatizations underlying "wear". We are now in a position to characterize the entire class. We will illustrate this by defining two different types of variants of "wear" – "chip" and "corrode".

"Chip" differs from "wear" in three ways: the bit of material removed in one abrasive event is larger (it need not be point-like), it need not happen because of a material hitting against the object, and "chip" does not require (though it does permit) a large collection of such events: one can say that some object is chipped if there is only one chip in it. Thus, we slightly alter the definition of *abr-event* to accommodate these changes:

$$(\forall e, m, o, b_0)chip'(e, m, o, b_0) \equiv$$
$$(\exists t, b, s, b_0, e_1, e_2, e_3)at(e, t)$$
$$\wedge consists\text{-}of(o, b, t) \wedge surface(s, b)$$
$$\wedge part(b_0, s) \wedge change'(e, e_1, e_2)$$
$$\wedge attached'(e_1, b_0, b) \wedge not'(e_2, e_1)$$

"Corrode" differs from "wear" in that the bit of material is chemically transformed as well as being detached by the contact event; in fact, in some way the chemical transformation causes the detachment. This can be captured by adding a condition to the abrasive event which renders it a (single) corrode event:

$$corrode\text{-}event(m, o, b_0) : fluid(m)$$
$$\wedge contact(m, b_0)$$
$$(\forall e, m, o, b_0)corrode\text{-}event'(e, m, o, b_0) \equiv \cdot \cdot$$
$$(\exists t, b, s, b_0, e_1, e_2, e_3)at(e, t)$$
$$\wedge consists\text{-}of(o, b, t) \wedge surface(s, b)$$
$$\wedge particle(b_0, s) \wedge change'(e, e_1, e_2)$$
$$\wedge attached'(e_1, b_0, b) \wedge not'(e_2, e_1)$$
$$\wedge cause(e_3, e) \wedge chemical\text{-}change'(e_3, m, b_0)$$

"Corrode" itself may be defined in a parallel fashion to "wear", substituting *corrode-event* for *abr-event*.

All of this suggests the generalization that abrasive events, chipping and corrode events all detach the bit in question, and that we may describe all of these as detaching events. We can then generalize the above axiom about abrasive events resulting in loss of material to the following axiom about detaching:

$$(\forall e, m, o, b_0, b_2, e_1, e_2, t_2)detach'(e, m, o, b_0)$$
$$\wedge change'(e, e_1, e_2) \wedge attached'(e_1, b_0, b)$$
$$\wedge not'(e_2, e_1) \wedge at(e_2, t_2)$$
$$\wedge consists\text{-}of(o, b_2, t_2)$$
$$\supset \neg(part(b_0, b_2))$$

# 4 Relevance and the Normative

Many of the concepts we are investigating have driven us inexorably to the problems of what is meant by "relevant" and by "normative". We do not pretend to have solved these problems. But for each of these concepts we do have the beginnings of an account that can play a role in analysis, if not yet in implementation.

Our view of relevance, briefly stated, is that something is relevant to some goal if it is a part of a plan to achieve that goal. [A formal treatment of a similar view is given in Davies and Russell, 1986.] We can illustrate this with an example involving the word "sample". If a bit of material $x$ is a sample of another bit of material $y$, then $x$ is a part of $y$, and moreover, there are *relevant* properties $p$ and $q$ such that it is believed that if $p$ is true of $x$ then $q$ is true of $y$. That is, looking at the properties of the sample tells us something important about the properties of the whole. Frequently, $p$ and $q$ are the same property. In our target texts, the following sentence occurs:

We retained an oil sample for future inspection.

The oil in the sample is a part of the total lube oil in the lube oil system, and it is believed that a property of the sample, such as "contaminated with metal particles", will be true of all of the lube oil as well, and that this will give information about possible wear on the bearings. It is therefore relevant to the goal of maintaining the machinery in good working order.

We have arrived at the following provisional account of what it means to be "normative". For an entity to exhibit a normative condition or behavior, it must first of all be a component of a larger system. This system has structure in the form of relations among its components. A pattern is a property of the system, namely, the property of a subset of these stuctural relations holding. A norm is a pattern which is established either by conventional stipulation or by statistical regularity. An entity is behaving in a normative fashion if it is a component of a system and instantiates a norm within that system. The word "operate" given above illustrates this. When we say that an engine is operating, we have in mind a larger system, the device the engine drives, to which the engine may bear various possible relations. A subset of these relations is stipulated to be the norm—the way it is supposed to work. We say it is operating when it is instantiating this norm.

## 5 Conclusion

The research we have been engaged in has forced us to explicate a complex set of commonsense concepts. Since we have done it in as general a fashion as possible, we may expect that it will be possible to axiomatize a large number of other areas, including areas unrelated to mechanical devices, building on this foundation. The very fact that we have been able to characterize words as diverse as "range", "immediately", "brittle", "operate" and "wear" shows the promise of this approach.

## Acknowledgements

The research reported here was funded by the Defense Advanced Research Projects Agency under Office of Naval Research contract N00014-85-C-0013. It builds on work supported by NIH Grant LM03611 from the National Library of Medicine, by Grant IST-8209346 from the National Science Foundation, and by a gift from the Systems Development Foundation.

## References

[1] Allen, James F., and Henry A. Kautz. 1985. "A model of naive temporal reasoning." *Formal Theories of the Commonsense World*, ed. by Jerry R. Hobbs and Robert C. Moore, Ablex Publishing Corp., 251-268.

[2] Croft, William. 1986. *Categories and Relations in Syntax: The Clause-Level Organization of Information* Ph.D. dissertation, Department of Linguistics, Stanford University.

[3] Davies, Todd R., and Stuart J. Russell. 1986. "A logical approach to reasoning by analogy." Submitted to the AAAI-86 Fifth National Conference on Artificial Intelligence, Philadelphia, Pennsylvania.

[4] Davis, Ernest. 1984. "Shape and Function of Solid Objects: Some Examples." Computer Science Technical Report 137, New York University. October 1984.

[5] Hager, Greg. 1985. "Naive physics of materials: A recon mission." In *Commonsense Summer: Final Report*, Report No. CSLI-85-35, Center for the Study of Language and Information, Stanford University.

[6] Hayes, Patrick J. 1979. "Naive physics manifesto." *Expert Systems in the Micro-electronic Age*, ed. by Donald Michie, Edinburgh University Press, pp. 242-270.

[7] Herskovits, Annette. 1982. *Space and the Prepositions in English: Regularities and Irregularities in a Complex Domain.* Ph.D. dissertation, Department of Linguistics, Stanford University.

[8] Hilbert, David. 1902. *The Foundations of Geometry* The Open Court Publishing Company.

[9] Hobbs, Jerry R. 1974. "A Model for Natural Language Semantics, Part I: The Model." Research Report #36, Department of Computer Science, Yale University. October 1974.

[10] Hobbs, Jerry R. 1985a. "Ontological promiscuity." *Proceedings, 23rd Annual Meeting of the Association for Computational Linguistics*, pp. 61-69.

[11] Hobbs, Jerry R. 1985b. "Granularity." *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, California, August 1985. 432-435.

[12] Hobbs, Jerry R. and Robert C. Moore, eds. 1985. *Formal Theories of the Commonsense World*. Ablex Publishing Corp.

[13] Hobbs, Jerry R. et al. 1985. *Commonsense Summer Final Report*, Report No. CSLI-85-35, Center for the Study of Language and Information, Stanford University.

[14] Katz, Jerrold J. and Jerry A. Fodor. 1963. "The structure of a semantic theory." *Language*, Vol. 39 (April-June 1963), 170-210.

[15] Lakoff, G. 1972. "Linguistics and natural logic". *Semantics of Natural Language*, ed. by Donald Davidson and Gilbert Harman, 545-665.

[16] McDermott, Drew. 1985. "Reasoning about plans." *Formal Theories of the Commonsense World*, ed. by Jerry R. Hobbs and Robert C. Moore, Ablex Publishing Corp., 269-318.

[17] Miller, George A. and Philip N. Johnson-Laird. 1976. *Language and Perception*, Belknap Press.

[18] Rieger, Charles J. 1974. "Conceptual memory: A theory and computer program for processing and meaning content of natural language utterances." Stanford AIM-233, Department of Computer Science, Stanford University.

[19] Schank, Roger. 1975. *Conceptual Information Processing*. Elsevier Publishing Company.

[20] Shoham, Yoav. 1985. "Naive kinematics: Two aspects of shape." In *Commonsense Summer: Final Report*, Report No. CSLI-85-35, Center for the Study of Language and Information, Stanford University.

[21] Stickel, M.E. 1982. "A nonclausal connection-graph resolution theorem-proving program." *Proceedings of the AAAI-82 National Conference on Artificial Intelligence*, Pittsburgh, Pennsylvania, 229-233.

[22] Talmy, Leonard. 1983. "How language structures space." *Spatial Orientation: Theory, Research, and Application*, ed. by Herbert Pick and Linda Acredolo, Plenum Press.

[23] Talmy, Leonard. 1985. "Force dynamics in language and thought." *Proceedings from the Parasession on Causatives and Agentivity, 21st Regional Meeting, Chicago Linguistic Society*, ed. by William H. Eilfort, Paul D. Kroeber, and Karen L. Peterson.

[24] Zahn, C. T., and R. Z. Roskies. 1972. "Fourier descriptors for plane closed curves." *IEEE Transactions on Computers*, Vol. C-21, No. 3, 269-281. March 1972.

# SECTION 5: RESEARCH CONTRIBUTIONS

*University of Massachusetts*

David D. McDonald

This is a description of Mumble's approach to natural language generation, excerpted from a technical survey of generation entitled "Natural Language Generation: complexities and techniques," which will appear in Nirenburg (ed.) **Theoretical and Methodological Issues in Machine Translation**, Cambridge University Press, to appear 1986.

## 8. Multi-Level, Description Directed Generation

The principal deficit of the direct replacement approach is its difficulties with grammar, i.e. the awkwardness of maintaining an adequate representation of the grammatical context, or of carrying out grammatically mediated text-level actions such as producing the correct syntactic form for an embedded clause. In other respects, however, the message-directed control flow that drives direct replacement has a great deal to recommend it. Compared with grammar-directed control schemes, message-directed control is more efficient, since every action will contribute to the eventual production of the text. Message-directed control also gives a planner a very clear semantic basis for its communication to the realization component, since the message can be viewed simply as a set of instructions to accomplish specific goals. The question then becomes: is there a way of elaborating the basic, message-directed framework so as to overcome the deficits that plague direct replacement approaches while still keeping the computational properties that have made it attractive?

A number of generation researchers have independently choosen the same solution: to interpose a level of explicitly linguistic representation between the message and the words of the text (McDonald 1975, 1984; Kempen and Hoenkamp 1982; Jacobs 1985; Swartout 1984). They believe that employing a syntactic description of the text under construction is the most effective means of introducing grammatical information and constraints into the realization process, in particular, that it is a better locus for grammatical processing than a separately stated, active grammar.

The specifics of their individual treatments differ, but a common thread is clearly identifiable: Realization is organized as choices made by specialists, where the form of the choice--the output of the specialist--is a linguistic representation of what is to be said, i.e. a structural annotation of the syntactic relations that govern the words (and embedded conceptual elements) to be said, rather than just a list of words. These representations are phrase structures of one or another sort--hierarchies of nodes and constituents--of essentially the same kind that a theoretical linguist would use. They employ functional terms like "subject" and "focus", and are most aptly characterized as a kind of "surface structure" in the generative linguist's sense, e.g. they undergo no derivation, and are a proper and complete description of the syntactic properties of the text that is produced.

It will be convenient to restrict the present discussion to only one examplar of this approach; taking advantage of an author's prerogative, I will describe my own (c.f. McDonald 1984; McDonald & Pustejovsky 1985; McDonald, Pustejovsky & Vaughan 1986). As it is the historical outgrowth of a direct replacement system,[1] it will be useful to organize the discussion in terms of how it extends that approach and addresses its deficits. This will

[1] This author's interest in natural language generation began in 1971 while he was working on extentions to the grammar and parser in Winograd's SHRDLU program. As already discussed, SHRDLU employed a classic direct replacement technique for its generation. It was observations of the shortcomings of that design that were the original motivation for the research. The influences of systemic grammar and data-directed programming style also stem from that time.

be folded into the standard description of how it deals with the three general concerns one should have in examining a generation system: how it organizes its knowledge of grammar; what its control structure is; and what its approach to realization is.

Referring to our approach as "multi-level, description-directed generation" emphasizes specific features of its architecture and control protocols that we consider important; it is, however, too large a phrase to use conveniently. The name of the computer program that implements the design, MUMBLE (McDonald 1977, 1983), will serve as a compact, agentive reference. Characterizing MUMBLE as multi-level draws attention to the fact that it carries out operations over three explicitly represented levels of representation simultaneously: message, surface structure, and word stream. Description-directed is the name we have given to its control protocol, which is a specialization of the common programming technique known as data-directed control. Under this protocol, the data in the representations at the three levels is interpreted directly as instructions to the virtual machine that constitutes the generator proper. Since each of these representational structures is also a valid description of the text at its own level of abstraction and theoretical vocabulary, this characterization of the protocol emphasizes the fact that the particulars of how the person developing messages or syntactic structures chooses to design them has immediate consequences for the generator's performance (McDonald 1984). The feedback that this gives a developer has proven to be invaluable in refining the notations and their computational interpretations in all parts of the system.

MUMBLE's virtual machine is the embodyment of our computational theory of generation. It consists of three interleaved processes that manage and carry out the transitions between the representational layers. (1) Phrase structure execution interprets the surface structure, maintaining an environment that defines the grammatical constraints active at any moment, and producing the word stream as its incremental output. (2) Attachment interprets the message, transferring its component units to positions within the surface structure according to the functional relationships between them and their role in the message. (3) Realization takes the individual elements of the message into surface structure phrases by selecting from linguistically motivated classes of parameterized alternative forms. A minor fourth process, operating over the word stream, morphologically specalizes individual words to suit their syntactic and orthographic contexts (e.g. the article"a" going to "an" before vowels); later versions of MUMBLE that produce speech should be much more active at this level.

Thus, as seen by the developer of a text planner that would pass messages to MUMBLE for it to produce texts from, the virtual machine appears as a very high level, task-specific language, with its own operators and intermediate representations. To a lesser extent this is true also for the linguist writing generation-oriented grammars for MUMBLE to execute, since the virtual machine includes no presumptions as to what specific syntactic categories, functional relations, or syntactic constructions the natural language includes. Instead it supplies a notation for defining them in terms of primitive notions including the dominates and proceeds relations of phrase structure, bound thematic relations, configural regularities such as head or complement from X-bar theory; and the tree combination rules of Tree Adjoining Grammars (Kroch & Joshi, 1985).

As a message-directed design, MUMBLE is best discussed by reference to a concrete example message, situation, and resulting output text. To miminize the distraction that introducing an actual underlying program from one of our generation projects would entail, a relatively obvious excerpt from a message will have to suffice. The figure shows a generated output paragraph describing a legal case from the UMass Counselor Project (McDonald & Pustejovsky 1986). The structure below it is the message responsible for its second sentence, which details the events that were relevant to the court's decision. Using this example, we will look at MUMBLE's knowledge of grammar: how it is manifest, and how it has its effects, interleaving discussion of realization and control at convenient places.

"In the Telex case, Telex was sued by IBM for misappropriating trade secrets about its product Merlin. One of the managers of the Merlin development project, Clemens, left IBM to work for Telex, where he helped to develop Telex's competing product, the 6830. The key fact in the case was that Clemens brought a copy of the source code with him when he switched jobs. The court held for IBM."

```
(temporal-sequence
  (left-to-work-for (*‹role *‹project-manager Merlin›› *‹Clemens›)
                    (named-company *‹IBM›)
                    (named-company *‹Telex›))
  (helped-to-develop (named-person *‹Clements›)
                     (*‹kind product› *‹competition-by *‹Telex››
                                      *‹name "6830"›) ))
```

As previously discussed, one of the concomitant features of a message-directed approach is that items[2] directly from the underlying program are part of the messages. (These are indicated here by enclosing angle brackets, *‹...›.) Once in a message, such items become instructions to the generator, and as such need interpretations, i.e. associated functions from the item, and the linguistic and pragmatic environment, to the surface specification of some text or text fragment. However, considered in terms of the space of texts that might realize them, real program objects are large and vague as present day programmers tend to use them: they stand in many different relationships to other objects and to the underlying program's state, and consequently can have many different interpretations depending on the context and the speaker's intent.

We take it to be part of the job of a text planner to choose among these relationships and to indicate in the message the perspective from which an object is to be viewed. (The perspective on the first occurance of Clemens, for example, is indicated to be his role as (former) manager of the Merlin project.) Adopting a specific perspective often amounts to selecting a specific wording (often just of the lexical head, e.g. "manager"; but also entire conventional phrases such as "leave ‹employer1› to work for ‹employer2›"). These examples indicate that many of the terms in a message are surface lexical relations (e.g. "helped to develop") rather than a more abstract conceptual vocabulary; this has the deliberate corollary that syntactic realization will usually occur <u>after</u> key words have been chosen. The text planner must therefore understand a good deal about how alternative word choices cover the semantic fields of the situation it is trying to communicate, and what emphasis and what presupposed inferencing by the audience a given choice of wording will convey. This appears to us to be a choice that is best made at a conceptual level (i.e. during message construction), since it does not depend in any crutial way on the details of the grammatical environment, the arguments of Danlos (1984) notwithstanding (cf. McDonald et al. 1986).

Even though the key lexical choices for an item will have occurred before it has been syntactically realized, these message-level lexical decisions can draw on the grammatical context in which the text for it is going to occur. In particular, grammatical constraints imposed by the syntactic relations in which the text will stand will filter out grammatically

[2] The word "item", and at other times the word "object", is intended as a general term that denotes representational data structures in an underlying program without regard to the kind of real world entity that they model: individuals, kinds, relations, constraints, attributes, states, actions, events, etc.

inconsistent possibilities from the planner's choice set.[3] This is possible because the realization of messages is hierarchical, following the message's compositional structure top down, i.e. the message is interpreted much as a conventional program would be. The surface syntactic realization of the higher, dominating conceptual elements of the message is thus available to define and constrain the interpretations (i.e. linguistic realizations) of the lower, more embedded elements. This protocol for "evaluation" of arguments is known as normal order, and is in direct contrast with the previously discussed applicative order protocol used in most direct replacement designs.

The perspective that the text planner chooses to impose on an item from the underlying program is represented at the message-level by designating the realization class to be used for it. Realization classes are MUMBLE's equivalent of the "specialist programs" in direct replacement. They are linguistic entities rather than conceptual, and are developed by the designer of the grammar using control and data structures defined in the virtual machine. New underlying programs are interfaced to MUMBLE by developing a (possibly very minimal) text planner and assigning program items (or item types) to pre-defined realization classes. A relatively self-contained example of a class, "locative-relation", developed originally for use with Jeff Conklin's program for describing pictures of house scenes (see Conklin, 1984) is shown below:

```
(define-realization-class LOCATIVE-RELATION
  :parameters (relation arg1 arg2)
  :choices
    ( (Arg1-is-Relation-Arg2)
      "The driveway is next to the house"
      clause focus(arg1) )
    ( (Arg2-has-Arg1-Relation-Arg2)
      "The house has a driveway in front of it"
      clause focus(arg2) )
    ( (There-is-a-Arg1-Relation-Arg2)
      "There is a driveway next to the house"
      root-clause shifts-focus-to(arg1) )
    ( (Relation-Arg2-is-Arg1)
      "Next to the house is a driveway"
      root-clause shifts-focus-to(arg1)
      final-position(arg1) )
    ( (with-Arg1-Relation-Arg2)
      "...with a driveway next to it"
      prepp modifier-to(arg1) ))
```

[3] This filtering is automatic if the relevant parts of the text planner are implemented using the same abstract control device as MUMBLE uses for its own decisions, i.e. parameterized, pre-computed annotated choice sets of the sort employed for realization classes (see text). The descriptions of the lingustic character and potential of the choices that the annotation provides are the basis for filtering out incompatible choices on grammatical grounds, just as occurs at the syntactic level in selections within a realization class.

This technique is proving convenient in our own work with some simple text planners; however we can see a point where the requirement that the full set of alternatives be pre-computed may be unnecessarily limiting or possibly psychologically unrealistic, in which case an alternative design, presumably involving dynamic construction of the choices, will be needed and an alternative means of imposing the grammatical constraints will have to be found. For a discussion of another planning-level control paradigm that has been used with Mumble, see Conklin (1984) or McDonald & Conklin (1983).

The choices grouped together in a realization class will all be effective in communicating the conceptual item assigned to the class, but each will be appropriate for a different context. This context-sensitivity is indicated in the annotation accompanying the choice, for example "focus", which will dictate the grammatical cases and surface order given to the arguments, or the functional role "modifier-to", which will lead to realization as a postnominal prepositional phrase. These annotating characteristics indicate the contexts in which a choice can be used. They act both as passive descriptions of the choice that are examined by other routines, and as active test predicates that sample and define the pragmatic situation in the text planner or underlying program. Such terms are the basis of MUMBLE's model of language use--the effects that can be achieved by using a particular linguistic form; as such they play the same kind of role as the "choosers" or the controlling functional features in a systemic grammar like Mann's NIGEL.

The surface structure level, the source of grammatical constraints on realization, is assembled top down as the consequence of the interpretation and realization of the items in the message. In the example message (repeated below), the topmost item is a "sequence" of two steps, each of which is a lexicalized relation over several program objects on which a particular perspective has been imposed.

```
(temporal-sequence
  (left-to-work-for (*‹role *‹project-manager Merlin›› *‹name "Clemens"›)
                    (named-company *‹IBM›)
                    (named-company *‹Telex›))
  (helped-to-develop (named-person *‹Clements›)
                     (*‹kind product› *‹competition-by *‹Telex››
                                      *‹name "6830"›) ))
```

One of the goals of a multi-level approach is to distribute the text construction effort and knowledge throughout the system so that no level is forced to do more of the work than it has the natural capacity for. Thus for example in the interpretation of the first item the message, temporal sequence, MUMBLE is careful to avoid taking steps that would exceed the intent of the planner's instruction by being overly specific linguistically: As a message-level instruction, temporal-sequence says nothing about whether the items it dominates should appear as two sentences or one; it says simply that they occured after one another in time and that their realizations should indicate this. Since there is no special emphasis marked, this can be done by having them appear in the text in the order that they have in the message. The decision about their sentential texture is postponed until a linguistic context is available and the decision can be made on an informed basis.

This delay is achieved by having the Attachment process, which moves items from the message to the surface structure according to their functional roles, wait to position the second item of the sequence until the first has been realized. Only the first item will be moved into the surface structure initially, and it will appear as the contents of the second sentence as shown below. Note that a message item is not realized until it has a position, and then not until all of the items above it and to its left have been realized and the item has been reached by the Phrase Structure Execution process that is traversing the surface structure tree and coordinating all of these activities. By enforcing this discipline one is sure that all the grammatical constraints that could affect an item's realization will have been determined before the realization occurs, and consequently the virtual machine does not need to make provisions for changing an item's realization after it is finished (see figure one).

Considered as a function, a realization class such as "Left-to-work-for" specifies the surface form of a grammatically coherent text fragment, which is instantiated when the class is executed and a specific version of that phrase selected. Given its lexical specificity, such a class is obviously not primitive. It is derived by sucessive specializations of two, linguistically primitive subcategorization frames: one built around the verb class that

includes "leave" (shown below) and the other around the class containing "work for". The specialization is done by a definition-time currying operation wherein arguments to the subcategorization frames are bound to constants (e.g. the verb "leave"), producing new realization classes of reduced arity. On its face, a class built around variants on the phrase "‹employee› leaves ‹company1› to work for ‹company2›" is more appropriate to a semantic grammar (cf. Burton & Brown 1977) than to a conventional syntactic phrase structure grammar. This choice of linguistic modularity does however reflect the actual conceptual modularity of the underlying program that drives the example,[4] and we believe this is an important benefit methodologically.

```
(define-phrase subject-verb-locative (subj vb loc)
   :specification (clause
                    subject subj
                    predicate (vp
                                verb vb
                                locative-complement loc )) )
```

Comparing MUMBLE's organization of grammatical knowledge with that of the two grammar-directed approaches that have been discussed, we see that it resembles an ATN somewhat and a NIGEL-style systemic grammar hardly at all. ATN designs are based on procedurally encoded surface structures, which are executed directly; MUMBLE represents surface structure explicitly and has it interpreted. ATNs select the surface form to be used via a recursive, phrase by phrase, topdown and left to right consideration of the total set of forms the grammar makes available (i.e. alternative arc sequences), and queries the state of the underlying program to see which form is most appropriate. MUMBLE also preceeds recursively, topdown and left to right, but the recursion is on the structure of an explicitly represented message. Conceptual items or item types, through the the realization classes that the planner associates with them, control the selection and instantiation of the appropriate surface forms directly.

MUMBLE "packages" linguistic relations into constituent phrases; it does not provide an unbundled, feature-based representation of them as a systemic grammar does. It cannot, for example, reason about tense or thematic focus apart from a surface structure configuration that exhibits them. This design choice is deliberate, and reflects what we take to be a strong hypothesis about the character of linguistic knowledge. This hypothesis is roughly that the space of valid feature configurations (to use systemic terms) is smaller, less arbitrary, and more structured than a feature-heap notation can express (see McDonald et al. 1986 for details). Since our notation for surface structure incorporates functional annotations as well as categorical, and especially since it is only one of three representational levels operated over in coordination, we believe that organizing linguistic reasoning in terms of packaged, natural sets of relations will provide a great deal of leverage in research on text planning and computational theories of language use and communicative intention.

Nowhere in MUMBLE is there a distinct grammar in the sense of a set of rules for deriving linguistic forms from primitive features. Rather it manipulates a collection of

[4] As it happens, Leave-to-work-at is a primitive conceptual relation in the legal reasoning system that serves here as the underlying program (Rissland & Ashley, submitted). The causal model that the phrase evokes in a person, i.e. that working for the new company is the reason why the employee is leaving (cf. "John washed his car to impress his girlfriend") is encapsulated in this relation, and suppresses the causal model from consideration by the legal reasoner's rules. This encapsulation is deliberate. Reasoning systems should function at the conceptual level best suited to the task. This does however imply that some component of the natural language interface must now bridge the conceptual ground between the internal model and the lexical options of the language; see Pustejovsky (this volume) for a discussion of how this may be done.

predefined linguistic objects--the minimal surface phrases of the language and the composite phrases derived from them. The phrases are grouped into the realization classes, the projected linguistic images of different conceptual types and perspectives. When selected and instantiated to form the surface structure they take on an active role (through interpretation by the three processes), defining the order of further actions by the generator, defining the contraints on the realization of the embedded items from the message now at some of its leaf positions, and defining the points where it may be extended through further attachments from the message level. The figure below shows a snapshot of the surface structure for the first part of the text in the example, and can illustrate these points. At the moment of this snapshot, the Phrase Structure Execution process has traversed the structure up to the item *‹telex› and produced the text shown; its next action will be to have that item realized, whereupon the realizing phrase (an NP like the one for *‹IBM›) will replace *‹telex› in the surface structure and the process will traverse it and move on (see figure two).

The first thing to consider is the differences in the details of this surface structure representation compared with the more conventional trees used by generative grammarians. Two of these are significant in this discussion. The first is the presence of functional annotations over each of the constituents (indicated by labels inside square brackets). Terms like "subject" or "prep-complement" are used principally to summarize the grammatical relations that the constituents are in by warrant of their configurational positions, which makes these labels the source of most of the grammatical constraints on message item realizations. The functional annotations also play a role in the dynamic production of the word stream: Here this includes providing access to the subject when the morphological process needs to determine the person/number agreement for tensed verbs, and supplying grammatical function words like "of" or the infinitive marker "to" directly into the word stream.[5]

Formally the representation is not a tree but a sequential stream (as indicated by the arrows): a stream of annotated positions that are interpreted, in order, as instructions to the Phrase Structure Execution process. The grammar writer defines the interpretation an annotating label is to have, e.g. specifying control of morphological effects or function words, constraints to be imposed on realizations, or establishing salient reference positions (like the subject). Various useful technical details are expedited by defining the surface structure as a stream rather than a tree (see McDonald & Pustejovsky 1985b). The stream design provides a clean technical basis for the work of the Attachment process, which extends the surface structure through the addition of successive items from the message. The extensions are integrated into the active grammatical environment by breaking interposition links in the stream and kniting in the new items along with any additional covering syntactic nodes or functional constituent positions needed to correctly characterize the linguistic relationship of the new material to the old.

In the present example, the second item of the message's temporal sequence item, the lexicalized relation "helped-to-develop", remains unattached--its position in the surface

---

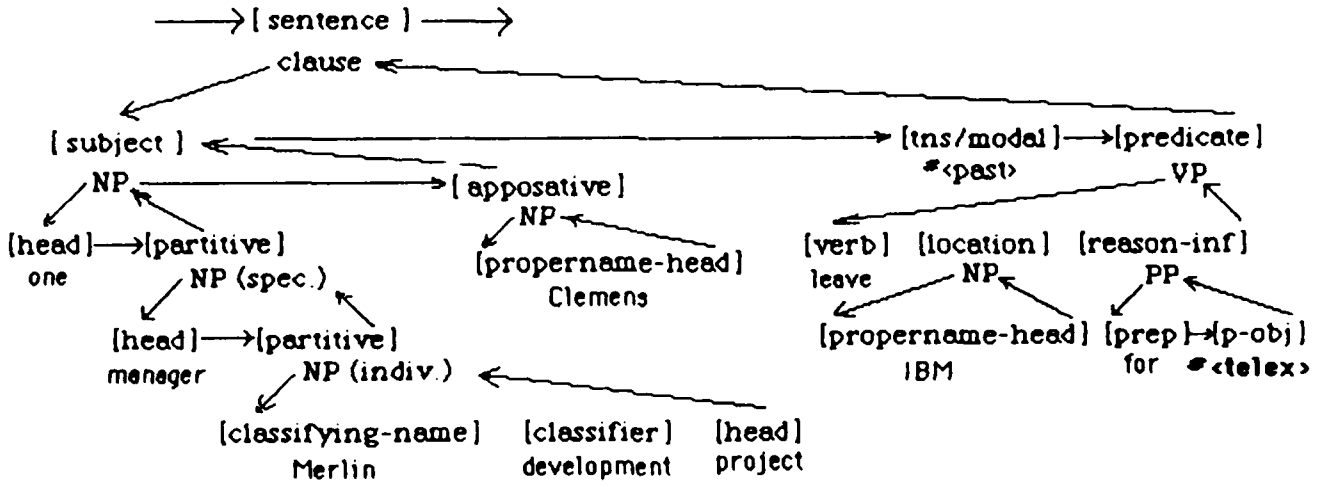
5 Introducing the closed class words that indicate syntactic function into the text as an active consequence of traversing the corresponding part of the surface structure tree, rather than having them first appear in constituent positions at the tree's leaves, is an experimentally motivated design decision. It is intended to explore the consequences of employing computational grammars that distinguish the sources of closed and open class words: positing that the open class words have a conceptual source and the closed class "function" words a purely syntactic source. The two word classes are distinguished psycholinguistically, e.g. they have very different behaviors in exchange errors (see Garrett 1975); if this empirical difference can be given a successful computational account, then that account can serve to anchor other aspects of the grammar's design and eventually lead to psycholinguistic predictions derived from the consequences of the computational design (McDonald 1984).

structure unestablished--until enough linguistic context has been established that a reasonable decision can be made about stylistic matters, e.g. whether the item should appear as an extension of the first item's sentence or start its own. Since the functional constraints on a temporal sequence's realization prohibit embedding the second item anywhere within the first, the only legal "attachment points" for it (i.e. links it could be knit in at) are on the trailing edge of the first item's sentence or as a following sentence. In terms of our theory of generation, attachment points are grammatical properties of phrasal configurations: places where the existing surface structure may be extended by splicing in "auxiliary" phrases (i.e. realizations of message items), for example adding an initial adjunct phrase to a clause or embedding the NP headed by "manager" inside the selector "one of". Every phrasal pattern (as indicated by the annotating labels) has specific places where it can be extended and still be a grammatically valid surface structure; the grammatical theory of such extensions is developed in studies of Tree Adjoining Grammars (Kroch & Joshi 1985).

What attachment points exist is a matter determined by the grammatical facts of the language; which points are actually used in a given situation is a matter of stylistic convention (see McDonald & Pustejovsky 1985a). In this case there is a very natural, compactly realized relationship between the first and second events: the final item in the realization of the first event, the Telex company, happens to be where the second event occurred. As neither clause is particularly complex syntactically, the attachment point that extends the final NP of the first event with a relative clause is taken and the second event knit into the surface structure there, to be realized when that position is reached in the stream.


The first item of the message in a top level position of the surface structure annotated as a 'sentence'

**FIGURE ONE**




**Said so far:**

"... One of the managers of the Merlin development project, Clemens left IBM for //"

**FIGURE TWO**

# TAG's as a Grammatical Formalism for Generation

David D. McDonald and James D. Pustejovsky

March, 1985

CPTM #5

This paper will be presented at and published in The Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics, July 8-12, 1985, University of Chicago.

## 1. Abstract

Tree Adjoining Grammars, or "TAG's", (Joshi, Levy & Takahashi 1975; Joshi 1983; Kroch & Joshi 1985) were developed as an alternative to the standard syntactic formalisms that are used in theoretical analyses of language. They are attractive because they may provide just the aspects of context sensitive expressive power that actually appear in human languages while otherwise remaining context free.

This paper describes how we have applied the theory of Tree Adjoining Grammars to natural language generation. We have been attracted to TAG's because their central operation—the extension of an "initial" phrase structure tree through the inclusion, at very specifically constrained locations, of one or more "auxiliary" trees—corresponds directly to certain central operations of our own, performance-oriented theory.

We begin by briefly describing TAG's as a formalism for phrase structure in a competence theory, and summarize the points in the theory of TAG's that are germaine to our own theory. We then consider generally the position of a grammar within the generation process, introducing our use of TAG's through a contrast with how others have used systemic grammars. This takes us to the core results of our paper: using examples from our research with well-written texts from newspapers, we walk through our TAG inspired treatments of raising and wh-movement, and show the correspondence of the TAG "adjunction" operation and our "attachment" process.

In the final section we discuss extensions to the theory, motivated by the way we use the operation corresponding to TAG's' adjunction in performance. This suggests that the competence theory of TAG's can be profitably projected to structures at the morphological level as well as the present syntactic level.

## 2. Tree Adjunction Grammars

The theoretical apparatus of a TAG consists of a primitively defined set of "elementary" phrase structure trees, a "linking" relation that can be used to define dependency relations between two nodes within an elementary tree, and an "adjunction" operation that combines trees under specifiable constraints. The elementary trees are divided into two sets: initial and auxiliary. *Initial trees* have only terminals at their leaves. *Auxiliary trees* are distinguished by having one non-terminal among their leaves; the category of this node must be the same as the category of the root. All elemental trees are "minimal" in the sense that they do not recurse on any non-terminal.

A node N1 in an elementary tree may be linked (co-indexed) to a second node N2 in the same tree provided N1 c-commands N2. Linking is used to indicate grammatically defined dependencies between nodes such as subcategorization relationships or filler-gap dependencies. Links are preserved (though "stretched out") when their tree is extended through adjunction; this is the mechanism TAG's use to represent unbounded dependencies.

Sentence derivations start with an initial tree, and continue via the adjunction of an arbitrary number of auxiliary trees. To adjoin an auxiliary tree A with root category X to a initial (or derived) tree T, we first select some node of category X within T to be the point at which the adjunction is to occur. Then (1) the subtree of T dominated by that instance of X (call it X′) is removed from T, (2) the auxiliary tree A is knit into T at the position where X′ had been located, and (3) the subtree dominated by X′ is knit into A to replace the second occurence of the category X at T's frontier. The two trees have now been merged by "splicing" A into T, displacing the subtree of T at the point of the adjunction to the frontier of A.

For example we could take the initial tree:

$[_{S'}$ Who$_i$ does $[_S$ John like $e_i$ ] ]

(the subscript "i" indicates that the "who" and the trace "e" are linked) and adjoin to it the auxiliary tree:

$[_S$ Bill believes S ]

to produce the derived tree:

$[_{S'}$ Who$_i$ does $[_S$ Bill believe $[_S$ John likes $e_i$ ] ] ]

Adjunction may be "constrained". The grammar writer may specify which specific trees may be adjoined to a given node in an elementary tree; if no specification is given the default is that there is no constraint and that any auxiliary tree may be adjoined to the node.

### 2.1 Key features of the theory of TAG's

**A TAG specifies surface structure.** There is no notion of derivation from deep structure in the theory of TAG's—the primitive trees are not transformed or otherwise changed once they are introduced into a text, only combined with other primitive trees. As Kroch and Joshi point out, this means that a TAG is incomplete as an account of the structure of a natural language, e.g. a TAG grammar will contain both an active and a passive form of the same verbal subcategorization pattern, without an theory-mediated description of the very close relationship between them.

To our minds this is by no means a deficit. The procedural machinery that generative grammars have traditionally carried with them to characterize relations like that of active to passive has only gotten in the way of employing those characterizations in processing models of generation. This is because a generation model, like any theory of performance, has a procedural structure of its own and cannot coexist with an incompatible one, at least not while still operating efficiently or while retaining a simple mapping from its actual machine to the virtual machine that its authors put forward as their account of psycholinguistic data.

Our own generator uses surface structure as its only explicitly represented linguistic level. Thus grammatical formalisms that dwell on the rules governing surface form are more useful to us than those that hide those rules in a deep to surface transformational process.

**A TAG involves the manipulation of very small elementary structures.** This is because of the stipulation that elementary trees may not include recursive nodes. It implies that the sentences one sees in everyday usage, e.g. newpaper texts, are the result of many sucessive adjunctions. This melds nicely with a move that we have made in recent years to view the conceptual representation from which generation proceeds as consisting of a heap of very small, redundantly related information units that have been deliberately selected by a text planning process from the total state of the knowledge base at the time of utterance; each such unit will correspond in the final text to a head lexical item plus selected thematic arguments—a linguistic entity that is easily projected onto the elementary trees of a TAG.

**TAG theory includes only one operation, adjunction, and otherwise makes no changes to the elementary trees that go into a text.** This comports well with the indelibility stipulation in our model of generation, since selected text fragments can be used directly as specified by the grammar without the need for any later transformation. The composition options delimited by the constraints on adjunction given with a TAG define a space of alternative text forms which can correspond directly in generation to alternative conceptual relations among information units, alternatives in rhetorical intent, and alternatives in prose style.

## 3. Adapting TAG's to Generation

The mapping from TAG's as a formalism for competence theories of language to our formalism for generation is strikingly direct. Their adjunction operation corresponds to our "attachment process"; their constraints on adjunction correspond to our "attachment points"; their surface structure trees correspond to our surface structure trees.[1] We further hypothesize that two quite strong correspondence claims can be made, though considerably more experimentation and theorizing will have to be done with both formalisms before these claims can be confirmed.

1. The primitive information units in realization specifications can be realized exclusively as one or another elementary tree as defined by a suitable TAG, i.e. linguistic criteria can be used in determining the proper modularity of the conceptual structure.[2]

2. Conversely, for <u>any</u> textual relationship which our generator would derive by the attachment of multiple information units into a single package, there is a corresponding rule of adjunction. Since we use attachment in the realization of nominal compounds like "*oil tanker*", this has the force of extending the domain of TAG analyses into morphology. (See section 7).

---

[1] Our model of generation does not employ the simple trees of labeled nodes that appear in most theoretical linguistic analyses. Our surface structure incorporates the semantic properties of trees, but it also includes reifications of constituent positions like "subject" or "sentence" and is better characterized overall as an "executable sequence of labeled positions". We discuss this further in section 5.1.

[2] If this hypothesis is sucessful, it has very consequential implications for the "size" of the information units that the text planner constructing the realization specification can use, e.g. they would not be realized as texts that include recursive nodes. We will discuss this and other implications in a later paper.

## 4. The Place of Grammar in a Theory of Generation

To understand why we are looking at TAG's rather than some other formalism, one must first understand the role of grammar within our processing model. The following is a brief summary of the model; a more complete description can be found in McDonald & Pustejovsky [1985b].

We have always had two complementary goals in our research: on the one hand our generation program has had to be of practical utility to the knowedge based expert systems that use it as part of a natural language interface. This means that architecturally our generator has always been designed to produce text from conceptual specifications, "plans", developed by another program and consequently has had to be sensitive to the limitations and varying approaches of the present state of the art in conceptual representation.

At the same time, we want the architecture of the virtual machine that we abstract out of our program to be effective as a source of psycholinguistic hypotheses about the actual generation process that humans use; it should, for example, provide the basis for predictive accounts of human speech error behavior and apparent planning limitations. To achieve this, we have restricted ourselves to a highly constrained set of representations and operations, and have adopted strong and suggestive stipulations on our design such as high locality, information encapsulation, online quasi-realtime runtime performance, and indelibility.[3] This restricts us as programmers, but disciplines us as theorists.

We see the process of generation as involving three temporally intermingled activities: (1) determining what goals the utterance is to achieve, (2) planning what information content and rhetorical force will best meet those goals given the context, and (3) realizing the specified information and rhetorical intent as a grammatical text. Our *linguistic component* (henceforth LC), the Zetalisp program MUMBLE, handles the third of these activities, taking a "realization specification"[4] as input, and producing a stream of morphologically specialized words[5] as output.

As described in [McDonald 1984], our LC is a "description-directed" process: it uses the structure of the realization specification it is given, plus the syntactic surface structure of the text in progress (which it extends incrementally as the specification is realized) to directly control its actions, interpreting them as though they were sequential computer programs. This technique imposes strong demands on the descriptive formalism used for representing surface structure. For example, nodes and category labels now designate actions the generator is to take (e.g. imposing scoping relations or constraining embedded decisions) and dictate the inclusion of function words and morphological specializations.

---

[3] "Indelibility" in a computation requires that no action of a process (making decisions, constructing representations, changing state, etc.) can be transparently undone once it has been performed. Many nonbacktracking, nonparallel program designs have this property; it is our term for what Marcus [1980] referred to as the property of being "strictly deterministic".

[4] A realization specification can informally be taken to correspond to what many researchers, particularly psychologists, think of as the "message level" representation of a text.

[5] Which is to say that it presently produces written rather than spoken texts. We expect to work with speech output shortly, however, and the need to support the representational basis of an intonational contour is beginning to influence our designs for constituency patterns in surface structure.

### 4.1 *Unbundling Systemic Grammars*

Of the established linguistic formalisms, systemic grammar [Halliday 1976] has always been the most important to AI researchers on generation. Two of the most important generation systems that have been developed, PROTEUS [Davey 1974] and NIGEL [Mann & Matthiessen 1983], use systemic grammar, and others, including our own, have been strongly influenced by it. The reasons for this enthusiasm are central to the special concerns of generation. Systemic grammars employ a functional vocabulary: they emphasize the uses to which language can be put—how languages achieve their speakers' goals—rather than its formal structure. Since the generation process begins with goals, unlike the comprehension process which begins with structure, this orientation makes systemic grammars more immediately useful than, for example, transformational generative grammars or even procedurally oriented AI formalisms for language such as ATN's.

The generation researcher's primary question is why use one construction rather than another—active instead of passive, "the" instead of "a". The principle device of a systemic grammar, the "choice system", supports this question by highlighting how the constructions of the language are grouped into sets of alternatives. Choice systems provide an anchoring point for the rules of a theory of language use since it is natural to associate the various semantic, discourse, or rhetorical criteria that bear on the selection of a given construction or feature with the choice system to which the construction belongs, thus providing the basis of a decision-procedure for selecting from its listed alternatives; the NIGEL system does precisely this in its "chooser" procedures.

*In our formalism we make use of the same information as a systemic grammar captures, however we have choosen to bundle it quite differently.* The underlying reason for this is that our concern for psycholinguistic modeling and efficient processing takes precedence in our design decisions about how the facts of language and language use should be represented in a generator. It is thus instructive to look at the different kinds of linguistic information that a network of choice systems carry. In our system we distribute these to separate computational devices.

- Dependencies among structural features: A generator must respect the constraints that dependencies impose and appreciate the impact they have on its realization options: for example that some subordinate clauses can not express tense or modality while main clauses are required to; or that a pronominal direct object forces particle movement while a lexical object leaves it optional.
- Usage criteria. The decision procedures associated with each choice system are not a part of the grammar per se, although they are naturally associated with it and organized by it. Also most systemic grammars include very abstract features such as "generic reference" or "completed action", which cross-correlate the language's surface features, and thus are more controllers of why a construct is used rather than constructs themselves.
- Coordinated structural alternatives. A sentence may be either active or passive, either a question or a statement. By grouping these alternatives into systems and using these systems exclusively when constructing a text, one is guaranteed not to combine inconsistent structural features.
- Efficient ordering of choices. The network that connects choice systems provides a natural path between decisions, which if followed strictly guarentees that a choice will not be made unless it is required, and that it will not be made before any of the choices that it is itself dependent upon, insuring that it can

be made indelibly.

- Typology of surface structure. Almost by accident (since its specification is distributed throughout all of the systems implicitly), the grammar determines the pattern of dominance and constituency relationships of the text. While not a principle of the theory, the trees of clauses, NPs, etc. in systemic grammars tend to be shallow and broad.

We believe, but have not yet established, that equivalence transformations can be defined that would take a systemic grammar as a specification to construct the alternative devices that we use in our generator (or augment devices that derive from other sources, e.g. a TAG) by decomposing the information in the systemic grammar along the lines just listed and redistributing it.

## 5. Example Analyses

One of the task domains we are currently developing involves newspaper reports of current events. We are "reverse engineering" leading paragraphs from actual newspaper articles to produce narrow but complex conceptual representation, and then designing realization specifications—plans—that will lead our LC to reconstruct the original text or motivated variations on it. We have adopted this domain because the news reporting task, with its requirement of communicating what is new and significant in an event as well as the event itself, appears to impose exceptionally rich constraints on the selection of what conceptual information to report and on what syntactic constructions to use in reporting it (see discussion in Clippinger & McDonald [1983]). We expect to find out how much complexity a realization specification requires in order to motivate such carefully composed texts; this will later guide us in designing a text planner with sufficient capabilities to construct such specifications on its own.

Our examples are drawn from the text fragment below (Associated Press, 12/23/84); the realization specification we use to reproduce the text follows.

---

*"LONDON - Two oil tankers, the Norwegian-owned Thorshavet and a Liberian-registered vessel, were reported to have been hit by missiles Friday in the Gulf.*

*The Thorshavet was ablaze and under tow to Bahrain, officials in Oslo said. Lloyds reported that two crewmen were injured on the Liberian ship."*

```
(the-day's-events-in-the-Gulf-tanker-war
    :events-require-certification-as-to-source
    (main-event #<same-event-type_varying-patient
                    #<hit-by-missiles Thorshavet>
                    #<hit-by-missiles Liberian> >
                :unusual  #<number-of-ships-hit 2>
                :identify-the-ships  )
    (particulars #<damage-report Thorshavet Oslo-officials>
                 #<damage-report Liberian Lloyds>  ))
```

**Figure 1**

---

This realization specification represents the structured object which gives the toplevel plan for this utterance. Symbols preceded by colons indicate particular features of the utterance. The two expressions in parentheses are the content items of the specification and are restricted to appear in the utterance in that order. The first symbol in each expression is a label indicating the function of that item within the plan; embedded items appearing in angle brackets are information units from the current-events knowledge base.

Obviously this plan must be considerably refined before it could serve as a proximal source for the text; that is why we point out that it is a "toplevel" plan. It is a specification for the general outline of the utterance which must be fleshed out by recursive planning once its realization has begun and the LC can supply a linguistic context to further constrain the choices for the units and the rhetorical features.

For present purposes, the key fact to appreciate about this realization specification is how different it is in form from the surface structure. One cannot produce the cited text simply by traversing and "reading out" the elements of the specification as though one were doing direct production. Structural rearrangements are required, and these must be done under the control of constraints which can only be stated in linguistic vocabulary with terms like "subject" or "raising".

The first unit in the specification, #<same-event-type...>, is a relation over two other units. It indicates that a commonality between the two has been noticed and deemed significant in the underlying representation of the event. The present LC always realizes such relations by merging the realizations of the two units. If nothing else occurred, this would give us the text *"Two oil tankers were hit by missiles"*.

As it happens, however, a pending rhetorical constraint from the realization specification, :events-require-certification-as-to-source will force the addition of yet another information unit,[6] the reporting event by the news service that announced the aledged event (e.g. a press release from Iraq, Reuters, etc.). In this case the "content" of the reporting event is the two damage-reports which have already been planned for inclusion in the utterance as part of the "particulars" part of the specification. Let us look closely at how that reportiing event unit is folded into surface structure.

When not itself the focus of attention, a reporting event is typically realized as "so-and-so said X", that is, the content of the report is more important than the report itself; whatever significance the report or its source has as news will be indicated subtly through which of the alternative realizations below is selected for it.[7]

---

[6] We will not discuss the mechanism by which features in the specification influence realization. Realization specifications of the complexity of this example are still very new in our research and we are unsure whether the process is better organized at the conceptual level directing a composition process within the planning component (during one of the recursive invocations) or within the LC mediating a selection between anticipated alternatives. At this point our design experiments are inconclusive.

[7] These sentences are artificial; actual ones would be considerably longer. Interestingly, certain other syntactically permissable versions such as *"It was reported that"* do not occur in any of the texts we have examined. Perhaps the "lead NP" position is too important to waste on a pronoun.

| Desired characteristic | Resulting text |
|---|---|
| **de-emphasize report** | *Two tankers were hit, Gulf shipping sources said.* |
| **source is given elsewhere** | *Two tankers were reported hit.* |
| **emphasize report** | *Iraq reported it hit two tankers.* |

**Figure 2 Possibilities for expressing report(source, info) in newspaper prose**

In our LC, these alternative "choices" are grouped together into a "realization class" as shown in Figure 3. Our realization classes have their historic origins in the choice systems of systemic grammar, though they are very different in almost every concrete detail. The most important difference of interest theoretically is that while systemic choice systems select among single alternative features (e.g. passive, gerundive), realization classes select among entire surface structure fragments at a time (which might be seen as prespecified realizations of bundles of features). That is, our approach to generation calls for us to organize our decision procedures so as to select the values for a number of linguistic features simultaneously in one choice where a systemic grammar would make the selection incrementally.[8]

[8] The standard technique of using choice systems to control the active selection of utterance features is employed by the most well-known applications of systemic grammars to generation (i.e. the work of Davey [1974] and Mann and Matthiessen [1983]). However very recent work with systemic grammars at Edinburgh by Patten [1985] departs from this technique. Patten uses a semantic-level planning component to directly select groups of features at the rightward, "output", side of a systemic network, and then works backwards through the network to determine what other, not sematically specified features must be added to the text for it to be grammatical; control is thus outside the grammar proper, with grammar rules relegated to constraint specification only. We are intrigued by this technique and look forward to its further development.

```
(define-realization-class  believe-verbs
  : parameters  (agent proposition verb)
  : choices
    (( (AGENT-VERBe-that-PROP agent verb prop)
       clause  focus(agent)  emphasize(self) )
       ; e.g. "Lloyds reports Iraq hit two tankers."
       ; encompasses variations with and without that, and
       ; also tenseless complements like "John believes him
       ; to be a fool."
     ( (raise-VERB-into-PROP  (passivize verb) prop)
       clause  focus((agent prop))  mentioned-elsewhere(agent) )
       ; "Two tankers were reported to have been hit"
     ( (it-VERB-PROP  verb prop)
       clause  inferable(agent) )
       ; e.g. "It is reported that 2 tankers were hit."
     ( (left-dislocated-PROP  agent verb prop)
       clause  de-emphasize(self) )
       ; "Two tankers were hit, Gulf sources said."
    ))
```

**Figure 3 Realization class assigned to report(Iraq,hit(...)**

Returning to our example, we are now faced now with the need to incorporate a unit denoting the report of the Iraqi attacks into the utterance to act as a certification of the #<hit-by-missiles> events. This will be done using the realization class believe-verbs; the class is applicable to any information unit of the form **report(source, info)** (and others). It determines the realization of such units both when they appear in isolation and, as in the present case, when they are to augment an utterance corresponding to one of their arguments.

From this realization class the choice raise-VERB-into-PROP will be selected since (1) the fact that two ships were hit is most significant, meaning that the focus will be on the information and not the source (n.b. when the class executes the source Iraq will be bound to the agent parameter and the information about the missile hits to the proposition parameter); (2) there is no rhetorical motivation for us to occupy space in the first sentence with the sources of the report since they have already been planned to follow. These conditions are sensed by attached procedures associated with the characteristics that annotate the choice (i.e. focus and mentioned-elsewhere).

Since the PROP is already in place in the surface structure tree, the LC will be interpreting raise-VERB-into-PROP as a specification of how it may fold the auxiliary tree for *reported* into the tree for *Two oil tankers were hit by missiles Friday in the Gulf*. This corresponds to the TAG analysis in Figure 4 [Kroch & Joshi 1985].

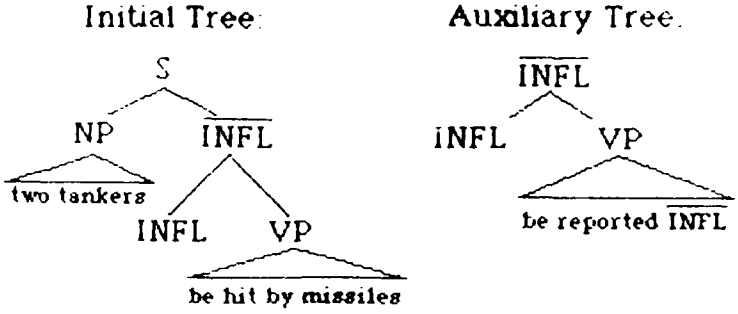



**Figure 4 Initial and auxilliary trees for Raising-to-subject**

The initial tree for *Two oil tankers were hit by missiles*, $\mathbf{I_1}$, may be extended at its INFL´ node as indicated by the constraint given in parenthesis by that node. Figure 5 shows the tree after the auxiliary tree $\mathbf{A_2}$, named by that constraint, has been adjoined. Notice that the original INFL´ of Figure 4 is now in the complement position of report, giving us the sentence *Two oil tankers were reported hit by missiles.*

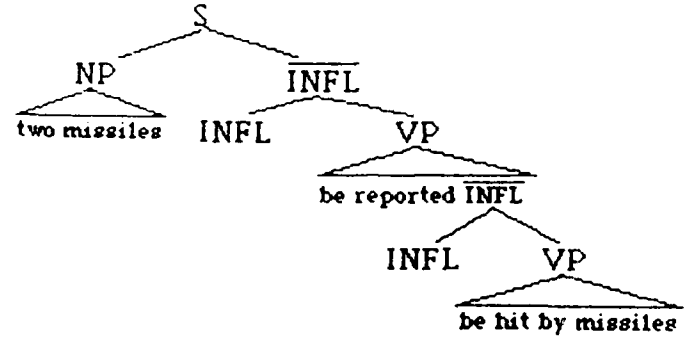



**Figure 5 After embedding *report***

### 5.1 Path Notation

As readers of any of our earlier papers are aware, we do not employ a conventional tree notation in our LC. A generation model places its own kinds of demands on the representation of surface structure, and these lead to principled departures from the conventions adopted by theoretical linguists. Figure 6 shows the surface structure as our LC would actually represent it just before the moment when the adjunction is made.

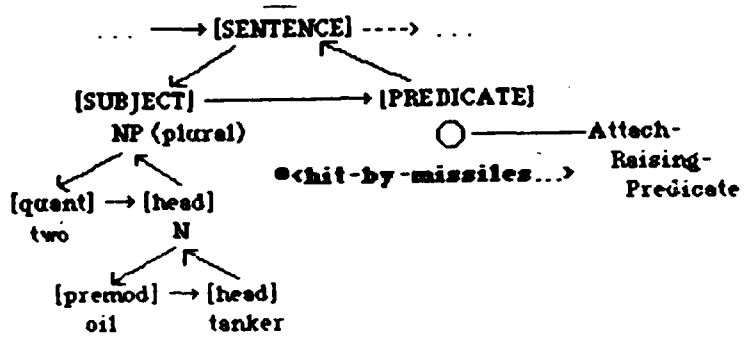



**Figure 6 Surface structure in path notation**

We call this representation *path notation* because it defines the path that our LC follows. Formally the structure is not a tree but a uni-directional linked list whose formation rules obey the axioms of a tree (e.g. any path "down" through a given node must eventually pass back "up" through that same node). The path consists of a stream of entities representing phrasal nodes, constituent positions (indicated by square brackets), instances of information units (in boldface), instances of words, and activated attachment points (the labeled circle under the predicate; see next section). The various symbols in the figure (e.g. **sentence**, **predicate**, etc.) have attached procedures that are activated as the point of speech moves along the path, a process we call "phrase structure execution". Phrase structure execution is the means by which grammatical constraints are imposed on embedded decisions and function words and grammatical morphemes are produced. (For discussion see McDonald [1984].)

Once one has begun to think of surface structure as a traversal path, it is a short step to imagining being able to cut the path and "splice in" additional position sequences.[9] This splicing operation inherits a natural set of constraints on the kinds of distortions that it can perform, since, by the indelibility stipulation, existing position sequences can not be destroyed or rethreaded. It is our impression that these constraints will turn out to be formally the same as those of a TAG, but we have not yet carried out the detailed analyses to confirm this.

[9] The possibility of cutting the surface structure and inserting new sequences that change the linguistic context of positions already in place has been in our theory of generation since 1978, when we used it to implement raising verbs whose rhetorical force was the same as "hedging" adverbs like *possibly*. Our present, much more extensive use of this device as the core of a distinct attachment process dates from the summer of 1984.

### 5.2 Attachment Points

The TAG formalism allows a grammar writer to define "constraints" by annotating the nodes of elementary trees with lists indicating what auxiliary trees may be adjoined to them (including "any" or "none").[10] In a similar manner the "choices" in our realization classes—which by our hypothesis can be taken to always correspond to TAG elementary trees—include specifications of the *attachment points* at which new information units can be incorporated into the surface structure path they define. Rather than being constraints on an otherwise freely applying operation, as in a TAG, attachment points are actual objects interposed in the path notation of the surface structure. A list of the attachment points active at any moment is maintained by the attachment process and consulted whenever an information unit needs to be added. Most units could be attached at any of several points, with the decision being made on the basis of what would be most consistent with the desired prose style (cf. McDonald and Pustejovsky [1985a]). When one of the points is selected it is instantiated, usually splicing in new surface structure in the process, and the new unit added at a designated position within the new structure. Figure 7 shows our present definition of the attachment point that ultimately leads to the addition of "was reported".

```
(define-attachment-point   attach-raising-predicate
   reference-points
      ((present-predicate (slot-contents 'predicate phrase)))
   required-characteristics-of-unit's-realization
      ( raising-verb-with-complement(present-predicate) )
   position-of-attachment-point
      ( (actual-slot 'predicate phrase)
        :attach-under )
      new-phrase-structure
        (submerge-existing-contents-into-new-structure
           (vp-infinitive-complement) ; specification of new phrase
           verb    ; where the unit being attached goes
           infinitive-complement)    ; where the existing contents go
   effect-on-other-pending-attachment-points
      none
   choices-that-introduce-it
      :choices-passing-test (includes-slot 'predicate))
```

**Figure 7 The attachment-point used by *was reported***

[10] Constraints of this sort are an inovation introduced in Kroch & Joshi [1985]. Previous versions of TAG theory allowed "context sensitive" constraint specifications that in fact were never exploited. The present constraints are more attractive formally since they must be stated locally to a single tree.

This attachment point goes with any choice (elementary tree) that includes a constituent position labeled **predicate**. It is placed in the position path immediately after (or "under") that position (see Figure 6), where it is available to any new unit that passes the indicated requirements.

When this attachment is selected, it builds a new VP node that has the old VP as one of its constituents, then splices this new node into the path in its place as shown in Figure 7.

The unit being attached, e.g. the report of the attack on the two oil tankers, is made the verb of the new VP. Later, once the phrase structure execution process has walked into the new VP and reached that verb position, the unit's realization class (**belief-verbs**) will be consulted and a choice selected that is consistent with the grammatical constraints of being a verb (i.e. a conventional variant on the raise-VERB-into-PROP choice), giving us *"was reported"*.

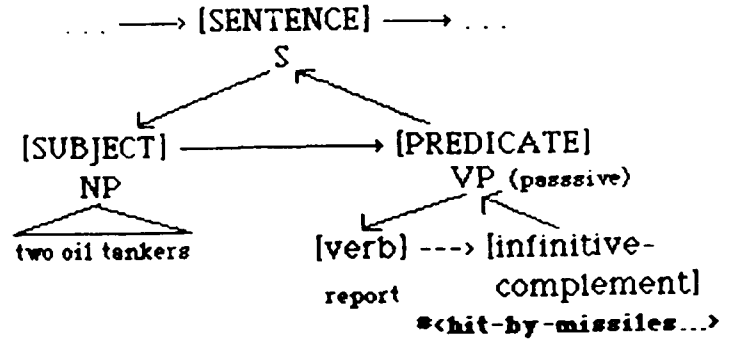



**Figure 8 The path after attachment**

From this discussion one can see that our treatment of attachment uses two structures, an attachment point and a choice, where a TAG would only use one structure, an auxiliary tree. This is a consequence of the fact that we are working with a performance model of generation that must show explicitly how conceptual information units are rendered into texts as part of a psycholinguistically plausible process, while a TAG is a formalism for competence theories that only need to specify the syntactic structure of the grammatical strings of a language. This is a significant difference, but not one that should stand in our way in comparing what the two theories have to offer each other. Consequently in the rest of this paper we will omit the details of the path notation and attachment point definitions to facilitate the comparison of theoretical issues.

## 6. Generating questions using a TAG version of wh-movement

Earlier we illustrated the TAG concept of "linking" by showing how one would start with an initial tree consisting of the *innermost* clause of a question plus the fronted wh-phrase and then build outward by successively adjoining the desired auxiliary phrases to the S node that intervenes between the wh-phrase and the clause. Wh-questions are thus built from the bottom up, as in fact is any sentence involving verbs taking sentential complements.

This analysis has the desirable property of allowing one to state the dependencies between the *Wh*-phrase and the gap as a local relation on a single elementary tree, eliminating the need to include any machinery for movement in the theory. All unbounded dependencies now derive from adjunctions (which, as far as the grammar is concerned, can be made without limit), rather than to the explicit migration of a constituent across clauses.

We also find this locality property to be desirable, and use an analogous procedure in our production of questions and other kinds of *Wh*questions and unbounded dependency constructions.

This "bottom-up" design has consequences for how the realization specifications for these constructions must be organized. In particular, the logician's usual representation of sentential complement verbs as higher operators is not tenable in that role. For example we cannot have the source of, say, *How many ships did Reuters report that Iraq had said it attacked?* be the expression:

**Lambda(quantity-of-ships) . report(Reuters, say(Iraq, attack(Iraq, quantity-of-ships)))**

Such an expression defines a natural *sequence of exposure* when used as realization specification, namely that one realize the **Lambda** operator first, the **report** operator second, the **say** third, and so on. A local TAG analysis of *Wh*-movement requires us to have the **Lambda** and the expression containing its matrix trace, **attach**, be present in a single "layer" of the specification, otherwise we would be forced to violate one of the strong principles of our theory of generation, namely that the characteristics in a realization class may "see" only the immediate arguments of the unit being realized; they may not look "inside" those arguments to subsequent levels of conceptual structure.

This principle has served us well, and we are disinclined to give it up without a very compelling reason. We elected instead to give up the internal representation of sentential complement verb te[illegible] single expressions. This move was easy for us to make since such expressions are awkward to manipulate in the "East Coast" style frame knowledge bases that we use in our own reasoning programs, and we have preferred a representational style with redundant, smaller sized conceptual units for quite some time.

The representation we use instead amounts to breaking up the logical expression into individual units and allowing them to include references to each other.

$$U_1 = \text{lambda(quantity-of-ships) . attack(Iraq,quantity-of-ships)}$$
$$U_2 = \text{say(Iraq, } U_1\text{)}$$
$$U_3 = \text{report(Reuters, } U_2\text{)}$$

Given such a network as the realization specification, the LC must have some principle by which to judge where to start: which unit should form the basis of the surface structure to which the others are then attached? A natural principle to adopt is to begin with the "basis" unit, i.e. the one that does not mention any other units in its definition. We are considering adopting the policy that such units should be allowed only realizations as initial trees while units whose definition involves "pointing to" (naming) other units should be allowed only realizations as auxiliary trees. We have not, however, worked through all of the ramifications such a policy might have on other parts of our generation model; without yet knowing whether it would improve or degrade the other parts of our theory, we are reluctant to assert it as one of our hypotheses relating our generation model to TAG's.

Given that three part source, the realization of the question is fairly straightforward (See Figure 9). The **Lambda** expression is assigned a realization class for clausal *Wh* constructions, whereupon the extracted argument **quantity-of-ships** is placed in COMP, and the body of the expression is placed in the HEAD position. At the same time, the two instances of **quantity-of-ships** are specially marked. The one in COMP is assigned to the realization class for *Wh* phrases appropriate to quantity (e.g. it will have the choice *how many X* and possibly related choices such as <quantity> *of which* and other variants appropriate to relative clauses or other positions where *Wh* constructions can be used). Simultaneously the instance of **quantity-of-ships** in the argument position of the head frame **attack** is assigned to the realization class for *Wh*-trace. These two specializations are the equivalent, in our model, of the TAG linking relation.

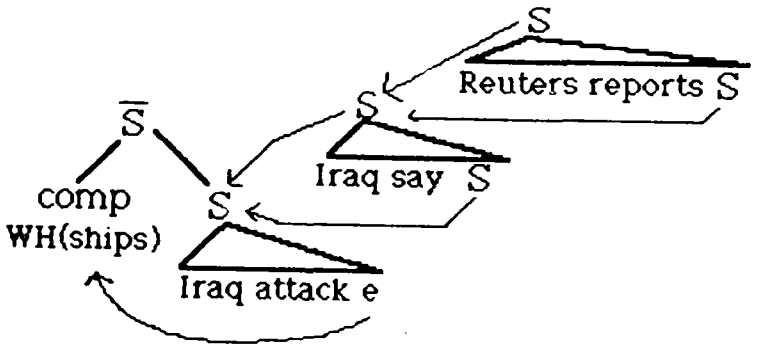



**Figure 9 Question formation with sentential complement verbs**

The two pending units, $U_2$ and $U_3$, are then attached to this matrix, submerging first the **attach** unit and then $U_2$ into complement positions.

## 7. Extensions to the Theory of TAG

Context-free grammars are able to express the word formation processes that seem to exist for natural languages (cf. Williams [1981], Selkirk [1982]). A TAG analysis of such a grammar seems like a natural application to the current version of the theory (cf. Pustejovsky (in preparation)). To illustrate our point, consider compounding rules in English. We can say that for a context-free grammar for word formation, $G_w$, there is a TAG, $T_w$, that is equivalent to $G_w$ (cf. Figures 10 and 11). Consider a fragment of $G_w$ below.[11]

```
N -> N | A | V | P  N
A -> N | A | P   A
V -> P V
```

**Figure 10 CFG Fragment for Word Formation**

The corresponding $G_w$ fragment would be:

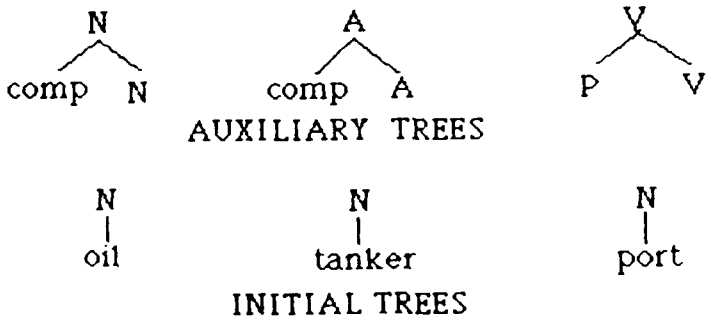



**Figure 11 TAG Fragment for Word Formation**

Now consider the compound , *"oil tanker terminal*, taken from the newspaper reporting domain, and its derivation in TAG theory, shown in Figure 12.

[11] Whether the word formation component should in fact have the power of a TAG or CFG is an open question. Langendoen [1981] discusses the possibility that a finite state grammar might be sufficient for the generative capacity of natural language word formation components.

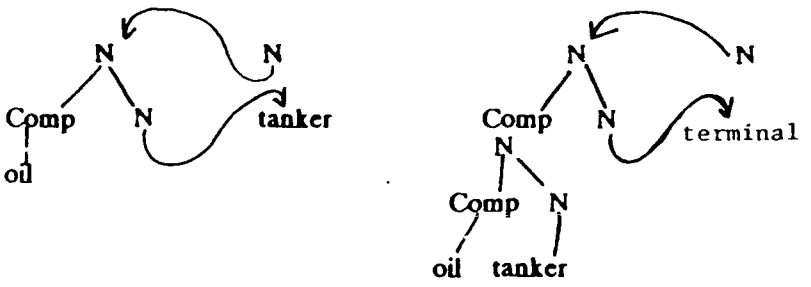



**Figure 12 TAG Derivation of *oil tanker terminal***

Let us compare this derivation to the process used by the LC. The underlying information units from which this compound is derived in our system are shown below. The planner has decided that the units below need to be communicated in order to adequately express the concept. The top-level unit in this bundle is #<terminal>.

$U_1$ = #<terminal>
$U_2$ = #<docks-at $U_1$ $U_3$>
$U_3$ = #<tanker>
$U_4$ = #<carries $U_3$ $U_5$>
$U_5$ = #<oil>

The first unit to be positioned in the surface structure is $U_1$, and appears as the head of an NP. There is an attachment point on this position, however, which allows for the possibility of expressing $U_2$ prenominally. One of the choices associated with this unit is a compound structure—expressed in terms of an auxiliary tree. A snapshot at this point in the derivation shows the following structure.

[N [Comp $U_2$ ] $U_1$ ]

The next unit opened up in this structure is $U_3$, which also allows for attachment prenominally. Thus an auxiliary tree corresponding to $U_4$ is introduced, giving us the structure below:

[N[Comp [Comp $U_4$ ] $U_3$]] $U_1$]

The selectional constraints imposed by the structural positioning of information unit $U_4$ allows only a compounding choice. Had there been no word-level compound realization option, we would have worked our way into a corner without expressing the relation between #<oil> and #<tanker>. Because of this it may be better to view units such as $U_4$ as being associated directly with a lexical compounded form, i.e. *oil tanker*. This partial solution, however, would not speak to the problem of active word formation in the language. Furthermore, it would be interesting to compare the strategic decisions made by a generation system with those planning mistakes made by humans when speaking. This is an aspect of generation that merits much further research.

## 8. Acknowledgements

This research has been supterminaled in part by contract N0014-85-K-0017 from the Defense Advanced Research Projects Agency. We would like to thank Marie Vaughan for help in the preparation of this text.

## 9. References

Clippinger, & McDonald (1983) "Why Good Writing is Easier to Understand", Proc. IJCAI-83, pp. 730-732.

Davey (1974) **Discourse Production**, Ph.D. Dissertation, Edinburgh University; published in 1979 by Edinburgh University Press.

Halliday (1976) **System and Function in Language**, Oxford University Press.

Joshi (1983) "How Much Context-Sensitivity is Required to Provide Reasonable Structural Descriptions: Tree Adjoining Grammars", preprint to appear in Dowty, Karttunen, & Zwicky (eds.) **Natural Language Processing: Psycholinguistic, Computational, and Theoretical Perspectives**, Cambridge University Press.

Kroch, T. and A. Joshi (1985) "The Linguistic Relevance of Tree Adjoining Grammar", University of Pennsylvania, Dept. of Computer and Information Science.

Langendoen, D.T. (1981) "The Generative Capacity of Word-Formation Components", **Linguistic Inquiry**, Volume 12.2

Mann & Matthiessen (1983) *Nigel: A Systemic Grammar for Text Generation*, in Freedle (ed.) **Systemic Perspectives on Discourse**, Ablex.

Marcus (1980) *A Theory of Syntactic Recognition for Natural Language*, MIT Press.

McDonald (1984) "Description Directed Control: Its Implications for Natural Language Generation", in Cercone (ed.) **Computational Linguistics**, Pergamon Press.

McDonald & Pustejovsky (1985a) "SAMSON: a computational theory of prose style in generation", Proceedings of the 1985 meeting of the European Association for Computational Linguistics.

_____ (1985b) "Description-Directed Natural Language Generation", Proceedings of IJCAI-85, W.Kaufmann Inc., Los Altos CA.

Patten T. (1985) "A Problem Solving Approach to Generating Text from Systemic Grammars", Proceedings of the 1985 meeting of the European Association for Computational Linguistics.

Pustejovsky, J. (In Preparation) "Word Formation in Tree Adjoining Grammars"

Selkirk (1982) **The Syntax of Words**, MIT Press.

Williams (1981) "Argument Structure and Morphology" *The Linguistic Review*, 1, 81-114.

# Hypotheticals as Heuristic Device

Edwina L. Rissland and Kevin D. Ashley
Department of Computer and Information Science
University of Massachusetts
Amherst, MA 01003

*Abstract*

In this paper we examine the use of hypotheticals as a heuristic device to assist a case-based reasoner test the strengths, weaknesses, and ramifications of an analysis or argument by exploring and augmenting the space of known cases and indirectly, the attendant spaces of doctrine and argument. Our program, HYPO, works in the task domain of the law, particularly, the area of trade secret protection for software. We describe how HYPO generates a constellation of legally-meaningful hypothetical fact situations ("hypos") which are "near" a given fact situation. This is done in two steps: analysis of the given situation and then generation of the hypos. We discuss the heuristics HYPO currently uses, which include: (1) make a case weaker or stronger; (2) generate an extreme case; (3) generate a near miss; (4) manipulate a near win; and (5) generate a case on a related "dimension".

## 1. Introduction

HYPO is a program to model reasoning with cases and hypotheticals ("hypos"). The program comprises a means of representing and indexing cases in a Case Knowledge Base ("CKB"), a computational definition of relevance in terms of "dimensions" which capture the utility of a case for making a particular kind of argument, a dimension-based method for comparing cases, and methods for generating hypotheticals to help an arguer formulate an argument, gather relevant facts, and explain his argument. HYPO's domain is legal argument where, as illustrated below with examples of oral arguments before the Supreme Court, cases and hypotheticals are primary tools.

In this paper, we concentrate on HYPO's creation of hypothetical new cases to accomplish such tasks as: (1) test the sensitivity of one's argument to absence or presence of certain facts; (2) locate and explore subspaces of relevant cases in the CKB; (3) augment and "flesh out" sparse areas of the CKB; (4) sample the space of implications of a given argument; (5) formulate refinements and refutations of an argument. Thus, we are using hypotheticals as a heuristic device to explore several "spaces" -- the CKB itself, and the spaces of legal doctrine and argument -- and to acquire new case knowledge. HYPO generates these hypotheticals heuristically using certain well-known general heuristics (e.g., examine extreme cases) as well as HYPO-specific ones (e.g., examine weaker/stronger cases along a HYPO dimension).

While HYPO is a program whose primary task domain is legal argument, the lessons learned from HYPO should prove useful for other case-oriented tasks like strategic planning and learning by experimentation. The posing and manipulating of hypotheticals is important in strategic planning where one must examine a proposed plan in light of telling *what if's* -- all too often the advocate of a plan only tells of its good points and and a devil's advocate is needed to unmask its weaknesses. In learning, some of the questions concerning how to *intelligently* select examples as training instances have a large overlap with our concerns here.

[1]This work supported in part by the Advanced Research Projects Agency of the Department of Defense and monitored by the Office of Naval Research under contract no. N00014-84-K-0017

In case-based systems, one cannot afford to wait passively for the "right" case to come along before grappling with a potential problem; one must create cases to reason in anticipation. So too in learning systems, one (i.e., the problem generator) must select or generate cases to drive the learning system. The heuristics we discuss here are the subject of another on-going project of ours on intelligent example selection for rule-learning systems like Buchanan and Fu's RL [1985].

Before going into details, we must mention that in the law there is a distinction between "real" and "hypothetical" cases. A real case is a case that has been litigated and decided; a hypo has not (even though it might be a very slight variation of one that has, or foretells of cases in the process of coming to light or just "waiting to happen") [Rissland, 1985]. Real cases are the basis of our Anglo-American legal system which reasons according to the standard of precedent, or *stare decisis*, which means roughly that like cases should be decided similarly and that one gives support for a legal outcome by citing other similar cases which share the desired conclusion and by distinguishing those that do not [Levi, 1949]. Of course, what counts as "similarity" is often up for grabs and one can apply the idea of precedent "loosely" allowing broad matches and interpretations or "strictly" allowing only narrow ones [Llewellyn, 1930, 1933]. Legal concepts are "open-textured", that is, they cannot be defined in a purely logical way with necessary and sufficient conditions [Hart, 1961; Gardner, 1984]. Further the meaning of concepts (and rules) changes over time, and, in fact, the law can be viewed as a system which learns (in a LEX-like manner) from the cases presented to it [Rissland & Collins, 1986].

These observations apply mostly to "common law" systems, like our own, which reason in a case-based manner. Others, such as the Continental systems (e.g., German or French) rely mostly on rules and to a much lesser degree on cases. However, even in the most rule-like legal orientations, like statute law, one must rely on cases since rarely is a statute so well-defined as to leave no room for ambiguity or interpretation [Levi, 1949].

Note, there are two situations where hypotheticals may actually be preferred over real cases: (1) law school teaching; and (2) aspects of litigation. In law school, hypos are used (sometimes unmercifully) to ferret out unspoken assumptions and prejudices of students, to focus attention on subtle or troublesome points, and to exercise the student's argumentative powers [Gewirtz, 1981; Rissland, 1984]. In litigation, hypos are used primarily at two points: (a) preparation and "debugging" of an argument in the way a strategic planner "dry runs" his plan, and (b) in oral argument. In oral argument, the hypos usually come from the judges trying to probe an advocate's position and the ramifications of it; once in a while, when a hypo is particularly strong or compelling an advocate might recite a hypo in support of his position, or he might present a "counter-example" hypo to refute or limit his opponent's position [Prettyman, 1975; Rissland, in press].

Our model of legal reasoning is based on actual verbatim data from experts, namely the Justices of the United States Supreme Court, on legal jurisprudential scholarship, and on scholarly analysis in legal journals. We have also gathered and analyzed interchanges from law school classes (at Harvard Law School) [Rissland, 1983], and interviews with a few of our own experts on software trade secret law [Werner, Ashley & Stucky, 1986].

## 2. Examples from Supreme Court Oral Arguments

The uses that attorneys and judges make of cases and hypotheticals as tools in argument are illustrated in the oral arguments before the United States Supreme Court. To the chagrin of counsel before the bar of the Supreme Court, the Justices frequently interrupt an attorney's presentation to pose hypotheticals. For example, in *Lynch v. Donnelly*, 104 S. Ct. 1355 (1984), a case involving the constitutionality of a Christmas creche display of a city on municipal land, the Justices posed the following hypotheticals:

To the attorney for the City:

> Q: Do you think ... that a city should display a nativity scene alone *without other* displays such as Santa Claus and Christmas trees...?
>
> Q: [C]ould the city display a cross for the celebration of Easter, under your view?

To the attorney opposing the display:

> Q: [S]supposing the creche were just one ornament on the Christmas tree and you could hardly see it unless you looked very closely, would that be illegal?
>
> Q: What if they had three wiseman and a star in one exhibit, say? Would that be enough? ... What if you had an exhibit that had not the creche itself, but just three camels out in the desert and a star up in the sky?
>
> Q: Well, the city could not display religious paintings or artifacts in its museum under your theory.
>
> Q: There is nothing self-explanatory about a creche to somebody ... who has never been exposed to the Christian religion.
>
> Q: Would the display up on the frieze in this courtroom of the Ten Commandments be unconstitutional then, in your view?
>
> Q: Several years ago ... there was a ceremony held on the Mall, which is federal property of course. ...[T]here were 200,000 or 300,000 people ... and the ceremony was presided over by Pope John Paul II. Would you say that was a step towards an establishment of religion violative of the religion clauses? ... Then you think it would be alright to put a creche over on the Mall? ... How do you distinguish a high mass from a creche? ... [T]here was a considerable involvement of government in that ceremony, hundreds of extra policeman on duty, streets closed... That was a considerable governmental involvement, was it not?

SUP, *Lynch v. Donnelly*, Case No. 82-1256, Fiche No. 5

In the above questions, one can see the Justices modifying the fact situation along various dimensions:

location, size, and focus of display
religious content of the display,
nature of the viewer,
degree of government involvement

Sometimes the purpose of the modifications (and thus the derivative hypos) is to compare the fact situation to actual cases previously decided by the Court to test whether the current case presents stronger or weaker facts.[2] Or a hypothetical case, like the Mall example, may be significant because it did not give rise to litigation.

Frequently, the Justices use hypotheticals to apply pressure to the rule proposed by an attorney for deciding the case. That can be seen in the Mall example above and in the following example from *New Jersey v. T.L.O*, 105 S.Ct. 733 (1985), a case involving the consitutionality of a high school principal's search of a female student's handbag for cigarettes after a teacher reported that she had been smoking in the girls' room. A Justice asked:

> Q: Do you think then that a male teacher could conduct a pat-down search of a young women at age sixteen to find the cigarettes?

In response, the attorney for the state took the position that the Fourth Amendment, which

---

[2]See e.g., *Stone v. Graham*, 449 U.S. 39 (1980): Posting copies of Ten Commandments in schools held unconstitutional; *Gilfillan v. City of Philadelphia*, 637 F. 2d 924 (CA3, 1980): City-financed platform and cross used by Pope John Paul II to celebrate public mass held unconstitutional; *McCreary v. Stone*, 575 F.Supp. 1112 (SDNY 1983): Not unconstitutional for village not to refuse permit to private group to erect creche in public park.

prohibits unreasonable searches by law enforcement authorities, does not apply to high school administrators. The Justice rejoined:

> Q: And does that mean that their authority then to make searches, if the Fourth Amendment is completely inapplicable, extends to any kind of search, strip search or otherwise?
>
> SUP, *New Jersey v. T.L.O*, 1984 Term, Fiche No. 5

In this *T.L.O.* example, the Justices have posed a short but typical "slippery slope" sequence of hypos, where each hypo is successively more extreme than its predecessor, and the culminating "reductio" case (of strip search) is clearly undesireable and suggests refutation of the attorney's position.

Another slippery slope -- this time involving the numerical range of a variable -- can be seen in the following exchange from oral argument from *Sony Corp. v. Universal City Studios*, 464 U.S. 417 (1984). The attorney was advocating the position that if Sony sold video recorders while knowing that consumers would use them to copy copyrighted materials, then Sony should be legally responsible to the owners of the copyrights:

> Q: Suppose ... that about 10 percent of all programming could be copied without interference by the producer or whoever owned the program...
>
> A: I don't think that would make any difference. I think 10% is too small of an amount.
>
> Q: Well, what about 50?

The attorney then asserted even if there were only one television program that was copyrighted, if Sony knew the program would be copied, it should be legally responsible. Finally, the Justice asked:

> Q: Under your test, supposing somebody tells the Xerox people that there are people who are making illegal copies with their machine and they know it. ... Xerox is a contributory infringer?
>
> A: To be consistent, Your Honor, I'd have to say yes.
>
> Q: A rather extreme position.
>
> SUP, *Sony Corp v. Universal City Studios*, Case No. 81-1687, Fiche No. 2

In these last two questions, although the altered fact situations posed by the Justice are still covered by the proposed rule, it is progressively harder for the attorney to justify his position because the hypotheticals present progressively *weaker* facts; the Justice has "stacked" the hypothetical with *extreme* facts. The attorney to keep his argument alive must distinguish the current Sony situation and the hypos. Indeed, the attorney failed. The Court held for Sony on the ground that the Betamax was capable of substantial noninfringing use because so many programs were not subject to copyright restrictions, 464 U.S. 417, 456.

To summarize, the above example illustrate how cases, especially hypotheticals, are used:

- To present, support and attack positions (e.g., by testing the consequences of a tentative conclusion, pressing an assertion to its limits, and exploring the meaning of a concept);
- To relate a fact situtation to significant past cases;
- To augment an existing case base with meaningful test or training cases;
- To factor a complex situation into component parts (e.g., by exagerating strengths, weaknesses or eliminating features);
- To control the course of argument (e.g., by focussing discussion on particular issues)

Such observations translate into our heuristics for using hypotheticals which we discuss after we present some background on the workings of HYPO.

## 3. Background on HYPO: Some definitions.

For the purposes of this research, *cases* are disputes between parties tried by a court, whose decisions are reported in published opinions. The opinion sets forth the facts of the case, the claims made by one party against the other, and the court's holding. *Facts* are statements about events associated with the dispute that were proved at trial or which the court assumed to be true. A *claim* is a recognized kind of complaint for which the courts will grant relief (e.g., breach of contract, negligence, trade secrets misappropriation, copyright infringement). The *elements of a claim* are generalized statements of what facts must be proven in order to prevail on the claim (e.g., the three elements for the existence of a trade secret: "novelty, secrecy, and value in the trade or business of the putative trade secret owner" [Gilburne & Johnson, 1982, p. 215]). The *holding* is the decision of the court as to the legal effect on each claim of the facts of the case, either in favor of the plaintiff or defendant.

In HYPO, cases are represented by a hierarchical cluster of frames (flavor instances) with slots for relevant features (plaintiff, defendant, claim, facts, etc.). Some features are in turn expanded and represented as frames (e.g., plaintiff) [Rissland, Valcarce, & Ashley, 1984]. The library of cases is called the *Case Knowledge Base* (*CKB*). HYPO's current CKB contains a dozen or so of the leading cases for trade secret law for software. See the Appendix Table 1 for a partial list of cases and a very brief indication of their content.

Besides the CKB and the understanding of the legal domain that this case representation implicitly contains, the other major source of domain-specific legal knowledge is in HYPO's *dimensions*. Dimensions capture the notion of legal relevance of a cluster of facts to the merits of a claim: that is, for a particular kind of case, what collections of facts represent *strengths* and *weaknesses* in a party's position. The short answer is that facts are relevant to a claim if there is a court that decided such a claim in a real case and expressly noted the presence or absence of such facts in its opinion. Examples of dimensions in HYPO's area of software trade secret law are: *Secrets-voluntarily-disclosed, Disclosure-subject-to-restriction, Competitive-advantage-gained, Vertical-knowledge.*

Each dimension has several facets:

1. Claims
2. Prerequisites
3. Focal-slots
4. Ranges
5. Direction-to-strengthen-plaintiff
6. Significance
7. Cases-indexed

For instance, the prerequisites of the *Secrets-voluntarily-disclosed* dimension are that two corporations, plaintiff and defendant, compete with respect to a product, plaintiff has confidential product information to which defendant has gained access and plaintiff has made some disclosures of the information to outsiders. The prerequisites are stated in terms of factual predicates, which indicate the presence or absence of a legal fact or attribute (e.g., existence of a product, existence of a non-disclosure agreement). The focal slot of this dimension is the number of disclosees and its range is a non-negative integer. To strengthen the plaintiff's position in a fact situation to which this dimension applies, decrease the number of disclosees; the best case being that with 0 disclosees. The significance of the dimension is that courts have found that the prerequisite facts are a reason for deciding a trade secrets misappropriation claim. This dimension indexes at least two cases in the CKB: *Midland-Ross* in which the court held for the defendant where the plaintiff disclosed the secret to 100 persons, and *Data-General* in which the court held for plaintiff where plaintiff disclosed to 6000 persons. Some of the dimensions relevant to this paper are summarized in the Appendix; HYPO knows about 30 dimensions in all (some of the others are described in [Rissland, Valcarce & Ashley, 1984]). The dimensions were gleaned from law journal articles describing

the state of the (case) law in this area [Gilburne & Johnson, 1982].

The overall flow of information in HYPO is presented in Figure 1. Particularly of interest to us here is HYPO's CASE-ANALYSIS module. In essence, this module works as a diagnostic engine to determine which dimensions apply to a fact situation. The prerequisites, in effect, define antecedent conditions and a dimension (i.e., a possible reason for deciding a claim in a particular way) is the consequent. To make an analogy with the medical domain and MYCIN-like diagnosis, the prerequisite facts are like symptomatic features and the dimensions are like intermediate disease classes. The other modules are described in more detail in [Ashley and Rissland, 1985] and in [Ashley, 1986].

The output of the CASE-ANALYSIS MODULE is the *Case-analysis-record* which contains:

1. applicable factual predicates
2. applicable dimensions
3. near-miss dimensions
4. applicable claims
5. relevant CKB cases
6. conflict examples
7. points-and-responses

The case-analysis-record is used by HYPO's ARGUMENT and HYPO-GEN modules. HYPO's argument task is to generate *3-ply arguments*, which means given the statement of the current facts, (1) side 1 generates a *point* which includes citation of supporting cases, in particular the one[s] HYPO considers the "best" supporting case, abstracting from it the "rule" of that case, and stating how it applies to the current facts; (2) side 2 generates a *response* which might include citation of a best opposing case, refutation of side 1's point with use of a single hypothetical or slippery slope sequence, re-explanation of side 1's best case in a way more in line with side 2's position; and (3) side 1's *counter-response* to side 2's response.

For the remainder of the paper, we concentrate on HYPO's ability to generate hypotheticals.

## 4. Heuristics for Generating Hypotheticals

Basically what HYPO does is to start with a given fact situation, or *seed* case, and generate legally relevant or plausible derivative hypotheticals by modifying the seed case. Since one cannot explore all the "legally" possible (in the sense of syntactic legal move), one needs to explore the space heuristically. Dimensions provide a handle on how to do this exploration in a legally meaningful way.

The process occurs in two steps:
(1) analyze the seed case;
(2) generate legally relevant derivative hypotheticals.

Step one is accomplished by the CASE-ANALYSIS module and results in the case-analysis-record described in the previous section. To recall, this is like a "legal-diagnosis".

Step two is accomplished by the HYPO-GEN module which given high level argument goals (e.g., generate a slippery slope sequence to refute side 1's position), uses the case-analysis-record, and heuristics like the following to generate hypotheticals derived from the seed case:

H1. Pick a near miss dimension and modify the facts to make it applicable.

H2. Pick an applicable dimension and make the case weaker or stronger along that dimension.

H3. Pick a dimension related to one of the applicable dimensions and apply 1 or 2.

H4. Pick an applicable dimension and make the case extreme with respect to that dimension.

H5. Pick a target case that is a win and, using 1 and 2, move the seed case toward it to create a near win.

In order to illustrate these methods, we will use the following hypothetical case, *Widget-King v. Cupcake*, whose facts are as follows:

> Plaintiff Widget-King and defendant Cupcake are corporations that make competing products. Widget-King has confidential information concerning its own product. Cupcake gained access to Widget-King's confidential information. Cupcake saved expense developing its competing product.

The parts of the case-analysis-record for *Widget-King v. Cupcake* that are relevant for the following sections are:

applicable dimensions: *competitive-advantage-gained*
near-miss dimensions: *secrets-voluntarily-disclosed*; *vertical-knowledge*
relevant CKB cases: *Telex v. IBM*

**4.1. Make a near miss dimension apply**

To make a hypothetical out of a fact situation according to this heuristic method, HYPO selects a near miss dimension and "fills in" the missing prerequisites. HYPO instantiates objects and makes appropriate cross references among objects' slots so that the missing factual predicates are satisfied. For example, *secrets-voluntarily-disclosed* would apply to *Widget-King* but for the fact that the confidential information had not been disclosed to anyone. The program instantiates, let us say, five disclosures and sets the subject of the disclosures to be the confidential information. As discussed below, the number of disclosures, five, may be derived from an actual case that the program is considering in the context of making up the hypothetical, or it may be somewhat arbitrarily chosen by the program from within the range of the dimension.

**4.2. Make a case weaker or stronger**

HYPO generates a derivative hypothetical weaker/stronger than the seed case by using the information it knows about dimensions. It can make a case weaker or stronger in two ways: (1) independently of the "caselaw" represented by the CKB; or (2) based on the CKB using a weak form of analogy. To accomplish a CKB-independent strengthening/weakening, HYPO simply changes the values of a focal slot in the manner specified by the direction-to-strengthen slot; the amount of change is somewhat arbitrary. To accomplish a CKB-based modification, for instance to strengthen, HYPO first chooses a case that (a) shares the dimension being manipulated, and (b) is further along the dimension in the stronger direction. HYPO then adjusts the values of the focal slots of the seed in the stronger direction so that the derivative case is stronger than the "precedent" chosen from the CKB. These changes can involve numerical, symbolic or Boolean values. For symbolic values, this means using a partial ordering on values.

Modifications can involve more than one focal slot, for instance a ratio. For example, given our fact situation involving Widget-King and Cupcake which involves some expenditure of money by Widget-King for product development, the *Telex v. IBM* case in the CKB is relevant. In *Telex* the ratio of paintiff's to defendant's expenditures was 2:1 (and the paintiff won). So to strengthen Widget-King's case, change ratio of Widget-King's to Cupcake's expenses to be at least 2:1. An example of such ratio manipulation can also be found in [McCarty & Sridharan, 1981].

Even a simple change in a single numerical focal slot value can have serious legal implcations. Again consider our Widget-King case, as modified by the introduction of 5 disclosees, and make it weaker along the *secrets-voluntarily-disclosed* dimension by using cases from the CKB. HYPO increases the number of Widget-King disclosees from 5 to 150 based on *Midland-Ross* which was decided for the defendant because there were too many disclosees (100) and now Widget-King has passed the 100-disclosee threshold. Note, Widget-King could

still rely on *Data-General* and argue that since the plaintiff won in that case (with 6000 disclosees), it should still win with only 150. HYPO could make the case weaker still by increasing the number of disclosees near or above 6000, the highest value in the CKB or even greater (in a CKB-independent way) to the highest value allowed by HYPO.

There are pros and cons to the two methods. The CKB-independent method is easy to do, but the precedential value of the derivative hypothetical is not predictable. The CKB-based method generates a hypo with known precedents; the drawback is that it can get complicated. HYPO tries to do CKB-based strengthening/weakening first. If it can't (e.g., because no relevant case exists in the CKB), it uses the CKB-independent approach. In either case, the task of actually generating the explanation (as we did above) why the hypo is stronger or weaker belongs to HYPO's EXPLANATION module.

**4.3. Generate a hypo on a related dimension**

The dimensions *disclosures-subject-to-restriction* and *secrets-voluntarily-disclosed* are related; in particular they conflict with one another. Dimensions conflict where there is a particular case to which the dimensions apply and the facts of the case make it strong for the plaintiff on one dimension and weak on the other. Such a case is called a *conflict-example*. Data-General is a conflict-example: it is weak for the plaintiff along the secrets-voluntarily-disclosed dimension (100 disclosees) and strong for the plaintiff along the disclosures-subject-to-restriction dimension (each disclosure subject to nondisclosure agreement). In *Data-General*, the conflict was resolved in favor of the plaintiff.

A hypothetical on a related dimension can be generated by taking the seed case and adding facts sufficient to make the related dimension apply to it in a manner similar to that with heuristic H1. For example, the Widget-King case, as modified by H1 and H2 above, can be further modified so that *disclosures-subject-to-restriction* applies by making all of the disclosures subject to nondisclosure agreements. In this example, the related dimension is also a near miss dimension but that need not always be true.

A hypothetical generated on a conflicting dimension is interesting because it is an example of a case where, at least arguably, facts associated with one dimension can override the effects of the other dimension's facts.

**4.4. Examine an extreme case**

To generate an extreme case, HYPO simply changes the value of a focal slot to be an extreme of its range of values. This can also be done in either a CKB-based or CKB-independent manner. The former method pushes the slot value to the extreme actually existing in a case in the CKB, the latter simply pushes the slot value to its permissible extreme.

For instance, the extreme case on the strongest end of the *secrets-voluntarily-disclosed* dimension for Widget-King is the facts as stated above with the exception that there are 0 disclosees. The other extreme is the maximum value for number of disclosees which in the CKB is 6000 and which in HYPO is 10,000,000.

**4.5. Manipulating a near win**

A near win hypo is one in which a seed fact situation is weak on behalf of, let us say, the plaintiff. It can be "moved" in the direction of a real target case from the CKB that has been decided in favor of the plaintiff. Using methods H1 through H3, HYPO endows the seed situation with the facts to make the case strong for the plaintiff. As a result, the target case becomes relevant to the seed hypothetical and an argument can be made, based on the pro-plaintiff target case, that the hypo should be decided in favor of the plaintiff. Correspondingly a near win hypo can start with a pro-plaintiff fact situation and be moved in the opposite direction away from the pro-plaintiff target case or towards a pro-defendant target case.

For example, consider two cases: *Telex*, which we have already seen above, and *Automated Systems*, where court held in favor of the defendant where the confidential information that

the plaintiff wanted to protect was about a customer's business operations, that is, the knowledge was about a "vertical market". Using the *Telex* case as a seed, and *Automated Systems* as target, HYPO could make *Telex* a near win by making IBM's confidential information be vertical knowledge (i.e., be about a customers business operations). As a result, an argument could be based on *Automated Systems* that, in the hypo, defendant Telex should win.

## 5. Examples of Heuristic HYPO Exploration

HYPO's heuristical y guided generation of hypotheticals makes it possible to explore a fact situation's legal significance in a manner not unlike the sequence of hypotheticals in the creche example from the *Lynch* case oral argument.

Suppose (a) the original *Widget-King* case is modified so that the confidential information is about customer business operations. Suppose on appeal to the Supreme Court, Cupcake's counsel, citing *Automated Systems*, has just argued to the Justices that his client should win because vertical knowledge is not protectible as a trade secret. One can imagine a Justice posing the following line of hypotheticals:

> Q: What never? Suppose (b) Widget-King's alleged trade secret information, eventhough it was vertical knowledge, helped it to produce its competing product in half the time like in the Telex case?
>
> Q: Suppose (c) the vertical knowledge allowed Widget-King to bring its product to market in one fourth the time and at one fourth the expense.
>
> Q: Suppose (d) that Cupcake paid a large sum to a former employee of Widget-King to use the information to build a competing product, as Telex did. Wouldn't the information be protectible as a trade secret then?.

In this example, heuristic methods 1,2,3 and 5 are at work. Near miss dimension vertical-knowledge is used with 1 to create the intial hypo (a). The modification at (b) is produced by 5 and 2 using the the *Telex* case as the target. Method 2 is used to make the stronger hypo at (c). Methods 5, 1 and 2 are used to create the hypo at (d) where the near miss dimension is *common-employee-paid-to-change-employers*.

It is interesting that a previous version of HYPO serendipitously generated a hypothetical very much like this. The starting point was a fact situation presenting a very strong position for the plaintiff along various dimensions: it involved alleged misappropriation of plaintiff's unique, novel technical knowledge about computer system hardware for a particular purpose, knowledge that was not learnable by an employee working for one of the plaintiff's competitor's and that conferred on the plaintiff a year's competitive advantage in bringing its product to market. Then, by accident, the hypo was changed by turning the technical knowledge about hardware into vertical knowledge about bank accounting practices. Although according to the *Automated Systems* case, the new hypo presented a very much weakened position for the plaintiff, it was immediately apparent to the attorney using the program that the *Automated Systems* case was distinguishable -- it did not involve the facts that the knowledge, though vertical, was unique, novel, not learnable elsewhere and conferred a substantial competitive advantage on its possessor -- and suggested the germ of an argument for the protectibility of vertical knowledge -- demonstrate that the vertical knowledge is unique, novel, etc.

Since that accidental discovery, we have provided the system with the above heuristic methods so that, given a case, it can generate a hypo that is distinguishable from the case in a legally significant way. Starting from a real case, methods 3 and 5, in particular, are recipes for creating hypo's with facts that justify a different holding from the real case. Our goal is for the system itself to realize that the hypo is significantly distinguishable and why and to generate such hypos on purpose to make points in an argument.

Having reached step (d) in the above extended example, a hypothetical has been constructed that is fairly strong for the plaintiff. But plaintiff's position can be eroded by moves along

other dimensions. One can imagine the scene at 11 p.m. in the oak-paneled library at 14 Wall Street as two first year associate attorneys, assigned to preparing an initial memorandum as to the strengths of Widget-King's claim against Cupcake, play devil's advocate with the facts:

> Q: Suppose (e) that Widget-King made disclosures to 100 outside persons as in the *Midland-Ross* case.
>
> Q: Well, maybe (f) all of the disclosees entered into nondisclosure agreements as in *Data-General*. Under that case, Widget-King (g) could have made restricted disclosures to as many as 6000 people.
>
> Q: What if (h) Widget-King made restricted disclosures to 10,000,000 people. Is it still a secret? (Not an idle hypothetical in this day of mass marketing of software.)
>
> Q: Are the nondisclosure agreements enforceable? What did all of these people get in exchange for agreeing not to disclose the secret? Suppose (i) that the disclosees did not receive anything of value for entering into the nondisclosure agreements?

With *secrets-voluntarily-disclosed* as near miss dimension and the *Midland-Ross* case as target, the hypo at (e) can be generated from (d) using methods 5, 1 and 2. (f) represents a method 3 move to a conflict dimension, *disclosures-subject-to-restriction*. We assume that the *Data-General* case has been recognized as a conflict-example. Otherwise this could be regarded as a method 5 move with *Data-General* as a target. Using method 4, the hypo at (g) has been moved to the extreme value in *Data-General* and at (h) to the extreme of the range of the dimension. The program does not know that a secret told to 10,000,000 people is not a secret, even if they promise not to tell anyone else, but the program does know that two dimensions conflict and that moving to an extreme on one dimension may cause the conflict to be moot. Having exhausted the possibilities for weakening the case along the secrets-voluntarily-disclosed dimension, the program moves, using methods 1 and 2, to a dimension that became a near miss as soon as nondisclosure agreements came into the hypo at (f), agreement-supported-by-consideration.

One can also analyze the sequence of hypotheticals about the civic creche display from the *Lynch* case oral argument in terms of the dimensional model and heuristics for building hypo's. The justices make the basic fact situation weaker and stronger along a dimension that might be called focus-of-attention: they remove all of the secular images leaving only the religious one, they physically shrink the symbol to an extreme and relegate it to a corner, they remove the religious symbols and leave the secular ones. They weaken plaintiff's case along the dimension of civic-content-message by moving it to a municipal art museum or the frieze of a courtroom. They compare the case along the dimension of government-involvement to an extreme example, the Pope's mass on the Mall.

## 6. Conclusions

In this paper, we have discussed an aspect of reasoning involving the use of hypothetical cases. In particular, we have discussed how our case-based legal reasoning program HYPO currently uses case examples, dimensions, and five or so heuristic methods to compare the legal consequences of facts and to generate hypothetical fact situations to augment and explore its case base. The hypos help accomplish analysis tasks, such as testing the sensitivity of positions and relating a fact situation to significant past cases, and argument tasks, such as generating a slippery slope to refine or refute an argument and controlling the course of argument. HYPO's heuristics involve (1) strengthening/weakening of a case; (2) taking the case to extremes; (3) making a near miss case a winning one; (4) manipulating a near win; and (5) examining a case along a related dimension.

As indicated earlier, one of our performance goals for HYPO is to have HYPO generate 3-ply argument exchanges which involve a heavy dose of case-based reasoning like distinguishing cases and using hypotheticals. Eventually we hope to bring together our descriptive work on argument moves and hypotheticals [Rissland, 1985; Stucky, 1985] with our computational 3-ply argument work. We also hope that this work on HYPO will cross-potentiate with work on the intelligent selection of examples for learning systems, a topic, we feel has been too often glossed over. The heuristic generation of hypotheticals is a step towards both these goals. However even as they now stand, HYPO's current hypothetical reasoning powers can be helpful in formulating, testing, debugging, and learning in case-based tasks.

## References


[1] Kevin D. Ashley. *Modelling Legal Argument: Reasoning with Cases and Hypotheticals – A Thesis Proposal.* Project Memo 10, The COUNSELOR Project, Department of Computer and Information Science, University of Massachusetts, 1986.

[2] Kevin D. Ashley. Reasoning by Analogy: A Survey of Selected A.I. Research with Implications for Legal Expert Systems. In Charles Walter, editor, *Computing Power and Legal Reasoning*, West Publishing Co., St. Paul, MN, 1985.

[3] Kevin D. Ashley and Edwina L. Rissland. Toward Modelling Legal Argument. In Antonio A. Martino and Fiorenza Socci Natali, editors, *Atti preliminari del II Convegno internazionale di studi su Logica Informatica Diritto*, pages 97–108, Consiglio Nazionale delle Ricerche, Istituto per la documentazione giuridica, Florence, Italy, September 1985.

[4] William J. Clancey. Classification Problem Solving. In *Proceedings of the Fourth National Conference on Artificial Intelligence*, American Association for Artificial Intelligence, Austin, TX, August 1984.

[5] A. vdL. Gardner. *An Artificial Intelligence Approach to Legal Reasoning.* PhD thesis, Department of Computer Science, Stanford University, 1984.

[6] Paul Gewirtz. The Jurisprudence of Hypotheticals. *American Bar Association Journal*, 67:864–866, 1981.

[7] M. R. Gilburne and R. L. Johnston. Trade Secret Protection for Software Generally and in the Mass Market. *Computer/Law Journal*, III(3), 1982.

[8] H.L.A. Hart. *The Concept of Law.* Clarendon Press, Oxford, 1961.

[9] Wendy G. Lehnert, David D. McDonald, and Edwina L. Rissland. Natural Language Generation in Battlefield Management, A Proposal for the DARPA Strategic Computing Program. University of Massachusetts, Department of Computer and Information Science, 1984.

[10] Edward H. Levi. *An Introduction to Legal Reasoning.* University of Chicago Press, 1949.

[11] Karl N. Llewellyn. Praejudizienrecht und Rechtssprechung in Amerika. Section 52, Certainty in Case Law; In Doubtful Cases the Legal Rule is not Decisive, pages 72–86. 1933.

[12] K.N. Llewellyn. *The Bramble Bush: On Our Law and Its Study.* Oceana Publications, Dobbs Ferry, NY, 1960 edition, 1930.

[13] L. Thorne McCarty and N.S. Sridharan. *A Computational Theory of Legal Argument.* Technical Report LRP-TR-13, Laboratory for Computer Science Research, Rutgers University, 1982.

[14] L. Thorne McCarty and N.S. Sridharan. The Representation of an Evolving System of Legal Concepts: II. Prototypes and Deformations. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, International Joint Conferences on Artificial Intelligence, Inc., Vancouver, B.C., August 1981.

[15] E. Barrett Prettyman, Jr. Opposing Certiorari in the United States Supreme Court. *Virginia Law Review*, 61:197–209, February 1975.

[16] Edwina L. Rissland. Hypotheticals from Oral Argument before the Supreme Court: an Analysis. 1986. in press.

[17] Edwina L. Rissland. Argument Moves and Hypotheticals. In Charles Walter, editor, *Computing Power and Legal Reasoning*, West Publishing Co., St. Paul, MN, 1985.

[18] Edwina L. Rissland. Hypothetically Speaking: Experience and Reasoning in the Law. In *Proceedings First Annual Conference on Theoretical Issues in Conceptual Information Processing*, Georgia Institute of Technology, Atlanta, GA, March 1984.

[19] Edwina L. Rissland. Examples in Legal Reasoning: Legal Hypotheticals. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, International Joint Conferences on Artificial Intelligence, Inc., Karlsruhe, Germany, August 1983.

[20] Edwina L. Rissland and Robert T. Collins. The Law as a Learning System. Submitted to the 8th Annual Conference of the Cognitive Science Society. 1986.

[21] Edwina L. Rissland, E. M. Valcarce, and Kevin D. Ashley. Explaining and Arguing with Examples. In *Proceedings of the Fourth National Conference on Artificial Intelligence*, American Association for Artificial Intelligence, Austin, TX, August 1984.

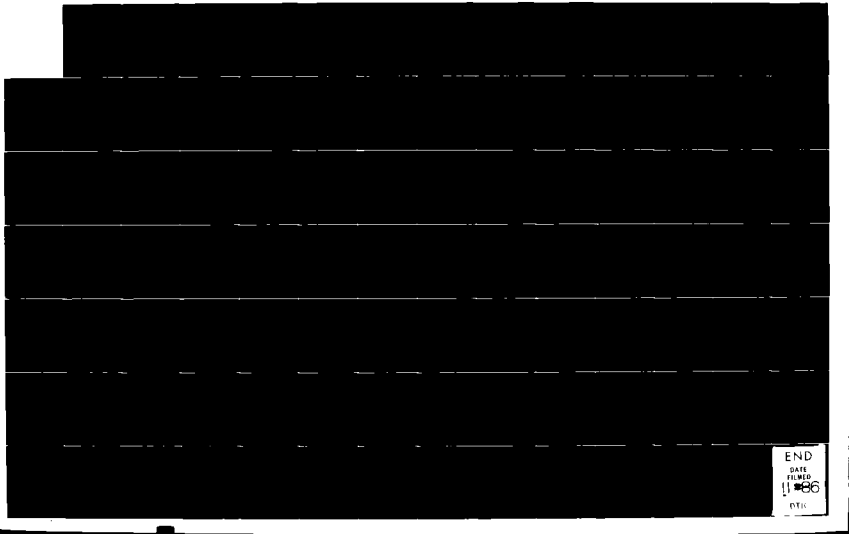
[22] Brian Stucky. *Understanding Legal Argument.* Project Memo 11, The COUNSELOR Project, Department of Computer and Information Science, University of Massachusetts, 1986.

[23] The Complete Oral Arguments of the Supreme Court of the United States. University Publications of America, Frederick, MD.

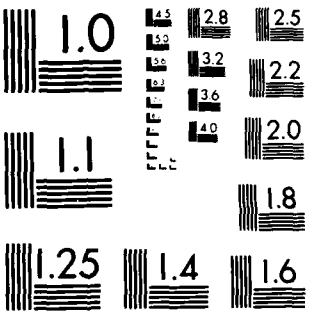
[24] Philip Werner, Kevin D. Ashley, and Brian Stucky. Analyzing Expert Discourse: A Knowledge Acquisition Strategy for Building an Expert Advice Giver. 1986. in preparation.

AD-A172 436 PROCEEDINGS OF THE STRATEGIC COMPUTING NATURAL LANGUAGE WORKSHOP HELD IN..(U) UNIVERSITY OF SOUTHERN CALIFORNIA MARINA DEL REY INFORMATION S.. N K SONDHEIMER MAY 86 3/3

UNCLASSIFIED ISI/SR-86-172 MDA903-81-C-0335 F/G 9/2 NL



END
DATE FILMED
11-86
DTIC




'CROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

## APPENDIX

*Telex Corp. v. IBM Corp.*, 510 F.2d 894 (5th Cir., 1975).
Held for plaintiff IBM on trade secrets misappropriation claim where Telex gained access to IBM's confidential product development information by hiring an IBM employee, paying him a large bonus to develop a competing product. The employee used development notes he brought from IBM. Telex saved time and expense developing the competing product.

*Midland-Ross Corp. v. Sunbeam Equipment Corp.*, 316 F.Supp. 171 (W.D. Pa., 1970).
Held for defendant Sunbeam on trade secrets misappropriation claim where Midland-Ross disclosed it's technical product developyment info to 100 persons.

*Data General Corr v. Digital Computer Controls, Inc.*, 357 A.2d 105 (Del. Ch. 1975).
Held for plaintiff Data General on trade secrets misappropriation claim where Data General disclosed its technical product development info to 6000 persons, all of whom were subject to nondisclosure agreements.

*Automated Systems, Inc. v. Service Bureau Corp.*, 401 F.2d 619 (10th Cir., 1968).
Held for defendant SBC on trade secrets misappropriation claim where Automated-Systems' confidential info was about customer's business operations (i.e., vertical info).

**Table 1: Sample Cases from Case Knowledge Base.**

**Secrets-voluntarily-disclosed:**
Significance: Plaintiff's (P's) position stronger the fewer persons to whom secrets disclosed.
Prerequisites: P and Defendant (D) compete; D had access to P's product information and gained some competitive advantage; some disclosures.
Focal slot: Number of disclosees. To Strengthen P: Decrease number of disclosees. Range: 0 to N. Cases Indexed: *Midland-Ross, Data-General*

**Disclosures-subject-to-restriction:**
Significance: P's position stronger the fewer disclosees not subject to nondisclosure agreements.
Prerequisites: Competition; access to info; some disclosures and nondisclosure agreements.
Focal slot: Number of disclosees subject to restriction. To Strengthen P: Increase percentage of disclosees subject to restriction. Range: 0 - 100 %. Cases Indexed: *Data-General*

**Competitive-advantage-gained:**
Significance: P's position stronger the greater competitive advantage gained by D.
Prerequisites: Competition; access to info; D saved some expense.
Focal slot: Development expense saved. To Strengthen P: Increase expense saved by D. Range: 0 - 100 %. Cases Indexed: *Telex v. IBM*

**Vertical-knowledge:**
Significance: P's position stronger if information technical, not vertical.
Prerequisites: P and D compete; D had access to P's product information; info about something.
Focal slot: What information is about. To Strengthen P: Make information about technical development of product. Range: {technical, vertical} Cases Indexed: *Automated Systems*, et al.

**Table 2: Sample Dimensions.**

## SECTION 6: RESEARCH CONTRIBUTIONS

*University of Pennsylvania*

# LIVING UP TO EXPECTATIONS: COMPUTING EXPERT RESPONSES[1]

Aravind Joshi and Bonnie Webber
Department of Computer and Information Science
Moore School/D2
University of Pennsylvania
Philadelphia PA 19104

Ralph M. Weischedel[2]
Department of Computer & Information Sciences
University of Delaware
Newark DE 19716

ABSTRACT

In cooperative man-machine interaction, it is necessary *but not sufficient* for a system to respond truthfully and informatively to a user's question. In particular, if the system has reason to believe that its planned response might mislead the user, then it must block that conclusion by modifying its response. This paper focusses on identifying and avoiding potentially misleading responses by acknowledging types of "informing behavior" usually expected of an expert. We attempt to give a formal account of several types of assertions that should be included in response to questions concerning the achievement of some goal (in addition to the simple answer), lest the questioner otherwise be misled.

## 1. Introduction]

In cooperative man-machine interaction, it is necessary *but not sufficient* for a system to respond truthfully and informatively to a user's question. In particular, if the system has reason to believe that its planned response might mislead the user to draw a false conclusion, then it must block that conclusion by modifying or adding to its response.

Such cooperative behavior was investigated in [5], in which a modification of Grice's *Maxim of Quality* - "Be truthful" - is proposed:

> If you, the speaker, plan to say anything which may imply for the hearer something that you believe to be false, then provide further information to block it.

This behavior was studied in the context of interpreting certain definite noun phrases. In this paper, we investigate this revised principle as applied to responding to users' plan-related questions. Our overall aim is to:

1. characterize tractable cases in which the system as respondent (R) can anticipate the possibility of the user/questioner (Q) drawing false conclusions from its response and hence alter it so as to prevent this happening;

2. develop a formal method for computing the projected inferences that Q may draw from a

[1]This work is partially supported by NSF Grants MCS 81-07290, MCS 82-05221, and IST 83-11400.

[2]At present visiting the Department of Computer and Information Science, University of Pennsylvania PA 19104.

particular response, identifying those factors whose presence or absence catalyzes the inferences;

3. enable the system to generate modifications of its response that can defuse possible false inferences and that may provide additional useful information as well.

In responding to any question, including those related to plans, a respondent (R) must conform to Grice's first *Maxim of Quantity* as well as the revised *Maxim of Quality* stated above:

> Make your contribution as informative as is required (for the current purposes of the exchange).

At best, if R's response is not so informative, it may be seen as uncooperative. At worst, it may end up violating the revised *Maxim of Quality*, causing Q to conclude something R either believes to be false or does not know to be true: the consequences could be dreadful. Our task is to characterize more precisely what this expected informativeness consists of. In question answering, there seem to be several quite different types of information, over and beyond the simple answer to a question, that are nevertheless expected. For example,

1. When a task-related question is posed to an expert (R), R is expected to provide additional information that he recognizes as necessary to the performance of the task, of which the questioner (Q) may be unaware. Such response behavior was discussed and implemented by Allen [1] in a system to simulate a train information booth attendant responding to requests for schedule and track information. In this case, not providing the expected additional information is simply uncooperative: Q won't conclude the train doesn't depart at any time if P fails to volunteer one.

2. With respect to discussions and/or arguments, a speaker contradicting another is expected to support his contrary contention. Again, failing to provide support would simply be viewed as uncooperative [2, 3].

3. With respect to an expert's responses to questions, if Q expects that R would inform him of P if P were true, then Q may interpret R's silence regarding P as implying P is not true.[3] Thus if R knows P to be true, his silence may lead to Q's being misled. This third type of expected informativeness is the basis for the potentially misleading responses that we are trying to avoid and that constitute the subject of this paper.

What is of interest to us is *characterizing* the Ps that Q would expect an expert R to inform him of, if they hold. Notice that these Ps differ from script-based expectations [8], which are based on what is taken to be the ordinary course of events in a situation. In describing such a situation, if the speaker doesn't explicitly reference some element P of the script, the listener simply assumes it is true. On the other hand, the Ps of interest here are based on normal cooperative discourse behavior, as set out in Grice's maxims. If the speaker doesn't make explicit some information P that the listener believes he would possess and inform the listener of, the listener assumes it is false.

In this paper, we attempt to give a formal account of a subclass of Ps that should be included (in addition to the simple answer) in response to questions involving Q's achieving some goal[4] - e.g., "Can I

---

[3] This is an interactional version of what Reiter [13] has called the "Closed World Assumption" and what McCarthy [9] has discussed in the context of "Circumscription".

[4] A companion paper [6] discusses responses which may mislead Q into assuming some default which R knows not to hold. Related work [4] discusses providing indirect or modified responses to yes/no questions where a direct response, while truthful, might mislead Q.

drop CIS577?", "I want to enrol in CIS577?", "How do I get to Marsh Creek on the Expressway?", etc., lest that response otherwise mislead Q. In this endeavor, our first step is to specify that knowledge that an expert R must have in order to identify the Ps that Q would expect to be informed of, in response to his question. Our second step is to formalize that knowledge and show how the system can use it. Our third step is to show how the system can modify its planned response so as to convey those Ps. In this paper, Section 2 addresses the first step of this process and Sections 3 and 4 address the second. The third step we mention here only in passing.

## 2. Factors in Computing Likely Informing Behavior]

Before discussing the factors involved in computing this desired system behavior, we want to call attention to the distinction we are drawing between *actions* and *events*, and between the *stated goal* of a question and its *intended goal*. We limit the term *action* to things that Q has some control over. Things beyond Q's control we will call *events*, even if performed by other agents. While events may be *likely* or even *necessary*, Q and R nevertheless can do nothing more than wait for them to happen. This distinction between actions and events shows up in R's response behavior: if an action is needed, R can suggest that Q perform it. If an event is, R can do no more than inform Q.

Our second distinction is between the *stated goal* or "S-goal" of a request and its *intended goal* or "I-goal". The former is the goal most directly associated with Q's request, beyond that Q know the information. That is, we take the S-goal of a request to be the goal directly achieved by using the information.

Underlying the stated goal of a request though may be another goal that the speaker wants to achieve. This *intended goal* or "I-goal" may be related to the S-goal of the request in any of a number of ways:

- The I-goal may be the same as the S-goal.
- The I-goal may be more abstract than the S-goal, which addresses only part of the I-goal. (This is the standard goal/sub-goal relation found in hierarchical planning [14].) For example, Q's S-goal may be to delete some files (e.g., "How can I delete all but the last version of FOO.MSS?"), while his I-goal may be to bring his file usage under quota. This more abstract goal may also involve archiving some other files, moving some into another person's directory, etc.
- The S-goal may be an enabling condition for the I-goal. For example, Q's S-goal may be to get read/write access to a file, while his I-goal may be to alter it.
- The I-goal may be more general than the S-goal. For example, Q's S-goal may be to know how to repeat a control-N, while his I-goal may be to know how to effect multiple sequential instances of a control character.
- Conversely, the I-goal may be more specific than the S-goal - for example, Q's S-goal may be to know how to send files to someone on another machine, while his I-goal is just to send a particular file to a local network user, which may allow for a specialized procedure.

Inferring the I-goal corresponding to an S-goal is an active area of research [1, Carberry83, 10, 11]. We assume for the purposes of this paper that R can successfully do so. One problem is that the relationship that Q believes to hold between his S-goal and his I-goal may not actually hold: for example, the S-goal

may not fulfill part of the I-goal, or it may not instantiate it, or it may not be a pre-condition for it. In fact, the S-goal may not even be possible to effect! This failure, under the rubric "relaxing the appropriate-query assumption", is discussed in more detail in [10, 11]. It is also reason for augmenting R's response with appropriate Ps, as we note informally in this section and more formally in the next.

Having drawn these distinctions, we now claim that in order for the system to compute both a direct answer to Q's request and such Ps as he would expect to be informed of, were they true, the system must be able to draw upon knowledge/beliefs about

- the events or actions, if any, that can bring about a goal
- their enabling conditions
- the likelihood of an event occuring or the enabling conditions for an action holding, with respect to a state
- ways of evaluating methods of achieving goals - for example, with respect to simplicity, other consequences (side effects), likelihood of success, etc.
- general characteristics of cooperative expert behavior

The roles played by these different types of knowledge (as well as specific examples of them) are well illustrated in the next section.

## 3. Formalizing Knowledge for Expert Response

In this section we give examples of how a formal model of user beliefs about cooperative expert behavior can be used to avoid misleading responses to task-related questions - in particular, what is a very representative set of questions, those of the form "How do I do X?". Although we use logic for the model because it is clear and precise, we are not proposing theorem proving as the means of computing cooperative behavior. In Section 4 we suggest a computational mechanism. The examples are from a domain of advising students and involve responding to the request "I want to drop CIS577". The set of individuals includes not only students, instructors, courses, etc. but also states. Since events and actions change states, we represent them as (possibly parameterized) functions from states to states. All terms corresponding to events or actions will be underlined. For these examples, the following notation is convenient:

| | |
|---|---|
| Q | the user |
| R | the expert |
| Sc | the current state of the student |
| RB(P) | R believes proposition P |
| RBQB(P) | R believes that Q believes P |
| admissible($e$(S)) | event/action $e$ can apply in state S |
| likely($a$,S) | $a$ is a likely event/action in state S |
| holds(P,S) | P, a proposition, is true in S |
| want(x,P) | x wants P to be true |

To encode the preconditions and consequences of performing an action, we adopt an axiomatization of STRIPS operators due to [Chester83, 7, 15]. The preconditions on an action being applicable are encoded using "holds" and "admissible" (essentially defining "admissible"). Namely, if c1, ..., cn are preconditions on an action $a$,

holds(c1,s) &...& holds(cn,s) ⇒ admissible($a$(s))

$a$'s immediate consequences p1, ..., pm can be stated as

admissible($a$(s)) ⇒ holds(p1, $a$(s)) & ... & holds(pm, $a$(s))

A frame axiom states that only p1, ..., pm have changed.

¬(p=p1) & ... & ¬(p=pm) & holds(p,s)3 & admissible($a$(s)) ⇒ holds,$a$(s))

In particular, we can state the preconditions and consequences of dropping CIS577. ($h$ and $n$ are variables, while C stands for CIS577.)

RB(holds(enrolled($h$, C, fall), $n$) & holds(date($n$)<Nov16, $n$)
⇒ admissible(*drop*($h$,C)($n$)))

RB(admissible(*drop*($h$,C)($n$)) ⇒ holds(¬enrolled($h$,C,fall),*drop*($h$,C)($n$)))

RB(¬(p=enrolled($h$,C,fall)) & admissible(*drop*($h$,C)($n$)) & holds(p,$n$)
⇒ holds(p,*drop*($h$,C)($n$)))

Of course, this only partially solves the frame problem, since there will be implications of p1, ..., pm in general. For instance, it is likely that one might have an axiom stating that one receives a grade in a course only if the individual is enrolled in the course.

Q's S-goal in dropping CIS577 is not being in the course. By a process of reasoning discussed in [10, 11], R may conclude that Q's likely intended goal (I-goal) is not failing it. That is, R may believe:

RBQB(holds(¬fail(Q,C), *drop*(Q,C)(Sc)))[5]

RB(want(Q,¬fail(Q,C)) )

What we claim is: (1) R must give a truthful response addressing at least Q's S-goal; (2) in addition, R may have to provide information in order not to mislead Q; and (3) R may give additional information to be cooperative in other ways. In the subsections below, we enumerate the cases that R must check in effecting (2). In each case, we give both a formal representation of the additional information to be conveyed and a possible English gloss. In that gloss, the part addressing Q's S-goal will appear in normal type, while the additional information will be underlined.

For each case, we give two formulae: a statement of R's beliefs about the current situation and an axiom stating R's beliefs about Q's expectations. Formulae of the first type have the form RB(P). Formulae of the second type relate such beliefs to performing an informing action. They involve a statement of the form

RB[P] ⇒ likely($i$, Sc),

where $i$ is an informing act. For example, if R believes there is a better way to achieve Q's goal, R is likely to inform Q of that better way. Since it is assumed that Q has this belief, we have

QB( RB[P] ⇒ likely($i$, Sc) ).

[5]It will also be the case that RBQB(admissible(*drop*(Q,C)(Sc))) if Q's asks "How can I drop CIS577?", but not if he asks "Can I drop CIS577?". In the latter case, Q must of course believe that it may be admissible, or why ask the question. In either case, R's subsequent behavior doesn't seem contingent on his beliefs about Q's beliefs about admissibility.

where we can equate "Q believes $i$ is likely" with "Q expects $i$." Since R has no direct access to Q's beliefs, this must be embedded in R's model of Q's belief space. Therefore, the axioms have the form (modulo quantifier placement)

$RBQB(\ RB[P] \Rightarrow likely(i, Sc)\ ).$

An informing act is meant to serve as a command to a natural language generator which selects appropriate lexical items, phrasing, etc. for a natural language utterance. Such an act has the form *inform-that(R,Q,P)* R informs Q that P is true.

### 3.1. Failure of enabling conditions

Suppose that it is past the November 15th deadline or that the official records don't show Q enrolled in CIS577. Then the enabling conditions for dropping it are not met. That is, R believes Q's S-goal cannot be achieved from Sc.

[1] $RB(want(Q,\neg fail(Q,C)) \ \&\ \neg admissible(drop(Q,C)(Sc)))$

Thus R initially plans to answer "You can't drop CIS577". Beyond this, there are two possibilities.

#### 3.1.1. A way

If R knows another action $b$ that would achieve Q's goals (cf. formula [2]), Q would expect to be informed about it. If not so informed, Q may mistakenly conclude that there is no other way. Formula [3] states this belief that R has about Q's expectations.

[2] $RB((\exists b)\ [admissible(b(Sc))\ \&\ holds(\neg fail(Q,C),\ b(Sc))])$

[3] $RBQB(RB[want(Q,\neg fail(Q,C))\ \&\ \neg admissible(drop(Q,C)(Sc))]\ \&$
$RB[(\exists b)[admissible(b(Sc))\ \&\ holds(\neg fail(Q,C),b(Sc))]]$
$\Rightarrow likely(inform\text{-}that(R,\ Q,$
$(\exists b)\ [admissible(b(Sc))\ \&\ holds(\neg fail(Q,C),b(Sc))\ \&$
$can(Q,b)),Sc)])$

R's full response is therefore "You can't drop 577; *you can b.*" For instance, $b$ could be changing status to auditor, which may be performed until December 1.

#### 3.1.2. No way

If R doesn't know of any action or event that could achieve Q's goal (cf. [4]), Q would expect to be so informed. Formula [5] states this belief about Q's expectations.

[4] $RB(\neg(\exists a)[admissible(a(Sc))\ \&\ holds(\neg fail(Q,C),a(Sc))])$

[5] $RBQB(RB(want(Q,\neg fail(Q,C))\ \&\ \neg(\exists a)[admissible(a(Sc))$
$\&\ holds(\neg fail(Q,C),\ a(Sc))])$
$\Rightarrow likely(inform\text{-}that(R,\ Q,\ \neg(\exists a)[admissible(a(Sc))$
$\&\ holds(\neg fail(Q,C),a(Sc))]),Sc))$

To say only that Q cannot drop the course does not exhibit expert cooperative behavior, since Q would be uncertain as to whether R had considered other alternatives. Therefore, R's full response is "You can't drop 577; *there isn't anything you can do to prevent failing.*"

Notice that R's analysis of the situation may turn up additional information which a cooperative expert

could provide that does not involve avoiding misleading Q. For instance, R could indicate enabling conditions that prevent there being a solution: suppose the request to drop the course is made after the November 15th deadline. Then R would believe the following, in addition to [1]

RB(holds(enrolled(Q,C,fall),Sc) & holds(date(Sc)>Nov15,Sc))

More generally, we need a schema such as the following about Q's beliefs:

RBQB(RB[want(Q,¬fail(Q,C))
& (holds(P1, S) &...& holds(Pn, S) ⇒ admissible(a(S)))
& (¬holds(Pi, S), for some Pi above)]
⇒ likely(*inform-that(R,Q,¬holds(Pi,S))*,S))

In this case the response should be "You can't drop 577; *Pi isn't true.*" Alternatively, the language generator might paraphrase the whole response as, "if Pi were true, you could drop."

Of course there are potentially many ways to try to achieve a goal: by a single action, by a single event, or by an event and an action, ... In fact, the search for a sequence of events or actions that would achieve the goal may consider many alternatives. If all fail, it is far from obvious which blocked condition to notify Q of, and knowledge is needed to guide the choice. Some heuristics for dealing with that problem are given in [12].

### 3.2. An nonproductive act

Suppose the proposed action does not achieve Q's I-goal, cf. [6]. For example, dropping the course may still mean that failing status would be recorded as a WF (withdrawal while failing). R may initially plan to answer "You can drop 577 by ...". However, Q would expect to be told that his proposed action does not achieve his I-goal. Formula [7] states R's belief about this expectation.

[6] RB(¬holds(¬fail(Q,C), *drop(Q,C)*(Sc)) & admissible(*drop(Q,C)*(Sc)) )

[7] RBQB(RB[ want(Q,¬fail(Q,C)) & ¬holds(¬fail(Q,C),*drop(Q,C)*(Sc))
& admissible(*drop(Q,C)*(Sc))]
⇒ likely(*inform-that*,R,Q,
*¬holds(¬fail(Q,C),drop(Q,C)(Sc)))*,Sc))

R's full response is, "You can drop 577 by .... *However, you will still fail.*" Furthermore, given the reasoning in section 3.1.1 above, R's full response would also inform Q if there is an action *b* that the user can take instead.

### 3.3. A better way

Suppose R believes that there is a better way to achieve Q's I-goal, cf. [8] - for example, taking an incomplete to have additional time to perform the work, and thereby not losing all the effort Q has already expended. Q would expect that R, as a cooperative expert, would inform him of such a better way, cf. [9]. If R doesn't, R risks misleading Q that there isn't one.

[8] RB((∃*b*)[holds(¬fail(Q,C), *b*(Sc)) &
admissible(*b*(Sc)) & better(*b*,*drop(Q,C)(Sc)*)])

[9] RBQB(RB[want(Q,¬fail(Q,C))] &
RB[(∃*b*)[holds(¬fail(Q,C), *b*(Sc)) & admissible(*b*(Sc)) &
better(*b*,*drop(Q,C)(Sc)*)]
⇒ likely(*inform-that(R,Q,*

*(∃b)[holds(¬fail(Q,C),b(Sc)) & admissible(b(Sc)) &*
*better(b,drop(Q,C)(Sc))]], Sc)]])*

R's direct response is to indicate how *f* can be done. R's full response includes, in addition, "*b is a better way.*"

Notice that if R doesn't explicitly tell Q that he is presenting a better way (i.e., he just presents the method), Q may be misled that the response addresses his S-goal: i.e., he may falsely conclude that he is being told how to drop the course. (The possibility shows up clearer in other examples - e.g., if R omits the first sentence of the response below

Q: How do I get to Marsh Creek on the Expressway?
R: It's faster and shorter to take Route 30. Go out
Lancaster Ave until....

Thus even when adhering to expert response behavior in terms of addressing an I-goal, we must keep the system aware of potentially misleading aspects of its modified response as well.

Note that R may believe that Q expects to be told the best way. This would change the second axiom to include within the scope of the existential quantifier

(∀a){¬(a=b) ⇒ [holds(¬fail(Q,C), a(Sc)) & admissible(a(Sc)) & better(b,a)]}

### 3.4. The only way

Suppose there is nothing inconsistent about what the user has proposed - i.e., ... preconditions are met and it will achieve the user's goal. R's direct response would simply be to tell Q how. However, if R notices that that is the only way to achieve the goal (cf. [10]), it could optionally notify Q of that, cf. [11].

[10] RB((∃!a)[holds(¬fail(Q,C),a(Sc)) & admissible(a(Sc)) & *a=drop(Q,C)(Sc)*])

[11] RBQB(RB(want(Q,¬fail(Q,C)))
& RB((∃!a)[holds(¬fail(Q,C), a(Sc)) & admissible(a(Sc)) & *a=drop(Q,C)(Sc)*])
⇒ likely(*inform-that(R, Q,*
*(∃!a)[holds(¬fail(Q,C),a(Sc))*
*& admissible(a(Sc)) & a=drop(Q,C)(Sc)]]*, Sc))

R's full response is "You can 'rop 577 by .... *That is the only way to prevent failing.*"

### 3.5. Something Turning Up

Suppose there is no appropriate action that Q can take to achieve his I-goal. That is,

RB( ¬(∃ a)[admissible(a(Sc)) & holds(g, a(Sc))])

There may still be some event *e* out of Q's control that could bring about the intended goal. This gives several more cases of R's modifying his response.

#### 3.5.1. Unlikely event

If *e* is unlikely to occur (cf. [12]), Q would expect R to inform him of *e*, while noting its implausibility, cf. [13]

[12] RB((∃e)[admissible(e(Sc)) & holds(¬fail(Q,C), e(Sc))
& ¬likely(*e*, Sc)])

[13] RBQB(RB(want(Q,¬fail(Q,C)) &
RB(¬(∃a)[admissible(a(Sc)) & holds(¬fail(Q,C),a(Sc))] &
(∃e)[admissible(e(Sc)) & holds(¬fail(Q,C),e(Sc))
& ¬likely(e,Sc)])
⇒ likely(*inform-that(R, Q,*
*(∃ e)[admissible(e,Sc) & holds(¬fail(Q,C), e(Sc))*
*& ¬likely(e, Sc)])*, Sc))

Thus R's full response is, "You can't drop 577. *If e occurs, you will not fail 577, but e is unlikely.*"

### 3.5.2. Likely event

If the event e is likely (cf. [14]), it does not seem necessary to state it, but it is certainly safe to do so. A formula representing this case follows.

[14] RB((∃e)[admissible(e(Sc)) &
holds(¬fail(Q,C),e(Sc)) & likely(e,Sc)])

R's beliefs about Q's expectations are the same as the previous case except that likely(e, Sc) replaces ¬likely(e, Sc). Thus R's full response may be "You can't drop 577. *However, e is likely to occur, in which case you will not fail 577.*"

### 3.5.3. Event followed by action

If event e brings about a state in which the enabling conditions of an effective action a are true, cf. [15]

[15] RB((∃e)(∃a)[admissible(e(Sc)) & admissible(a(e(Sc))) &
holds(¬fail(Q,C), a(e(Sc)))])

[16] RBQB(RB((∃e)(∃a)[want(Q,¬fail(Q,C)) & admissible(e(Sc))
& admissible(a(e(Sc))) & holds(¬fail(Q,C),a(e(Sc)))])
⇒ likely(*inform-that(R,Q,*
*(∃e)(∃a)[holds(¬fail(Q,C),a(e(Sc)))) &*
*admissible(a(e(Sc)))]*),Sc))

then the same principles about informing Q of the likelihood or unlikelihood of e apply as they did before. In addition, R must inform Q of a, cf. [16]. Thus R's full response would be "You can't drop 577. *If e were to occur, which is (un)likely, you could a and thus not fail 577.*"

## 4. Reasoning

Our intent in using logic has been to have a precise representation language whose syntax informs R's reasoning about Q's beliefs. Having computed a full response that conforms to all these expectations, R may go on to 'trim' it according to principles of brevity that we do not discuss here.

Our proposal is that the informing behavior is "pre-compiled". That is, R does not reason explicitly about Q's expectations, but rather has compiled the conditions into a case analysis similar to a discrimination net. For instance, we can represent informally several of the cases in section 3.

**if** admissible(*drop(Q,C)*(Sc))
**then if** ¬holds(¬fail(Q,C),*drop(Q,C)*(Sc))
**then begin** nonproductive act
**if** (∃b)[admissible(b(Sc)) & holds(¬fail(Q,C),b(Sc))]
**then** a way
**else** no way
**end**
**else if** (∃b)[admissible(b(Sc)) &

holds(¬fail(Q,C),*b*(Sc)) & better(*b*,*f*)]
**then** a better way
**else if** (∃ *b*)[admissible(*b*(Sc)) & holds(¬fail(Q,C), *b*(Sc))]
**then** a way
**else** no way
...

Note that we are assuming that R assumes the most demanding expectations by Q. Therefore, R can reason solely within its own space without missing things.

## 5. Conclusion

Since the behavior of expert systems will be interpreted in terms of the behavior users expect of cooperative human experts, we (as system designers) must understand such behavior patterns so as to implement them in our systems. If such systems are to be truly cooperative, it is not sufficient for them to be simply truthful. Additionally, they must be able to predict limited classes of false inferences that users might draw from dialogue with them and also to respond in a way to prevent those false inferences. The current enterprise is a small but non-trivial step in this direction. In addition to questions about achieving goals, we are investigating other cases where a cooperative expert should prevent false inferences by another agent, including preventing inappropriate default reasoning [6, JWW84nonmon].

Future work should include

- identification of additional cases where an expert must prevent false inferences by another agent,
- formal statement of a general principle for constaining the search for possible false inferences, and
- design of a natural language planning component to carry out the informing acts assumed in this paper.

ACKNOWLEDGEMENTS

We would like to thank Martha Pollack, Deborah Dahl, Julia Hirschberg, Kathy McCoy and the AAAI program committee reviewers for their comments on this paper.

# References


**1.** Allen, J. Recognizing Intentions from Natural Language Utterances. In *Computational Models of Discourse*, M. Brady, Ed., MIT Press, Cambridge MA, 1982.

**2.** Birnbaum, L., Flowers, M. & McQuire, R. Towards an AI Model of Argumentation. Proceedings of 1980 Conference, American Assoc. for Artificial Intelligence, Stanford CA, August, 1980.

**3.** Cohen, R. A Theory of Discourse Coherence for Argument Understanding. Proceedings of the 1984 Conference, Canadian Society for Computational Studies of Intelligence, University of Western Ontario, London Ontario, May, 1984, pp. 6-10.

**4.** Hirschberg, J. Scalar Implicature and Indirect Responses in Question-Answering. Proc. CSCSI-84, London, Ontario, May, 1984.

**5.** Joshi, A.K. M[illegible]l Beliefs in Question Answering Systems. In *Mutual Belief*, N. Smith, Ed., Academic Pre[illegible]w York, 1982.

**6.** Joshi, A[illegible] Webber, B. & Weischedel, R. Preventing False Inferences. Proceedings of COLING-84, Stanford CA, July, 1984.

**7.** Kowalski, Robert. *Logic for Problem Solving*. North Holland, New York, 1979.

**8.** Lehnert, W. A Computational Theory of Human Question Answering. In *Elements of Discourse [illegible]standing*, A. Joshi, B. Webber & I. Sag, Ed., Cambridge University Press, 1981.

**9.** McCarthy, John. "Circumscription -- A Form of Non-Monotonic Reasoning". *Artificial Intelligence 13* (1980), 27-39.

**10.** Pollack, Martha E. *Goal Inference in Expert Systesm*. MS-CIS-84-07, University of Pennsylvania, 1984. Doctoral dissertaion proposal..

**11.** Pollack, M. Good Answers to Bad Questions. Proc. Canadian Society for Computational Studies of Intelligence (CSCSI), Univ. of Western Ontario, Waterloo, Canada, May, 1984.

**12.** Ramshaw, Lance and Ralph M. Weischedel. Problem Localization Strategies for Pragmatics Processing in Natural Language Front Ends. Proceedings of COLING-84, July, 1984.

**13.** Reiter, R. Closed World Databases. In *Logic and Databases*, H. Gallaire & J. Minker, Ed., Plenum Press, 1978, pp. 149-177.

**14.** Sacerdoti, Earl D.. *A Structure for Plans and Behavior*. American Elsevier, New York, 1977.

**15.** Warren, D.H.D. WARPLAN: A System for Generating Plans. Proceedings of IJCAI-75, August, 1975.

# THE ROLE OF PERSPECTIVE IN RESPONDING TO PROPERTY MISCONCEPTIONS

MS-CIS-85-31
May 1985

Kathleen F. McCoy
Department of Computer & Information Science
University of Pennsylvania
Philadelphia, PA 19104

This work is partially supported by the ARO grant DAA20-84-K-0061 and by the NSF grant #MCS81-07290.

This paper appears in The Proceedings of IJCAI-85, August 18-23, 1985, University of California, Los Angeles, Ca.

## Abstract

In order to adequately respond to misconceptions involving an object's properties, we must have a context-sensitive method for determining object similarity. Such a method is introduced here. Some of the necessary contextual information is captured by a new notion of *object perspective*. It is shown how object perspective can be used to account for different responses to a given misconception in different contexts.

## 1. Introduction

As a user interacts with a database or an expert system, s/he may attribute a property or property value to an object that that object does not have. For instance, imagine the following query to a database.

U. Give me the HULL-NO of all DESTROYERS whose MAST-HEIGHT is above 190.

If a system were to evaluate such a query, it might find that there are no such ships in the database. The reason for this is that the user has queried a value of the property MAST-HEIGHT that it cannot have for the object DESTROYER. I term this error a *property misconception*. Upon encountering such a query, even a very cooperative system could only respond:

S. There are no DESTROYERS in the database with a MAST-HEIGHT above 190. Would you like to try again?

In most cases, however, this is not the way a human would respond. A study of human/human transcripts reveals that a human conversational partner often tries to get at the cause of the misconception and offer additional information to correct the wrong information. The additional information *often* takes the form of a correct query that is a possible alternative to the user's query. In this paper I describe some of the knowledge and reasoning that are necessary for a natural language interface to a database or expert system to mimic this human behavior.

In the above query, since there is an object similar to a DESTROYER that has the value of HULL-NO given, the user's misconception may result from his/her confusing the two objects. Hence a reasonable response would be:

S. All DESTROYERS in the database have a MAST-HEIGHT between 85 and 90. Were you thinking of an AIRCRAFT-CARRIER?

Notice the strategy used to correct the misconception is to (1) deny (implicitly) the property/value given, (2) give the corresponding correct information, (3) suggest an alternative query containing the object the user may have confused with the misconception object.

In other situations, a reasonable alternative query might involve the same object the user asked about, with a different property/value pair. This is the case in the following query.

U. Give me the HULL-NO of all DESTROYERS whose MAST-HEIGHT is above 3500.

S. All DESTROYERS in the database have a MAST-HEIGHT between 85 and 90. Were you thinking of the DISPLACEMENT?

This response is similar to the one given above except that the alternative query suggests an attribute rather than an object which may have been confused.

In general, there can be two major reasons why a wrong attribution may occur. Either (1) the user has the wrong object -- that is, s/he has confused the object being discussed with a similar object or has reasoned (falsely) by analogy from a similar object; or (2) the user has the wrong attribute -- that is, s/he has confused the attribute being discussed with a similar attribute. If one of these two can be seen as likely in a given situation, then a revised query can be suggested which mentions the similar object or the similar attribute.

To propose alternative queries, a system must have a method for determining similarity of objects and attributes. In this paper I will focus on responses involving object confusion; thus I will examine a similarity metric for objects. In the next section such a similarity metric is introduced. The following section introduces a new notion of object perspective which is needed to provide the similarity metric with some necessary contextual information, in particular, attribute salience ratings. Finally, an example of how perspective information and the similarity metric can be used to give reasonable responses to misconceptions involving object properties is given.

## 2. Object Similarity

As was shown above, in order to respond effectively to property misconceptions, we must have a method for determining object similarity. Object similarity has previously been shown to be important in tasks such as organizing explanations [6], offering cooperative responses to pragmatically ill-formed queries [2], and identifying metaphors [9]. In the above systems the similarity of two objects is based on the distance between the objects in the generalization hierarchy. One problem with this approach is that it is *context invariant.*[*] That is, there is no way for contextual information to affect similarity judgments.

However, Tversky [8] proposes a measure of object similarity based on common and disjoint features/properties of the objects involved, which enables contextual

[*] See [5] for additional problems and discussion of this point.

information to be taken into account. Tversky's similarity rating for two objects **a** and **b**, where **A** is the set of properties associated with object **a** and **B** is the set of properties associated with object **b**, can be expressed as:

$$s(a,b) = \theta f(A \cap B) - \alpha f(A - B) - \beta f(B - A)$$

for some $\theta, \alpha, \beta \geq 0$. This equation actually defines a family of similarity scales where $\theta$, $\alpha$, and $\beta$ are parameters which alter the importance of each piece of the equation, and f maps over the features and yields a salience rating for each. The equation states that the similarity of two objects is some function of their common features minus some function of their disjoint features. The importance of each feature involved (determined by the function f) and the importance of each piece of the equation (determined by $\theta$, $\alpha$, and $\beta$) may change with context.

Previous work [4, 7] has discussed the effect of "focus" on the prominence of objects. Focusing algorithms can be adapted to set the values of $\theta$, $\alpha$, and $\beta$. For instance, if object **a** is "in focus" and object **b** is not, then the features of **a** should be weighted more heavily than the features of **b**. Thus we should choose $\alpha > \beta$ so that the similarity is reduced more by features of **a** that are not shared by **b** than vice versa.

The problem then is to determine f. Other work [3, 9] has hand encoded salience values for the attributes of individual objects in the knowledge base, effectively setting the f function once and for all. This approach, however, is not sufficient since salience values must change with context. The following examples in which two objects (Treasury Bills and Money Market Certificates) are compared in two different circumstances, illustrate the importance of context on the similarity rating.

Consider someone calling an expert financial advisor to see if she can better invest her money. She begins by telling the expert where her money is:

U. We have $40,000 in money market certificates. One is coming due next week for $10,000... I was wondering if you think this is a good savings...

E. Well, I'd like to see you hold that $10,000 coming due in a money market fund and then get into a longer term money market certificate.

U. Hm... well I was just wondering, what about a treasury bill instead?

E. That's not a bad idea but it doesn't replace your money market certificate in any way -- it's an exact duplicate. They're almost identical types of instruments - so one, as far as I'm concerned, is about the same as another.

Now consider how the same two objects can be seen quite differently when viewed

in a different way. Imagine the following conversation:

U. I am interested in buying some US Government Securities. Now I was thinking of Money Market Certificates since they are the same as Treasure Bills.

E. But they're not - they are two very different things. A Treasury Bill is backed by the U.S. Government: you have to get it from the federal reserve. A Money Market Certificate, on the other hand, is backed by the individual bank that issues it. So, one is a Government Security while the other is not.

In the first example the objects are viewed as savings instruments. This view highlights attributes such as interest-rates and maturity-dates that are common to Treasury Bills and Money Market Certificates. This highlighting causes the two instruments to be seen as "identical". In contrast, the second example views the objects as instruments issued by a particular company or organization. In this case attributes such as issuing-company and purchase-place are highlighted. Since these highlighted attributes are different for the two objects, the objects are seen as being quite different.

As the examples illustrate, a context-free metric of similarity is not sufficient; contextual information is needed. A notion of object perspective, introduced below, can capture the needed contextual information. In particular, perspective accounts for how the f function (the assignment of salience values to various attributes) changes with context.

## 3. Perspective

[4, 1] note that the same object may be viewed from different perspectives. For instance a particular building may be viewed as an architectural work, a home, a thing made with bricks, etc. According to this work, an object viewed from a particular perspective is seen as having one particular superordinate, although in fact it may have many superordinates. The object inherits properties only from the superordinate in perspective. Therefore different perspectives on the same object cause different properties to be highlighted.

Although this notion of perspective is intuitively appealing, in practice its use is rather difficult since it hinges on the use of a limited inheritance mechanism. The problem is that attributes may be inherited from the top of the generalization hierarchy, not just from immediate superordinates. So, an object's perspective involves not just one superordinate but a chain of superordinates. Therefore one must not only determine what perspective a particular object is being viewed from, but also what perspective its

superordinate is viewed from, and so on. As one continues up the hierarchy in this fashion, the definition of perspective as viewing an object as a member of a particular superordinate becomes less and less appealing.

In addition, this notion of object perspective says nothing about the density of the generalization hierarchy. That is, in some situations the immediate superordinate of an object (and the properties it contributes) may be ignored. For example, even though a whale is a cetacean (a class of aquatic mammals including whales and porpoises), this classification (and all attributes contributed by the classification) may be ignored in some situations in which the important attributes instead are inherited from a superordinate of cetacean, say, mammal. In other situations, the class "cetacean" may be central. The notion of object perspective outlined above has no way of determining whether or not certain superordinates should be ignored or included.

Here I introduce a new notion of perspective which is able to handle both the assignment of differing salience values and the density problem. In this notion, *perspectives sit orthogonal* to the generalization hierarchy. Each comprises a set of properties and their salience values. A number of perspectives must be defined *a priori* for the objects in a particular domain. The specification of perspectives, just like the specification of an object taxonomy, must be done by a domain expert. Knowledge of useful perspectives in a domain then, is part of the domain expertise.

With this new notion of perspective, when an object is viewed through a particular perspective, the perspective essentially acts as a filter on the properties which that object inherits from its superordinates. That is, properties are inherited with the salience values given by the perspective. Thus properties of the object which are given a high salience rating by the perspective will be highlighted, while those which are given a low salience value or do not appear in the perspective will be suppressed. The density problem is handled by ignoring those superordinate concepts which contribute only attributes suppressed by the current perspective.

## 4. Using Perspective to Determine Responses

Perspective information can be used with Tversky's similarity metric to help determine alternative queries to a query containing a misconception. To see how this works, consider a domain containing the following three objects with the attributes shown:

Money Market Certificates
Maturity: 3 months
Denominations: $1,000
Issuer: Commercial Bank
Penalty for Early Withdrawal: 10%
Purchase Place: Commercial Bank
Safety: Medium

Treasury Bills
Maturity: 3 months
Denominations: $1,000
Issuer: US Government
Purchase Place: Federal Reserve
Safety: High

Treasury Bond
Maturity: 7 years
Denominations: $500
Issuer: US Government
Penalty for Early Withdrawal: 20%
Purchase Place: Federal Reserve
Safety: High

and the following perspectives:

Savings Instruments
Maturity -- high
Denominations -- high
Safety -- medium

Issuing Company
Issuer -- high
Safety -- high
Purchase Place -- medium

Notice that the perspective of Savings Instruments highlights Maturity and Denominations, and somewhat highlights Safety. This indicates that when people are discussing securities as savings instruments, they are most interested in how long their money will be tied up and in what denominations they can save their money. The perspective of Issuing Company, on the other hand, highlights different attributes. When securities are discussed from this perspective, things like who the issuer of the security is and how safe a security issued from that company is, become important.

Suppose the perspective is Savings Instruments and the user says:

U. What is the penalty for early withdrawal on a Treasury Bill?

This query indicates that the user has a misconception since s/he has attributed a property to Treasury Bills that they do not have. One reasonable correction to the query would contain an alternative query which to replaces Treasury Bills with another object that has the property specified and is similar to Treasury Bills. The system may reason that both Money Market Certificates and Treasury Bonds have the penalty specified, and so check to see if either of these objects is similar to Treasury Bills. Notice that the Savings Instruments perspective highlights attributes common to Treasury Bills and Money Market Certificates (they have the same Maturity and Denominations), as well as attributes disjoint to Treasury Bills and Treasury Bonds (they have different Maturity and Denominations). Using these salience values, the similarity metric will find that Money Market Certificates are very similar to Treasury Bills while Treasury Bonds are very different. Thus Money Market Certificates will be deemed a probable object of confusion and the following correction may be offered:

S. Treasury Bills do not have a penalty for early withdrawal. Were you thinking of a Money Market Certificate?

Notice that if the perspective had instead been Issuing Company, which highlights attributes common to Treasury Bills and Treasury Bonds and disjoint to Treasury Bills and Money Market Certificates, the most reasonable response would be:

S. Treasury Bills do not have a penalty for early withdrawal. Were you thinking of a Treasury Bond?

Selecting the appropriate perspective is in itself a difficult question which is currently under investigation and will be reported in [5]. Certainly important in the selection procedure will be the attributes that have entered into the conversation so far: these attributes should be of fairly high salience in the selected perspective. Other clues to the selection process include the objects under discussion, the superordinates which contribute the attributes under discussion to these objects, and the current goals of the user.

## 5. Conclusion

In this paper we have seen that a context-dependent similarity metric is needed in order to respond adequately to misconceptions involving the properties of an object. Such a metric has been suggested and a notion of perspective has been introduced to account for some of the contextual information required by the metric. These notions have been shown to account for differences in the way a particular misconception is best

corrected in two different circumstances.

## 6. Acknowledgements

I would like to thank Julia Hirschberg, Aravind Joshi, Martha Pollack, Ethel Schuster, and Bonnie Webber for their many comments and discussions concerning the direction of this research and the content and style of this paper.

## 7. References

[1] Bobrow, D. G. and Winograd, T. "An Overview of KRL, a Knowledge Representation Language." *Cognitive Science 1*, 1 (January 1977), 3-46.

[2] Carberry, Sandra M. Understanding Pragmatically Ill-Formed Input. 10th International Conference on Computational Linguistics & 22nd Annual Meeting of the Association of Computational Linguistics, Coling84, Stanford University, Ca., July, 1984, pp. 200-206.

[3] Carbonnell, Jaime R. & Collins, Allan M. Mixed-Initiative Systems For Training and Decision-Aid Applications. Tech. Rept. ESD-TR-70-373, Electronics Systems Division, Laurence G. Hanscom Field, US Air Force, Bedford, Ma., November, 1970.

[4] Grosz, B. Focusing and Description in Natural Language Dialogues. In *Elements of Discourse Understanding*, A. Joshi, B. Webber & I. Sag, Ed.,Cambridge University Press, Cambridge, England, 1981, pp. 85-105.

[5] McCoy, K.F. Correcting Object-Related Misconceptions. 1985. Forthcoming University of Pennsylvania doctoral thesis

[6] McKeown, K. . *Generating Natural Language Text in Response to Questions About Database Structure*. Ph.D. Th., University of Pennsylvania, May 1982.

[7] Sidner, C. L. Focusing in the Comprehension of Definite Anaphora. In *Computational Models of Discourse*, Michael Brady and Robert Berwick, Ed.,MIT Press, Cambridge, Ma, 1983, pp. 267-330.

[8] Tversky, A. "Features of Similarity." *Psychological Review 84* (1977), 327-352.

[9] Weiner, E. Judith. "A Knowledge Representation Approach to Understanding Metaphors." *Computational Linguistics 19*, 1 (January - March 1984), 1-14.

# Adapting MUMBLE:
# Experience with Natural Language Generation

Robert Rubinoff
Computer and Information Science Department
Moore School of Electrical Engineering
University of Pennsylvania
Philadelphia, PA 19104
April 24, 1986

**Abstract**

This paper describes the construction of a MUMBLE-based [McDonald 83b] tactical component for the TEXT text generation system [McKeown 85]. This new component, which produces fluent English sentences from the sequence of structured message units output from TEXT's strategic component, has produced a 60-fold speed-up in sentence production. Adapting MUMBLE required work on each of the three parts of the MUMBLE framework: the interpreter, the grammar, and the dictionary. It also provided some insight into the organization of he generation process and the consequences of MUMBLE's commitment to a deterministic model.

Track: Engineering
Topic: Natural Language Generation

# 1 TEXT's Message Vocabulary

The TEXT system [McKeown 85] is designed to answer questions about the structure of a database. It is organized into two relatively independent components: a strategic component which selects and organizes the relevant information into a discourse structure, and a tactical component which produces actual English sentences from the strategic component's output. The original tactical component [Bossie 81] used a functional grammar [Kay 79]; it is this component that has been replaced.[1]

A tactical component for TEXT must be tailored to the form in which TEXT's strategic component organizes information. The strategic component responds to a query with a list of rhetorical propositions. A rhetorical proposition indicates some information about the database and the rhetorical function the information TEXT intends it to perform. For example, the rhetorical proposition:

```
(identification GUIDED PROJECTILE
                (restrictive (TRAVEL-MEANS SELF-PROPELLED))
                (non-restrictive (ROLE PROJECTED-OBJECT)))
```

indicates that TEXT wants to identify guided missiles by saying that they are projectiles and that they have certain attributes. This same information might be presented with a different rhetorical function such as attributive, i.e. attributing certain information to guided missiles rather than using it to identify them.

The information in the propositions generally consists of objects and attributes from TEXT's database model, indicating attributes of the mentioned objects and sub-type relationships between the objects. Some of the rhetorical functions allow other sorts of information. Inference propositions, for example, can indicate comparisons between database values:

```
(inference OCEAN-ESCORT CRUISER
           (HULL_NO (1 2 DE) (1 2 CA))
           (smaller DISPLACEMENT)
           (smaller LENGTH)
           (PROPULSION STMTURGRD STMTURGRD))
```

Here TEXT infers that ocean escorts have smaller length and displacement than cruisers, that the two kinds of ships have the same form of propulsion and that their hull numbers differ in their first two letters.

The strategic component also produces focus information for each proposition to insure that the individual sentences will form a coherent paragraph when combined. Following Sidner's model [Sidner 83], TEXT indicates a discourse focus and potential focus list for each proposition. The tactical component uses this information to decide when to pronominalize and what sentence-level syntactic structure to use.

[1] No attempt was made to investigate changing the overall division into strategic and tactical components. In part this was because the task of adapting the MUMBLE system to work with an independently developed text planner seemed like an interesting experiment in itself. Also, TEXT's strategic component was in the process of being ported from a VAX to a Symbolics 3600, and was thus already in a state of flux.

# 2 Adapting MUMBLE to TEXT

MUMBLE is a general-purpose generation framework which has been used with several domains and message representations[McDonald 83b,Karlin 85].[2] MUMBLE-based systems are constructed out of three components: the interpreter, the grammar, and the dictionary. The interpreter controls the overall generation process, co-ordinating the propagation and enforcement of constraints and the (incremental) translation of the message.[3] The grammar enforces grammatical constraints and maintains local grammatical information. The dictionary indicates, for each term in the vocabulary of the message formalism, the various ways it can be expressed in English. In adapting MUMBLE to this new domain, each of these three components had to be modified to a different degree.

## 2.1 The Interpreter

The interpreter is a domain-independent embodiment of MUMBLE's approach to generation [McDonald 83a]. The translation process is guided by a depth-first traversal of the surface structure tree. Each position in the tree has one or more labels, which may indicate procedures to be run when the traversal enters or leaves that position. The leaves of the tree will either be words, which are printed out after morphological processing, or pieces of the original message. In the latter case, the interpreter looks up the message in the dictionary to find a realization for the message that satisfies the local constraints. The result is a new piece of surface structure tree which is spliced into the tree, possibly with part(s) of the original message as new leaves. In this way, the entire message is gradually translated and printed out.

Because the interpreter actually does some additional work beyond guiding the generation process, some modification to it was required. In particular, the routine that handles word morphology needed changes to the way it determined noun phrase plurality. A noun phrase was considered to be plural if it was derived from a message element that represented more than one object. This was adequate when the domain contained only specific objects, as has been the case in past uses of MUMBLE. In TEXT, however, many terms represent *generic concepts*, e.g. SHIP, which represents the concept of a ship rather than any particular ship. Generic concepts can be expressed using either singular or plural, for example "A ship is a water-going vehicle" vs. "Ships are water-going vehicles". Thus the morphology routine had to be modified to look at the surface structure tree to see how the term had actually been realized. (The grammar and dictionary also had to be modified to always explicitly mark plural noun phrases in the tree). This was the only modification necesary to the interpreter.

However, not all of the interpreter was used. In addition to the traversal and incremental expansion of the surface structure tree, MUMBLE provides a mechanism for subsequent messages to be combined with the original message as it is translated. This is done via

[2]The version of MUMBLE used with TEXT dates from March 1985 and was originally set up to translate the output of the GENARO scene description system[McDonald 83a].

[3]A "message" is simply an expression that the text planner (here TEXT's strategic component) sends to MUMBLE to be translated. This is the same as a "realization specification" in [McDonald 83a].

"attachment points"[McDonald 85] that are marked in the tree; a new message from the planner can be added at an attachment point if there is a way to realize it that satisfies the attachment point's grammatical constraints. For example, in translating messages from GENARO, MUMBLE puts an ATTACH-AS-ADJECTIVE attachment point before the head noun in noun phrases. This allows MUMBLE to combine the messages such as (introduce house_1) and (red house_1) and generate the single sentence "This is a picture of a red house" instead of "This is a picture of a house. It is red."

This attachment mechanism is not used with the TEXT output.[4] Originally this decision was made because TEXT's strategic component organizes its messages into sentence-size packets (the propositions), and there seemed little reason to split these up and then have MUMBLE recombine them.

It turned out, though, that there was one case where attachment points would have been useful. The attribute-value pair (TARGET-LOCATION X) (where X is the type of target location, e.g. SURFACE or WATER) can be translated as either "a target location <X as a prep. phrase>" or "a <X as an adjective> target location". The latter form is preferred, but can only be used if X can be realized as an adjective. Thus MUMBLE can produce "a surface target location", but must resort to "a target location in the water". The problem is that since the interpreter traverses the tree in depth-first order, MUMBLE must decide which form to use for (TARGET-LOCATION X) *before* determining whether X has a realization as an adjective. This is the one case where it was necessary to circumvent MUMBLE's control strategy. Attachment points could have solved this problem; the value X could have been a separate message which would have been attached ahead of "target location" only if it had a possible realization as an adjective.

Unfortunately, there was a problem that prevented the use of attachment points. Attachment can be constrained so that the result will be grammatical and so that the attached message will be together with the proper objects. For example, (red house_1) will only be attached as an adjective in a noun phrase describing house_1. But there was no principled way to force several messages to be combined into a single phrase. To see why this is a problem, consider a simple rhetorical proposition:

```
(identification SHIP WATER-VEHICLE (restrictive (TRAVEL-MODE SURFACE)))
```

("restrictive" indicates that this attribute distinguishes SHIP from other kinds of WATER-VEHICLE.) This is intended to produce something like "a ship is a water-going vehicle that travels on the surface". There are really two pieces of information here: that ships are water-going vehicles, and the ships travel on the surface. If we separate these out, the first would become (identification SHIP WATER-VEHICLE), and the second would become something like (attributive SHIP (TRAVEL-MODE SURFACE)). The problem is that there is no way to force MUMBLE to combine these back to get something like the original sentence. Instead, MUMBLE might translate these as "A ship is a water-going vehicle. Ships travel on the surface." The precise characterization of ships has been diluted. Even worse, if the next proposition is about ships, the travel-mode information may be

[4] Actually, attachment points are used to attach each proposition as a new sentence. This is simply a convenience to allow MUMBLE to be invoked once on a list of propositions; the results are exactly as they would be if MUMBLE were invoked individually on each proposition.

combined with it instead, completely destroying the rhetorical structure intended by the strategic component.

Of course, there is no immediately apparent advantage to splitting up identification propositions (although it does suggest the possibility of letting more of the structural decisions be made by MUMBLE). But the same problems arise in trying to solve the problem with `(TARGET-LOCATION X)` discussed above. Attachment would allow the system to choose correctly between "a surface target location" and "a target location on the surface". But then instead of "The missile has a surface target location. Its target location is indicated by the DB attribute DESCRIPTION", MUMBLE might produce "The missile has a target location. Its surface target location is indicated by the DB attribute DESCRIPTION."

What is needed is a way to constrain the attachment process to build several messages into a single phrase. In fact, this capacity *has* been added to MUMBLE, although it is not present in the version used with TEXT [McDonald, personal communication]. It is possible to create "bundles" of messages that can have additional constraints on their overall realization while allowing the individual messages to be reorganized by the attachment process. This facility would make it feasible to use attachment with TEXT.

## 2.2 The Grammar

A MUMBLE grammar is not simply a declarative specification of valid surface structure like the rules in a context-free grammar. Rather, it consists of procedures that enforce (local) constraints and update info about current grammatical environment. The grammar provides the low-level control on realization as the interpreter traverses the tree. Grammar in the more conventional sense is a by-product of this process.

The grammar operates via "constituent-structure labels". These labels are placed on positions in the surface structure tree to identify their grammatical function. Some, such as `adjective` and `np`, are purely syntactic. Others, such as `compound` and `name`, have more of a semantic flavor (as used with TEXT). Labels constrain the generation process through an associated "grammatical constraint". This is a LISP predicate that must be satisfied by a proposed realization. Whenever the interpreter tries to translate a message, it checks that the constraints associated with all the labels at the current tree position are satisfied. These constraints can depend on both the proposed realization and the current environment. The labels also provide for local operations such as propagation of constraints through the tree and production of purely grammatical words such as the "to" in infinitival complements and the "that" in relative clauses. As with the constraints, this is done by associating procedures with labels. Each label has several "grammar routines" to be run at various times (such as when the interpreter enters or leaves a node, or after a message is realized). For example, the `rel-clause` label prints "that" when the interpeter enters a node it labels.

The labels handle local aspects of the grammar; global aspects are managed via "grammar variables". These keep track of global information (i.e. information needed at more than one tree position). For example, there are grammar variables that record the current

subject and the current discourse focus. These "variables" are actually stacks so that embedded phrases can be handled properly. The grammar variables are maintained by the grammar routines associated with the labels. The `clause` label, for example, updates the current subject whenever the interpreter enters or leaves a clause. The grammar variables enable information to be passed from one part of the tree to another.

Adapting MUMBLE to TEXT required considerable modification and extension to the grammar. A number of new syntactic structures had to be added. Some, such as appositives, simply required adding a new label. Others were more complex; relative clauses, for example, required a procedure to properly update the current subject grammar variable (if the relative pronoun is serving as the subject of the relative clause) as well as a procedure to produce the initial "that". Also, some of the existing grammar had to be modified. Post-nominal modifiers, for example, previously were always introduced via attachment and realized as prepositional phrases. When working from TEXT, they are introduced as part of the original noun phrase, and they can sometimes be realized as relative clauses, so the constraints had to be completely redesigned.

The grammar was also augmented to handle some constraints that were more semantic than syntactic. These were included in the grammar because it is the only mechanism by which decisions made at one place in the tree can affect subsequent decisions elsewhere. In fact, there is really nothing inherently grammatical about the "grammar"; it is a general mechanism for enforcement of local constraints and propagation of information through the tree. It serves well as a mechanism for enforcing grammatical constraints, of course, but it is also useful for other purposes. For example, the grammar variable `current-entity-type` keeps track of whether the current clause is dealing with specific or generic concepts.

## 2.3 The Dictionary

The dictionary stores the various possible ways each kind of message can be realized in English. Dictionary entries provide the pieces of surface structure that are organized by the interpreter and constrained by the grammar. The dictionary has two parts: a look-up function and a set of "realization classes" (or "rclasses"). The look-up function determines which rclass to use for a message and how to construct its arguments (which are usually either sub-parts of the message or particular words to use in the English realization of the message). An rclass is a list of possible surface structures, generally parameterized by one or more arguments.

The look-up function is intended to be domain-dependent. However, the look-up function that was developed for GENARO, which has a fairly simple keyword strategy, seemed adequate for TEXT as well. The keyword is the first element of the message if the message is a list; otherwise it is the message itself. The function then simply looks up the keyword in a table of terms and rclasses. Using an existing function was convenient, but it did cause a few problems because it required that keywords be added to TEXT's formalism in a few cases. For example, numbers had to be changed to `(number #)` so they would have a keyword. Some straightforward modifications to the look-up function, however, would allow MUMBLE to generate from the original TEXT formalism.

The realization classes vary greatly in their generality. Some of them are very general. The rclass SVO, for example, produces simple transitive clauses; the subject, verb, and object are arguments to the rclass. At the other extreme, the rclass TRAVEL-MEANS-CLASS is only useful for a particular attribute as used by TEXT; even if another system had an attribute called TRAVEL-MEANS, it is unlikely to mean exactly the same thing.

Intuitively, it might seem that there would be a number of general realization classes like SVO. In fact, though, SVO was the *only* pre-existing rclass used in that was used for TEXT. None of the other rclasses proved useful.

One source of this lack of generality is that concepts that seem similar are often expressed quite differently in natural language. For example, of the eight generic attributes (e.g. TRAVEL-MEDIUM, TRAVEL-MEANS, and TARGET-LOCATION) in the dictionary, three require special rclasses because the general translation won't work for them. Inside TEXT's domain model, TRAVEL-MEDIUM and TRAVEL-MEANS are considered similar sorts of concepts. But in English, the two concepts are expressed differently. TEXT's notion of generic attribute simply doesn't correspond to any natural linguistic category.

Furthermore, different message formalisms will tend to capture different generalizations. GENARO can use a CONDENSE-ON-PROPERTY rclass[McDonald 83a] because it has a particular notion of what a property is and how it gets translated into English. TEXT doesn't have anything that exactly corresponds to GENARO's properties (and even if it did, it couldn't condense things because the properties would be buried inside the rhetorical propositions).

The crux of the matter is that while there are linguistic generalizations that might be captured in realization classes, they usually cut across the grain of the classes of expressions in a message formalism, and cut differently for different formalisms. Thus whatever generalizations can be encoded into the rclasses for one formalism are unlikely to be useful with a different formalism.

For example, TEXT can produce attribute expressions of the form:

```
(HULL_NO (1 2 DE))
```

which means, roughly, "characters 1 through 2 of the HULL_NO are DE". This is a very idiosyncratic sort of message; it is unlikely that another (independently developed) text planner would have a message form with even the same meaning, let alone the same syntax. Thus the dictionary entry for this message is unlikely to be of use with any system other than TEXT. Many of TEXT's messages were similarly idiosyncratic, because its message formalism was designed around the needs of its particular task. Similarly, other generation systems will have their own idiosyncratic message formalism. Thus they will need their own highly specific dictionaries to work with MUMBLE.

# 3 Using MUMBLE to produce text

## 3.1 Examples from TEXT

The new MUMBLE-based tactical component has been very successful. It can process all of the examples in the appendix to [McKeown 85] and produce comparable English text. Furthermore, it can process all 57 sentences in the appendix in about 5 minutes; the old tactical component took that long to produce a single sentence.

For example, TEXT's strategic component responds to a request to describe the ONR database with:

```
(attributive db OBJECT (name REMARKS))
(constituency OBJECT (VEHICLE DESTRUCTIVE-DEVICE))
(attributive db VEHICLE
              (based-dbs (SOME-TYPE-OF TRAVEL_MEANS)
                         (SOME-TYPE-OF SPEED_INDICES)))
(attributive db DESTRUCTIVE-DEVICE
              (based-dbs (SOME-TYPE-OF LETHAL_INDICES)))
```

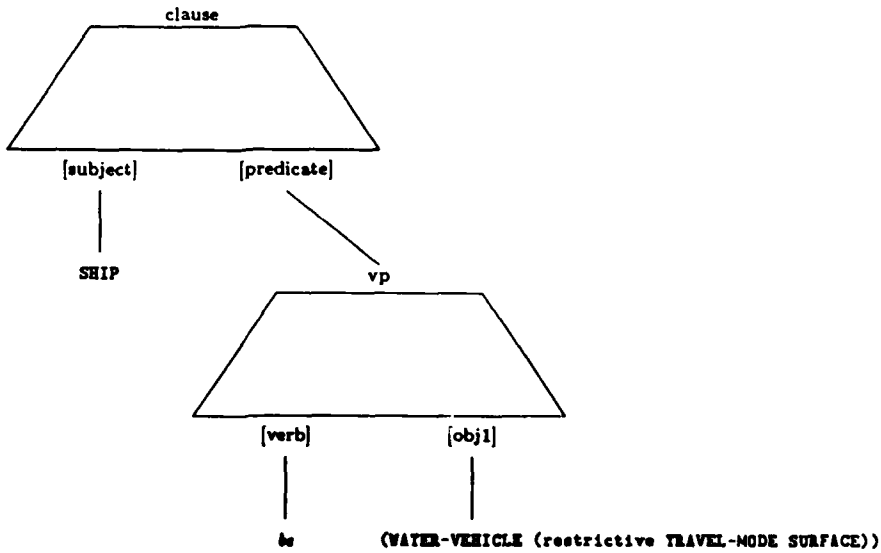
which is then translated into English by MUMBLE as:

> All entities in the ONR database have DB attributes REMARKS. There are 2 types of entities in the ONR database: vehicles and destructive devices. The vehicle has DB attributes that provide information on SPEED_INDICES and TRAVEL_MEANS. The destructive device has DB attributes that provide information on LETHAL_INDICES.

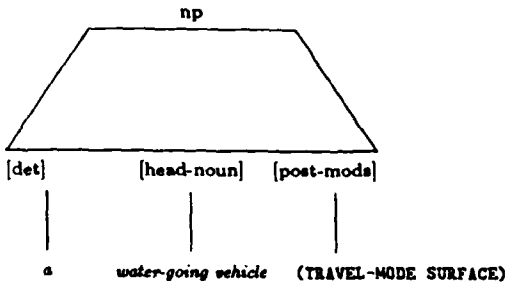
This translation is guided and controlled by the various sub-components that make up the MUMBLE tactical component, as can be seen in a more detailed example. The message:

```
(identification SHIP WATER-VEHICLE (restrictive TRAVEL-MODE SURFACE))
```

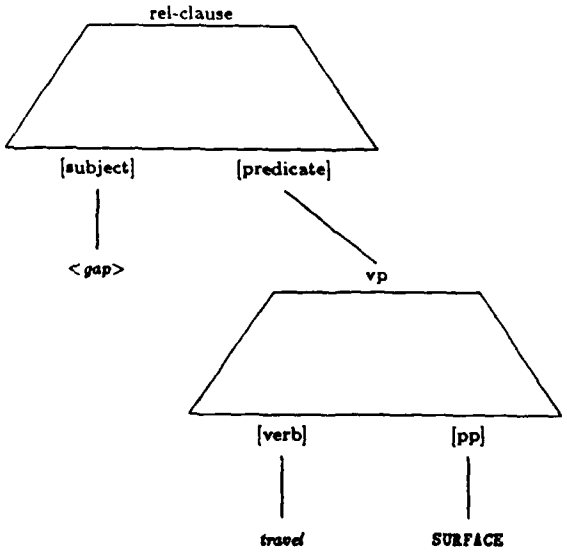
when received by MUMBLE, is first looked up in the dictionary, which indicates that the overall structure of the sentence will be:



The interpreter then traverses this (partially filled-out) surface structure tree, soon reaching the still untranslated message element SHIP. The first possibility listed for this in the dictionary is the noun phrase "a ship"; since no constraints rule it out, this choice is selected. The interpreter continues, printing the words "a" and "ship" as it reaches them. The morphology routine converts "be" to "is" by checking the number of the current subject and whether any deviation from simple present tense (the default) has been arranged for. Next the interpreter reaches the object, another message element which is translated (via dictionary lookup) as:




"A" and "water-going vehicle" are simply printed when passed through. The treatment of (TRAVEL-MODE SURFACE is more complicated. This message element can be translated in many ways, such as a noun phrase, a bare noun, a verb phrase, and so on. The post-mods label, however will allow only two possibilities: a prepositional phrase or a relative clause. Since the dictionary indicates that relative clauses are preferred over prepositional phrases (for this message) and there are no other constraints blocking it, the relative clause form is chosen:




The interpreter continues on through the relative clause in a similar fashion, eventually

producing "that travels on the surface". (Note, incidentally, that the word "that" is not explicitly in the tree; rather it is printed out by an attached routine associated with the `rel-clause` label.) The complete translation produced by MUMBLE is:

> A ship is a water-going vehicle that travels on the surface.

All three elements of the overall MUMBLE framework have worked together to produce the final English text.

## 3.2 Mumble and the Generation Process

The fundamental constraint that MUMBLE places on generation is, of course, that it is deterministic; this is the guiding principle driving its design, and has been discussed at length elsewhere[McDonald 83b,McDonald 83a]. There are, however, several other interesting constraints that MUMBLE places on the overall design of the generation process:

1. **The information used to guide the generation process is centered around the message formalism, not language.**
   MUMBLE's knowledge of how language expresses things is stored in the dictionary, organized around the possible expressions in the message formalism. Thus the "dictionary" does not list meanings of words, but rather possible (partial) phrases that can express a message. Similarly, the grammar is not set up primarily to express whether a sentence is grammatical but rather to constrain the choice of realizations as the sentence is generated. The grammatical constraints depend in part on the message being translated and the current grammatical environment (i.e. the grammar variables), none of which is preserved in the generated English sentence. Thus it may not be possible to tell whether a given sentence satisfies the grammar's constraints (at least without knowing a message it could have been generated from).

   This organization is a natural consequence of MUMBLE's purpose: to generate text. In language understanding, it is important to know about language, because that is what the system must be able to decipher. MUMBLE is also set up to know about its input, but its input is the message formalism, not natural language. What MUMBLE needs to know is not what a particular word or construction means, but rather when to generate it.

2. **Generation is incremental and top-down.**
   Large messages are partially translated incrementally, with sub-messages left to be translated later as the interpreter reaches them. Thus it is easy for large-scale structure to influence more local decisions, but harder (or impossible) for local structures to constrain the global structure that contains them. This asymetry is a direct consequence of determinism; *something* has to be decided first.

3. **Constraints can be associated both with the surface structure being built up and with possible realizations.**
   Thus the existing structure can constrain what further structures are built, and

candidate structures can constrain where they can be placed. This allows some of the bidirectionality that would seem to be ruled out by determinism. For example; transitive verbs can insist on only being used with direct objects, and verb phrases with direct objects can insist on getting transitive verbs. Note though that the decision to use a transitive verb phrase would still be made first, before the verb was selected.

4. **Constraints are largely local, with all global constraints anticipated in advance.**
   Most constraints are handled locally by constraint predicates that are attached to the surface structure tree or to the possible realization. Any global constraints must have been anticipated and prepared for, either by passing information down to the local node as the tree is traversed, or by storing the information in globally accessable grammar variables. Furthermore, all constraints are still locally enforced; global information can only constrain decisions if there are local constraints that use it.

# 4 Conclusion

The new MUMBLE-based tactical component has been very successful; it produces equivalent English text approximately 60 times faster than TEXT's old tactical component. Its construction, however, required modifications to each of the three parts of MUMBLE: the dictionary needed new entries for the new types of messages that TEXT produced; the grammar needed expansion to handle additional constructions and to implement new constraints that were needed for TEXT; and the interpreter was modified to handle a new criterion for noun phrase number. Furthermore, the new component sheds some light on how MUMBLE organizes the generation process and the consequences of its commitment to determinstic generation.

# References


[Bossie 81] Bossie, Steve. *A Tactical Component for Text Generation: Sentence Generation Using a Functional Grammar.* Technical Report MS-CIS-81-5, CIS Department, University of Pennsylvania, Philadelphia, PA, 1981.

[Karlin 85] Karlin, Robin. *Romper Mumbles.* Technical Report MS-CIS-85-41, CIS Department, University of Pennsylvania, Philadelphia, PA, 1985.

[Kay 79] Kay, Martin. Functional Grammar. In *Proceedings of the 5th Annual Meeting of the Berkeley Linguistic Society.* 1979.

[McDonald 83a] McDonald, David D. Description Directed Control: Its Implications for Natural Language Generation. In Brady, N. (editor), *Computational Linguistics*, pages 111–129. Pergamon Press, 1983.

[McDonald 83b] McDonald, David D. Natural Language Generation as a Computational Problem. In Brady, M. and Berwick, Bob (editors), *Computational Models of Discourse*, pages 209–265. MIT Press, 1983.

[McDonald 85] McDonald, David D. and Pustejovsky, James. TAGs as a Grammatical Formalism for Generation. In *Proceedings of the 23rd Annual Meeting of the ACL*, Association for Computational Linguistics, pages 94–103. Chicago, 1985.

[McKeown 85] McKeown, Kathleen R. *TEXT GENERATION: Using Discourse Strategies and Focus Constraints to Generate Natural Language*. Cambridge University Press, 1985.

[Sidner 83] Sidner, C. L. Focusing in the Comprehension of Definite Anaphora. In Brady, M. and Berwick, Bob (editors), *Computational Models of Discourse*, pages 267–329. MIT Press, 1983.

# SOME COMPUTATIONAL PROPERTIES OF TREE ADJOINING GRAMMARS*

K. Vijay-Shankar and Aravind K. Joshi

Department of Computer and Information Science
Room 268 Moore School/D2
University of Pennsylvania
Philadelphia, PA 19104

## ABSTRACT

Tree Adjoining Grammar (TAG) is a formalism for natural language grammars. Some of the basic notions of TAG's were introduced in [Joshi,Levy, and Takahashi 1975] and by [Joshi, 1983]. A detailed investigation of the linguistic relevance of TAG's has been carried out in [Kroch and Joshi,1985]. In this paper, we will describe some new results for TAG's, especially in the following areas: (1) parsing complexity of TAG's, (2) some closure results for TAG's, and (3) the relationship to Head grammars.

## 1. INTRODUCTION

Investigation of constrained grammatical systems from the point of view of their linguistic adequacy and their computational tractability has been a major concern of computational linguists for the last several years. Generalized Phrase Structure grammars (GPSG), Lexical Functional grammars (LFG), Phrase Linking grammars (PLG), and Tree Adjoining grammars (TAG) are some key examples of grammatical systems that have been and still continue to be investigated along these lines.

Some of the basic notions of TAG's were introduced in [Joshi, Levy, and Takahashi,1975] and [Joshi,1983]. Some preliminary investigations of the linguistic relevance and some computational properties were also carried out in [Joshi,1983]. More recently, a detailed investigation of the linguistic relevance of TAG's were carried out by [Kroch and Joshi,1985].

In this paper, we will describe some new results for TAG's, especially in the following areas: (1) parsing complexity of TAG's, (2) some closure results for TAG's, and (3) the relationship to Head grammars. These topics will be covered in Sections 3, 4, and 5 respectively. In section 2, we will give an introduction to TAG's. In section 6, we will state some properties not discussed here. A detailed exposition of these results is given in [Vijay-Shankar and Joshi,1985].

*This work was partially supported by NSF Grants MCS-8210116-CER, MCS-82-07294. We want to thank Carl Pollard, Kelly Roach, David Searl, and David Weir. We have benefited enormously by valuable discussions with them.

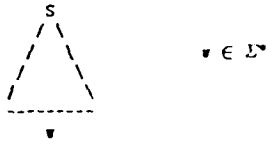
## 2. TREE ADJOINING GRAMMARS--TAG's

We now introduce tree adjoining grammars (TAG's). TAG's are more powerful than CFG's, both weakly and strongly.[1] TAG's were first introduced in [Joshi, Levy, and Takahashi,1975] and [Joshi,1983]. We include their description in this section to make the paper self-contained.

We can define a tree adjoining grammar as follows. A tree adjoining grammar G is a pair (I,A) where I is a set of initial trees, and A is a set of auxiliary trees.

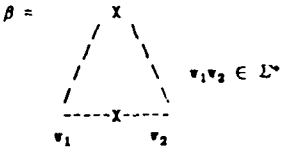
A tree $\alpha$ is an initial tree if it is of the form

$\alpha =$




That is, the root node of $\alpha$ is labelled S and the frontier nodes are all terminal symbols. The internal nodes are all non-terminals A tree $\beta$ is an auxiliary tree if it is of the form

$\beta =$




That is, the root node of $\beta$ is labelled with a non-terminal X and the frontier nodes are all labelled with terminals symbols except one which is labelled X. The node labelled by X on the frontier will be called the foot node of $\beta$. The frontiers of initial trees belong to $\Sigma^*$, whereas the frontiers of the auxiliary trees belong to $\Sigma^* N \Sigma^* \cup \Sigma^+ N \Sigma^*$.

We will now define a composition operation called adjoining, (or adjunction) which composes an auxiliary tree $\beta$ with a tree $\gamma$. Let $\gamma$ be a tree with a node n labelled X and let $\beta$ be an auxiliary tree with the root labelled with the same symbol X. (Note that $\beta$ must have, by definition, a node (and only one) labelled X on the frontier.)

[1] Grammars G1 and G2 are weakly equivalent if the string language of G1, L(G1) = the string language of G2, L(G2). G1 and G2 are strongly equivalent if they are weakly equivalent and for each w in L(G1) = L(G2), both G1 and G2 assign the same structural description to w. A grammar G is weakly adequate for a (string) language L, if L(G) = L. G is strongly adequate for L if L(G) = L and for each w in L, G assigns an "appropriate" structural description to w. The notion of strong adequacy is undoubtedly not precise because it depends on the notion of appropriate structural descriptions

Adjoining can now be defined as follows. If $\beta$ is adjoined to $\gamma$ at the node n then the resulting tree $\gamma_1'$ is as shown in Fig. 2.1 below.

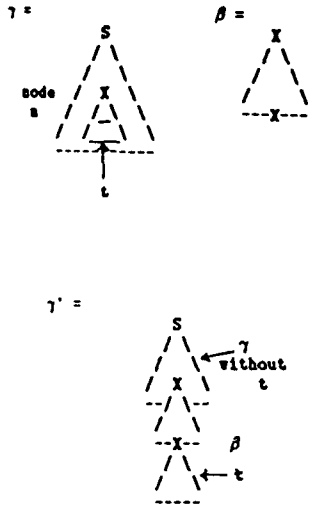



Figure 2.1

The tree t dominated by X in $\gamma$ is excised, $\beta$ is inserted at the node n in $\gamma$ and the tree t is attached to the foot node (labelled X) of $\beta$, i.e., $\beta$ is inserted or adjoined to the node n in $\gamma$ pushing t downwards. Note that adjoining is not a substitution operation.

We will now define

T(G): The set of all trees derived in G starting from initial trees in I. This set will be called the tree set of G.

L(G): The set of all terminal strings which appear in the frontier of the trees in T(G). This set will be called the string language (or language) of G. If L is the string language of a TAG G then we say that L is a Tree-Adjoining Language (TAL). The relationship between TAG's , context-free grammars, and the corresponding string languages can be summarized as follows ([Joshi, Levy, and Takahashi, 1975], [Joshi, 1983]).

Theorem 2.1: For every context-free grammar, G', there is an equivalent TAG, G, both weakly and strongly.

Theorem 2.2: For every TAG, G, we have the following situations:

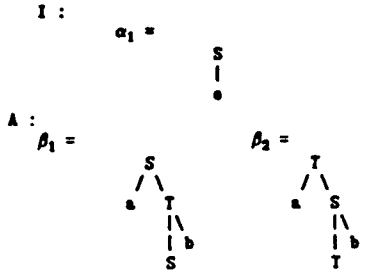
a. L(G) is context-free and there is a context-free grammar G' that is strongly (and therefore weakly) equivalent to G.

b. L(G) is context-free and there is no context-free grammar G' that is equivalent to G. Of course, there must be a context-free grammar that is weakly equivalent to G.

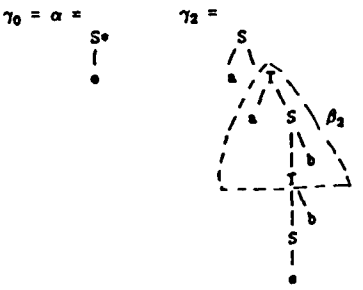
c. L(G) is strictly context-sensitive. Obviously in this case, there is no context-free grammar that is weakly equivalent to G.

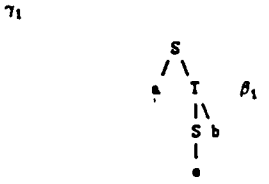
Parts (a) and (c) of Theorem 2.2 appear in ([Joshi, Levy, and Takahashi, 1975]). Part (b) is implicit in that paper, but it is important to state it explicitly as we have done here because of its linguistic significance. Example 2.1 illustrates part (a). We will now illustrate parts (b) and (c).

Example 2.2: Let G = (I,A) where




Let us look at some derivations in G.




$\gamma_1 = \gamma_0$ with $\beta_1$ adjoined at S as indicated in $\gamma_0$.

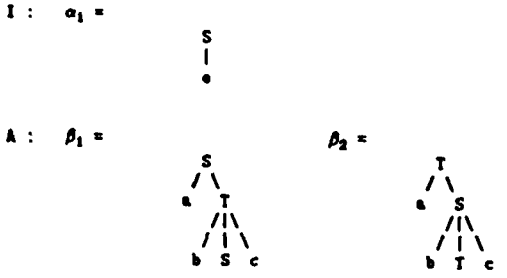
$\gamma_2 = \gamma_1$ with $\beta_2$ adjoined at T as indicated in $\gamma_2$.

Clearly, L(G), the string language of G is

$$L = \{a^n e b^n / n \geq 0\}$$

which is a context-free language. Thus, there must exist a context-free grammar, G', which is at least weakly equivalent to G. It can be shown however that there is no context-free grammar G' which is strongly equivalent to G, i.e., T(G) = T(G'). This follows from the fact that the set T(G) (the tree set of G) is non-recognizable, i.e., there is no finite state bottom-up tree automaton that can recognize precisely T(G). Thus a TAG may generate a context-free language, yet assign structural descriptions to the strings that cannot be assigned by any context-free grammar.

Example 2.3: Let G = (I,A) where



The precise definition of L(G) is as follows:

$L(G) = L_1 = \{ w \in c^n / n \geq 0$, w is a string of a's and b's such that

(1) the number of a's = the number of b's = n, and

(2) for any initial substring of w, the number of a's $\geq$ the number of b's. $\}$

$L_1$ is a strictly context-sensitive language (i.e., a context-sensitive language that is not context-free). This can be shown as follows. Intersecting L with the regular language $a^* b^* c^*$ results in the language

$$L_2 = \{ a^n b^n c^n / n \geq 0 \} = L_1 \cap a^* b^* c^*$$

$L_2$ is well-known strictly context-sensitive language. The result of intersecting a context-free language with a regular language is always a context-free language; hence, $L_1$ is not a context-free language. It is thus a strictly context-sensitive language. Example 2.3 thus illustrates part (c) of Theorem 2.2.

TAG's have more power than CFG's. However, the extra power is quite limited. The language $L_1$ has equal number of a's, b's and c's; however, the a's and b's are mixed in a certain way. The language $L_2$ is similar to $L_1$, except that a's come before all b's. TAG's as defined so far are not powerful enough to generate $L_2$. This can be seen as follows. Clearly, for any TAG for $L_2$, each initial tree must contain equal number of a's, b's and c's (including zero), and each auxiliary tree must also contain equal number of a's, b's and c's. Further in each case the a's must precede the b's. Then it is easy to see from the grammar of Example 2.3, that it will not be possible to avoid getting the a's and b's mixed. However, $L_2$ can be generated by a TAG with local constraints (see Section 2.1) The so-called copy language.

$$L = \{ w e w / w \in \{a,b\}^* \}$$

also cannot be generated by a TAG, however, again, with local constraints. It is thus clear that TAG's can generate more than context-free languages. It can be shown that TAG's cannot generate all context-sensitive languages [Joshi ,1984].

Although TAG's are more powerful than CFG's, this extra power is highly constrained and apparently it is just the right kind for characterizing certain structural descriptions. TAG's share almost all the formal properties of CFG's (more precisely, the corresponding classes of languages), as we shall see in section 4 of this paper and [Vijay-Shankar and Joshi,1985]. In addition,the string languages of TAG's can also be parsed in polynomial time, in particular in $O(n^6)$. The parsing algorithm is described in detail in section 3.

### 2.1. TAG's with Local Constraints on Adjoining

The adjoining operation as defined in Section 2.1 is "context-free". An auxiliary tree, say,

$\beta =$


is adjoinable to a tree t at a node, say, n, if the label of that node is X. Adjoining does not depend on the context (tree context) around the node n. In this sense, adjoining is context-free.

In [Joshi ,1983], local constraints on adjoining similar to those investigated by [Joshi and Levy ,1977] were considered.These are a generalization of the context-sensitive constraints studied by [Peters and Ritchie ,1969]. It was soon recognized, however, that the full power of these constraints was never fully utilized, both in the linguistic context as well as in the "formal languages" of TAG's. The so-called proper analysis contexts and domination contexts (as defined in [Joshi and Levy ,1977]) as used in [Joshi ,1983] always turned out to be such that the context elements were always in a specific elementary tree i.e., they were further localized by being in the same elementary tree. Based on this observation and a suggestion in [Joshi, Levy and Takahashi ,1975], we will describe a new way of introducing local constraints. This approach not only captures the insight stated above, but it is truly in the spirit of TAG's. The earlier approach was not so, although it was certainly adequate for the investigation in [Joshi ,1983]. A precise characterization of that approach still remains an open problem.

G = (I,A) be a TAG with local constraints if for each elementary tree $t \in I \cup A$, and for each node, n, in t, we specify the set $\beta$ of auxiliary trees that can be adjoined at the node n. Note that if there is no constraint then all auxiliary trees are adjoinable at n (of course, only those whose root has the same label as the label of the node n). Thus, in general, $\beta$ is a subset of the set of all the auxiliary trees adjoinable at n.

We will adopt the following conventions.

1. Since, by definition, no auxiliary trees are adjoinable to a node labelled by a terminal symbol, no constraint has to be stated for node labelled by a terminal.

2. If there is no constraint, i.e., all auxiliary trees (with the appropriate root label) are adjoinable at a node, say, n, then we will not state this explicitly.

3. If no auxiliary trees are adjoinable at a node n, then we will write the constraint as $(\phi)$, where $\phi$ denotes the null set.

4. We will also allow for the possibility that for a node at least one adjoining is obligatory, of course, from the set of all possible auxiliary trees adjoinable at that node.

Hence, a TAG with local constraints is defined as follows. G = (I, A) is a TAG with local constraints if for each node, n. in each tree t, be specify one (and only one) of the following constraints.

1. Selective Adjoining (SA:) Only a specified subset of the set of all auxiliary trees are adjoinable at n. SA is written as (C), where C is a subset of the set of all auxiliary trees adjoinable at n.

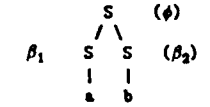
   If C equals the set of all auxiliary trees adjoinable at n, then we do not explicitly state this at the node n.

2. Null Adjoining (NA:) No auxiliary tree is adjoinable at the node N. NA will be written as $(\phi)$.

3. Obligating Adjoining (OA:) At least one (out of all the auxiliary trees adjoinable at n) must be adjoined at n. OA is written as (OA), or as O(C) where C is a subset of the set of all auxiliary trees adjoinable at n.

Example 2.4: Let G = (I,A) be a TAG with local constraints where

I: $\alpha =$

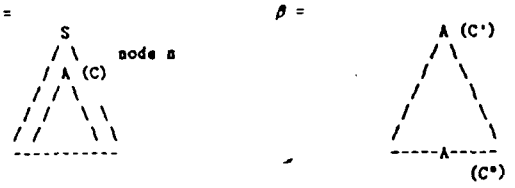


A: $\beta_1$ = $\beta_2$ =


In $\alpha_1$ no auxiliary trees can be adjoined to the root node. Only $\beta_1$ is adjoinable to the left S node at depth 1 and only $\beta_2$ is adjoinable to the right S node at depth 1. In $\beta_1$ only $\beta_1$ is adjoinable at the root node and no auxiliary trees are adjoinable at the foot node. *Similarly for* $\beta_2$.

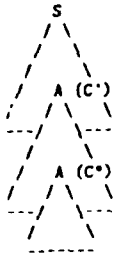
We must now modify our definition of adjoining to take care of the local constraints. given a tree $\gamma$ with a node, say, u, labelled A and given an auxiliary tree, say, $\beta$, with the root node labelled A, we define adjoining as follows. $\beta$ is adjoinable to $\gamma$ at the node n if $\beta \in \beta$, where $\beta$ is the constraint associated with the node n in $\gamma$. The result of adjoining $\beta$ to $\gamma$ will be as defined in earlier, except that the constraint C associated with n will be replaced by C', the constraint associated with the root node of$\beta$ and by C*, the constraint associated with the foot node of $\beta$. Thus, given

$\gamma$ = $\beta$ =




The resultant tree $\gamma'$ is

$\gamma'$ =




We also adopt the convention that any derived tree with a node which has an OA constraint associated with it will not be included in the tree set associated with a TAG, G. The string language L of G is then defined as the set of all terminal strings of all trees derived in G (starting with initial trees) which have no OA constraints left-in them.

Example 2.5: Let G = (I,A) be a TAG with local constraints where

I : $\alpha$ =




A : $\beta$ =

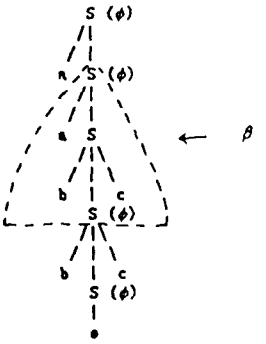

There are no constraints in $\alpha_1$. In $\beta$ no auxiliary trees are adjoinable at the root node and the foot node and for the center S node there are no constraints.

Starting with $\alpha_1$ and adjoining $\beta$ to $\alpha_1$ at the root node we obtain

$\gamma$ =


Adjoining $\beta$ to the center S node (the only node at which adjunction can be made) we have

$\gamma'$ =




It is easy to see that G generates the string language

$$L = \{ a^n b^n c^n / n \geq 0 \}$$

Other languages such as $L' = \{a^{n^2} | n \geq 1\}$, $L^* = \{a^{n^2} | n \geq 1\}$ also cannot be generated by TAG's. This is because the strings of a TAL grow linearly (for a detailed definite of the property called "contact growth" property, see [Joshi ,1983].

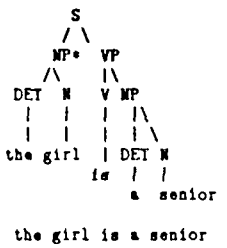
For those familiar with [Joshi, 1983], it is worth pointing out that the SA constraint is only abbreviating, i.e., it does not affect the power of TAG's. The NA and OA constraints however do affect the power of TAG's. This way of looking at local constraints has only greatly simplified their statement, but it has also allowed us to capture the insight that the 'locality' of the constraint is statable in terms of the elementary trees themselves!

### 2.3. Simple Linguistic Examples

We now give a couple of linguistic examples. Readers may refer to [Kroch and Joshi, 1985] for details.

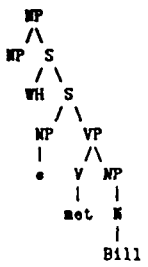
1. Starting with $\gamma_1 = \alpha_1$ which is an initial tree and then adjoining $\beta_1$ (with appropriate lexical insertions) at the indicated node in $\alpha_1$, we obtain $\gamma_2$.

$\gamma_1 = \alpha_1 =$

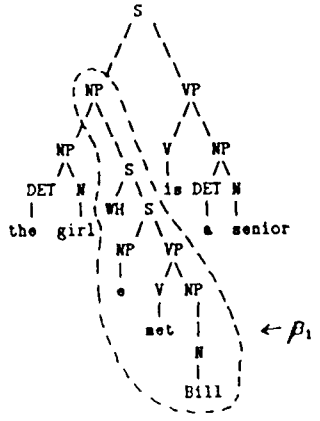



the girl is a senior

$\beta_1 =$



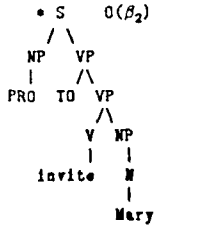

$\gamma_2 =$




The girl who met Bill is a senior

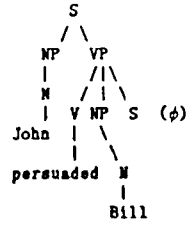
2. Starting with the initial tree $\gamma_1 = \alpha_2$ and adjoining $\beta_2$ at the indicated node in $\alpha_2$ we obtain $\gamma_2$.

$\gamma_1 = \alpha_2 =$



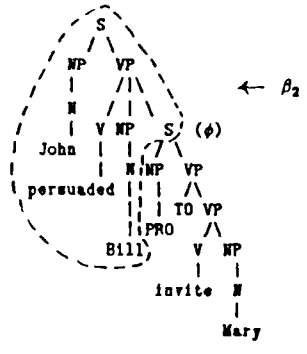

PRO to invite Mary

$\beta_2 =$




John persuaded Bill S

$\gamma_2 =$




John persuaded Bill to invite Mary

Note that the initial tree $\alpha_2$ is not a matrix sentence. In order for it to become a matrix sentence, it must undergo an adjunction at its root node, for example, by the auxiliary tree $\beta_2$ as shown above. Thus, for $\alpha_2$ we will specify a local constraint O($\beta_2$) for the root node, indicating that $\alpha_2$ requires for it to undergo an adjunction at the root node by an auxiliary tree $\beta_2$. In a fuller grammar there will be, of course, some alternatives in the scope of O( ).

## 3. PARSING TREE-ADJOINING LANGUAGES

### 3.1. Definitions

We will give a few additional definitions. These are not necessary for defining derivations in a TAG as defined in section 2. However, they are introduced to help explain the parsing algorithm and the proofs for some of the closure properties of TAL's.

**DEFINITION 3.1** Let $\gamma,\gamma'$ be two trees.We say $\gamma \vdash \gamma'$ if in $\gamma$ we adjoin an auxiliary tree to obtain $\gamma'$.
$\vdash^*$ is the reflexive,transitive closure of $\vdash$.

**DEFINITION 3.2** $\gamma'$ is called a derived tree if $\gamma \vdash^* \gamma'$ for some elementary tree $\gamma$.
We then say $\gamma' \in D(\gamma)$.

The frontier of any derived tree $\gamma$ belongs to either $\Sigma^* N \Sigma^+ \cup \Sigma^+ N \Sigma^*$ if $\gamma \in D(\beta)$ for some auxiliary tree $\beta$, or to $\Sigma^*$ if $\gamma \in D(\alpha)$ for some initial tree $\alpha$. Note if $\gamma \in D(\alpha)$ for some initial tree $\alpha$, then $\gamma$ is also a sentential tree.

If $\beta$ is an auxiliary tree, $\gamma \in D(\beta)$ and the frontier of $\gamma$ is $w_1$ X $w_2$ (X is a nonterminal,$w_1,w_2 \in \Sigma^*$) then the leaf node having this non-terminal symbol X at the frontier is called the foot of $\gamma$.

Sometimes we will be loosely using the phrase "adjoining with a derived tree" $\gamma \in D(\beta)$ for some auxiliary tree $\beta$. What we mean is that suppose we adjoin $\beta$ at some node and then adjoin within $\beta$ and so on, we can derive the desired derived tree $\in D(\beta)$ which uses the same adjoining sequence and use this resulting tree to "adjoin" at the original node.

### 3.2. The Parsing Algorithm

The algorithm, we present here to parse Tree-Adjoining Languages (TALs), is a modification of the CYK algorithm (which is described in detail in [Aho and Ullman,1973]), which uses a dynamic programming technique to parse CFL's. For the sake of making our description of the parsing algorithm simpler, we shall present the algorithm for parsing without considering local constraints. We will later show how to handle local constraints.

We shall assume that any node in the elementary trees in the grammar has atmost two children. This assumption can be made without any loss of generality, because it can be easily shown that for any TAG G there is an equivalent TAG $G_1$ such that any node in any elementary tree in $G_1$ has atmost two children. A similar assumption is made in CYK algorithm. We use the terms ancestor and descendant, throughout the paper as a transitive and reflexive relation, for example, the foot node may be called the ancestor of the foot node.

The algorithm works as follows. Let $a_1 \ldots a_n$ be the input to be parsed. We use a four-dimensional array A; each element of the array contains a subset of the nodes of derived trees. We say a node X of a derived tree $\gamma$ belongs to A[i,j,k,l] if X dominates a sub-tree of $\gamma$ whose frontier is given by either $a_{i+1} \ldots a_j$ Y $a_{k+1} \ldots a_n$ (where the foot node of $\gamma$ is labelled by Y) or $a_{i+1} \ldots a_n$ (i.e., j = k. This

corresponds to the case when $\gamma$ is a sentential tree). The indices (i,j,k,l) refer to the positions between the input symbols and range over 0 through n. If i = 5 say, then it refers to the gap between $a_5$ and $a_6$.

Initially, we fill A[i,i+1,i+1,i+1] with those nodes in the frontier of the elementary trees whose label is the same as the input $a_{i+1}$ for $0 \leq i \leq n-1$. The foot nodes of auxiliary trees will belong to all A[i,i,j,j], such that $i \leq j$.

We are now in a position to fill in all the elements of the array A. There are five cases to be considered.

Case 1. We know that if a node X in a derived tree is the ancestor of the foot node, and node Y is its right sibling, such that $X \in A[i,j,k,l]$ and $Y \in A[l,m,m,n]$, then their parent, say, Z should belong to A[i,j,k,n], see Fig 3.1a.

Case 2. If the right sibling Y is the ancestor of the foot node such that it belongs to A[l,m,n,p] and its left sibling X belongs to A[i,j,j,l], then we know that the parent Z of X and Y belongs to A[i,m,n,p], see Fig 3.1b

Case 3. If neither X nor its right sibling Y are the ancestors of the foot node ( or there is no foot node) then if $X \in A[i,j,j,l]$ and $Y \in A[l,m,m,n]$ then their parent Z belongs to A[i,j,j,n].

Case 4. If a node Z has only one child X, and if $X \in A[i,j,k,l]$, then obviously $Z \in A[i,j,k,l]$.

Case 5. If a node $X \in A[i,j,k,l]$, and the root Y of a derived tree $\gamma$ having the same label as that of X, belongs to A[m,i,l,n], then adjoining $\gamma$ at X makes the resulting node to be in A[m,j,k,n], see Fig 3.1c.

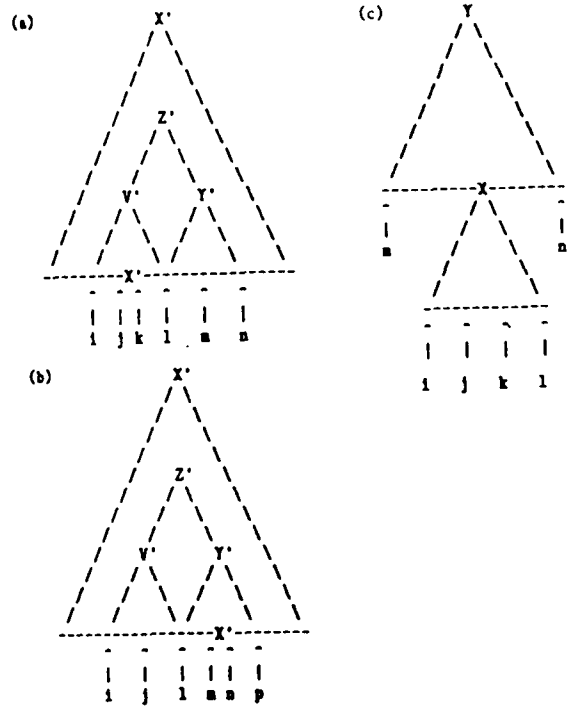



Figure 3.1

Although we have stated that the elements of the array contain a subset of the nodes of derived trees, what really goes in there are the addresses of nodes in the elementary trees. Thus the the size of any set is bounded by a constant, determined by the grammar. It is hoped that the presentation of the algorithm below will make it clear why we do so.

### 3.3. The algorithm

The complete algorithm is given below

```
Step 1 For i=0 to n-1 step 1 do

Step 2     put all nodes in the frontier of elementary
trees whose label is a_{i+1} in A[i,i+1,i+1,i+1].

Step 3 For i=0 to n-1 step 1 do

Step 4   for j=i to n-1 step 1 do

Step 5      put foot nodes of all auxiliary trees in
A[i,i,j,j]

Step 6  For l=0 to n step 1 do

Step 7     For i=l to 0 step -1 do

Step 8        For j=i to l step 1 do

Step 9           For k=l to j step -1 do

Step 10             do Case 1

Step 11             do Case 2

Step 12             do Case 3

Step 13             do Case 5

Step 14             do Case 4

Step 15  Accept if root of some initial tree ∈ A[0,j,j,n],
 0 ≤ j ≤ n
```

where,

(a) Case 1 corresponds to situation where the left sibling is the ancestor of the foot node. The parent is put in A[i,j,k,l] if the left sibling is in A[i,j,k,m] and the right sibling is in A[m,p,p,l], where $k \leq m < l$, $m \leq p$, $p \leq l$. Therefore Case 1 is written as

```
   For m=k to l-1 step 1 do

    for p= m to l step 1 do

      if there is a left sibling in A[i,j,k,m] and the
right sibling in A[m,p,p,l] satisfying appropriate
restrictions then put their parent
in A[i,j,k,l].
```

(b) Case 2 corresponds to the case where the right sibling is the ancestor of the foot node. If the left sibling is in A[i,m,m,p] and the right sibling is in A[p,j,k,l], $i \leq m < p$ and $p \leq j$, then we put their parent in A[i,j,k,l]. This may be written as

```
  For m=i to j-1 step 1 do

   For p=m+1 to j step 1 do

     for all left siblings in A[i,m,m,p] and right
siblings
in A[p,j,k,l] satisfying appropriate restrictions put
their parents
in A[i,j,k,l].
```

(c) Case 3 corresponds to the case where neither children are ancestors of the foot node. If the left sibling $\in$ A[i,j,j,m] and the right sibling $\in$ A[m,p,p,l] then we can put the parent in A[i,j,j,l] if it is the case that ( $i < j \leq m$ or $i \leq j < m$) and ( $m < p \leq l$ or $m \leq p < l$ ). This may be written as

```
for m = j to l-1 step 1 do
    for p = j to l step 1 do
        for all left siblings in A[i,j,j,m] and
```

right siblings in A[m,p,p,l] satisfying the appropriate restrictions put their parent in A[i,j,j,l].

(e) Case 5 corresponds to adjoining. If X is a node in A[m,j,k,p] and Y is the root of a auxiliary tree with same symbol as that of X, such that Y is in A[i,m,p,l] (($i \leq m \leq p < l$ or $i < m \leq p \leq l$) and ($m < j \leq k \leq p$ or $m \leq j \leq k < p$)). This may be written as

```
for m = i to j step 1 do
    for p = m to l step 1 do
        if a node X ∈ A[m,j,k,p] and the root of
auxiliary tree is in A[i,m,p,l] then put X in A[i,j,k,l]
```

Case 4 corresponds to the case where a node Y has only one child X If X $\in$ A[i,j,k,l] then put Y in A[i,j,k,l]. Repeat Case 4 again if Y has no siblings.

### 3.4. Complexity of the Algorithm

It is obvious that steps 10 through 15 (cases a-e) are completed in $O(n^2)$, because the different cases have at most two nested for loop statements, the iterating variables taking values in the range 0 through n. They are repeated atmost $O(n^4)$ times, because of the four loop statements in steps 6 through 9. The initialization phase (steps 1 through 5) has a time complexity of $O(n + n^2) = O(n^2)$. Step 15 is completed in O(n). Therefore, the time complexity of the parsing algorithm is $O(n^6)$.

### 3.5. Correctness of the Algorithm

The main issue in proving the algorithm correct, is to show that while computing the contents of an element of the array A, we must have already determined the contents of other elements of the array needed to correctly complete this entry. We can show this inductively by considering each case individually. We give an informal argument below.

Case 1: We need to know the contents of A[i,j,k,m], A[m,p,p,l] where $m < l$, $i < m$, when we are trying to compute the contents of A[i,j,k,l]. Since l is the variable itererated in the outermost loop (step 6), we can assume (by induction hypothesis) that for all $m < l$ and for all p,q,r, the contents of A[p,q,r,m] are already computed. Hence, the contents of A[i,j,k,m] are known. Similarly, for all $m > i$, and for all p,q, and $r \leq l$, A[m,p,q,r] would have been computed. Thus, A[m,p,p,l] would also have been computed.

Case 2: By a similar reasoning, the contents of A[i,m,m,p] and A[p,j,k,l] are known since $p < l$ and $p > i$.

Case 3: When we are trying to compute the contents of some A[i,j,j,l], we need to know the nodes in A[i,j,j,p] and A[p,q,q,l]. Note $j > i$ or $j < l$. Hence, we know that the contents of A[i,j,j,p] and A[p,q,q,l] would have been computed already.

Case 5: The contents of A[i,m,p,l] and A[m,j,k,p] must be known in order to compute A[i,j,k,l], where ( $i \leq m \leq p < l$ or $i < m \leq p \leq l$ ) and ( $m \leq j \leq k < p$ or $m < j \leq k \leq p$ ). Since either $m > i$ or $p < l$, contents of A[m,j,k,p] will be known. Similarly, since either $m < j$ or $k < p$, the contents of A[i,m,p,l] would have been computed.

### 3.6. Parsing with Local Constraints

So far,we have assumed that the given grammar has no local constraints. If the grammar has local constraints, it is easy to modify the above algorithm to take care of them. Note that in Case 5, if an adjunction occurs at a node X, we add X again to the element of the array we are computing. This seems to be in contrast with our definition of how to associate local constraints with the nodes in a sentential tree. We should have added the root of the auxiliary tree instead to the element of the array being computed, since so far as the local constraints are concerned,this node decides the local constraints at this node in the derived tree. However, this scheme cannot be adopted in our algorithm for obvious reasons. We let pairs of the form (X,C) belong to elements of the array, where X is as before and C represents the local constraints to be associated with this node.

We then alter the algorithm as follows. If $(X,C_1)$ refers to a node at which we attempt to adjoin with an auxiliary tree (whose root is denoted by $(Y,C_2)$), then adjunction would determined by $C_1$. If adjunction is allowed, then we can add $(X,C_2)$ in the corresponding element of the array. In cases 1 through 4, we do not attempt to add a new element if any one of the children has an obligatory constraint.

Once it has been determined that the given string belongs to the language, we can find the parse in a way similar to the scheme adopted in CYK algorithm.To make this process simpler and more efficient, we can use pointers from the new element added to the elements which caused it to be put there. For example, consider Case 1 of the algorithm (step 10 ). If we add a node Z to A[i,j,k,l], because of the presence of its children X and Y in A[i,j,k,m] and A[m,p,p,l] respectively, then we add pointers from this node Z in A[i,j,k,l] to the nodes X, Y in A[i,j,k,m] and A[m,p,p,l]. Once this has been done, the parse can be found by traversing the tree formed by these pointers.

A parser based on the techniques described above is currently being implemented and will be reported at time of presentation.

## 4. CLOSURE PROPERTIES OF TAG's

In this section, we present some closure results for TALs. We now informally sketch the proofs for the closure properties. Interested readers may refer to [Vijay-Shankar and Joshi,1985] for the complete proofs.

### 4.1. Closure under Union

Let $G_1$ and $G_2$ be two TAGs generating $L_1$ and $L_2$ respectively. We can construct a TAG G such that $L(G)=L_1 \cup L_2$.

Let $G_1 = ( I_1, A_1, N_1, S )$, and $G_2 = ( I_2, A_2, N_2, S )$. Without loss of generality, we may assume that the $N_1 \cap N_2 = \phi$. Let $G = ( I_1 \cup I_2, A_1 \cup A_2, N_1 \cup N_2, S )$. We claim that $L(G) = L_1 \cup L_2$.

Let $x \in L_1 \cup L_2$. Then $x \in L_1$ or $x \in L_2$. If $x \in L_1$, then it must be possible to generate the string x in G , since $I_1$ , $A_1$ are in G. Hence $x \in L(G)$. Similarly if $x \in L_2$ , we can show that $x \in L(G)$. Hence $L_1 \cup L_2 \subseteq L(G)$. If $x \in L(G)$, then x is derived using either only $I_1$, $A_1$ or only $I_2$, $A_2$ since $N_1 \cap N_2 = \phi$. Hence, $x \in L_1$ or $x \in L_2$. Thus, $L(G) \subseteq L_1 \cup L_2$. Therefore, $L(G) = L_1 \cup L_2$.

### 4.2. Closure under Concatenation

Let $G_1 = (I_1, A_1, N_1, S_1)$, $G_2 = (I_2, A_2, N_2, S_2)$ be two TAGs generating $L_1$, $L_2$ respectively, such that $N_1 \cap N_2 = \phi$. We can construct a TAG $G = (I, A, N, S)$ such that $L(G) = L_1 . L_2$. We choose S such that S is not in $N_1 \cup N_2$. We let $N = N_1 \cup N_2 \cup \{S\}$, $A = A_1 \cup A_2$. For all $t_1 \in I_1$, $t_2 \in I_2$, we add $t_{12}$ to I, as shown in Fig 4.2.1. Therefore, $I = \{ t_{12} / t_1 \in I_1, t_2 \in I_2 \}$, where the nodes in the subtrees $t_1$ and $t_2$ of the tree $t_{12}$ have the same constraints associated with them as in the original grammars $G_1$ and $G_2$. It is easy to show that $L(G) = L_1 . L_2$, once we note that there are no auxiliary trees in G rooted with the symbol S, and that $N_1 \cap N_2 = \phi$.

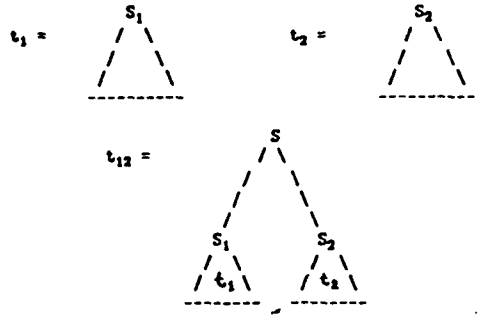



Figure 4.2.1

### 4.3. Closure under Kleene Star

Let $G_1 = (I_1, A_1, N_1, S_1)$ be a TAG generating $L_1$. We can show that we can construct a TAG G such that $L(G) = L_1^*$. Let S be a symbol not in $N_1$, and let $N = N_1 \cup \{S\}$. We let the set I of initial trees of G be $\{t_e\}$, where $t_e$ is the tree shown in Fig 4.3a. The set of auxiliary trees A is defined as

$$A = \{t_{1A} / t_1 \in I_1\} \cup A_1.$$

The tree $t_{1A}$ is as shown in Fig 4.3b, with the constraints on the root of each $t_{1A}$ being the null adjoining constraint, no constraints on the foot, and the constraints on the nodes of the subtrees $t_1$ of the trees $t_{1A}$ being the same as those for the corresponding nodes in the initial tree $t_1$ of $G_1$.

To see why $L(G) = L_1^*$, consider $x \in L(G)$. Obviously, the tree derived (whose frontier is given by x ) must be of the form shown in Fig 4.3c, where each $t_i'$ is a sentential tree in $G_1$,such $t_i' \in D(t_i)$, for an initial tree $t_i$ in $G_1$. Thus, $L(G) \subseteq L_1^*$.

On the other hand, if $x \in L_1^*$, then $x = w_1 \ldots w_n$, $w_i \in L_1$ for $1 \leq i \leq n$. Let each $w_i$ then be the frontier of the sentential tree $t_i'$ of $G_1$ such that $t_i' \in D(t_i)$, $t_i \in I_1$. Obviously, we can derive the tree T, using the initial tree $t_e$, and have a sequence of adjoining operations using the auxiliary trees $t_{iA}$ for $1 \leq i \leq n$. From T we can obviously obtain the tree T' the same as given by Fig 4.3c, using only the auxiliary trees in $A_1$. The frontier of T' is obviously $w_1 \ldots w_n$. Hence, $x \in L(G)$. Therefore, $L_1^* \in L(G)$. Thus $L(G) = L_1^*$.

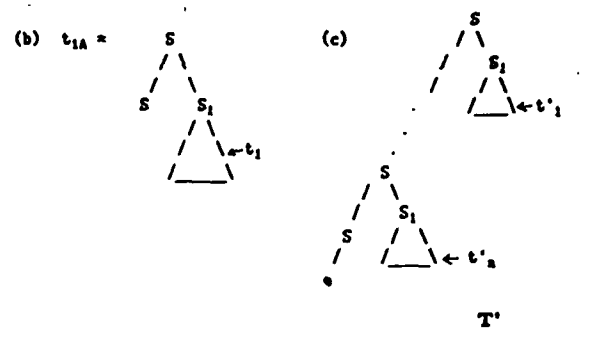




Figure 4.3

### 4.4. Closure under Intersection with Regular Languages

Let $L_T$ be a TAL and $L_R$ be a regular language. Let G be a TAG generating $L_T$ and $M = (Q, \Sigma, \delta, q_0, Q_F)$ be a *finite state automaton* recognizing $L_R$. We can construct a grammar G and will show that $L(G_1) = L_T \cap L_R$.

Let $\alpha$ be an elementary tree in G. We shall associate with each node a quadruple $(q_1, q_2, q_3, q_4)$ where $q_1, q_2, q_3, q_4 \in Q$. Let $(q_1, q_2, q_3, q_4)$ be associated with a node X in $\alpha$. Let us assume that $\alpha$ is an auxiliary tree, and that X is an ancestor of the foot node of $\alpha$, and hence, the ancestor of the foot node of any derived tree $\gamma$ in $D(\alpha)$. Let Y be the label of the root and foot nodes of $\alpha$. If the frontier of $\gamma$ ($\gamma$ in $D(\alpha)$) is $w_1\ w_2\ Y\ w_3\ w_4$, and the frontier of the subtree of $\gamma$ rooted at Z, which corresponds to the node X in $\alpha$ is $w_2\ Y\ w_3$. The idea of associating $(q_1, q_2, q_3, q_4)$ with X is that it must be the case that $\delta^*(q_1, w_2) = q_2$, and $\delta^*(q_3, w_3) = q_4$. When $\gamma$ becomes a part of the sentential tree $\gamma'$ whose frontier is given by $u\ w_1\ w_2\ v\ w_3\ w_4\ w$, then it must be the case that $\delta^*(q_2, v) = q_3$. Following this reasoning, we must make $q_2 = q_3$, if Z is not the ancestor of the foot node of $\gamma$, or if $\gamma$ is in $D(\alpha)$ for some initial tree $\alpha$ in G.

We have assumed here, as in the case of the parsing algorithm presented earlier, that any node in any elementary tree has atmost two children.

From G we can obtain $G_1$ as follows. For each initial tree $\alpha$, associate with the root the quadruple $(q_0, q, q, q_f)$ where $q_0$ is the initial state of the finite state automaton M, and $q_f \in Q_F$. For each auxiliary tree $\beta$ of G, associate with the root the quadruple $(q_1, q_2, q_3, q_4)$, where $q, q_1, q_2, q_3, q_4$ are some variables which will later be given values from Q. Let X be some node in some elementary tree $\alpha$. Let $(q_1, q_2, q_3, q_4)$ be associated with X. Then, we have to consider the following cases.

Case 1: X has two children Y and Z. The left child Y is the ancestor of the foot node of $\alpha$. Then associate with Y the quadruple ( p, $q_2$, $q_3$, q ), and ( q, r, r, s ) with Z, and associate with X the constraint that only those trees whose root has the quadruple ( $q_1$, p, s, $q_4$ ), among those which were allowed in the original grammar, may be adjoined at this node. If $q_1 \neq p$, or $q_4 \neq s$ , then the constraint associated with X must be made obligatory. If in the original grammar X had an obligatory constraint associated with it then we retain the obligatory constraint regardless of the relationship between $q_1$ and p, and $q_4$ and s. If the constraint associated with X is a null adjoining constraint, we associate ( $q_1$, $q_2$, $q_3$, q ), and ( q, r, r, $q_4$ ) with Y and Z respectively, and associate the null adjoining constraint with X. If the label of Z is a, where $a \in \Sigma$, then we choose s and q such that $\delta$ ( q, a ) = s. In the null adjoining constraint case, q is chosen such that $\delta$ ( q, a ) = $q_4$.

Case 2: This corresponds to the case where a node X has two children Y and Z, with $(q_1,q_2,q_3,q_4)$ associated at X. Let Z ( the right child ) be the ancestor of the the foot node the tree $\alpha$. Then we shall associate $(p,q,q,r)$, $(r,q_2,q_3,s)$ with Y and Z. The associated constraint with X shall be that only those trees among those which were allowed in the orignal grammar may be adjoined provided their root has the quadruple $(q_1,p,s,q_4)$ associated with it. If $q_1 \neq p$ or $q_4 \neq r$ then we make the constraint obligatory. If the original grammar had obligatory constraint we will retain the obligatory constraint. Null constraint in the original grammar will force us to use null constraint and not consider the cases where it is not the case that $q_1 = p$ and $q_4 = s$. If the label of Y is a terminal 'a' then we choose r such that $\delta^*(p,a) = r$. If the constraint at X is a null adjoining constraint, then $\delta^*(q_1,a) = r$.

Case 3: This corresponds to the case where neither the left child Y nor the right child Z of the node X is the ancestor of the foot node of $\alpha$ or if $\alpha$ is a initial tree. Then $q_2 = q_3 = q$. We will associate with Y and Z the quadruples $(p,r,r,q)$ and $(q,s,s,t)$ resp. The constraints are assigned as before , in this case it is dictated by the quadruple $(q_1,p,t,q_4)$. If it is not the case that $q_1 = p$ and $q_4 = t$, then it becomes an OA constraint. The OA and NA constraints at X are treated similar to the previous cases, and so is the case if either Y or Z is labelled by a terminal symbol.

Case 4: If $(q_1,q_2,q_3,q_4)$ is associated with a node X, which has only one child Y, then we can deal with the various cases as follows. We will associate with Y the quadruple $(p,q_2,q_3,s)$ and the constraint that root of the tree which can be adjoined at X should have the quadruple $(q_1,p,s,q_4)$ associated with it among the trees which were allowed in the original grammar, if it is to be adjoined at X. The cases where the original grammar had *null or obligatory* constraint associated with this node or Y is labelled with a terminal symbol, are treated similar to how we dealt with them in the previous cases.

Once this has been done, let $q_1,\ldots,q_m$ be the independent variables for this elementary tree $\alpha$, then we produce as many copies of $\alpha$ so that $q_1,\ldots,q_m$ take all possible values from Q. The only difference among the various copies of $\alpha$ so produced will be constraints associated with the nodes in the trees. Repeat the process for all the elementary trees in $G_1$. Once this has been done and each tree given a unique name we can write the constraints in terms of these names. We will now show why $L(G_1) = L_T \cap L_R$.

Let $w \in L(G_1)$. Then there is a sequence of adjoining operations starting with an initial tree $\alpha$ to derive w. Obviously, $w \in L_T$, also since corresponding to each tree used in deriving w, there is a corresponding tree in G, which differs only in the constraints associated with its nodes. Note, however, that the constraints associated with the nodes in trees in $G_1$ are just a restriction of the corresponding ones in G, or an obligatory constraint where there was none in G. Now, if we can assume ( by induction hypothesis ) that if after n adjoining operations we can derive $\gamma' \in D(\alpha')$, then there is a corresponding tree $\gamma \in D(\alpha)$ in G, which has the same tree structure as $\gamma'$ but differing only in the constraints associated with the corresponding nodes, then if we adjoin at some node in $\gamma'$ to obtain $\gamma_1'$, we can adjoin in $\gamma$ to obtain $\gamma_1$ (corresponding to $\gamma_1'$). Therefore, if w can be derived in $G_1$, then it can definitely be derived in G.

If we can also show that $L(G_1) \subseteq L_R$, then we can conclude that $L(G_1) \subseteq L_T \cap L_R$. We can use induction to prove this. The induction hypothesis is that if all derived trees obtained after $k \leq n$ adjoining operations have the property P then so will the derived trees after n + 1 adjoinings where P is defined as,

Property P: If any node X in a derived tree $\gamma$ has the foot-node of the tree $\beta$ to which X belongs labelled Y as a descendant such that $w_1$ Y $w_2$ is the frontier of the subtree of $\beta$ rooted at X, then if $(q_1,q_2,q_3,q_4)$ had been associated with X, $\delta^*(q_1,w_1) = q_2$ and $\delta^*(q_3,w_2) = q_4$, and if w is the frontier of the subtree under the foot node of $\beta$ in $\gamma$ is then $\delta^*(q_2,w) = q_3$. If X is not the ancestor of the foot node of $\beta$ then the subtree of $\beta$ below is of the form $w_1w_2$. Suppose X has associated with it $(q_1,q,q,q_2)$ then $\delta^*(q_1,w_1) = q$, $\delta^*(q,w_2) = q_2$.

Actually what we mean by an adjoining operation is not necessarily just one adjoining operation but the minimum number so that no obligatory constraints are associated with any nodes in the derived trees. Similarly, the base case need not consider only elementary trees, but the smallest (in terms of the number of adjoining operations) tree starting with elementary trees which has no obligatory constraint associated with any of its nodes. The base case can be seen easily considering the way the grammar was built (it can be shown formally by induction on the height of the tree) The inductive step is obvious. Note that the derived tree we are going to use for adjoining will have the property P, and so will the tree at which we adjoin; the former because of the way we designed the grammar and assigned constraints, and the latter because of induction hypothesis. Thus so will the new derived tree. Once we have proved this, all we have to do to show that $L(G_1) \subseteq L_R$ is to consider those derived trees which are sentential trees and observe that the roots of these trees obey property P.

Now, if a string $x \in L_T \cap L_R$, we can show that $x \in L(G)$. To do that, we make use of the following claim.

Let $\beta$ be an auxiliary tree in G with root labelled Y and let $\gamma \in D(\beta)$. We *claim* that there is a $\beta'$ in $G_1$ with the same structure as $\beta$, such that there is a $\gamma'$ in D(beta())') where $\gamma'$ has the same structure as $\gamma$, such that there is no OA constraint in $\gamma'$. Let X be a node in $\beta_1$ which was used in deriving $\gamma$. Then there is a node X' in $\gamma'$ such that X' belongs to the auxilliary tree $\beta_1'$ (with the same structure as $\beta_1$. There are several cases to consider -

Case 1: X is the ancestor of the foot node of $\beta_1$, such that the frontier of the subtree of $\beta_1$ rooted at X is $w_3Yw_4$ and the frontier of the subtree of $\gamma$ rooted at X is $w_3w_1Zw_2w_4$. Let $\delta^*(q_1,w_3) = q$, $\delta^*(q,w_1) = q_2$, $\delta^*(q_3,w_2) = r$, and $\delta^*(r,w_4) = q_4$. Then X' will have $(q_1,q,r,q_4)$ associated with it, and there will be no OA constraint in $\gamma'$.

Case 2: X is the ancestor of the foot node of $\beta_1$, and the frontier of the subtree of $\beta_1$ rooted at X is $w_3Yw_4$. Let the frontier of the subtree of $\gamma$ rooted at X is $w_3w_1w_2w_4$. Then we claim that X' in $\gamma'$ will have associated with it the quadruple $(q_1,q,r,q_4)$, if $\delta^*(q_1,w_3) = q$, $\delta^*(q,w_1) = p$, $\delta^*(p,w_2) = r$, and $\delta^*(r,w_4) = q_4$.

Case 3: Let the frontier of the subtree of $\beta_1$ (and also $\gamma$) rooted at X is $w_1w_2$. Let $\delta^*(q,w_1) = p$, $\delta^*(p,w_2) = r$. Then X' will have associated with it the quadruple $(q,p,p,r)$.

We shall prove our claim by induction on the number of adjoining operations used to derive $\gamma$. The base case (where $\gamma = \beta$) is obvious from the way the grammar $G_1$ was built. We shall now assume that for all derived trees $\gamma$, which have been derived from $\beta$ using k or less adjoining operations, have the property as required in our claim. Let $\gamma$ be a derived tree in $\beta$ after k adjunctions. By our inductive hypothesis we may assume the existence of the corresponding derived tree $\gamma' \in D(\beta')$ derived in $G_1$. Let X be a node in $\gamma$ as shown in Fig. 4.4.1. Then the node X' in $\gamma'$ corresponding to X will have associated with it the quadruple $(q_1',q_2',q_3',q_4^*)$. Note we are assuming here that the left child Y' of X' is the ancestor of the

foot node of $\beta'$. The quadruples $(q_1', q_2', q_3', p)$ and $(p, p_1, p_1, q_4'')$ will be associated with Y' and Z' (by the induction hypothesis). Let $\gamma_1$ be derived from $\gamma$ by adjoining $\beta_1$ at X as in Fig. 4.4.2. We have to show the existence of $\beta_1'$ in $G_1$ such that the root of this auxiliary tree has associated with it the quadruple $(q, q_1', q_4'', s)$. The existence of the tree follows from induction hypothesis ($k = 0$). We have also got to show that there exists $\gamma_1'$ with the same structure as $\gamma'$ but one that allows $\beta_1'$ to be adjoined at the required node. But this should be so, since from the way we obtained the trees in $G_1$, there will exist $\gamma_1''$ such that $X_1'$ has the quadruple $(q, q_2', q_3', r)$ and the constraints at $X_1'$ are dictated by the quadruple $(q, q_1', q_4'', s)$, but such that the two children of $X_1'$ will have the same quadruple as in $\gamma'$. We can now adjoin $\beta_1'$ in $\gamma_1''$ to obtain $\gamma_1'$. It can be shown that $\gamma_1'$ has the required property to establish our claim.

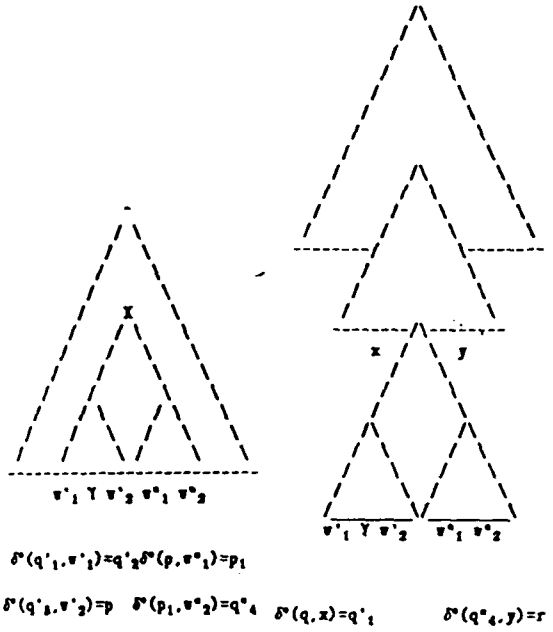



Figure 4.4.1




Figure 4.4.2

Firstly, any node below the foot of $\beta_1'$ in $\gamma_1'$ will satisfy our requirements as they are the same as the corresponding nodes in $\gamma_1''$. Since $\beta_1'$ satisfies the requirement, it is simple to observe that the nodes in $\beta_1'$ will, even after the adjunction of $\beta_1'$ in $\gamma_1''$. However, because the quadruple associated with $X_1'$ are different, the quadruples of the nodes above $X_1'$ must reflect this change. It is easy to check the existence of an auxiliary tree such that the nodes above $X_1'$ satisfy the requirements as stated above. It can also be argued on the basis of the design of grammar $G_1$, that there exists trees which allow this new auxiliary tree to be adjoined at the appropriate place. This then allows us to conclude that there exist a derived tree for each derived tree belongin to $D(\beta)$ as in our claim. The next step is to extend our claim to take into account all derived trees (i.e., including the sentential trees). This can be done in a manner similar to our treatment of derived trees belonging to $D(\beta)$ for some auxiliary tree $\beta$ as above. Of course, we have to consider only the case where the finite state automaton starts from the initial state $q_0$, and reaches some final state $q_f$ on the input which is the frontier of some sentential tree in G. This, then allows us to conclude that $L_T \cap L_R \subseteq L(G_1)$. Hence, $L(G_1) = L_T \cap L_R$.

## 5. HEAD GRAMMARS AND TAG's

In this section, we attempt to show that Head Grammars (HG) are remarkably similar to Tree Adjoining Grammars. It appears that the basic intuition behind the two systems is more or less the same. Head Grammars were introduced in [Pollard,1984], but we follow the notations used in [Roach,1984]. It has been observed that TAG's and HG's share a lot of common formal properties such as almost identical closure results, similar pumping lemma.

Consider the basic operation in Head Grammars - the Head Wrapping operation. A derivation from a non-terminal produces a pair $(i, a_1 \ldots a_i \ldots a_n)$ (a more convenient representation for this pair is $a_1 \ldots a_i a_{i+1} \ldots a_n$ ). The arrow denoter the head of the string, which in turn determines where the string is split up when wrapping operation takes place. For example, consider $X \rightarrow LL_2(A,B)$, and let $A \Rightarrow^* wh_1x$ and $B \Rightarrow^* ug_1v$. Then we say, $X \Rightarrow^* whug_1vx$.

We shall define some functions used in the HG formalism, which we need here. If A derives in 0 or more steps the headed string $\overset{\uparrow}{w}h\overset{\uparrow}{x}$ and B derives $\overset{\uparrow}{u}g\overset{\uparrow}{v}$, then

1) if $X \rightarrow LL_1(A,B)$ is a rule in the grammar then X derives $\overset{\uparrow}{w}hug\overset{\uparrow}{v}x$

2) if $X \rightarrow LL_2(A,B)$ is a rule in the grammar then X derives $\overset{\uparrow}{w}hug\overset{\uparrow}{v}x$

3) if $X \rightarrow LC_1(A,B)$ is a rule in the grammar then X derives $\overset{\uparrow}{w}hxug\overset{\uparrow}{v}$

4) if $X \rightarrow LC_2(A,B)$ is a rule in the grammar then X derives $\overset{\uparrow}{w}hxug\overset{\uparrow}{v}$

Now consider how a derivation in TAGs proceeds -

Let $\beta$ be an auxilliary tree and let $\alpha$ be a sentential tree as in Fig 5.1. Adjoining $\beta$ at the root of the sub-tree $\gamma$ gives us the sentential tree in Fig 5.1. We can, now see how the string whx has "wrapped around" the sub-tree i.e,the string ugv. This seems to suggest that there is something similiar in the role played by the foot in an auxilliary tree and the head in a Head Grammar how the adjoining operations and head-wrapping operations operate on strings. We could say that if X is the root of an auxilliary tree $\beta$ and $a_1 \ldots a_i$ X $a_{i+1} \ldots a_n$ is the frontier of a derived tree $\gamma \in D(\beta)$, then the derivation of $\gamma$ would correspond to a derivation from a non-terminal X to the string $a_1 \ldots a_i \; a_{i+1} \ldots a_n$ in HG and the use of $\gamma$ in some sentential tree would correspond to how the strings $a_1 \ldots a_i$ and $a_{i+1} \ldots a_n$ are used in deriving a string in HL.

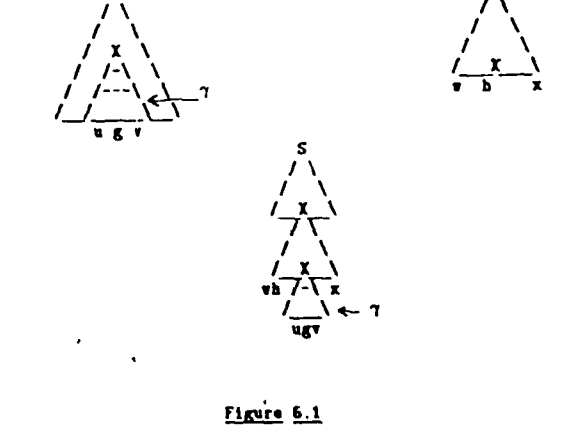



Figure 5.1

Based on this observation, we attempt to show the close relationship of TAL's and HL's. It is more convinient for us to think of the headed string $(i,a_1...a_n)$ as the string $a_1...a_n$ with the head pointing in between the symbols $a_i$ and $a_{i+1}$ rather than at the symbol $a_i$. The definition of the derivation operators can be extended in a straightforward manner to take this into account. However, we can acheive the same effect by considering the definitions of the operators LL,LC,etc. Pollard suggests that cases such as $LL_2(\overline{x},\overline{\lambda})$ be left undefined. We shall assume that if $\overline{x}$ =why then $LL_2(\overline{x},\overline{\lambda}) = wh\overset{\uparrow}{\lambda}y$, $LL_2(\overline{\lambda},\overline{x}) = \overline{x}$, $LC_2(\overline{x},\overline{\lambda}) = x\overset{\uparrow}{\lambda}$, $LC_2(\overline{\lambda},\overline{x}) = \overline{x}$, $LC_1(\overline{x},\overline{\lambda}) = \overline{x}$, and $LC_1(\overline{\lambda},\overline{x}) = \overset{\uparrow}{\lambda}x$.
We, then say that if G is a Head Grammar, then $w_1 = wh\overset{\uparrow}{x}$ belongs to L(G) if and only if S derives the headed string $wh\overset{\uparrow}{x}$ or $wh\overset{\uparrow}{\lambda}x$. With this new definition, we shall show, without giving the proof, that the class of TAL's is contained in the class of HL's. by systematically converting any TAG G to a HG G'. We shall assume, without loss of generality, that the constraints expressed at the nodes of elementary trees of G are -

1) Nothing can be adjoined at a node (NA),

2) Any appropriate tree (symbols at the node and root of the auxilliary tree must match) can be adjoined (AA), or

3) Adjoining at the node is obligatory (OA).

It is easy to show that these constraints are enough, and that selective adjoining can be expressed in terms of these and additional non-terminals. We know give a procedural description of obtaining an equivalent Head Grammar from a Tree-Adjoining Grammar. The procedure works as follows. It is a recursive procedure (Convert_to_HG) which takes in two parameters, the first representing the node on which it is being applied and the second the label appearing on the left-hand side of the HG productions for this node. If X is a nonterminal, for each auxiliary tree $\beta$ whose root has the label X, we obtain a sequence of productions such that the first one has X on the left-hand side. Using these productions, we can derive the string $w_1\overset{\uparrow}{\lambda}w_2$ where a derived tree in $D(\beta)$ has a frontier $w_1Yw_2$. If Y is a node with with label X in some tree where adjoining is allowed, we introduce the productions

Y' -> $LL_2$(X,N') (so that a derived tree with root label X may wrap around the string derived from the subtree below this node)

N' -> $LC_1(A_1,\ldots,A_j)$ (assuming that there are j children of this node and the $i^{th}$ child is the ancestor of the foot node. By calling the procedure recursively for all the j children of Y with $A_k$,k ranging from 1 through j, we can derive from N' the frontier of the subtree below Y)

Y' -> N' ( this is to handle the case where no adjunction takes place at Y)

If G is a TAG then we do the following –

Repeat for every Initial tree

Convert_to_HG(root,S') {S' will be the start symbol of the new Head Grammar}.

Repeat for each Auxilliary tree

Convert_to_HG(root,rootsymbol)

where Convert_to_HG(node,name) is defined as follows

if node is an internal node then

case 1 If the constraint at the node is AA

add productions Sym->$LL_2$(node symbol,N'),

N'->$LC_1(A_1',\ldots,A_1',\ldots,A_j')$

Sym->$LC_1(A_1',\ldots,A_1',\ldots,A_j')$

where N',$A_1'$,$A_2'$,...$A_j'$ are new non-terminal symbols,$A_1,\ldots,A_j$ correspond to the j children of the node and i=1 if foot node is not a descendant of node else =1 such that the $1^{th}$ child of node is ancestor of foot node,j=number of children of node

for k=1 to j step 1 do

Convert_to_HG($k^{th}$ child of node,$A_k'$).

Case 2 The constraint at the node is NA.

Same as Case 1 except don't add the productions
Sym->$LL_1$(node symbol,N'),
N'->$LC_1(A_1',\ldots,A_j')$,

Case 3 The constraint at the node is OA.

Same as Case 1 except that we don't add
Sym->$LC_1(A_1',\ldots,A_j')$

else if the node has a terminal symbol a,

then add the production Sym ->$\overline{a}$

else {it is a foot node }

if the constraint at the foot node is AA then add the productions
Sym ->$LL_2$(node symbol,$\overline{\lambda}$)/$\overline{\lambda}$

if the constraint is OA then add only the production
Sym ->$LL_2$(node symbol,$\overline{\lambda}$)

if the constraint is NA add the production
Sym ->$\lambda$

We shall now give an example of converting a TAG G to a HG. G contains a single initial tree $\alpha$, and a single auxiliary tree $\beta$ as in Fig. 5.2.

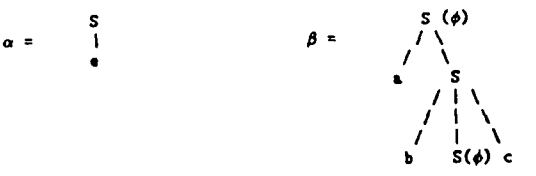



Figure 5.2

Obviously, $L(G) = \{a^n b^n c^n \mid n \geq 0\}$

Applying the procedure Convert_to_HG to this grammar we obtain the HG whose productions are given by-

S'-> $LL_2(S,A)$
A -> $\overline{\lambda}$
S -> $LC_2(B,C)$
B -> $\overline{a}$
C -> $LL_2(S,D)/D$
D -> $LC_2(E,F,G)$
E -> $\overline{b}$
F -> $\overline{\lambda}$
G -> $\overline{c}$

which can be rewritten as

S' -> $S/\overline{\lambda}$
S -> $LC_2(a,A')$
A' -> $LL_2(S,b\overset{\wedge}{\lambda}c)$ or A' -> $LL_2(S,\overset{\wedge}{bc})$

It can be verified that this grammar generates exactly L(G).

It is worth emphasising that the main point of this exercise was to show the similarities between Head Grammars and Tree Adjoining Grammars. We have shown how a HG G' (using our extended definitions) can be obtained in a systematic fashion from a TAG G. It is our belief that the extension of the definition may not necessary. Yet, this conversion process should help us understand the similarities between the two formalisms.

## 6. OTHER MATHEMATICAL PROPERTIES OF TAG's

Additional formal properties of TAG's have been discussed in [Vijay-Shankar and Joshi,1985]. Some of them are listed below

1) Pumping lemma for TAG's
2) TAL's are closed under substitution and homomorphisms
3) TAL's are not closed under the following operations

   a) intersection with TAL's
   b) intersection with CFL's
   c) complementation

Some other properties that have been considered in [Vijay-Shankar and Joshi,1985] are as follows

1) closure under the following properties
   a) inverse homomorphism
   b) gsm mappings
2) semilinearity and Parikh-boundedness.

**References**


1. Aho,A.V., and Ullman,J.D., 1973 "Theory of Parsing, Translation, and Compiling, Volume 1: Parsing, Prentice-Hall, Englewood Cliffs, N.J., 1973.

2. Joshi,A.K., 1983 "How much context-sensitivity is necessary for characterizing structural descriptions - tree adjoining grammars" in Natural Language Processing - Theoretical, Computational, and Psychological Perspectives (ed. D.Dowty, L.Karttunen, A.Zwicky), Cambridge University Press, New York, (originally presented in 1983) to appear in 1985.

3. Joshi,A.K., and Levy,L.S., 1977 "Constraints on Structural Descriptions: Local Transformations", SIAM Journal of Computing June 1977.

4. Joshi,A.K., Levy,L.S., and Takahashi, M., 1975 "Tree adjoining grammars", Journal of Computer Systems and Sciences, March 1975

5. Kroch, T., and Joshi, A.K., 1985 "Linguistic relevance of tree adjoining grammars", Technical Report, MS-CIS-85-18, Dept. of Computer and Information Science, University of Pennsylvania, April 1985

6. Pollard, C., 1984 "Generalized Phrase Structure Grammars, Head Grammars, and Natural language", Ph.D dissertation, Stanford University, August 1984

7. Roach, K., 1984 "Formal Properties of Head Grammars", unpublished manuscript, Stanford University, also presented at the Mathematics of Languages workshop at the University of Michigan, Ann Arbor, Oct. 1984.

8. Vijay-Shankar,K., Joshi,A.K., 1985 "Formal Properties of Tree Adjoining Grammars", Technical Report, Dept. of Computer and Information Science, University of Pennsylvania, July 1985.

# GUMS$_1$ : A General User Modeling System

Tim Finin
Computer and Information Science
University of Pennsylvania
Philadelphia, PA

David Drager
Arity Corporation
Concord, MA

**Abstract**

This paper describes a general architecture of a domain independent system for building and maintaining *long term models* of *individual users*. The user modeling system is intended to provide a well defined set of services for an *application system* which is interacting with various users and has a need to build and maintain models of them. As the application system interacts with a user, it can acquire knowledge of him and pass that knowledge on to the user model maintenance system for incorporation. We describe a prototype *general user modeling system* (hereafter called *GUMS*$_1$) which we have implemented in Prolog. This system satisfies some of the desirable characteristics we discuss.

## Introduction - The Need for User Modeling

Systems which attempt to interact with people in an intelligent and cooperative manner need to know many things about the individuals with whom they are interacting. Such knowledge can be of several different varieties and can be represented and used in a number of different ways. Taken collectively, the information that a system has of its users is typically refered to as its *user model*. This is so even when it is distributed through out many components of the system.

Examples that we have been involved with include systems which attempt to provide help and advice [4, 5, 15], tutorial systems [14], and natural language interfaces [16]. Each of these systems has a need to represent information about individual users. Most of the information is acquired incrementaly through direct observation and/or interaction. These systems also needed to infer additional facts about their users based on the directly acquired information. For example, the WIZARD help system [4, 15] had to represent which VMS operating system objects (e.g. commands, command qualifiers, concepts, etc) a user was familiar with and to infer which other objects he was likely to be familiar with.

We are evolving the e design of a general user model maintenance system which would support the modeling needs of the projects mentioned above. The set of services which we envision the model maintenance system performing includes:

- maintaining a data base of observed facts about the user.
- infering additional true facts about the user based on the observed facts.
- infering additional facts which are likely to be true based on default facts and default rules.
- informing the application system when certain facts can be infered to be true or assumed true.
- maintaining the consistency of the model by retracting default information when it is not consistent with the observed facts.
- providing a mechanism for building hierarchies of *stereotypes* which can form initial, partial user models.
- recognizing when a set of observed facts about a user is no longer consistent with a given stereotype and suggesting alternative stereotypes which are consistent.

This paper describes a general architecture for a domain independent system for building and maintaining *long term models* of *individual users*. The user modeling system is intended to provide a well defined set of services for an *application system* which is interacting with various users and has a need to build and maintain models of them. As the application system interacts with a user, it can acquire knowledge of him and pass that knowledge on to the user model maintenance system for incorporation. We describe a prototype *general user modeling system* (hereafter called *GUMS*$_1$) which we have implemented in Prolog. This system satisfies some of the desirable characteristics we discuss.

## What is a User Model?

The concept of encorporating user models into interactive systems has become common, but what has been meant by a user model has varied and is not always clear. In trying to specify what is being refered to as a user model, one has to answer a number of questions: who is being modeled; what aspects of the user are being modeled; how is the model to be initially acquired; how will it be maintained; and how will it be used. In this section we will attempt to characterize our own approach by answering these questions.

### Who is being modeled?

The primary distinctions here are whether one is modeling individual users or a class of users and whether one is attempting to construct a short or long term model. We are interested in the aquisition and use of long term models of individual users. We want to represent the knowledge and beliefs of individuals and to do so in a way that results in a persistent record which can grow and change as neccessary.

It will be neccessary, of course,to represent generic facts which are true of large classes (even all) of users. In particular, such facts may include inference rules which relate a person's belief, knowledge or understanding of one thing to his belief, knowledge and understanding of others. For example in the context of a timeshared computer system we may want to include a rule like:

> *If a user U believes that machine M is running, then U will believe that it is possible for him to log onto M.*

It is just this sort of rule which is required in order to support the kinds of cooperative interactions studied in [6] and [7], such as the following:

```
User:   Is UPENN-LINC up?

System: Yes, but you can't log on now.
        Preventative maintenance is being
        done until 11:00am.
```

### What is to be modeled?

Our current work is focused on building a general purpose, domain *independent model maintenance system*. Exactly what *information* is to be modeled is up to the application. For example, a natural language system may need to know what language terms a user is *likely to be familiar with* [16], *a CAI system for second language* learning may need to model a user's knowledge of grammatical rules [14], an intelligent database query system may want to model which *fields of a data base relation a user is interested in* [10], and an expert system may need to model a user's domain goals [11].

### How is the model to be aquired and maintained?

We are exploring a system in which an initial model of the user will be selected from a set of stereotypical user models [13]. Selecting the most appropriate stereotype from the set can be accomplished by a number of techniques, from letting the user select one to surveying the user and having an expert system select one. Once an initial model has been selected, it will be updated and maintained as direct knowledge about the user is aquired from the interaction. Since the use of stereotypical user models is a kind of *default reasoning* [12], we will use *truth maintenance* techniques [9] for maintaining a consistent model.

In particular, if we learn something which contradicts a fact in the our current model of the user than we need to update the model. Updating the model may lead to an inconsistency which must be squared away. If the model can be made consistent by changing any of the *default* facts in the model, then this should be done. If there is *a choice of which defaults to alter*, then a mechanism must be provided to do this (e.g. through further dialogue with the user). If there are no defaults which can be altered to make the model consistent then the stereotype must be abandoned and a new one sought.

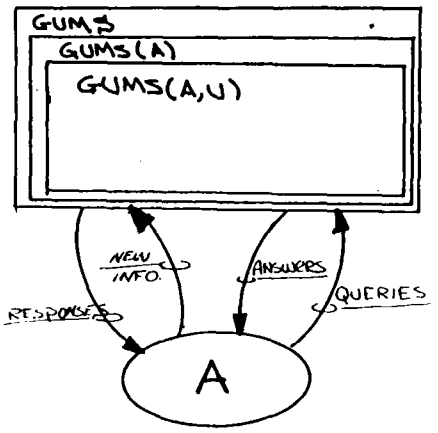
### How is the model to be used?

The model can be accessed in two primary ways: facts can be added, deleted or updated from the model and facts can be looked up or infered. A forward chaining component together with a truth maintenance system can be used to update the default assumptions and keep the model consistent.

## Architectures for User Modeling Systems

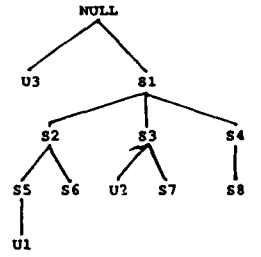
Our goal is to provide a general user modeling utility organized along the lines shown in figures 1 and 2. The user modeling system provides a service to an application program which interacts directly with a user. This application program gathers information about the user through this interaction and choses to store some of this information in the user model. Thus, one service the user model provides is accepting (and storing!) new information about the user. This information may trigger an inferential process which could have a number of outcomes:

- The user modeling system may detect an inconsistency and so inform the application.
- The user model may infer a new fact about the user which triggers a demon causing some action (e.g. informing the application).




A: an Application
GUMS: General User Modeling System
GUMS(A): Modeling System for Application A
GUMS(A,U): Model for User U in Application A

Figure 1: A General Architecture for a User Modeling Utility




NULL: the Empty Stereotype
Si: Stereotype i
Ui: User i

Figure 2: A User Modeling System for an Application

- The user model may need to update some previously inferred default information about the user

Another kind of service the user model must provide is answering queries posed by the application. The application may need to look up or deduce certain information about its current user.

We are currently experimenting with some of these ideas in a system called $GUMS_1$. This system is implemented in prolog and used a simple default logic together with a backward chaining interpreter rather than a truth maintenance system and a forward chaining engine. The next section describes $GUMS_1$ and its use of default logic.

## Default Logic and User Modeling

A user model is most useful in a situation where the application does not have complete information about the knowledge and beliefs of its users. This leaves us with the problem of how to model a user given we have only a limited amount of knowledge about him. Our approach involves using several forms of default reasoning techniques: stereotypes, explicit default rules, and failure as negation.

We assume that the $GUMS_1$ system will be used in an application which incrementaly gains new knowledge about its users throughout the interaction. But the mere ability to gain new knowledge about the user is not enough. We can not wait until we have full knowledge about a user to reason about him. Fortunately we can very often make generalization about users or classes of users. We call a such a generalization a stereotype. A stereotype consists of a set of facts and rules that are believed to applied to a class of users. Thus a stereotype gives us a form of default reasoning.

Stereotypes can be organized in hierarchies in which one stereotype subsumes another if it can be thought to be more general. A stereotype $S_1$ is said to be more general than a stereotype $S_2$ if everything which is true about $S_1$ is neccessarily true about $S_2$. Looking at this from another vantage point, a stereotype inherits all the facts and rules from every stereotype that it is subsumed by. For example, in the context of a *programmer's apprentice* application, we might have stereotypes corresponding to different classes of programmer, as is suggested by the the hierarchy in figure 2.

In general, we will want a stereotype to have any number of immediate ancestors, allowing us to compose a new stereotype out of several existing ones. In the context of a *programmers apprentice*, gor example, we may wish to describe a particular user as a *SymbolicsWizard* and a *UnixNovice* and a *ScribeUser*. Thus, the stereotype system should form a general lattice. Our current system constrains the system to a tree.

Within a stereotype we can have default information as well. For instance, we can be sure that a programmer will know what a *file* is, but we can only guess that a programmer will know what a *file directory* is. If we have categorized a given user under the *programmer stereotype* and discover[1] that he is not familiar with the concept of a *file* then we can concludethat we had improperly chosen a stereotype and must choose a new one. But if we got the information that he did not know what a *file directory* was, this would not rule out the possibility of him being a programmer. Thus $GUMS_1$

[1] perhaps through direct interaction with her

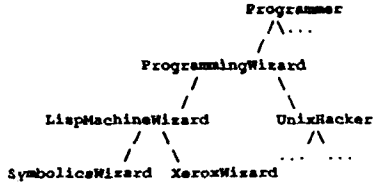



Figure 3: A Hierachy of Stereotypes

allows rules and facts within a stereotype to be either definitely true or true by default (i.e. in the absence of information to the contrary.)

In $GUMS_1$ we use the certain/1 predicate to introduce a definite fact or rule and the default/1 predicate to indicate a default fact or rule, as in:

| | |
|---|---|
| certain(P). | a definite fact: P is true. |
| certain(P if Q). | a definite rule: P is true if Q is definitely true and P is assumed to be true if Q is only assumed to be true. |
| default(P). | a default fact: P is assumed to be true unless it is known to be false. |
| default(P if Q). | a default rule: P is assumed to be true if Q is true or assumed to be true and there is no definite evidence to the contrary. |

As an example, consider a situation in which we need to model a persons familiarity with certain terms. This is a common situation in systems which need to produce text as explanations or in response to queries and in which there is a wide variation in the users' familiarity with the domain. We might use the following rules

(a) default(understandsTerm(ram)).

(b) default(understandsTerm(rom)
if understandsTerm(ram)).

(c) certain(understandsTerm(pc)
if understandsTerm(ibmpc)).

(d) certain(~understandsTerm(cpu)).

to represent these assertions, all of which are considered as pertaining to a particular user with respect to the stereotype containing the rules:

(a) Assume the user understands the term *ram* unless we know otherwise.

(b) Assume the user understands the term *rom* if we know or believe he understands the term *ram* unless we know otherwise.

(c) This user understands the term *pc* if he understands the term *ibmpc*.

(d) This user does understand the term *cpu*.

$GUMS_1$ also treats negation as failure in some cases as a default rule. In general, logic is interpreted using an open world assumption That is, the failure to be able to prove a proposition is not taken as evidence that it is not true. Many logic programming languages, such a prolog, encourage the interpretation of unprovability as logical negation. Two approaches have been forwarded to justify the

negation as failure rule. One approach is the closed world assumption [2]. In this case we assume that anything not interable from the database is by necessity false. One problem with this assumption is that this is a metalevel assumption and we do not know what the equivalent object level assumptions are. A second approach originated by Clark is based upon the concept of a completed database [1]. A completed database is the database constructed by rewriting the set of clauses defining each predicate to an if and only if definition that is called the completion of the predicate. The purpose of the completed definition is to indicate that the clauses that define a predicate define every possible instance of that predicate.

Any approach to negation as failure requires that a negated goal be ground before execution, (actually a slightly less restrictive rule could allow a partially instantiated negated goal to run but would produce the wrong answer if any variable was bound.) Thus we must have some way of insuring that every negated literal will be bound. In *GUMS*$_1$ we have used a simple variable typing scheme to achieve this, as will be discussed later.

We have used a variant of the completed database approach to show that a predicate within the scope of a negation is closed. A predicate is closed if and only if it is defined by an iff statement and every other predicate in the definition of this predicate is closed. We allow a metalevel statement *completed(P)* that is used to signify that by predicate P we really intend the iff definition associated with P. This same technique was used by Kowalski [8] to indicate completion. *By default we believe competed(P) where not indicated.* So if *P* is not explicitly closed *not P* is decided by default.

Thus in *GUMS*$_1$ we have the ability to express that a default should be taken from the lack of certain information (i.e. negation as failure) as well as from the presence of certain information (i.e. default rules). For example, we can have a default rule for the programmer stereotype that can conclude knowledge about linkers from knowledge about compilers, as in:

```
default(knows(linkers) if knows(compilers))
```

We can also have a rule that will take the lack of knowledge about compilers as an indication that the user probably knows about interpreters, as in:

```
certain(knows(interpreters)
        if ~ knows(compilers))
```

This system also allows explicit negative facts and default facts. When negation is proved in reference to a negative fact then negation is not considered a default case. Similarly negation as failure is not considered a default when the predicate being negated is closed. Such distinctions are possible because the *GUMS*$_1$ interpreter is based on a four value logic.

The distinction between truth or falsity by default (i.e. assumption) and truth or falsity by logical implication is an important one to this system. The central predicate of the system is the two argument predicate *show* which relates a goal *G* expressed as a literal to a truth value. Thus **show(Goal,Val)** returns in the variable *Val* the current belief in the literal *Goal*. The variable *Val* can be instantiated to **true**, **false**, **assume(true)**, or **assume(false)**. The meanings of these values are as follows:

| | |
|---|---|
| **true** | definitely true according to the current database. |
| **assume(true)** | true by assumption (i.e. true by default) |
| **assume(false)** | false by assumption |
| ***false*** | ***definitely not true.*** |

These values represent truth values for a given user with respect to a given stereotype. If the stereotype is not appropriate, then even definite values may have to change.

Having a four value logic allows us to distinguish conclusions made from purely logical information from those dependent on default information. Four value logic also allows a simple type of *introspective reasoning* that may be useful for modeling the beliefs of the user. We currently use a default rule to represent an uncertain belief about what the user knows or believes, but we could imagine a situation where we would like to model uncertainties that the user has in his beliefs or knowledge. One such predicate is an embeded *show* predicate. For example we might have a rule that a user will use a operating system command that he believe might erase a file only if he is certain that he knows how to use that command. This might encode as:

```
certain(okay_to_use(Command) if
          can_erase_files(Command),
           show(know(Command),true)).
```

Another predicate *assumed(Pred)* will evaluate the truth of *Pred* and "strengthen" the result. That is

```
demo(assumed(P),V) :-
  demo(P,V2),
  strengthen(V2,V).
```

where the strengthen relation maps assumed values into definite values (e.g. assume(true) becomes true, assume(false) becomes false and true and false remain unchanged). The *assumed* predicate is used to express a certain belief from an uncertain knowledge or belief. For example we might want to express a rule that a user will always want to use a screen editor if he believes one may be available.

```
certain(willUse(screenEditor) if
     assumed(available(screenEditor))).
```

The interpreter that *GUMS*$_1$ is base on is a metalevel interpreter written in Prolog. The interpreter must generate and compare many possible answers to each subquery, because of the multiple value logic and the presence of explicit negative information. Strong answers to a query (i.e. *true* and *false*) are sought first, followed by weak answers (i.e. *assume(true)* and *assume(false)*). Because strong answers have precedence over weak ones, it is not necessary to remove weak information that contradicts strong information.

Another feature of this system is that we can specify the types of arguments to predicates. This type information can be used to allow the system to handle non-ground goals. In our system, a type provides a way to enumerate a complete set of possible values subsumed by that type. When the top-level *show* predicate is given a partially instantiated goal to solve, it uses the type information to generate a stream of consistent fully instantiated goals. These ground goals are tried sequentially.

That goals must be fully intantiated follows from the fact that negation as failure is built into the evaluation algorithm. Complex terms will be instantiated to every pattern allowed by the datatype given the full power of unification. To specify the type information, one should specify argument types for a predicate, subtype information and type instance information. For example, the following says that the **canProgram** predicate ranges over instances of the ***type person and programmingLanguage***, that the type **functionalLanguage** is a sub-type of **programmingLanguage** and

that the value **scheme** is an instance of the type **functionalLanguage**:

```
declare(canProgram(person,
                   programmingLanguage)).

subtype(programmingLanguage,
        functionalLanguage).

inst(functionalLanguage,scheme).
```

## Limitations of the Present System

Our current system has several limitations. One problem is that it does not extract all of the available information from a new fact learned of the user. If we assert that a predicate is *closed*, we are saying that the set of (certain) rules for the predicate form a definition, i.e. a neccessary and sufficient description. In our current system, however, the information still only flows direction! For example, suppose that we would like to encode the rule that a user knows about *I/O redirection* if and only of they know about *files* and about *pipes*. Further, let's suppose that the default is that a person in this stereotype does not know about *files* or *pipes*. This can be expresses as:

```
certain(knows(io_redirection) if
          knows(pipes),
          knows(files)).

default(~knows(pipes)).

default(~knows(files))

closed(knows(io_redirection)).
```

If we learn that a particular user does know about *I/O redirection* then it should follow that she neccessarily knows about both *files* and *pipes*. Adding the assertion

```
certain(knows(io_redirection))
```

however, will make no additional changes in the data base. The values of **knows(pipes)** and **knows(files)** will not change! A sample run after this change might be :

```
?- show(knows(io_redirection),Val).
        Val = true

?- show(knows(pipes),Val).
        Val = assume(false)

?- show(knows(files),Val).
        Val = assume(false).
```

The reason for this problem is that the current interpreter was designed to be able to incorporate new information without actually using a full truth maintenance system. Before a fact *F* with truth value *V* is to be added to the data base, *GUMS*$_1$ checks to see if an inconsistent truth value *V'* can be derived for *F*. If one can be, then a new stereotype is sought in which the contradiction goes away. New knowledge that does not force an obvious inconsistency within the database is added as is. Neither redundant information or existing default information effect the correctness of the interpreter. Subtler inconsistencies are possible, of course.

Another limitation of the current system its inefficiency. The use of default rules requires us to continue to search for solutions for a goal until a strong one is found or all solutions have been checked. These two limitations may be addressable by redesigning the system to be based on a forward chaining truth maintenance system. The question is whether the relative efficiency of forward chaining will offset the relative inefficiency of truth maintenance. The use of an assumption based truth maintenance system [3] is another alternative that we will investigate.

## The *GUMS*$_1$ Command Language

Our current experimental implementation provides the following commands to the application.

**show(Query,Val)** succeeds with *Val* as the strongest truth value for the gaol *Query*. A Query is a partially or fully instantiated positive or negative literal. Val is return and is the value the current belief state If *Query* is partially instantiated then it will return more answers upon backtracking if possible. In general one answer will be provided for every legal ground substitution that agrees with current type declarations.

**add(Fact,Status)** sets belief in *Fact* to true. If *Fact* or any legal instance of it contradicts the current belief state then the user model adopts successively higher stereotypes in the hierarchy until one is found in which all of the added facts are consistent. If no stereotype is successful then no stereotype is used, all answers will be based entirely on added facts. *Fact* must be partially or fully instantiated and can be either a positive or negative literal. *Status* must be uninstantiated and will be bound to a message describing the result of the addition (e.g. one of several error messages, *ok*, the name of a new stereotype, etc.).

**create_user(UserName,Stereotype,File,Status)** stores the current user if necessary and creates a new user who then is the current user. *UserName* is instantiated to the desired name. *Stereotype* is the logical name of the stereotype that the system should assume to hold. *File* is the name of the file that information pertaining to the user will be stored. *Status* is instantiated by the system and returns error messages. A user must be created in order for the system to be able to answer queries.

**store_current(Status)** stores the current users information and clears the workspace for a new user. *Status* is instantiated by the system on an error.

**restore_user(User,Status)** restores a previous user after saving the current user if necessary. *User* is the name of the user. *Status* is instantiated by the system to pass error messages.

**done** stores the system state of the user modeling system, saving the current user if necessary. This command should be the last command issued and needs to be issued at the end of every session.

## Conclusions

Many interactive systems have a strong need to maintain models of individual users. We have presented a simple architecture for a general user modeling utility which is based on the ideas of a default logic. This approach provides a simple system which can maintain a database of known information about users as well as use rules and facts which are associated with a stereotype which is believed to be appropriate for this user. The stereotype can contain definite facts and define rules of inference as well as default information and rules. The rules can be used to derive new information, both definite and assumed, from the currently believed information about the user.

We believe that this kind of system will prove useful to a wide range of applications. We have implemented an initial version in Prolog and are planning to use it to support the modeling needs of several projects. We are also exploring a more powerful approach to user modeling based on the notion of a *truth maintenance system*.

## Bibliography


1. Clark, Keith L. Negation as Failure. In *Logic and Databases*, J. Minker and H. Gallaire, Ed., Plenum Press, New York, 1978.

2. Reiter, R. Closed World Databases. In *Logic and Databases*, H. Gallaire & J. Minker, Ed., Plenum Press, 1978, pp. 149-177.

3. DeKleer, J. An Assumption Based Truth Maintenance System. Proceedings of IJCAI-85, IJCAI, August, 1985.

4. Finin, T.W. Help and Advice in Task Oriented Systems. Proc. 7th Int'l. Joint Conf. on Art. Intelligence, IJCAI, August, 1982.

5. Howe, A. and T. Finin. Using Spreading Activation to Identify Relevant Help. Proceeding of the 1984 Canadian Society for Computational Studies of Intelligence, CSCSI, 1984. also available as Technical Report MS-CIS-84-01, Computer and Information Science, U. of Pennsylvania.

6. Joshi, A., Webber, B. & Weischedel, R. Preventing False Inferences. Proceedings of COLING-84, Stanford CA, July, 1984.

7. Joshi, A., Webber, B. & Weischedel, R. Living Up to Expectations: Computing Expert Responses. Proceedings of AAAI-84, Austin TX, August, 1984.

8. Kowalski, Robert. *Logic for Problem Solving*. North-Holland, New York, 1979.

9. McDermott, D and J. Doyle. "Non-Monotonic Logic I". *Artificial Intelligence 13*, 1-2 (1980), 41 - 72.

10. Motro, A. Query Generalization: A Method for Interpreting Null Answers. In Larry Kerschberg, Ed., *Expert Database Systems*, Benjamin/Cummings, Menlo Park CA, 1985.

11. Pollack, M. Information Sought and Information Provided. Proceedings of CHI'85, Assoc. for Computing Machinery (ACM), San Francisco CA, April, 1985, pp. 155-160.

12. Reiter, Ray. "A Logic for Default Reasoning". *Artificial Intelligence 13*, 1 (1980), 81-132.

13. Rich, Elaine. "User Modeling via Stereotypes". *Cognitive Science 3* (1979), 329-354.

14. Schuster, E. and T. Finin. VP2: The Role of User Modelling in Correcting Errors in Second Language Learning. Proc. Conference on Artificial Intelligence and the Simulation of Behavior, AISB, 1985.

15. Shrager, J. and T. Finin. An Expert System that Volunteers Advice. Proc. Second Annual National Conference in Artificial Intelligence, AAAI, August, 1982.

16. Webber, B. and T. Finin. In Response: Next Steps in Natural Language Interaction. In *Artificial Intelligence Applications for Business*, W. Reitman, Ed., Ablex Publ. Co., Norwood NJ, 1984.


## Appendix - The Demo Predicate

This appendix defines the *demo* predicate which implements the heart of the *GUMS*$_1$ interpreter. The relation

`show(Goal,Value)`

holds if the truth value of proposition *Goal* can be shown to be *Value* for a particular ground instance of *Goal*. The *show* predicate first makes sure that *Goal* is a ground instance via a call to the *bindVars* predicate and then invokes the meta-evaluator *demo*. The relation

`demo(Goal,Value,Level)`

requires that *Goal* be a fully instantiated term and *Level* be an integer that represents the level of recursion within the demo predicate. The relation holds if the "strongest" truth value for *Goal* is *Value*.

---

```
:- op(950,fy,'~').
:- op(1150,xfy,'if').

show(P,V) :- bindVars(P), demo(P,V,0).

% truth values
demo(P,P,_) :- truthValue(P),!.

% reflection...
demo(demo(P,V1),V,D) :-
  !,
  nonvar(V1),
  demo(P,V1,D) -> V=true;V=false.

% disjunction ...
demo((P;Q),V,D) :- !,
  demo(P,V1,D),
  demo(Q,V2,D),
  upperbound(V1,V2,V).

% conjunction ...
demo((P,Q),V,D) :- !,
  demo(P,V1,D),
  demo(Q,V2,D),
  lowerbound(V1,V2,V).

% negation ...
demo(~P,V,D) :- !,
  demo(P,V1,D),
  negate(V1,V,P).

% assumption ...
demo(assumed(P),V,D) :- !,
  demo(P,V1,D),
  strengthen(V1,V).

% call demo1 with deeper depth and then cut.
demo(P,V,Depth) :-
  Deeper is Depth+1,
  demo1(P,V,Deeper),
  retractall(temp(_,Deeper)),
  !.

% definite facts...
demo1(P,true,_) :- certain(P).
demo1(P,false,_) :- certain(~P).

% find a definite rule that yields TRUE or FALSE.
demo1(P,V,D) :-
  forsome(certain(P if Q),(demo(Q,V,D),demoNote(V,D))).

demo1(P,V,D) :-
  forsome(certain(~P if Q),
          (demo(Q,V1,D),
           negate(V1,V,P),
           demoNote(V,D))).

% stop if the best so far was ASSUME(TRUE).
demo1(P,assume(true),D) :-
  retract(temp(assume(true),D)).

% default positive facts.
demo(P,assume(true),_) :- default(P).

% try default rules 'til one gives a positive value.
demo1(P,assume(true),D) :-
  forsome(default(P if Q),(demo(Q,V,D),positive(V))).

% default negative facts.
demo(P,assume(false),_) :- default(~P).

% default negative rules.
```

```
demo1(P,assume(false),D) :-
  forsome(default(~P if Q),(demo(Q,V,D),positive(V))).

% if P is closed, then its false.
demo1(P,false,_) :- closed(P),!.

% the default answer.
demo1(P,assume(false),_).

% demoNote(X,D) succeeds if X is TRUE or FALSE,
% otherwise it fails after updating temp(A,_)
% to be the strongest value known so far.

demoNote(V,_) :- known(V).
demoNote(V,D) :-
  not(temp(_,D)),
  !,
  assert(temp(V,D)),
  fail.
demoNote(assume(true),D) :-
  retract(temp(_,D)),
  !,
  assert(temp(assume(true),D)),
  fail.
```

## % Relations on Truth Values

```
positive(X) :- X == true ; X == assume(true).

known(X) :- X == true ; X == false.

higher(true,_).
higher(assume(true),assume(false)).
higher(_,false).

upperbound(X,Y,Z) :- higher(X,Y) -> Z=X ; Z=Y.

lowerbound(X,Y,Z) :- higher(X,Y) -> Z=Y ; Z=X.

strengthen(assume(X),X).
strengthen(true,true).
strengthen(false,false).

% negation is relative to a predicate.
negate(true,false,_).
negate(assume(true),assume(false),_).
negate(assume(false),assume(true),_).
negate(false,true,P) :- closed(P).
negate(false,assume(true),P) :- not(closed(P)).

truthValue(true).
truthValue(false).
truthValue(assume(X)) :- truthValue(X).
```

## % The Type System

```
% isSubtype(T1,T2) iff type T1 has an
% ancestor type T2.
isSubtype(T1,T2) :- subtype(T1,T2).
isSubtype(T1,T2) :-
  subtype(T1,T),
  isSubtype(T,T2).

% true if instance I is descendant from type T.
isInstance(I,T) :- inst(I,T).
isInstance(I,T) :-
  isSubtype(T1,T),
  isInstance(I,T1).

% true if T is a type.
isType(T) :- inst(_,T).
isType(T) :- subtype(T,_).
isType(T) :- subtype(_,T).
```

## % Grounding Terms

```
% bindVars(P) ensures that all variables
%  in P are bound or it fails.
bindVars(P) :- var(P),!,fail.
bindVars(P) :- atomic(P),!.
bindVars(P) :-
  scheme(P,PS),
  P =.. [_|Args],
  PS =.. [_|Types],
  bindArgs(Args,Types).

bindArgs([],[]).
bindArgs([Arg|Args],[Type|Types]) :-
  bindArg(Arg,Type),
  bindArgs(Args,Types).

bindArg(Arg,Type) :-
  var(Arg),
  isInstance(Arg,Type).
bindArg(Arg,_) :- bindVars(Arg).

% scheme(P,S) is true if S is the schema for P, eg
% schema(give(john,X,Y),give(person,person,thing)).

% find a declared schema.
schema(P,S) :-
  functor(P,F,N),
  functor(S,F,N),
  declare(S),
  !.

% use the default schema F(thing,thing,...).
schema(P,S) :-
  functor(P,F,N),
  functor(S,F,N),
  for(I,1,N,arg(I,S,thing)),
  !.
```

## SECTION 7: RESEARCH CONTRIBUTIONS

*University of Southern California*

# A LOGICAL-FORM AND KNOWLEDGE-BASE DESIGN FOR NATURAL LANGUAGE GENERATION[1,2]

Norman K. Sondheimer
USC/Information Sciences Institute
Marina del Rey, CA

Bernhard Nebel
Technische Iniversitaet Berlin
Berlin, West Germany

**ABSTRACT**

This paper presents a technique for interpreting output demands by a natural language sentence generator in a formally transparent and efficient way. These demands are stated in a logical language. A network knowledge base organizes the concepts of the application domain into categories known to the generator. The logical expressions are interpreted by the generator using the knowledge base and a restricted, but efficient, hybrid knowledge representation system. This design has been used to allow the NIGEL generator to interpret statements in a first-order predicate calculus using the NIKL and KL-TWO knowledge representation systems. The success of this experiment has led to plans for the inclusion of this design in both the evolving Penman natural language generator and the Janus natural language interface.

## 1. INTRODUCTION

We have as a general goal the development of natural language generation capabilities. Independent software systems will state demands to the generation facility in a mutually convenient form. The generator will use those demands to create natural language sentences. Instead of merging generation with other functions of the overall computer system, this design allows for reuse of the generator in other systems, specialized processing of linguistic information, and modular development.

Our design requires a notation to represent expressive demands. The notation should be of general applicability. For example, a good notation ought to be acceptable as the output of a natural language parser. The notation should have a well-defined semantics. In addition, the generator has to have some way of interpreting the demands. This interpretation has to be efficient.

In our research, we have used formal logic as a demand language. Network knowledge-bases are used to define the domain of discourse in order to help the generator interpret the logical forms. And a restricted, hybrid knowledge representation is utilized to analyze demands for expression using the knowledge base.

Arguments for these decisions include the following: Formal logic is a well established means of expressing information with a well-defined semantics. Furthermore, it is commonly used in natural language analyzers and discourse processors, as well as other AI systems. Network knowledge-base notations have been shown to be effective and efficient in language processing. Work on network representations has shown that they too can be given formal semantics [Schmolze and Lipkis 83]. Finally, recent work on hybrid knowledge representation systems has shown how to combine the reasoning of logic and network systems [Brachman 85]. Restricted-reasoning hybrid systems have shown this reasoning can be done efficiently.

On our project, we have:

1. Developed a demand language based on first order logic,
2. Structured a NIKL (New Implementation of KL-ONE) network [Kaczmarek 86] to reflect conceptual distinctions observed by functional systemic linguists.
3. Developed a method for translation of expression demands into a propositional logic database,
4. Employed KL-TWO [Vilain 85] to analyze the translated demands, and
5. Used the results of the analyses to provide directions to the Nigel English sentence generation system [Mann & Matthiessen 83].

This paper presents our design and some of our experiences with it.

---

[1] This research is supported by the Defense Advanced Research Projects Agency under Contract No MDA903 81 C 0335 and by the Air Office of Scientific Research under FQ8671-84-01007. Views and conclusions contained in this report are the author's and should not be interpreted as representing the official opinion or policy of DARPA, AFOSR, the U.S. Government, or any person or agency connected with them.

[2] A revised version of this paper will appear in the Proceedings of the National Conference on Artificial Intelligence, August 11 - 15, 1986, Philadelphia, PA.

Others have attempted to design an interface between a linguistic generation engine and an associated software system using an appropriate information representation [Goldman 75, Appelt 83, Hovy 85, Kukich 85, Jacobs 85, McKeown 85]. Still others have depended on information demand representations with similar well-defined semantics and expressive power,e.g., [Shapiro 79]. However, he produces a logician's reading of expressions rather than colloquial English. For example, the popular song "Every man loves a woman.", might be rendered "For all men there exist a woman that they love.".

The generation component of HAM-ANS [Hoeppner et al. 83] and one effort of McDonald's [McDonald 83] are probably closest to our design. HAM-ANS also uses a logical language (the same one used for representing the analyzed input), has an extensive network domain model, and has a separable linguistic engine (although not as broad in coverage as Nigel). However, the interface language is close to surface linguistic representation, e.g., there are particular expressions for tense and voice. So while it is easier to generate sentences from such structures, it is correspondingly **harder for software systems to produce demands for expressions** without having access to significant amounts of linguistic knowledge.

McDonald accepts statements in the first order predicate calculus, processes them with a grammar, and outputs excellent English forms. It is hard to evaluate the coverage of McDonald's grammar, however, the program does depend on extensive procedurally-encoded domain-dependent lexical entries. Our domain dependancies are limited to the correct placement of concepts in the NIKL hierarchy and the association of lexical entries with the concepts. These lexical entries are only characterized by syntactic features.

In Section 2, we present the component technologies we have applied. Section 3 presents the method by which they are combined. Section 4 presents several examples of their use. We conclude with a section describing the open problems identified by our experiences and our plans for future work.

## 2. BASIC COMPONENTS

The processes and representations we have employed include a unique linguistic component (Nigel), a frame-based network knowledge representation (NIKL), a propositional reasoner that can take advantage of the network knowledge representation (KL-TWO), and our own first order logic meaning representation.

### 2.1. Nigel

The Nigel grammar and generator realizes the functional systemic framework [Halliday 76] at the level of sentence generation. Within this framework, language is viewed as offering a set of grammatical choices to its speakers. Speakers make their choices based on the information they wish to convey and the discourse context they find themselves in. Nigel captures the first of these notions by organizing minimal sets of choices into **systems**. The grammar is actually just a collection of these systems. The factors the speaker considers in evaluating his communicative goal are shown by questions called **inquiries** [Mann 83a]. A choice alternative in a system is chosen according to the responses to one or more of these inquiries.

For example, because processes with addressees are grammatically different from other processes, the grammar has an inquiry, **VerbalProcessQ**, to test whether the process is one of communication. Elsewhere, as part of deciding on number, Nigel has an inquiry **MultiplicityQ** that determines whether an object being described is unary or multiple. These are examples of **information characterization** inquiries.

Another type of inquiry, called **information decomposition**, picks out of the environment the conceptual entities to be described. For example, at appropriate times, Nigel asks for descriptions of the causers of events, **CauserID**, or the objects affected in them, **AffectedID**.

One very special inquiry, **TermSpecificationID**, establishes the words that will be used. Nigel asks the environment for a set of lexical entries that can be used to describe an entity. So Nigel might find itself being told to describe some event as a "Send" or some object as a "Message".

Nigel currently has over 230 systems and 420 inquiries and covers a large subset of English.

Up until the effort described here, the developers of Nigel had only identified the inquiries of the grammar, but not implemented them.

## 2.2. NIKL

NIKL is a network knowledge-base system descended from KL-ONE [Brachman and Schmolze 85]. This type of reasoner supports description of the categories of objects, actions, and states of affairs that make up a domain. The central components of the notation are sets of concepts and roles, organized in IS-A hierarchies. The concepts are used to identify the categories of entities. The roles are associated with concepts (as "role restrictions"), and identify the relationships that can hold between actual individuals that belong to the categories. The IS-A hierarchies identify when membership in one category (or the holding of one relationship) entails membership in (or the holding of) another.

We have experimented with a mail and calendar NIKL domain model developed for the Consul project [Kaczmarek, Mark, and Sondheimer 83]. It has a concept **Send** that is meant to identify the activity of sending messages. **Send** IS-A type of **Transmit** (intended to identify the general activity of transmission of information). **Send** is distinguished from **Transmit** by having a role restriction **actee** that relates **Sends** to **Messages**. The concept of **Message** is defined as being a kind of a communication object, through the IS-A relation to a concept **Communication**. In addition, role restrictions connect **Message** to properties of messages which serve to distinguish it from other communication objects. The overall model has over 850 concepts with over 150 roles.

In flavor, NIKL is a frame system, with the concepts equivalent to frames and the role restrictions to slots. However, the NIKL representation can be given a formal semantics.

## 2.3. KL-TWO

KL-TWO is a hybrid knowledge representation system that uses NIKL's formal semantics. KL-TWO links another reasoner, PENNI, to NIKL. For our purposes, PENNI can be viewed as restricted to reasoning using propositional logic[3]. As such, PENNI is more restricted than those systems that use first order logic and a general purpose theorem prover. But it is also more efficient.

PENNI can be viewed as managing a data base of propositions of the form (P a) and (Q a b) where the forms are variable free[4]. The first item in each ordered pair is the name of a concept in an associated NIKL network and the first item in each ordered triple is the name of a role in that network. So the assertion of any form (P a) is a statement that the individual a is a kind of thing described by the concept P. Furthermore, the assertion (Q a b) states that the individuals a and b are related by the abstract relation described by Q.

NIKL adds to PENNI the ability to do taxonomic reasoning. Assume the NIKL database contained the concepts just described in discussing NIKL. Assume that we assert just the following three facts: (**Transmit** x), (**actee** x y) and (**Message** y). Using the knowledge base, PENNI is able to recognize that any **Transmit**, all of whose actees are **Messages**, is a **Send**. So if we ask if (**Send** x) is true, KL-TWO will reply positively.

KL-TWO can also retrieve information from its database. For example, if asked what individuals were the **actees** of **x**, it could respond with **y**.

## 2.4. THE LOGICAL LANGUAGE

Our logical language is based on first order logic. To it, we have added restricted quantification, i.e., the ability to restrict the set quantified over. In addition, we allow for equality and some related quantifiers and operators, such as the quantifier for "there exists exactly one ..." (∃!) and the operator for "the one thing that ..." (ι). We permit the formation and manipulation of sets, including a predicate for set membership (**ELEMENT-OF**). And we have some quantifiers and operators based on Habel's η operator [Habel 82].

Figure 2-1 gives three examples of the forms accepted. Included are a few individuals: two people (RICHARD and SMITH), the computer (COMPUTER), a set of messages (MM33), and the current time (NOW). Later on, we will show how these are turned into English by our system.

We include in our language a theory of the categories of conceptual entities and their relationships. We have taken what is often referred to as a Davidson approach [Davidson 67]. This is marked by quantification over events and state of affairs. We refer to these as **ActionOccurrences** and **RelationOccurrences**, respectively. We associate time and place with these

[3] PENNI is an enhanced version of RUP [McAllester 82]

[4] PENNI actually works with the quantifier-free predicate calculus with equality. It has a demon-like facility capable of some quantificational reasoning as well.

A. (∃x ∈ ActionOccurrence)((∃p ∈ Past)(timeofoccurrence(x,p))∧
(∃y ∈ Transmit)(records(x,y)∧actor(y,SMITH)∧(∃z ∈ Message)actee(y,z)))

B. (∃x ∈ ActionOccurrence)((∃t ∈ Future)(timeofoccurrence(x,t))∧
(∃y ∈ Display)(records(x,y)∧actor(y,RICHARD)∧
requestedobject(y,ιq((∃z ∈ ActionOccurrence)((∃p ∈ Past)(timeofoccurrence(z,p))∧
(∃w ∈ Send)(records(z,w)∧actor(w,SMITH)∧actee(w,q)))))∧
beneficiary(y,COMPUTER)))

C. (∃z ∈ ActionOccurrence)
(∃d ∈ Display)(records(z,d)∧actor(d,RICHARD)∧beneficiary(d,COMPUTER)∧
(∀m ∈ MM33)(∃! r ∈ {r|(∃s ∈ RelationOccurrence)
(timeofoccurrence(s,NOW)∧
(∃t ∈ InspectionStatus)(records(s,t)∧range(t,r)∧domain(t,m)))})
(requestedobject(d,r))))

Figure 2-1: Example Logical Expressions

entities. We differ from Davidson by identifying a class of abstract **Actions** and **Relations** that are recorded by **ActionOccurrences** and **RelationOccurrences**[5]. With **Actions** and **Relations**, we associate the participants and circumstances of the actions and states-of-affairs, e.g., the actor and actee.

In addition to using the logical language for the demands for expression, we use it to maintain a database of factual information. Besides the "facts of the world", we assume the availability of such knowledge as:

- Hearer, speaker, time and place.
- The theme of the ongoing discussion.
- Objects assumed to be identifiable to the hearer.

Work on maintaining this database is proceeding in parallel with the work reported here.

Finally, we have allowed for a speech act operator to be supplied to the generation system along with the logical form. This can be **ASSERT, COMMAND, QUERY, ANSWER** or **REFER. ANSWER** is used for Yes/No answers. The others are given the usual interpretation.

## 3. CONNECTING LANGUAGE AND LOGIC

Restating the general problem in terms of our basic components, a logical form submitted as a demand for expression must be interpreted by the Nigel inquiries. Nigel must be able to decompose the expressions and characterize their parts.

To achieve this, we have used NIKL to categorize the concepts (or terms) of the domain in terms of Nigel's implicit categorizations. We have written Nigel inquiries which use the structure of the logical language and the NIKL model to analyze the logical forms. To do this efficiently, we have developed a way to translate the logical form into a KL-TWO database and use its reasoning mechanisms.

### 3.1. Functional Systemic Categorizations in a NIKL Knowledge Base

Our NIKL knowledge base is structured in layers. At the top are concepts and roles that reflect the structure we impose on our logical forms. Here we find concepts like **ActionOccurrence** and **Action**, as well as roles like **records**. At the bottom are the entities of the domain. Here we find concepts like **Transmit** and **Send**, as well as roles like **requestedobject**. All of these concepts and roles must be shown as specializing concepts at a third, intermediate level, which we have introduced to support Nigel's generation.

Functional systemic linguists take a Whorfian view: that there is a strong connection between the structures of thought and language. We follow them in categorizing domain concepts in a way that reflects the different linguistic structures that describe them. For example, we have distinguished three types of actions (verbal, mental and material) because the clauses that describe these actions differ in structure. We have at least three types of relations (ascription, circumstantial and

[5] This approach is inspired by representations that associate time and place indices with formulas [Montague 74].

generalized possession) for the same reason[6].

Some of these categories are shown graphically in Figure 3-1. The double arrows are the IS-A links. The single arrows are role restrictions.

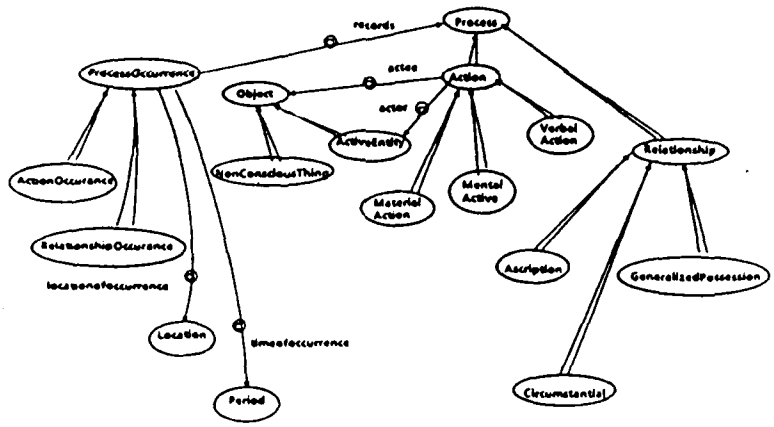



Figure 3-1: Example Upper and Intermediate Layer Categories

Relating these distinctions to our earlier examples, the concepts Transmit and Send are modelled as subclasses of MaterialAction. Message is a kind of NonConsciousThing.

This modelling extends to the role hierarchy, as well. For example, the role requestedobject is modelled as a kind of actee role.

The insertion of systemic distinctions does not compromise other factors, since non-linguistic categorizations can co-exist in the knowledge base with the systemic categories.

Once the domain model is built we expect the systems using the generator to never have to refer to our middle level concepts. Furthermore, we expect Nigel to never refer to any domain concepts.

Since the domain concepts are organized under our middle level, we can note that all domain predicates in logical forms are categorized in systemic terms. To be complete, the domain model must identify each unary predicate with a concept and each binary predicate with a role. The concepts in a logical form must either reflect the highest, most general, concepts in the network or the lowest layer. The domain predicates must therefore relate through domain concepts to systemic categories.

### 3.2. Logical Forms in KL-TWO

Gary Hendrix [Hendrix 75] developed the notion of Partitioned Semantic Networks in order to add the representational power of quantifier scoping, belief spaces, etc., to the semantic network formalism. This does not pay off in terms of faster inferences, but it allows us to separate the two structures inherent in logical formulas, the quantification scopes and the connections of terms. In partitioned networks, these are represented by hierarchically ordered partitions and network arcs, respectively.

This separation of the scope and connection structure is needed. The connection structure can be used to evaluate Nigel's inquiries against the model, and the scope structure can be used to infer additional information concerning quantification.

We translate a logical form into an equivalent KL-TWO structure. All predications appearing in the logical form are put into the PENNI database as assertions. Figure 3-2 shows the set of assertions entered for the formula in Figure 2-1A. These are

[6] Roughly, ascription relates an object to an intrinsic property, such as its color. Circumstantials involve time, place, instrument, etc. In addition to ownership, generalized possession includes such relationships as part/whole and social association.

shown graphically in Figure 3-3 which includes the partitions. KL-TWO does not support partitions. Instead of creating scope partitions, a tree is created which reflects the variable scoping[7].

(ActionOccurrence x) (Past p) (timeofoccurrence x p) (Transmit y)
(records x y) (actor y SMITH) (Message z) (actee y z)

Figure 3-2: Sample PENNI Assertions

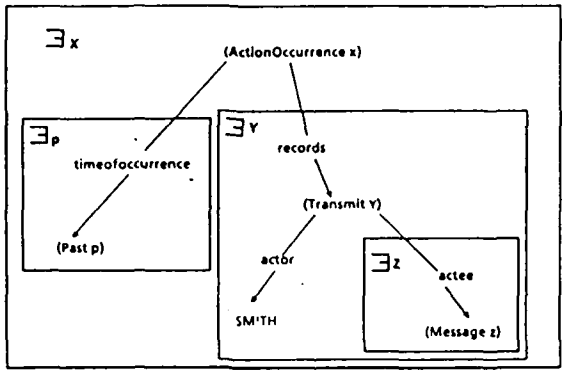



Figure 3-3: Sample Partition Structure

During the translation, the variables and constants are given unique names so that these assertions are not confused with true assertional knowledge (this is not shown in our examples.). These new entities may be viewed as a kind of hypothetical object that Nigel will describe, but the original logical meaning may still be derived by inspecting the assertions and the scope structure.

### 3.3. Implementation of Nigel Inquiries

Our implementation of Nigel's inquiries using the connection and scope structures with the NIKL upper structure is fairly straightforward to describe. Since the logical forms reflecting the world view are in the highest level of the NIKL model, the information decomposition inquiries use these structures to do search and retrieval. With all of the predicates in the domain specializing concepts in the functional systemic level of the NIKL model, information characterization inquiries that consider aspects of the connection structure can test for the truth of appropriate PENNI propositions. The inquiries that relate to information presented in the quantification structure of the logical form will search the scope structure. Finally, to supply lexical entries, we associate lexical entries with NIKL concepts as attached data and use the retrieval methods of PENNI and NIKL to retrieve the appropriate terms.

Let's consider some examples. The generation activity begins with a pointer to the major **ProcessOccurrence**. By the time **CauserID** is asked, Nigel has a pointer to what it knows to be a caused **Action**. **CauserID** is realized by a procedure that finds the thing or things that are in **actor** type relationships to the **Action**. **AffectedID** works similarly through the **actee** predicate. When **VerbalProcessQ** is asked, Nigel simply asks PENNI if a proposition with **VerbalAction** and the **Action** is true.

These examples emphasize the use of the connection structure to analyze what functional systemic grammarians call the **ideational** content of an utterance. In addition, utterances are characterized by **interpersonal** content, e.g., the relation between the hearer and the speaker, and **textual** content, e.g., relation to the last utterance. We have been developing methods for storing this information in a PENNI database, so that interpersonal and textual inquiries can also be answered by asking questions of PENNI.

**MultiplicityQ** is an example of a more involved process. When it is invoked, Nigel has a pointer to an individual to be described. The inquiry identifies all sets as multiple and any non-set individuals as unitary. For non-set variables, it explores their scoping environment. Its most interesting property involves an entity whose quantification suggests an answer of unitary. If the entity is shown in the logical form as a property of or a part of some entity and it is inside the scope of the quantifier that binds that entity and this second entity must be treated as multiple, then both entities are said to be multiple.

---

[7] Here we diverge from Hendrix because of the demands of our language. Separate scopes are kept for the range restriction of a quantification and its predication. In addition the scope of the term forming operators, ι and η are kept in the scope structure.

**TermSpecificationID** is unique in that it explores the NIKL network directly. It is given a pointer to a PENNI individual. It accesses the most specific generic concept PENNI has constructed to describe the individual. It looks at this concept and then up through more general categories until it finds a lexical entry associated with a concept.

## 4. EXAMPLE SENTENCES

Space constraints forbid presentation ofa complete example. Let's look at a few points involved in transforming the three example logical forms in Figure 2-1 into English. Assume for Example 2-1A, that, at this moment, the COMPUTER wishes to communicate to RICHARD the information as an assertion, and that SMITH is known by name through the PENNI database. The flow starts with x identified as the central **ProcessOccurrence**. From there, y is identified as describing the main process.

**TermSpecificationID** is applied to y in one of the first inquiries processed. This is stated to be a **Transmit**. However, we are also told that its **actee** is a **Message**. Assuming the model described in Section 2.2, PENNI concludes that y is not just a **Transmit**, but a **Send** as well. This leads **TermSpecificationID** to look first at **Send** for a lexical entry.

Next, Nigel asks for a pointer to the time being referred to and receives back p. Later this is evaluated against the speaking time to establish the tense.

Further on, Nigel characterizes the process. The inquiries attempt to prove, in turn, that y is a **Relation**, a **MentalActive**, and a **VerbalAction**. When none of these queries are answered positively , it concludes that y is a **MaterialAction**.

After establishing that y is a kind of event that is caused, Nigel uses **CauserID** and **AffectedID**. It receives back SMITH and z, respectively.

The actual decision on how to describe SMITH and z are arrived at during separate passes through the grammar. During the pass for SMITH, **TermSpecificationID** returns his name, "Smith". **MultiplicityQ** is invoked and returns unitary. During the pass for z, **TermSpecificationID** returns "message", while **MultiplicityQ** returns unitary.

In the end, the sentence "Smith sent a message." is generated.

Looking at Example 2-1B, one difference on the level of the outermost **ActionOccurrence** is the absence of an **actee** relation. However, ***requestedobject*** is shown in the model as a type of **actee** relation and **AffectedID** returns q. In order to describe q the grammar forms a relative clause, "which was sent by Smith". There is no overt indication of the type of entity q is. However, from the model of **Send**, PENNI infers that **(Message z)** is true. **TermSpecificationID** for z returns "message". Treating the sentence as a command and assuming "show" is associated with **Display**, Nigel will produce "Show me the message which was sent by Smith.".

Example 2-1C allows us to consider the use of the scope structure in deciding the multiplicity of r. We are required to describe the displaying of the single inspection status (or read status) that is found for each message in a set of messages. As noted, we have modelled **InspectionStatus** as an **Ascription** relation. The grammar uses "of" to describe this sort of relation in a noun phrase. **MultiplicityQ** evaluates m as multiple. Because r is in m's scope, it too is evaluated as multiple and the noun phrase is rendered as "the read statuses of the messages". If the scopings were reversed, the logical form would indicate that there was only one read status for all the messages. **MultiplicityQ** would evaluate r as unitary and the noun phrase would be "the read status of the messages". If both the quantifiers were existential, then each scoping would result in **MultiplicityQ** evaluating both as unitary. The noun phrase would be rendered as "the read status of the message". If m were bound by an $\exists!$, bound by an $\iota$ or replaced by a unitary constant, and r bound by a universal quantifier, the rendering would be "the read statuses of the message".

In Figure 4-1, we display a set of sentences to give the reader some idea of the generator's range as of January 1986. Nigel played the part of both participants in the dialogue, which used hand constructed logical forms and dialogue contexts in the absence of an associated software system.

## 5. CONCLUSION

*User: What is the read status of the message which was sent by Jones?
*System: Message 40 is unread.
*User: Are any messages which were sent by Jones in the tape file?
*System: Yes. There are messages which were sent by Jones. However, the tape is in use.
*User: Did Smith send any messages?
*System: Message 45 was sent by Smith.
*User: Did Smith schedule his New York meeting with me?
*System: Yes.
*User: What does the window show?
*System: The window shows a request by Jones.
*User: Did I create the window?
*System: The window was created when you asked me to display Message 50. You deleted the message, then you asked me to display it.

Figure 4-1: A Sample Set of Generated Sentences

### 5.1. Summary

To summarize, we have developed a first-order predicate-calculus language which can be used to make demands for expressions to the Nigel grammar. This works by translating the logical forms into two separate structures that are stored in a PENNI database. Nigel inquiries are evaluated against these structures through the aid of a NIKL knowledge base. Discourse context is also stored in the data base and lexical entries are obtained from the knowledge base.

Adding this facility to Nigel seems to have added only 10 to 20 percent to Nigel's run time.

### 5.2. Limitations and Future Plans

For the sake of presentation, we have simplified our description of the working system. Other facilities include an extensive tense, aspect and temporal reference system. There is also a facility for dynamically constructing logical forms for referring expressions. This is used when constants are found in other logical forms that cannot be referred to by name or through pronoun.

There are also certain limitations in our approach. One of which may have occurred to the reader is that the language our system produces is ambiguous in ways formal logic is not. For example, "the read statuses of the messages" has one reading which is different from the logical form we used in our example. While scope ambiguities are deeply ingrained in language, they are not a problem in most communication situations.

Related to this problem is a potentially important mismatch between logic and functional systemic grammars. These grammars do not control directly for quantification scope. They treat it as only one aspect of the decision making process about determiner choice and constituent ordering. Certainly, there is a great deal of evidence that logical scoping is not often a factor in the interpretation of utterances[8].

Another set of problems concern the limits we place on logical connectives in logical forms. One limit is the the position of negations: we can only negate **ProcessOccurrences**, e.g., "John didn't send a message.". Negation on other forms, e.g., "John sent no messages.", affects the basic connection with the NIKL model. Furthermore, certain conjunctions have to be shown with a conjunctive **Relation** as opposed to logical conjunction. This includes conjunctions between **ProcessOccurrences** that lead to compound sentences, as well as all disjunctions.

Furthermore, we impose a condition that a demand for expression must concern a single connected set of structures. In operation the system actually ignores parts of the logical form that are independent of the main **ProcessOccurrences**. Because the underlying grammar can only express one event or state of affair(not counting dependent processes) and its associated circumstances at a time, in order to fit in one sentence all the entities to be mentioned must be somehow connected to one event or state of affair.

We expect that the limitations in the last two paragraphs will be overcome as we develop our text planning system,

[8] For example, Keenan and Faltz state "We feel that the reason for the poor correspondence is that NP scope differences in natural language are not in fact coded or in general reflected in the derivational history of an expression. If so, we have a situation where we need something in LF which really doesn't correpond to anything in SF." [Keenan 85, p. 21].

Penman [Mann 83b]. A theory of text structure is being developed at USC/ISI that will take less restrained forms and map them into multi-sentence text structures [Mann 84]. The use of this intermediate facility will mediate for logical connectives and connectivity by presenting the sentence generator with normalized and connected structures.

The word choice decisions the system makes also need to be enhanced. It currently takes as specific a term as possible. Unfortunately, this term could convey only part of the necessary information. Or it could convey more information than that conveyed by the process alone, e.g., in our transmit/send example, "send", unlike "transmit", conveys the existence of a message. We are currently developing a method of dealing with word choice through descriptions in terms of primitive concepts that will support better matching between demands and lexical resources.

A related limit is the requirement in the current NIKL that all aspects of a concept be present in the logical form in order for the NIKL classifier to have effect. For example, the logical forms must show all aspects of a **Send** to identify a **Transmit** as one. A complete model of **Send** will certainly have more role restrictions than the **actee**. However, just having an **actee** which is a **Message** should be sufficient to indicate that a particular **Transmit** is a **Send**. We are working with the developers of NIKL to allow for this type of reasoning.

Two other areas of concern relate directly to our most important current activity. First, it is not clear that first-order logic will be sufficiently expressive for all possible situations. Second, it is not clear the use of hand-built logical forms is sufficient to test our design to its fullest extent.

### 5.3. JANUS

The success of our work to date has led to plans for the inclusion of this design in the **Janus** natural language interface. Janus is a joint effort between USC/ISI and BBN, Inc., to build the next generation natural language interface within the natural language technology component of the Strategic Computing Initiative [Walker 85]. One feature of the system will be the use of higher-order logics. Plans are underway to test the system in actual use. The future direction of the work presented here will be largely determined by the demands of the Janus effort.

**Acknowledgments**

We gratefully acknowledge the assistance of our colleagues Bill Mann, Richard Whitney, Tom Galloway, Robert Albano, Susanna Cumming, Lynn Poulton, Christian Matthiessen and Marc Vilain.

## References

[Appelt 83] Douglas E. Appelt, "Telegram: a grammar formalism for language planning," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pp. 595-599, IJCAI, Aug 1983.

[Brachman 85] R. J. Brachman, V. P. Gilbert, H. J. Levesque, "An Essential Hybrid Reasoning System: Knowledge and Symbol Level Accounts of KRYPTON," in *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pp. 532-539, Los Angeles, CA, August 1985.

[Brachman and Schmolze 85] Brachman, R.J., and Schmolze, J.G., "An Overview of the KL-ONE Knowledge Representation System," *Cognitive Science*, August 1985, 171-216.

[Davidson 67] D. Davidson, "The Logical Form of Action Sentences," in N. Rescher (ed.), *The Logic of Decision and Action*, pp. 81-95, The University of Pittsburgh Press, Pittsburgh, 1967.

[Goldman 75] Goldman, N. M., "Conceptual generation," in R. C. Schank (ed.), *Conceptual Information Processing*, North-Holland, Amsterdam, 1975.

[Habel 82] Christopher Habel, "Referential nets with attributes," in Horecky (ed.), *Proc. COLING-82*, North-Holland, Amsterdam, 1982.

[Halliday 76] Halliday, M. A. K., *System and Function in Language*, Oxford University Press, London, 1976.

[Hendrix 75] G. Hendrix, "Expanding the Utility of Semantic Networks through Partitioning," in *Advance Papers of the Fourth International Joint Conference on Artificial Intelligence*, pp. 115-121, Tbilisi, September 1975.

[Hoeppner et al. 83] Wolfgang Hoeppner, Thomas Christaller, Heinz Marburger, Katharina Morik, Bernhard Nebel, Mike O'Leary, Wolfgang Wahlster, "Beyond domain-independence: experience with the development of a German natural language access system to highly diverse background systems," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pp. 588-594, IJCAI, Aug 1983.

[Hovy 85] E. H. Hovy, "Integrating Text Planning and Production in Generation," in *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pp. 115-121, Los Angeles, CA, August 1985.

[Jacobs 85] Paul Jacobs, *A Knowledge-Based Approach to Language Production*, Ph.D. thesis, University of California, Berkeley, CA, August 1985.

[Kaczmarek 86] T. Kaczmarek, "Recent Developments in NIKL," in *Workshop on Expert Systems*, DARPA, Asilomar, CA, April 1986.

[Kaczmarek, Mark, and Sondheimer 83] T. Kaczmarek, W. Mark, and N. Sondheimer, "The Consul/CUE Interface: An Integrated Interactive Environment," in *Proceedings of CHI '83 Human Factors in Computing Systems*, pp. 98-102, ACM, December 1983.

[Keenan 85] Edward L. Keenan, Leonard M. Faltz, *Boolean Semantics for Natural Language*, Reidel, Boston, 1985.

[Kukich 85] Karen Kukich, "Explanation Structures in XSEL," in *Proceedings of the 23rd Annual Meeting*, ACL, Jul 1985.

[Mann 83a] Mann, W. C., "Inquiry semantics: A functional semantics of natural language grammar," in *Proceedings of the First Annual Conference*, Association for Computational Linguistics, European Chapter, September 1983.

[Mann 83b] Mann, W. C., "An overview of the Penman text generation system," in *Proceedings of the National Conference on Artificial Intelligence*, pp. 261-265, AAAI, August 1983. Also appears as USC/Information Sciences Institute, RR-83-114.

[Mann 84] Mann, W., *Discourse Structures for Text Generation*, USC/Information Sciences Institute, Marina del Rey, CA, Technical Report RR-84-127, February 1984.

[Mann & Matthiessen 83] William C. Mann & Christian M I.M. Matthiessen, *Nigel: A Systemic Grammar for Text Generation*, USC/Information Sciences Institute, Technical Report ISI/RR-83-105, Feb 1983.

[McAllester 82] D.A. McAllester, *Reasoning Utility Package User's Manual*, Massachusetts Institute Technology , Technical Report, April 1982.

[McDonald 83] David D McDonald, "Natural language generation as a computational problem: an introduction," in Brady & Berwick (eds.), *Computational Problems in Discourse*, pp. 209-264, MIT Press, Cambridge, 1983.

[McKeown 85] Kathleen R McKeown, *Text generation: using discourse strategies and focus constraints to generate natural language text*, Cambridge University Press, Cambridge, 1985.

[Montague 74] R. Montague, *Formal Philosophy,* Yale University Press, New Haven, CN, 1974.

[Schmolze and Lipkis 83] James Schmolze and Thomas Lipkis, "Classification in the KL-ONE Knowledge Representation System," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, IJCAI, 1983.

[Shapiro 79] Shapiro, S. C., "Generalized augmented transition network grammars for generation from semantic networks," in *Proceedings of the Seventeenth Meeting of the Association for Computational Linguistics*, pp. 25-29, August 1979.

[Vilain 85] M. Vilain, "The Restricted Language Architecture of a Hybrid Representation System," in *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pp. 547-551, Los Angeles, CA, August 1985.

[Walker 85] E. Walker, R. Weischedel, N. Sondheimer, "Natural Language Interface Technology," in *Strategic Systems Symposium*, DARPA, Monterey, CA, October 1985.

# The Lexicon in Text Generation

**Susanna Cumming, USC/ISI**

## 1. Introduction[1]

In this paper I will review the state of the text generation lexicon. I have two primary goals: 1) to give the reader an idea of what is currently being done, by setting out some of the alternatives designers of generation lexicons have faced, the choices they have made, and the implications of these choices for the types of lexical phenomena they have been able to represent. 2) to suggest what a generation lexicon *could* do, i.e. what range of lexical phenomena is relevant to the generation task. These issues will be addressed more or less in parallel throughout this paper, with more attention to the first goal in the first two sections, and to the second in the last three sections.

The remainder of this Introduction discusses the issue of what kind of linguistic knowledge can be considered lexical knowledge, and what kind of lexical knowledge is most relevent to the generation task. The text generation systems which I have been able to compare are briefly described in section 1.3. Section 2 distinguishes between phrasal and word-based lexicons, and draws some finer distinctions within these two groups. Section 3 sets out the range of cooccurrence phenomena that that a lexicon can treat; section 4 deals with lexical choice and lexical semantics in generation systems. Section 5 presents a summary of the kinds of information an ideal generation lexicon could cover.

There are many aspects of lexical representation which I have chosen not to cover in this paper. I haven't given much space to a description of morphological information, because most of the systems I have information on generate English, which isn't very interesting from a morphological point of view. Since I haven't looked at any systems which generate speech, there is no discussion here of how to represent phonological or phonetic information.

### 1.1. What is lexical knowledge?

A brief examination of a few text generation systems reveals what seem to be staggering differences in the content of the component labelled "lexicon" or "dictionary". Treatments range from dictionaries which contain only information about the endings of nouns and verbs, to systems which store entire sentences as single units in the lexicon; from systems which insert lexical material as a last stage in the derivation process, to systems with lexicons that do the major part of structure-building work. However, this apparent diversity is to a large degree illusory: systems represent the same basic kind of information in different ways and in different components. For instance, information about restrictions on the modifiers a word can take can be treated as part of syntax, as part of semantics, or as a purely idiosyncratic component of a lexical entry. This diversity has its origin in the diversity of practical goals and theoretical underpinnings of the text generation systems which I studied.

The diversity of approaches to lexical representation in linguistic theory is not just an artifact of notational differences; it in turn stems at least partly from the fact that the appropriate characterization of a "word" is different in different subsystems of language. In other words, "word" must be differently defined for the purposes of phonological, orthographic, morphological, syntactic and semantic regularities, although there is a partial overlap (which accounts for the fact that we can frequently get away with using the same term for all these different units). For most of the systems discussed in this paper, the only crucial mismatches are those between the syntactic word and the semantic word (though the orthographic word and the morphological word do occasionally have to be dealt with as well).

Because of this complexity, for the purposes of this paper I will avoid answering in any absolute way the question posed in the title of this section, simply characterizing as "lexical knowledge" that knowledge which at least one of the systems which I review contains in a component called a "lexicon" or "dictionary", while discussing the connections between the structure of particular systems and the decisions made in those systems about whether and how to represent particular pieces of lexical information.

---

[1] This research is supported by the Defense Advanced Research Projects Agency under Contract No. MDA903 81 C 0335. Views and conclusions contained in this report are the author's and should not be interpreted as representing the official opinion or policy of DARPA, the U.S. Government, or any person or agency connected with them.

This paper has benefitted greatly from interaction with a number of my colleagues, most notably Bill Dolan, Cecile Paris, Bob Ingria, Johanna Moore, Lynn Poulton and Sandy Thompson. Special thanks are due to Christian Matthiessen. Any misconceptions or inadequacies that remain are my own.

### 1.2. Understanding vs. generation: different priorities

Before I begin, however, I would like to address the issue of the extent to which the directionality of linguistic processing -- i.e. whether it is a matter of understanding or generation -- influences the content of the lexicon. According to one ideal, in which the language processing system models all of the linguistic knowledge of a human speaker, the relevant information should be the same; and some systems which are bidirectional[2] use the same lexicon for both understanding and generation. However, in practice the two types of lexicon tend to be rather different in the information they encode; even in the bidirectional systems, some of the lexical information is only used in one direction. This is due to differences in the type of demands that apply to most actual understanding and generation projects.[3] A text understanding system has to be able to accept whatever input it gets from the user; this requirement dictates a grammar which is comprehensive at least with respect to a given domain, and a dictionary which is both lexically comprehensive (contains a large number of words) and syntactically comprehensive (supports all the syntactic distinctions that the grammar can make). However, it can assume a fluent and cooperative interlocuter; it doesn't have to weed out input which is textually non-cohesive, unidiomatic, uncooperative, or otherwise "awkward" (with the exception perhaps of gross syntactic ungrammaticality). A generator, on the other hand, doesn't need a full range of syntactic capabilities (one way of saying whatever it needs to say may be enough); nor does it need a very large lexicon (one word for everything it needs to say, and fewer syntactic distinctions corresponding to a smaller syntactic component). But it has to know more about the syntax and lexicon it does have: it has to have a basis for choosing between syntactic alternatives and lexical items so as to be not only conceptually appropriate and grammatical, but also cooperative, idiomatic, non-redundant, and otherwise fluent.[4] Thus, we can say that the generation task sets different priorities for the lexicon: roughly speaking, a generation lexicon has to put depth before breadth, while the reverse is true for understanding.

In this paper I will naturally concentrate my attention to those aspects of lexical specification which are most particular to the generation task.

### 1.3. Systems surveyed

In order to make more concrete the comparison between systems which I will present in the body of this paper, I first give a very brief sketch of each of the systems which I have been able to find out about, with particular attention to the structure and function of the lexicon within the system. More detailed discussion of the interesting features of various of these lexicons will be given in the body of the paper. Citations for the sources from which I have drawn my information are all given in this section; hereafter I will refer to systems by name without repeating the citations. (For the convenience of the reader who may not be familiar with all these systems, I will upper-case system names throughout the text of the paper even when this is not the conventional spelling of the system name, so as to distinguish them from the names of the researchers who developed them. In this section, systems are listed alphabetically for easy reference. In some cases, I have assigned a name to unnamed systems.)

I should add that in most cases, I have not had an opportunity to examine the actual listings for the lexicons I discuss.[5] My statements as to the contents of these listings are inferred from the published descriptions of the systems; frequently only incomplete or suggestive information is provided about the lexicon.[6] Therefore, my comments should be taken as reflecting potential capabilities of particular lexicon formalisms, which may not be fully exploited in the working versions of each system. As the interesting issues have to do with what is possible rather than with what has been done, I don't see this as a liability.

**ANA**: [Kukich 83, ... 83] Generates English text from numerical data about the stock-market. The lexicon contains entries for whole subjects and predicates. Each entry contains morphological information, semantic information matching certain patterns in the data, and stylistic information (which aids in lexical selection) as well as lexical material. The predicate entries contain subject slots, with semantic restrictions on the fillers of these slots.[7] Thus there are predicate entries like "display a hesitant mood early in the day"

---

[2] E.g. JANUS, the VIE LANG system, and PHRED. (References for these and the other systems mentioned in this paper are all given in section 1.3 below.)

[3] This remark, as most of the observations in this paper, applies only to natural language systems which are intended to take one side in a communicative exchange with a user. It does not necessarily apply to systems such as ILIAD, which produces sentences for the purpose of language drill, or to systems which generate random sentences in order to test grammar rules.

[4] An analogy can be made to the experience of a human learning a second language: typically the range of the language which the learner can produce appropriately is much smaller than the range the learner can comprehend.

[5] The exceptions are the lexicons of the JANUS system (which I have worked on), TEXT, and ILIAD.

[6] In many systems, especially those with a case frame orientation, the information available applies only to verb entries. I have much less information about the representation of nouns and even less about other categories.

[7] While there are also slots within predicate entries, these are only for quantitative elements which are inserted from the statistical summary.

and "display a hesitant mood late in the day", and subject entries like "the indexes" and "stock indexes".

ILIAD: [Bates & Wilson 81, Bates, Beinashowitz, Brown, Dougherty, Ingria, Shaked, Simpson & Wilson 81, Bates, Beinashowitz, Ingria & Wilson 81, Bates & Ingria 81]. Generates English sentences designed to test language ability in deaf children. The lexicon contains semantic information relating the entry to a conceptual hierarchy, case-frame information with semantic restrictions on the fillers of the slots, and morphological information.

JANUS: [Cumming & Albano 86, Cumming 86, Mann & Matthiessen 83, Matthiessen 84]. A natural language interface which includes the Nigel systemic generation grammar developed at ISI, and the RUS parser developed at BBN. The parser and the generation grammar share various data structures, including the lexicon. The JANUS lexicon (ML, or Master Lexicon) contains lexical entries which are single words or continuous multi-word phrases; each entry has a feature specification (which contains morphological as well as syntactic features), a semantic specification which is the name of one or more concepts in the knowledge base, and possibly some properties which provide cross-indexing with other lexical entries, values for case and number of pronouns, etc. The features include all the feature information required by the Nigel and RUS grammars; thus some features are used only by one of the grammars. In this discussion my remarks about JANUS feature specification will be aimed primarily at the subset of features used by Nigel.

The features of the Master Lexicon are arranged hierarchically in a tree; they can thus be thought of as defining wordclasses. The wordclass organization contains information about which features are compatible with which other features, and what can constitute a complete feature specification. A word can belong to any number of wordclasses. Thus in some respects the feature hierarchy of the JANUS system is similar to the feature systems represented by the "word ranks" of some other systemic generation grammars (e.g. PROTEUS and SLANG).

KAMP: [Appelt 85a, Appelt 85b, Appelt 83]. Combines a planner with a "teleological grammar" (Telegram) written in Kay's unification framework ( [Kay 79]). The lexical entries map semantic material to lexical material annotated by syntactic features. Unlike some other grammars written in this framework (e.g. McKeown's grammar), lexical entries apparently do not contain internal structure.

MUMBLE: [McDonald 80, McDonald 85, McDonald 83]. This system produces English text from a variety of input meaning representations. It contains two main knowledge structures, the "dictionary" and the "grammar". The dictionary builds structures by matching an element of the semantic representation to a structure containing lexical material and labelled slots. More than one realization of the semantic representation may be specified, so dictionary entries contain "decision-rules" which choose between alternatives on the basis of context; the various possible outcomes are called "choices". The grammar performs realizations on the structures that emerge from the dictionary.[8]

PHRED: [··· 85, Jacobs 83]. The generation half of a natural language dialogue system. Its principal knowledge structure is the "pattern-concept pair", where the pattern is a phrasal unit which specifies structures, features, and lexical material, linked to the "concept", a semantic representation; this may be thought of as the lexicon. The same knowledge is used in understanding and generation.

PROTEUS: [Davey 78]. A systemic grammar which generates descriptions of tic-tac-toe games. It treats the lexicon as a "word rank", as proposed in [Halliday 61]; according to this view of lexis[9], lexical choices are represented in exactly the same way grammatical choices are, as a system network, with their own "rank". In Davey's system, verbs are treated a little differently: the lexical item corresponding to the verb is chosen within the verbal group rather than in the word rank. For convenience, there is also a "lexicon proper" which contains morphological information about the lexical items which inflect.[10]

---

[8] The structure of the MUMBLE dictionary seems to have changed somewhat in the version described in [McDonald 85], with the introduction of domain-independent "realization-classes" which contain some of the more general decision-rule/choice correspondences and which can be referred to by dictionary entries.

[9] Systemic linguists prefer the term "lexis" to "the lexicon", since the latter term evokes images of a single repository of lexical information which is organized around words rather than choices. I'll discuss this distinction farther in section 2.3.

[10] When writing about English, I use the term "inflection" to refer to the addition of endings to nouns, verbs and adjectives to indicate number, tense, person, and degree

**SLANG:** [Patten 86]. Another systemic grammar, which generates from a systemic semantic stratum. Like PROTEUS, it represents lexical distinctions in a word rank of the grammar. However, in SLANG, inflected forms are handled as separate words in the grammar, rather than storing inflectional information in a separate component and doing morphology via a routine.

**SMRAD:** [Kittredge & Mel'chuk 83] . A proposed system which would incorporate the ideas on dictionary content represented in [Mel'chuk et al. 83, Mel'chuk & Zholkovsky 84, Mel'chuk 81], etc. In addition to semantic, syntactic (including case frames), phonological and morphological information, a lexical entry contains *lexical functions* which relate the word being defined to other words which conventionally cooccur with it or have certain other types of semantic relationship with it.

**TEXT:** [McKeown 85, McKeown 83, Derr & McKeown 84]. Generates English text in response to user questions about the structure of a database. The system consists of several components, of which the most important are the strategic component (which creates strings of propositions by selecting a schema and filling it with propositions from the knowledge base with guidance from focus constraints), the dictionary, and the "tactical component", which contains a unification-style grammar and some realization routines. The "dictionary" is intermediate between the strategic component and the Unification-style grammar; it matches semantic predicates to verb entries containing lexical material and argument structures, and fills in the arguments from entires corresponding to the arguments of the semantic representation. The grammar performs transformations and syntactic realization on the output of the dictionary. There is also a "lexicon", which contains morphological information used in realization.

**VIE-LANG:** [Steinacker & Buchberger 83, Buchberger, Steinacker, Trappl, Trost & Leinfellner 82, Steinacker & Trost 83]. A bi-directional German dialogue system; the lexicons (of which one contains morphological information, and the other contains syntactic/semantic information) are shared between the parser and the generator. The syntactic lexicon contains pairs (similar to the "pattern-concept pairs" of PHRED) which match semantic representations to syntactic patterns including lexical material and case structures.

**GAT** [Danlos 85, Gross 84, Danlos 84]. (As far as I can tell, this system is unnamed; I've given it the acronym GAT from the name of [Danlos 85].) Generates reports of terrorist attacks in English and French, from summaries of the attacks. It uses the lexicon/grammar developed by M. Gross and others at the LADL project in Paris: the lexicon can be thought of as a list of all the "simple sentences" which exist in the language, with labelled slots for the noun phrase arguments. The "simple sentences" have features specifying the transformations they can undergo, characteristics of the arguments that can fill the slots, etc. These "simple sentences" are such things as "ACTOR explode EXPLOSIVE in VICTIM'S:LOCATION", or "ACTOR open fire on VICTIM'S:VEHICLE".[11]

## 2. Phrasal lexicons and word-based lexicons

The lexicons used in text generation systems can be roughly grouped into two classes, according to what is represented in a typical lexical entry (unit of the lexicon). One class contains lexicons whose entries are typically single words, like the lexicons of traditional linguistic theory; the other class contains lexicons whose entries typically represent larger constituents, phrases or even sentences, with some lexical material (by which I mean orthographically-realized words which will appear in the output string), and usually also some slots or variables which can be instantiated with further lexical material or lexical entries. The distinction between these two types isn't always clear-cut. Some systems, as mentioned above, have both types, in which typically the phrasal lexicon represents syntactic and semantic information, and the word-based lexicon represents morphological information;[12] others can easily provide either type of representation, and it is a question of practice which alternative is chosen in any given case.

---

[11] My translation of Danlos' examples. The upper case words are the slots, which are filled in from the event summaries.

[12] This is a much more efficient style of representation where there is a lot of morphological information to be expressed, since in most systems different senses of the same (orthographic or phonological) word will receive different lexical entries, but the inflection will be the same. For example, *be* as a passive auxiliary (as in *the bug was eaten by the bat*) and *be* as a copula (as in *the bug was a spider*) are very different syntactically and semantically, but they share the same inflected forms (i.e. *am, are, is, was, were, been, being*), as do all the other uses of the verb spelling *be*. If a morphological and a syntactic/semantic lexicon are distinguished, the information about the forms of *be* only needs to be represented once. In English, the amount of inflectional information that needs to be specified is so small that this may not be an important consideration (*be* is an extreme example), but in other Indo-European languages it becomes much more important.

### 2.1. Phrasal lexicons

Perhaps the most important factors distinguishing generation lexicons are the size of the lexical item, the amount of structure it contains, and the role of lexical selection in the system. In text generation, as opposed to understanding, there seems to be a tendency towards a large size, complex structure, and powerful role for the lexical item. In this section, I will discuss the reasons for each of these tendencies and their implications for text generation; in section 3, I will describe how more traditional word-based lexicons handle the same range of phenomena.

#### 2.1.1. Size

While traditional dictionaries are primarily organized around small linguistic units -- words or even morphemes -- many computational lexicons have entire syntactic constituents stored as their basic unit, all the way up to multi-clausal units. (These lexicons can conveniently be described as "phrasal", although as we will see the kind of unit which counts as a "phrase" varies widely. An argument for this treatment can be found in [Becker 75].) This has several advantages in text generation: 1) all kinds of subcategorization and selectional restrictions which need to be stated as properties of particular lexical items can easily be handled without any special mechanism: the allowed patterns are listed in the lexicon, and the disallowed patterns aren't. Any combination of complement types may be represented without the necessity of deciding beforehand on a particular inventory of possibilities. 2) Similarly, all kinds of idioms and collocational restrictions can potentially be handled by specifying the exact wording of the lexical phrase. 3) An indefinitely large syntactic range may be "simulated" by treating syntactic constructions which can't be generated by the grammar as idioms, thus adding to the syntactic variety of the output text. This principle may be extended to the point where the lexicon "takes over" most of the grammar, i.e. all or almost all grammatical patterns are represented only in the specification for the lexical items which they apply to.

The disadvantages of this method are merely the flip side of the advantages. Generally speaking, the more phenomena which are represented as idiosyncratic properties of lexical items, the fewer phenomena are treated in a general way (although some systems have the flexibility to represent the same phenomena as either idiosyncratic or as general). This has two related consequences: 1) lexicons must be much larger; 2) making additions to the lexicon is a much more lengthy and difficult process, as properties of lexical items which may in fact be predictable (on the basis of either other lexical properties or semantic properties of the item) must be specified anyway.

#### 2.1.2. Structure

Phrasal lexicons differ in the amount of internal structure they can encode within their phrases. Thus, there is a difference between encoding an idiom like *go mad* as a verb or predicate with no internal structure indicated and knowing that *go* is a verb and *mad* is a resultative adjective phrase. If there is internal structure indicated, it is possible to store each of these variants as a single lexical item (which may be desirable, since the phenomenon is not generally productive), and yet still do some syntactic variation, e.g., add intervening adverbials (*go quietly mad*), inflect the verb (*I go mad, he goes mad*), or relate it to other syntactically similar expressions (*go crazy, run dry*). Another reason one might want information about the internal structure of phrases is for stylistic control, e.g. to allow control of the amount of variation in lexical choice and syntactic structure.[13] The lexicons of TEXT, PHRED, VIE-LANG, and MUMBLE all allow any amount of internal structure to be specified in a lexical item, in contrast to GAT and ANA; while these two systems contain slots for other elements (various arguments in GAT, subjects only in ANA), they cannot indicate any further structural complexity.

#### 2.1.3. Depth of lexical selection

Another important parameter which distinguishes generation lexicons is the amount of influence lexical choice has over other kinds of choices, for example syntactic, rhetorical or stylistic choices, made in the system. To cite some instances of restrictions imposed by lexical items on clause syntax: some verbs with direct objects can't be passivized (e.g. *the candy bar cost a quarter*), verbs (and to a lesser degree adjectives and nouns) restrict the syntax of their complement clauses in various ways (e.g. *I insist that he come* vs. **I insist that he comes*, but *I hope that he comes* vs. **I hope that he come*) and some pronouns can be modified by relative clauses while others can't (e.g. *Anyone who wants to can come* but **We who want to can come*[14]). Naturally, the degree of constraint the availability of lexical items can impose on grammatical choice is directly related to the stage in the generation process (or "depth", in terms of the metaphor current in transformational grammar) at which lexical choice is made. If lexical choice is made late in the generation process, it can have little input into other decision-making, unless some kind of backtracking is allowed for.

In many systems, the lexicon acts as the intermediary between semantic and syntactic representations, and the step of "lexical

---

[13] Kukich discusses this point in [... 83], p. 124)

[14] The latter example may be grammatical with a nonrestrictive reading, but it is not possible with a restrictive reading.

insertion" is actually the step at which syntactic structure is built. (This is the case for MUMBLE, TEXT, PHRED, VIE-LANG, and GAT.) This generally works by matching the predicate of the semantic representation with the lexical entry for a verb, and then filling in the argument slots of the verb with arguments from the semantic representation. (It may also be more complicated than this: in both TEXT and MUMBLE, for instance, the way this matching is done may involve information from contextual information such as focus history or preceding reference; in ANA stylistic factors such as length are considered; etc.) In these systems, the structure built by the lexicon then undergoes further syntactic realization (e.g. transformations, morphological adjustments). Since the lexical item has already been chosen when these realizations are performed, properties of the lexical item have the opportunity to constrain the way these realizations occur. For example, in the TEXT system, routines in the dictionary itself control the choice of syntactic construction (active, passive or existential) as well as the basic sentence structure. This avoids problems such as a text plan calling for passive syntax when the verb in question can't be passivized. In KAMP, syntactic processing (including lexical insertion) is alternated with planning in such a way that plans can be modified in response to the set of choices made available by a particular lexical item. In GAT, all the decisions are made simultaneously by the selection of a particular schema which includes lexical, (clause-level) syntactic and clause-combining specifications.

Of course, if a grammar is sufficiently rich to treat as regular (i.e. as predictable from aspects of the specification of the sentence) a large range of syntactic phenomena, a correspondingly small range needs to be treated as idiosyncratic to a lexical item (i.e., as dependent on a particular lexical choice). This is another form of the tradeoff between grammar and lexicon: the more complete a grammar is, the less dependent it is on early lexical specification to do its job right. Thus, in Nigel, most of the the syntactic properties of a lexical item are taken to be predictable from its semantic properties, following Halliday's analysis; so, although a particular lexical item isn't chosen until after syntactic planning has occurred, the syntactic plan is made with reference to the same semantic categories that constrain lexical choice.[15] For example, non-subjunctive "that" clauses, since they refer to reports about the world, are restricted to verbs of saying and thinking.

### 2.2. Word-based lexicons

Many of the models of language to come out of linguistics until recently assume a word-based lexicon in which syntactic information is specified in the form of features; word choice is constrained both on the basis of meaning and the fit between the syntactic features of the word and the syntactic environment it is supposed to fit into. In these models, rather than having the powerful role it has in the systems discussed above, the lexicon is primarily viewed as an appendage to the syntax, where information which can't be predicted by general rules is stored. The units represented are small (usually morphemes), and the amount of internal structure which can be represented within an item is minimal. Systems surveyed here which have this traditional type of lexicon are ILIAD, KAMP,[16] and Mel'chuk's system.[17]

In some ways, the difference in practice between a low-level word-based lexicon with features and a highly structured phrasal lexicon is smaller than it appears. For example, a case-frame representation can be mapped onto a feature representation in which the feature corresponds to a particular case pattern -- e.g. the feature "transitive" can be mapped onto a case frame containing a direct object slot. The major difference is that the case frame representation allows more freedom than is available with a small set of features (as mentioned above); on the other hand, since features can be thought of as corresponding to classes of lexical items, a single lexical feature may efficiently encode a range of possible case frames that tend to cooccur with a particular type of word. In the lexical feature specifications referred to by Nigel, all of the subcategorizational possibilities of a particular sense of a verb are taken to be predictable from a single feature representing its wordclass membership.[18] Thus, verbs such as "see", "hear" etc. have the feature "perception"; the grammar knows that these verbs can be generated with either a direct object, with a complement clause in which the verb is in its stem form without "to" (e.g. "I saw you arrive", "I heard her come in"), or with a complement clause in which the verb is in its present participle form ("I saw you arriving", "I heard her coming in"). This particular configuration of possible complements is restricted to verbs which refer to sense perception, and thus it is redundant to list each of these possibilities separately for all the perception verbs.

Of course, to take advantage of this type of generalization one must have a detailed theory of the wordclasses of a language,

---

[15] This statement, of course, is relative to a particular view of the characterization of both syntax and semantics; for more discussion of this point, see section 4 below.

[16] Although the Unification formalism used in KAMP allows for lexical entries containing further structure, just as in Lexical Functional Grammar representations, as far as I know Appelt doesn't exploit this possibility in his system.

[17] Although Mel'chuk's dictionaries contain an unusual degree of cross-referencing between entries, they are still primarily organized around entries for single words.

[18] These features are related to the semantic type of the verb as represented in the position of the corresponding concept in the semantic network; however, the relationship is not direct. As we will see in section 3.2 below, case-frame phenomena and selectional restrictions are also handled in the JANUS system; however, they are treated purely as part of knowledge about word meanings, and therefore represented in the semantic net rather than the lexicon.

such as is available in the systemic framework; and indeed, it's clear that a reasonably complete grammar must make referenc to a very large set of such wordclasses. This is another case of a tradeoff between having a relatively complex rule system and treating few things as "irregular" or unpredictable, and having a relatively simple rule system and treating many things as irregular. In the computational context, the first option implies a large development effort in the area of grammar, while the second implies a large effort in the area of lexicon. Which option is preferable depends on the goals of the system.

### 2.3. Systemic grammars

The systemic approach to lexical classification exemplified in SLANG and PROTEUS doesn't fall easily into either of the categories described above, although in practice these two systems, like Nigel, have the closest affinity with word-based systems, since neither supports phrasal lexical items.

The "word rank" of a systemic grammar represents alternatives among word classes in the same way the grammar represents grammatical alternatives; the result is a highly structured feature system. Within the word rank, successive choices lead to actual words in the case of closed class items or "function words" such as prepositions, verbal auxiliaries, and connectives; these can be thought of as words with unique feature specifications. As mentioned above, the wordclass hierarchy of JANUS is similar in some ways to a word rank; however, it is more limited in the kinds of relationships it can represent between features.

In systemic theory, choices between open class items fall into the area called "lexis", often envisioned as an entirely separate level of grammar ( [Halliday, McIntosh, & Strevens 64, Berry 77, Halliday 76]). It has been proposed that lexis could ultimately be entirely incorporated into the grammar -- that is, that finer and finer (or, as systemicists say, "more and more delicate") decisions could ultimately distinguish every word from every other word -- but this "dream" (as Halliday has called it, [Halliday 61]) has never been completely realized.

## 3. Approaches to cooccurrence phenomena

Now that we have surveyed the various kinds of lexicon and the way they interact with the systems of which they form a part, we can take a look at the range of phenomena that they express, and consider the implications of these phenomena for optimal lexicon design. Most of the syntactic information (and some of the semantic information) that needs to be specified about lexical items can be subsumed under the term "cooccurrence information", i.e. information about which other linguistic elements (lexical items or syntactic types) a particular item can "go with". I will discuss here four distinct types of cooccurrence phenomena: subcategorization, selectional restrictions, collocation, and idioms[19] By "subcategorization" I mean specification of the syntactic or semantic frame(s) an item can occur in, such as the fact that *think* can take a clausal complement with *that* but not a complement with *to*. By "selectional restrictions" I mean semantic restrictions on the fillers of subcategorization frames, such as the restriction on the subject of the verb *elapse* that it refer to a period of time. By "collocation" I mean lexical restrictions (restrictions which are not predictable from the syntactic or semantic properties of the items) on the modifiers of an item; for example, you can say *answer the door* but not *answer the window*. By "idiom" I mean a fixed phrase whose meaning is noncompositional, i.e. not predictable from the meanings of its parts, e.g. *a one-track mind*; an idiom may be "ungrammatical" (i.e., not generatable by independently motivated rules) if interpreted compositionally, e.g. *all of a sudden*.

A consideration of these definitions will at once suggest that the extension of these classes of phenomena depends largely on the particular model to which they are applied. Whether something needs to be treated as compositional or not will depend on the rules that are available to generate it; there are large numbers of constructions which apply to very limited classes of words. For example, there is a set of expressions *hundreds and hundreds, thousands and thousands* etc; this construction is limited to the number words that act like common nouns in that they can be plural and take articles (so we get *a dozen, several dozens, dozens and dozens* but not **a twelve, *several twelves, *twelves and twelves*), and also to other kinds of quantity expressions, e.g. *barrels and barrels*. While this could be treated as a regular grammatical construction, it is sufficiently limited in generality that few computational grammars will include it in their syntactic scope; it may be more cost-effective to treat this kind of phenomenon as idiomatic. Similarly, what could be stated as a selectional restriction if one has the right semantic classes in one's model, may have to be stated as a set of collocations or idioms otherwise. And the line between selection and subcategorization is blurred when syntactic properties are taken to be predictable from semantic classes.

---

[19] My use of the terms "subcategorization" and "selectional restriction" is largely derived from their use in classical transformational theory. "Collocation" in this sense can be traced back to [Firth 57]; my sense is related most specifically to Firth's "general or usual collocations". "Idiom" as used here is more restricted than the sense it is given in e.g. *Longman Dictionary of English Idioms*, [Longman 79] (which includes collocations, standard metaphors, proverbs etc. as well); it is closer to what are characterized as "traditional idioms" in the introduction to Longman's.

### 3.1. Subcategorization

The handling of subcategorization in several models has been touched on above, in section 2.1.1. To reiterate, most phrasal or case-frame lexicons indicate subcategorization by using slots in a lexical entry. The following lexical entry from PHRED ( [--- 85], p. 221) for the verb *remove* is fairly representative:

```
<agent> <root = remove> <physob>
<<word = from> <container>>
```

This entry contains the information that the verb "*remove*" takes a subject (which is an agent), a direct object, and prepositional phrase with *from*. (It also places certain semantic restrictions on the fillers of these slots.)

Word-based lexicons, on the other hand, generally deal with subcategorization by providing lists of features. The entry from the JANUS lexicon for the same verb contains the following syntactic (and morphological) information:

```
(make-lexical-item
 :name 'REMOVE
 :spelling "remove"
 :features '(VERB INFLECTABLE UNITARYSPELLING S-D LEXICAL
CASEPREPOSITIONS OBJECTPERMITTED PASSIVE DOVERB
DISPOSAL EFFECTIVE) )
```

### 3.2. Selectional restrictions

Some lexicons can handle selectional restriction by attaching semantic restrictions to lexical entry slots. The labels *agent*, *physob* and *container* in the Phred example above can be thought of as selectional restrictions. ILIAD lexical entries contain similar restrictions; for examples, the entry for (the verb) "grease" is as follows:

```
(GREASE SYNCASES
 ((SUBJ (HEADCONCEPT T) (MUST-BE (OR (ADULT CHILD))))
  (OBJ (HEADCONCEPT T) (MUST-BE VEHICLE))))
```

This says that the subject of "*grease*" must be a word that refers to an adult or a child, while the object must refer to a vehicle. ANA's predicate contain feature restrictions on their subjects (e.g., the entry for *display a hesitant mood early in the day* has the features **tsubjtype NAME tsubjclass MKT**, indicating that the subject must be a name for the stock market), and the slots in the "simple sentence" lexical items of the LADL grammar may have semantic feature restrictions such as +HUMAN associated with them.

In other lexicons, including that of JANUS and TEXT, selectional restrictions aren't directly represented in the lexicon at all; rather, these restrictions are in fact captured in another part of the system -- the semantic network. This option is available to systems that are based on semantic networks composed of hierarchically-arranged concepts, related to one another by "case roles" (which specify the semantic roles a concept has and the other concepts that represent possible fillers of each role). In systems that use a semantic net as the source of the representations which go to the grammar, selectional restrictions are already enforced in the representation that goes to the grammar for expression. This is equivalent to saying that selection, unlike subcategorization, derives from knowledge about the meanings of words rather than lexical knowledge specific to the linguistic expressions of those meanings.[20]

### 3.3. Collocation

The phenomenon which I've called collocation is of particular interest in the context of a paper on the lexicon in text generation because this particular type of idiom is something which a generator needs to know about, while a parser may not. For example, consider the expression *wreak havoc*. This can be parsed compositionally as a verb and its object without any special knowledge; but a generator must know about the special connection between these words, since neither word is found very often in any other context; we need to avoid generating *wreak a mess*, *make havoc*. (Many more examples of this kind of expression can be found in [Makkai 72, Chafe 68, Fillmore 79, Fillmore, Kay & O'Conner 84].) Because of this, this set of phenomena has been labelled "idioms of encoding",[21] i.e. expressions which are compositional, and may seem semantically

---

[20] Systems differ, however, in how close the mappings are between concepts and words, semantic role specifications and syntactic case frames; in some systems it would be hard to make an argument that the properties of the "concepts" of the semantic net aren't simply properties of the words used to express those concepts in a particular language, or that the "semantic roles" on those concepts aren't really labels for syntactic arguments. For more discussion of this issue, see section 4 below.

[21] I believe the term comes from [Makkai 72].

### 3.1. Subcategorization

The handling of subcategorization in several models has been touched on above, in section 2.1.1. To reiterate, most phrasal or case-frame lexicons indicate subcategorization by using slots in a lexical entry. The following lexical entry from PHRED ( [--- 85], p. 221) for the verb *remove* is fairly representative:

```
<agent> <root = remove> <physob>
<<word = from> <container>>
```

This entry contains the information that the verb "remove" takes a subject (which is an agent), a direct object, and prepositional phrase with *from*. (It also places certain semantic restrictions on the fillers of these slots.)

Word-based lexicons, on the other hand, generally deal with subcategorization by providing lists of features. The entry from the JANUS lexicon for the same verb contains the following syntactic (and morphological) information:

```
(make-lexical-item
 :name 'REMOVE
 :spelling "remove"
 :features '(VERB INFLECTABLE UNITARYSPELLING S-D LEXICAL
CASEPREPOSITIONS OBJECTPERMITTED PASSIVE DOVERB
DISPOSAL EFFECTIVE) )
```

### 3.2. Selectional restrictions

Some lexicons can handle selectional restriction by attaching semantic restrictions to lexical entry slots. The labels *agent*, *physob* and *container* in the Phred example above can be thought of as selectional restrictions. ILIAD lexical entries contain similar restrictions; for examples, the entry for (the verb) "grease" is as follows:

```
(GREASE SYNCASES
 ((SUBJ (HEADCONCEPT T) (MUST-BE (OR (ADULT CHILD))))
  (OBJ (HEADCONCEPT T) (MUST-BE VEHICLE))))
```

This says that the subject of "grease" must be a word that refers to an adult or a child, while the object must refer to a vehicle. ANA's predicate contain feature restrictions on their subjects (e.g., the entry for *display a hesitant mood early in the day* has the features **tsubjtype NAME tsubjclass MKT**, indicating that the subject must be a name for the stock market), and the slots in the "simple sentence" lexical items of the LADL grammar may have semantic feature restrictions such as +HUMAN associated with them.

In other lexicons, including that of JANUS and TEXT, selectional restrictions aren't directly represented in the lexicon at all; rather, these restrictions are in fact captured in another part of the system -- the semantic network. This option is available to systems that are based on semantic networks composed of hierarchically-arranged concepts, related to one another by "case roles" (which specify the semantic roles a concept has and the other concepts that represent possible fillers of each role). In systems that use a semantic net as the source of the representations which go to the grammar, selectional restrictions are already enforced in the representation that goes to the grammar for expression. This is equivalent to saying that selection, unlike subcategorization, derives from knowledge about the meanings of words rather than lexical knowledge specific to the linguistic expressions of those meanings.[20]

### 3.3. Collocation

The phenomenon which I've called collocation is of particular interest in the context of a paper on the lexicon in text generation because this particular type of idiom is something which a generator needs to know about, while a parser may not. For example, consider the expression *wreak havoc*. This can be parsed compositionally as a verb and its object without any special knowledge; but a generator must know about the special connection between these words, since neither word is found very often in any other context; we need to avoid generating *wreak a mess*, *make havoc*. (Many more examples of this kind of expression can be found in [Makkai 72, Chafe 68, Fillmore 79, Fillmore, Kay & O'Conner 84].) Because of this, this set of phenomena has been labelled "idioms of encoding",[21] i.e. expressions which are compositional, and may seem semantically

---

[20] Systems differ, however, in how close the mappings are between concepts and words, semantic role specifications and syntactic case frames; in some systems it would be hard to make an argument that the properties of the "concepts" of the semantic net aren't simply properties of the words used to express those concepts in a particular language, or that the "semantic roles" on those concepts aren't really labels for syntactic arguments. For more discussion of this issue, see section 4 below.

[21] I believe the term comes from [Makkai 72].

transparent to a hearer but require specialized knowledge on the part of a speaker to produce correctly; non-compositional cooccurrence phenomena like *kick the bucket*, the ones which I call "idioms" here, correspond to Fillmore's "idioms of decoding"; both a parser and a generator must have knowledge of these.

Collocation phenomena aren't explicitly handled as such by any of the systems discussed so far.[22] They can, of course, be handled after a fashion, either by treating them as cases of selection (as the JANUS system does) or as cases of idioms (as in the PHRED system). If they're handled as selection, the distinction between idiosyncratic lexical properties and general semantic properties is lost; and if they're handled as idioms, the regular syntactic behaviour and semantic compositionality of these phrases isn't expressed. Thus, neither of these solutions is perfectly satisfactory, although one or the other may be adequate for a small domain in which full generality isn't crucial.

The only system I'm aware of which addresses this kind of phenomenon in a thorough and explicit way is Mel'chuk's proposal. He has proposed a device called the "lexical function", which he uses extensively to relate dictionary entries in his "explanatory and combinatorial" dictionaries of Russian and French. There are a large number of these lexical functions (62 "standard" ones, and an arbitrary number of "non-standard" ones), but they can be roughly divided into two groups: those that deal with paradigmatic relationships between words (meaning relationships such as hyponymy, synonymy, antonymy etc., plus words with related meanings but permuted argument structures; for more discussion of some of these phenomena, see section 4 below), and those that deal with syntagmatic relations -- standard words for the various arguments and modifiers of a term. It is this latter group of lexical functions that can be taken as expressing collocational phenomena. For example, there is a function *Magn* which relates a word with a modifier which has the meaning "to a great degree"; the words "shave", "easy", "scoundrel" have as Magns "close", "as pie", and "unmitigated" respectively. Presumably these lexical functions will be exploited in the SMRAD text generation system proposed in [Kittredge & Mel'chuk 83].[23]

### 3.4. Idioms

Idioms have been discussed in some detail in section 2.1.1 above and in the preceding paragraphs of this section. To reiterate, most phrasal lexicons can generally handle idioms without any special provisions, either by treating all pieces of the idiom as part of the same word as in Kukich's system, or (in case-frame lexicons) by having some of the slots filled in with lexical material. For example, in PHRED, *tell (someone) to get lost* is

⟨person⟩ ⟨root = tell⟩ ⟨person⟩
⟨word = to⟩ ⟨word = get⟩ ⟨word = lost⟩

(Note that there is relatively internal structure to this idiom; in particular, "to get lost" is not a clause.)

In the word-based systems I've surveyed, idioms can only be handled as single words, with no intervening material (thus *kick the bucket* can be handled -- as an intransitive verb -- but *knock (someone's) block off* can't be, and *kicked/kicks the bucket* may or may not be.) JANUS can't handle internal inflection, so idioms which are verb phrases aren't possible at all; however, anything that doesn't have to inflect internally is allowed, such as many noun phrase idioms (such as *red herring*, which can pluralize appropriately as *red herrings* and such things as complex prepositions (such as *face to face with*, *on account of*) can be handled.

## 4. Lexical semantics and lexical choice

If the phenomena treated in the previous section are characterized as phenomena of syntagmatic organization -- i.e. facts about what a lexical item can occur next to -- then the facts discussed in this section can be thought of as facts about paradigmatic organization -- i.e. facts about what a lexical item can occur instead of, or facts about lexical choice and meaning relations between words of the same class. The topic of lexical semantics will be treated only rather briefly in this paper (relative, at least, to the amount that has been said about it in the theoretical literature), since not all systems have an identifiable component of lexical semantics -- separate, that is, from whatever organizing principles underlie the elements of the demands for expression that are interpreted by the generator. Similarly, not all systems have an explicit strategy for lexical choice, relying instead on a one-to-one mapping between items in the lexicon and possible elements of the semantic representation to obviate the need for decision procedures.

---

[22] While Jacobs discusses these phenomena in [... 85], he doesn't actually distinguish them from idioms (of decoding) in his system.

[23] Hudson distinguishes idioms and collocations more or less the same way I do here in his "Word Grammar" theory, [Hudson 84] (and his abstract for this conference); his theory is actually quite similar to Mel'chuk's dependency grammar. However, as far as I know Hudson has no proposal for a text generator, so a discussion of his account would be out of place here.

I'll divide my discussion of semantic phenomena into two sections, of which the first is principally concerned with semantic classification and the second with how lexical choices are made.

### 4.1. Semantic classification

The two basic methods by which systems notate semantic classification of lexical items are by feature systems and by taxonomies. (While Mel'chuk's paradigmatic lexical functions might appear to represent a third system, they are based on an underlying taxonomy.) The only lexicon which uses a pure feature system is that of ANA: the phrases of ANA's systemare represented as feature clusters (or, more accurately, as clusters of attribute-value pairs). For example, the four entries **display a hesitant mood early in the day, display a hesitant mood late in the day, creep upward early in the session,** and **creep upward late in the session,** for example, are distinguished by the values of the two attributes **ttim** (time) and **tdeg** (degree).

Explicit taxonomic concept hierarchies represent (at least) relations of inclusion among word meanings. Thus, a taxonomy can represent the fact that a cat is a kind of animal; i.e. that the set of cats is included in the set of animals. Taxonomies also can represent the inheritance of properties from more general to less general concepts; thus, if a cat is an animal and an animal can have young, then a cat can have young. Taxonomies are composed of concepts, each of which may be associated with one or more lexical entries; the lexicon is generally the place where the correspondence between concepts and words is stated. In the above example, we can say that the concept which is associated with the word "cat" is a **subconcept** of the concept which is associated with the word "animal", and that the concept which is associated with the word "animal" is a **superconcept** of the concept which is associated with the word "cat". In the following discussion, I will use upper case for concept names to avoid confusing them with their associated lexical items.

Systems with taxonomies use taxonomic information in radically different ways. In TEXT, a taxonomy is actually the source of the semantic representations (propositions) from which sentences are generated, since the purpose of the generator is to describe the taxonomy. In JANUS, taxonomic information is used in the reasoning performed by the grammar during the generation of sentences. Thus, if the system is generating the sentence "Jones sent the message", the grammar will look at the taxonomy to see if SEND is the kind of process that typically has an agent. In fact, SEND is a subconcept of the concept DIRECTED ACTION, and since the grammar knows that directed actions have agents it will construct an agent noun phrase. Thus, the taxonomy employed in the JANUS system contains all the category distinctions relevant to grammatical choice.

In ILIAD, since its function is to provide grammar drills, the demand for expression consists of a syntactic form; the semantic taxonomy is used to ensure that the sentence which is finally generated is semantically coherent, i.e., doesn't violate selectional restrictions. Thus, lexical choice is primarily conditioned by selectional restrictions stated in terms of the taxonomy. For instance, in the example in section 3.2 above, once "grease" had been chosen as a main verb the only lexical items which would be considered for the direct object would be those associated with subconcepts of VEHICLE. (Since the actual semantic content of the generated sentence is unimportant in ILIAD, once selectional restrictions have been satisfied, lexical choice is essentially random.)

Mel'chuk's system contains a richer specification of paradigmatic relations than any of the systems so far discussed. In addition to hyponymy (the relation between a concept and its superconcept), he has functions for different kinds of synonyms, antonyms, words which have the same basic meaning but with the syntactic roles of the arguments interchanged (e.g. "buy" and "sell"), and many others that aren't so easily classifiable. This richness is vital in a system whose primary goal is paraphrase or translation, since it gives the system access to a great deal of knowledge about what expressions can be considered semantically equivalent, something not available from a simple taxonomy.

### 4.2. Lexical choice

As described above, some systems do all their lexical choice in what might be called the semantics -- that is, by the time they've decided what to say and before they've looked into the lexicon, they've already committed themselves to a particular wording. Systemic grammars with word ranks, conversely, treat lexical choice as part of the grammar (often referred to by systemicists as "lexico-grammar" for this reason. This is true even in JANUS for closed class items, since these are uniquely selected in various ranks of the grammar.) However, there are some systems which have routines for performing lexical choice built into the lexicon itself.

TEXT has choice routines built into the dictionary, but they are limited to choice of syntactic category: a given element in the demand for expression can have lexical realization in more than one category. For example, SURFACE can be realized as "surface" if it is an adjective or a noun, or as "on the surface" if it is a prepositional phrase. MUMBLE's decision rules combine grammatical choices with stylistic choices. ANA provides in the lexicon for choosing in order to enhance stylistic

variation of various kinds. Each entry is annotated for its length in syllables, and other things being equal, the grammar chooses so as to alternate two long sentences with one short one; similarly, each subject entry is annotated for "hyponym level", so that on the first mention of a given referent a more specific or more heavily modified phrase is used, and on subsequent mentions more general or briefer phrases are used; for example, *the Dow*, *the industrials average*, and *the Dow Jones average of 30 industrials* have successively lower hyponym levels.

## 5. Some goals for the generation lexicon

In this section I would like to both summarize the directions which have already been touched on for the generation lexicon, and add a few new goals to the wish list. These are intended to be goals which system implementors, regardless of the overall design or underlying linguistic framework of the system, might consider handling somewhere in the system. Some of these goals are met in some of the systems described here; others as far as I know have not been adequately dealt with in any working text generation system, and can thus be considered fruitful areas for future research. Many of them will only be relevant in a really comprehensive text generation system, and can easily be ignored in systems which operate in highly restricted domains.

### 5.1. Syntactic range

This isn't, of course, strictly a lexicon issue, but one that has repercussions for lexicon design. Most current systems are able to give quite detailed specifications for the subcategorizational properties of verbs, but other syntactic categories also impose subcategorization restrictions on their modifiers. For example, nouns and adjectives[24] can take postmodifying clauses with *that* (*the fact **that** the world is round is well known, it's good **that** you could make it*) just as can certain verbs. Similarly, all of the systems I looked at know about the inflections of verbs (e.g. *run/runs/ran/run/running*) and nouns (e.g. *book/books* or *goose/geese*), and some know about the inflections of adjectives (e.g. *large/larger/largest*, but none that I know of can generate inflected adverbs, which have the same possibilities as adjectives in English (e.g. *He ran fast/faster/fastest.*).[25] For a complete coverage, these possibilities must be allowed for.

### 5.2. The intelligent lexicon

It is a common observation that human languages have many words for things that their speakers commonly talk about -- cf. the famous claim (attributed to Whorf) that the Eskimos have twenty words for snow. Less universally accepted is the converse claim, that people tend to think/talk about things that their language has many words for. Whether or not this is the case, it seems to me that it is a desireable goal for a text generation system that it should not plan to say things which it does not have the lexical resources to actually produce.[26] In order to assure that this does not happen, the lexical resources of a system should be consulted along with the grammar, semantics, and strategic components in planning what to say, so that if it is not possible to say something using a single word a periphrastic expression can be planned. As mentioned above, work has been done on this problem in JANUS; KAMP and MUMBLE also both allow for some interaction between planning and linguistic realization such that this kind of negotiation is feasible.

### 5.3. Cooccurrence phenomena

Ideally, a text generation system should be able to handle all of the phenomena discussed above -- subcategorization, selectional restriction, collocation, and idiom -- in such a way that the different degrees of productivity and the different restrictions on these phenomena are distinguished. Moreover, the ideal system should have the flexibility to treat idioms and "fixed expressions" which are grammatical either productively (i.e., generate them according to general rules) or store them as units for the sake of efficiency, depending on the requirements of a given domain. Thus e.g. the phrase *We must conclude that...* can be stored as an idiom with a sense equivalent to "therefore", or generated "from first principles" as a clause with a first person plural subject, a modal of necessity etc. In such a system the tradeoff between productive capability and efficient processing could be avoided, much the way it presumably is in human language use.

---

[24] These are the nouns and adjectives that refer to or are predicated of reports of states of affairs; hence the term "factive" which is sometimes applied to them.

[25] There are also differences between systems in whether every inflected form must be listed for every inflectable word or phrase, or whether some cases are treated as predicatble.

[26] This is not an uncontroversial statement. [McDonald 80] and [-- 83] both argue that the fact that their systems are occasionally "at a loss for words" is a positive feature, since it accurately models the behaviour of the human language user.

### 5.4. Metaphor

A large range of phenomena which have been treated as idiosyncratic to individual lexical items -- i.e. as idioms or collocations -- could perhaps be treated in a more motivated way in a system which had a notion of standard metaphor. (This proposal is cogently stated in [--- 85]; the sense of metaphor involved here is that presented in e.g. [Lakoff & Johnson 80].) For example, consider the metaphor "time is money". In a system which had a way of representing this association, a number of collocations involving time -- "spend time", "waste time", "lose time" etc. -- are not random, but can be predicted from the collocations involving money. Another set of expressions involving time, e.g. "time passed", "time flies", "the days marched by in weary succession" etc., are derived from another standard metaphor for time, namely "time is a moving object". While some of Mel'chuk's lexical functions have to do with standard metaphors of this sort, as far as I know his is the only system that treats them systematically as such, although any system based on a taxonomic hierarchy with inheritance can simulate metaphor after a fashion. For example, there is a popular metaphor "a computer is a conscious being", which is involved when we refer to computers as agents of processes that normally only take conscious agents, e.g. "the computer deleted my files". In the Janus system, the only convenient way to represent this is by classifying the concept COMPUTER under CONSCIOUS BEING in the semantic taxonomy. Ideally, however, it would be preferable not to commit one's taxonomy to the *claim* that a computer is literally a conscious being, since we also talk about computers as unconscious objects; e.g. we usually say "the computer that just went down", not "the computer who just went down".

### 5.5. Choice

Ideally, a system should have some way of choosing between lexical items on other than purely grammatical and denotational grounds. Human speakers take a variety of factors into consideration when making lexical decisions. We use different words for the same things depending on who we're talking to, what we're talking about, where we are, and what role we're playing. A simple example is the observation that in more formal contexts English speakers tend to use Latinate words such as "expunge, remove, infer" instead of Anglo-Saxon phrasal verbs like "wipe out, take off, figure out". In addition to simply responding to social context in the way we choose words, we can use words in a way which evoke or create a context for our utterances; for instance, we can use borrowings from French in order to sound suave, or surfer slang in order to sound cool. We use more general or more specific terms for the same thing depending on which of its characteristics we're interested in: if we see a friend careening towards a tree, we're more likely to say "watch out for that tree!" than "watch out for that eucalyptus!" or "watch out for that plant!" And so on. We're a long way from having natural language generators that have the degree of control over any level of linguistic choice, grammatical or lexical, that a serious treatment of these considerations would entail; but we can design our systems such that such distinctions could be accomodated when we have the analyses to support them.

### 5.6. Conclusion

Lexicons play a wide varieties of roles in text generation systems, from the very central one of providing the primary link between form and meaning, to the quite peripheral one of finishing up after the grammar is done. Lexical phenomena such as semantic relationships, syntactic classes, collocation and idioms have received vastly different amounts of attention in different systems, while other phenomena such as metaphor and non-denotational meaning have received virtually none. Looking at the capabilities of a wide range of generation lexicons provides an exhilirating sense of the potential for future systems, both from the variety of phenomena that existing systems have dealt with, and from the challenges that still remain. I hope that bringing a few of these phenomena to light in this paper will succeed in sparking the interest necessary to ensure the lexicon the attention it warrants in text generation research.

## References


[Appelt 83] Douglas E. Appelt, "Telegram: a grammar formalism for language planning," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pp. 595-599, IJCAI, Aug 1983.

[Appelt 85a] Douglas E. Appelt, *Planning English Sentences*, Cambridge University Press, Cambridge, 1985.

[Appelt 85b] Douglas E. Appelt, "Planning English referring expressions," *Artificial Intelligence* 26, 1985, 1-33.

[Bates & Ingria 81] M. Bates, J. Beinashowitz, R. Ingria, & K. Wilson , "Controlled Transformational Sentence Generation," in *Proceedings of the 1981 Meeting of the Association for Computational Linguistics*, ACL, 1981.

[Bates & Wilson 81] Madeleine Bates and Kirk Wilson, *ILIAD: Interactive Language Instruction Assistance for the Deaf*, BBN, 10 Moulton St., Cambridge, MA 02138, Technical Report 4771, Sep 1981.

[Bates, Beinashowitz, Brown, Dougherty, Ingria, Shaked, Simpson & Wilson 81] M. Bates, J. Beinashowitz, D. Brown, D. Dougherty, R. Ingria, V. Shaked, W. Simpson, & K. Wilson, *ILIAD Database Reference*, BBN, 10 Moulton St., Cambridge, MA 02138, Supplement to Tech Report 4771, Sep 1981.

[Bates, Beinashowitz, Ingria & Wilson 81] M. Bates, J. Beinashowitz, R. Ingria, & K. Wilson , "Generative Tutorial Systems," in *Proceedings of the 1981 Meeting of the Association for the Development of Computer-Based Instructional Systems*, 1981.

[Becker 75] Becker, J.D., "The phrasal lexicon," in Schank & Webber (eds.), *Theoretical Issues in Natural Language Processing*, , Cambridge, 1975.

[Berry 77] M. Berry, *Introduction to Systemic Linguistics*, Batsford, London, 1977.

[Buchberger, Steinacker, Trappl, Trost & Leinfellner 82] Ernst Buchberger, Ingeborg Steinacker, Robert Trappl, Harald Trost, Elisabeth Leinfellner, "VIE-LANG: A German Language Understanding System," in *Cybernetics and Systems Research*, pp. 869-874, North-Holland, Amsterdam, 1982.

[Chafe 68] Wallace Chafe, "Idiomaticity as an anomaly in the Chomskyan paradigm," *Foundations of Language* 6, (1), 1968.

[Cumming 86] Susanna Cumming, *Design of a Master Lexicon*, USC/Information Sciences Institute, Technical Report ISI/RR-85-163, Feb 1986.

[Cumming & Albano 86] Susanna Cumming and Robert Albano, *A guide to lexical acquisition in the JANUS system*, USC/Information Sciences Institute, Technical Report ISI/RR-85-162, Feb 1986.

[Danlos 84] Laurence Danlos, "Conceptual and linguistic decisions in generation," in *Proceedings of Coling84*, pp. 501-504, COLING, July 1984.

[Danlos 85] Laurence Danlos, *Generation automatique de textes en langues naturelles*, Masson, Paris, 1985.

[Davey 78] Anthony Davey, *Discourse Production*, Edinburgh University Press, Edinburgh, 1978.

[Derr & McKeown 84] Marcia A. Derr and Kathleen R. McKeown, "Using fucus to generate complex and simple sentences," in *Proceedings of Coling84*, pp. 319-326, COLING, July 1984.

[Fillmore 79] Charles Fillmore, "Innocence: a second idealization for linguistics," in *Proceedings of the 5th Annual Meeting of the Berkeley Linguistics Society*, BLS, 1979.

[Fillmore, Kay & O'Conner 84] Charles Fillmore, Paul Kay & M.C. O'Conner, *Regularity and idiomaticity in grammar: the case of let alone*, University of California, Coginitive Science Working Paper, 1984.

[Firth 57] J.R. Firth, *Modes of Meaning*, Oxford University Press, Oxford, , 1957.

[Gross 84] Maurice Gross, "Lexicon-gr[illegible]mar and the syntactic analysis of French," in *Proceedings of Coling84*, pp. 275-282, COLING, Jul 1984.

[Halliday 61] M.A.K. Halliday, "Categories of the Theory of Grammar," *Word* 17, 1961.

[Halliday 76] Halliday, M.A.K., "Lexical Relations," in G.R. Kress (ed.), *Halliday: system and function in language*, Oxford University Press, London, 1976.

[Halliday, McIntosh, & Strevens 64] M.A.K. Halliday, Angus McIntosh, & Peter Strevens, *The Linguistic Sciences and Language Teaching*, Indiana University Press, Bloomington, 1964.

[Hudson 84] Richard Hudson, *Word Grammar*, Blackwell, Oxford, 1984.

[Jacobs 83] Paul S. Jacobs, "Generation in a natural language interface," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pp. 610-612, IJCAI, Aug 1983.

[Kay 79] Martin Kay, "Functional Grammar," in *Proceedings of the 5th Annual Meeting of the Berkeley Linguistics Society*, pp. 142-158, BLS, 1979.

[Kittredge & Mel'chuk 83] Richard Kittredge & Igor Mel'chuk, "Towards a computable model of meaning-text relations within a natural sublanguage," in , pp. 657-659, IJCAI, 1983.

[Kukich 83] Karen Kukich, "Design of a knowledge-based report generator," in *Proceedings of the 21st Annual Meeting*, ACL, Jun 1983.

[--- 83] Karen Kukich, *Knowledge-based report generation*, Ph.D. thesis, University of Pittsburgh, Interdisciplinary Department of Information Science, Aug 1983.

[Lakoff & Johnson 80] George Lakoff & David Johnson, *Metaphors We Live By*, University of Chicago Press, 1980.

[Longman 79] Longman Group Ltd., *Longman Dictionary of English Idioms*, Longman, Harlow and London, 1979.

[Makkai 72] Adam Makkai, *Idiom Structure in English*, Mouton, The Hague, 1972.

[Mann & Matthiessen 83] William C. Mann & Christian M.I.M. Matthiessen, *Nigel: A Systemic Grammar for Text Generation*, USC/Information Sciences Institute, Technical Report ISI/RR-83-105, Feb 1983.

[Matthiessen 84] Christian M.I.M. Matthiessen, *Systemic Grammar in Computation: the Nigel case*, USC/Information Sciences Institute, Technical Report ISI/RR-83-121, Feb 1984.

[McDonald 80] David D. McDonald, *Natural language productions as a process of decision-making under constraints*, Ph.D. thesis, Massachusetts Institute of Technology, Aug 1980.

[McDonald 83] David D. McDonald, "Natural language generation as a computational problem: an introduction," in Brady & Berwick (eds.), *Computational Problems in Discourse*, MIT Press, Cambridge, 1983.

[McDonald 85] David D. McDonald, "Description-directed natural language generation," in *Proceedings of the Ninth International Joint Conference on Artificial Intelligence, IJCAI, 1985.*

[McKeown 83] Kathleen R. McKeown, "Focus constraints on language generation," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pp. 582-586, IJCAI, 1983.

[McKeown 85] Kathleen R. McKeown, *Text generation: using discourse strategies and focus constraints to generate natural language text*, Cambridge University Press, Cambridge, 1985.

[Mel'chuk 81] Igor Mel'chuk, "Meaning-text models: a recent trend in Soviet linguistics," *Annual Review of Anthropology* 10, 1981, 27-62.

[Mel'chuk & Zholkovsky 84] Igor Mel'chuk & Alexander K. Zholkovsky, *Explanatory Combinatorial Dictionary of Modern Russian*, Wiener Slawistischer Almanach, Vienna, 1984.

[Mel'chuk et al. 83] Igor Mel'chuk, Lidija Iordanskaja, Nadia Arbatchewsky-Jumarie, and Adele Lessard, "Trois principes de description semantique d'une unite lexicale dans un dictionnaire explicatif et combinatoire," *Canadian Journal of Linguistics* 28, (2), 1983, 105-121.

[Patten 86] Terry Patten, *Interpreting Systemic Grammar as a Computational Representation: a problem solving approach to text generation*, Ph.D. thesis, University of Edinburgh, 1986.

[--- 85] Paul S. Jacobs, "PHRED: a generator for natural language interfaces," *ACL* 11, (4), 1985, 219-242.

[Steinacker & Buchberger 83] Ingeborg Steinacker & Ernst Buchberger, "Relating Syntax and Semantics: the syntactico-semantic lexicon of the system VIE-LANG," in *Proceedings of the First Conference of the European Chapter*, pp. 96-100, ACL, Sep 1983.

[Steinacker & Trost 83] Ingeborg Steinacker & Harald Trost, "Structural relations -- a case against case," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pp. 627-629, IJCAI, Aug 1983.

# Assertions from Discourse Structure

William C. Mann
USC Information Sciences Institute

and

Sandra A. Thompson
UCLA Linguistics Department

## Introduction

As part of an ongoing study of discourse structure of natural texts, we have identified a particular class of propositions that affect the hearer's perception of the coherence and communicated content of texts. As an example, if the text (spoken in a suitable situation) is:

"I'm hungry. Let's go to the Fuji Gardens."

then the most natural interpretation is that the Fuji Gardens is a restaurant at which the speaker would like to eat with the hearer. The text is heard as exhibiting a problem-and-solution structure. Consequently, we can say that there is a proposition which says that there is a "solutionhood" relation between the two sentences. In this case, going to the Fuji Gardens (partially) solves the hunger problem.

The solutionhood construct is one type of *relational propositions*. Note that the proposition about solutionhood is not stated explicitly in the text.

Although phenomena resembling relational propositions have been recognized, there is no widely accepted explanation of how they arise from text. This paper characterizes relational propositions and presents a theory of discourse structure to explain them. In this Rhetorical Structure Theory (RST), relational propositions arise in direct correspondence to particular elements of the structure of a discourse.

We present Rhetorical Structure Theory progressively during analysis of a published, two-paragraph political advocacy text. The text involves substructures for informing, giving evidence, conceding, requesting an action, justifying a presentation, asserting conditionally, and others.

The two elements that form the basis for this paper, relational propositions and Rhetorical Structure Theory, have both been described in more detail elsewhere. The explanatory relation between them, however, has not [Mann & Thompson 83, Mann 84].

[Mann & Thompson 83] examines various other theoretical constructs, including implicature, presupposition and indirect speech acts, to see whether they account for the textual properties of relational propositions. It concludes that these constructs do not account for them. The paper also discusses relationships between relational propositions and the work of Grimes, Hobbs, van Dijk, Martin, Longacre, Beekman and Callow, and it includes analyses of several texts.

# 1 The Phenomenon of Relational Propositions

The *Fuji Gardens* statement and the *solutionhood* relation have already illustrated that relational propositions need not be signalled explicitly in order to be recognized.[1] Extending the example, we now describe properties relational propositions hold more generally, giving special attention to those properties that can be accounted for by Rhetorical Structure Theory.

## 1.1 Relational Propositions Assert

In our informal presentations of relational propositions, virtually everyone recognizes that texts such as the political letter analyzed in this paper convey the particular relational propositions that we attribute to them, even though it does not represent them explicitly. If the text author were to deny a particular relational proposition, most readers would be surprised--and puzzled about the status of the part of the text containing that proposition.

This general consensus testifies that the relational proposition is conveyed. Further evidence lies in the apparent redundancy or somewhat bizarre markedness that occurs when the relational proposition is asserted explicitly by adding a clause to the text:

> "I'm hungry. Let s go to Fuji Gardens. Our going to Fuji Gardens would contribute significantly to solving the problem of my hunger."

## 1.2 Relational Propositions are Coherence Producing

One way to demonstrate that a relational proposition is coherence-producing is to insert a denial of the relational proposition into the text. Doing so makes some portion of the text a non-sequitur:

> "I'm hungry. Let's go to the Fuji Gardens. Of course, going to the Fuji Gardens won't do anything about my hunger."

---

[1] We use constructed examples in this section only, for expository reasons. RST is a theory of natural text, it was developed entirely on natural texts, such as the political advocacy text analyzed below.

The second sentence above has become a non-sequitur, and as a result the text as a whole is incoherent. Relational propositions are *always* coherence-producing in this way. We will see later that this is a consequence predictable from RST, particularly from the structural forms that RST posits. Also, relational propositions are *always* present in coherent multisentence texts.

### 1.3 Other Kinds of Relational Propositions

The list below names several kinds of relational propositions besides solutionhood, and gives an example of an asserted, coherence-producing proposition for each. These are drawn form the larger collection of [Mann & Thompson 83]; we believe that still more kinds of relational propositions could be discovered or perhaps created.[2]

EVIDENCE: They're having a party again next door. I couldn't find a parking space.

ELABORATION: I love to collect classic automobiles. My favorite car is my 1899 Duryea.

MOTIVATION: Take Bufferin. The buffering component prevents excess stomach acid.

THESIS/ANTITHESIS: Players want the referee to balance a bad call benefiting one team with a bad call benefiting the other. As a referee, I just want to call each play as I see it.

CONCESSION: I know you have great credentials. I'm looking for someone with great experience.

CONDITION: Give her a subscription to *Science* magazine. She'll be in seventh heaven.

REASON: I'm going to the corner. We're all out of milk.

JUSTIFICATION: Let me make one thing perfectly clear. I am not a crook.

We desire a theory that will answer two questions about relational propositions:

1. What relational propositions are possible?

---

[2] We have abundant natural correlates for these constructed examples. They have been designed to illustrate the fact that the relations and relational propositions are identifiable without any explicit signalling, such as a clause, conjunction, or lexical selection.

2. What relational propositions does a particular text assert?

The answers come from studying discourse.

## 2 Rhetorical Structure Theory

RST has not been developed simply to account for relational propositions; it arose from a much broader desire to understand text and communication and to learn how texts may be created. We identified and began studying relational propositions only after RST had largely assumed its present shape.

We wanted a theory of text organization--a way to describe what kinds of parts a text can have, how they can be arranged, and how parts can be connected to form a whole text. We especially valued the following attributes.

1. **Comprehensiveness**: The theory should apply to many kinds of text.

2. **Functionality**: The theory should be informative in terms of how text achieves its effects for the writer.

3. **Scale insensitivity**: The theory should apply to a wide range of sizes of text and should be capable of describing all of the various sized units of text organization that occur in a large text.

4. **Definiteness**: The theory should lend itself to formalization and computer programming;

5. **Constructive potential**: The theory should be usable in text construction as well as text description.

We developed this theory primarily in response to small written texts, although it has also been applied to larger texts. We have constructed RST descriptions for a variety of texts, including:

- Administrative memos
- Personal letters
- Advertisements
- Editorial letters in magazines
- Complete *Scientific American* articles
- Newspaper articles
- Public notices in magazines
- Research technical reports
- Travel brochures
- Cookbook recipes

To introduce the theory, let us consider the analysis of a text that appeared in a political newsletter, The Insider, Vol. 2.1, July 1982.[3] The Insider is the California Common Cause state newsletter. This text was the "con" part of a "pro" and "con" pair of letters on the issue of California Common Cause's endorsement of the nuclear freeze initiative, which was then on the California state ballot.

The text has been reformatted, numbered, and divided the text into units. Units are roughly equivalent to clauses, except that that relative clauses and complement clauses are considered to be part of the unit in which their governing item appears, rather than as independent units. As long as the whole text is analyzed, the size of the minimal units can vary without affecting the larger analysis.

1. I don't believe that endorsing the Nuclear Freeze Initiative is the right step for California CC.

2. Tempting as it may be,

3. we shouldn't embrace every popular issue that comes along.

4. When we do so

5. we use precious, limited resources where other players with superior resources are already doing an adequate job.

6. Rather, I think we will be stronger and more effective

7. if we stick to those issues of governmental structure and process, broadly defined, that have formed the core of our agenda for years.

8. Open government, campaign finance reform, and fighting the influence of special interests and big money, these are our kinds of issues.

---

[3] Letter used by permission

9. (New paragraph) Let's be clear:

10. I personally favor the initiative and ardently support disarmament negotiations to reduce the risk of war.

11. But I don't think endorsing a specific nuclear freeze proposal is appropriate for CCC.

12. We should limit our involvement in defense and weaponry to matters of process, such as exposing the weapons industry's influence on the political process.

13. Therefore, I urge you to vote against a CCC endorsement of the nuclear freeze initiative.

(signed) Michael Asimow, California Common Cause Vice-Chair and UCLA Law Professor

How is this text organized? At the most general level, the text as a whole functions as a request to vote in a certain way. At its coarsest level of decomposition, it has two parts. One part presents the request, presented in segment 13, and the remainder supports that presentation.

The theory has a number of patterns, called *rhetorical schemas*, that represent organizational information about text. To represent the particulars of two-part decomposition of the text, we use one of these rhetorical schemas, the Request Schema, Figure 1.

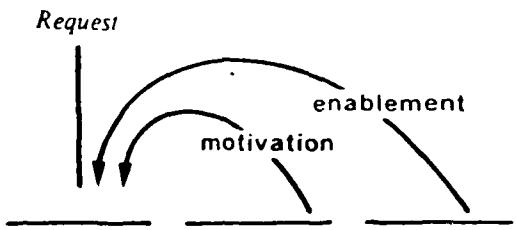



Figure 1: Request Schema

A text that instantiates the Request Schema has a nuclear part, called the *nucleus*, that presents a request. It also has one or more supplementary parts, called *satellites*, that are functionally related to the nucleus. Satellites are related to the nucleus by a named *relation*. Here we have relations named *motivation* and *enablement*.

Let us illustrate the parts of a Request Schema in a short example:

"Call me. I have a surprise for you. My extension number is 110."

The nucleus is "Call me," the motivation satellite is "I have a surprise for you," and the enablement satellite is "My extension number is 110." These elements can be arranged in any order and still be an instance of the Request Schema. Schemas do not encode the order of segments; in this case, the segments can be rearranged freely without disturbing their meaning or structural relation.

Satellites are all optional, so we can delete either one in this example and still instantiate the Request Schema--but there must be at least one satellite. The political text has a motivation satellite, segments 1 through 12, but no enablement satellite.

We analyze each of the two top-level segments of the political text in the same way. The final segment is a single unit, so we don't try to divide it. The first segment, 12 units long, consists of a claim (unit 1) and two arguments that give evidence for the claim. We analyze this arrangement with the Evidence Schema (Figure 2), in which the claim is the nucleus and an *evidence* relation connects the nucleus to the satellite. Figure 3 shows the resulting structure.

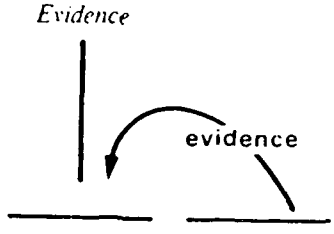



**Figure 2:** Evidence Schema

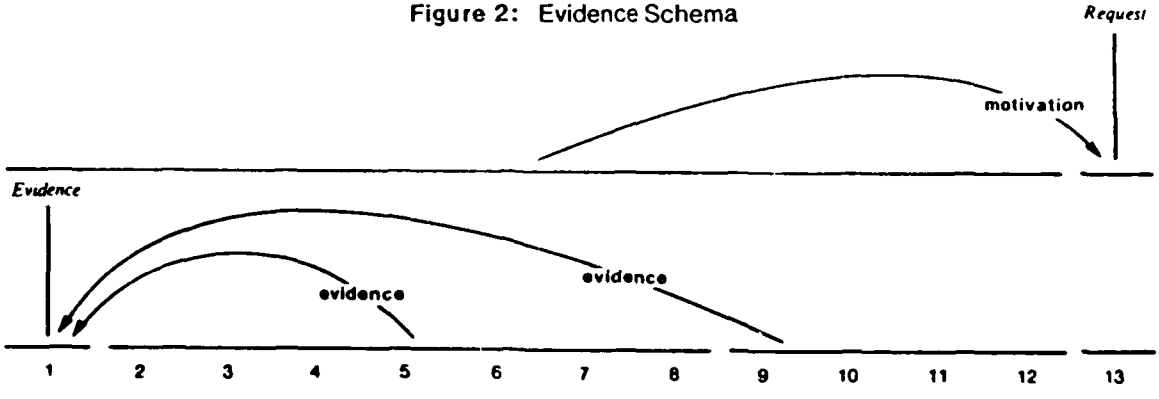



**Figure 3:** The Upper Structure of the Political Text

Both of the nuclei obey what we call the *Most Favorable Audience Rule*: For the most knowledgeable and positively predisposed hearer, the nucleus alone would be sufficient to perform the function of the structure; the satellites function to increase the likelihood that the nucleus will succeed. This rule is a summary of many observations about the rhetorical structures of texts. It does not always hold, but there is a strong, unexplained tendency for it to hold.

In this case of the Request Schema, presenting the request (to vote in a certain way) to a favorably predisposed hearer would be enough to get that reader to vote as desired; the supporting argument makes the desired vote more likely for most readers.

This application of the Evidence Schema contains two arguments: One says that the proposal is wasteful, and the other says that better alternatives exist. They make the reader more likely to accept the claim that endorsement is not right.[4]

The analysis goes on, down to single units. Figure 4 shows the additional schemas used. They are drawn from a larger set of about 25 schemas, defined through use of about 30 relations.

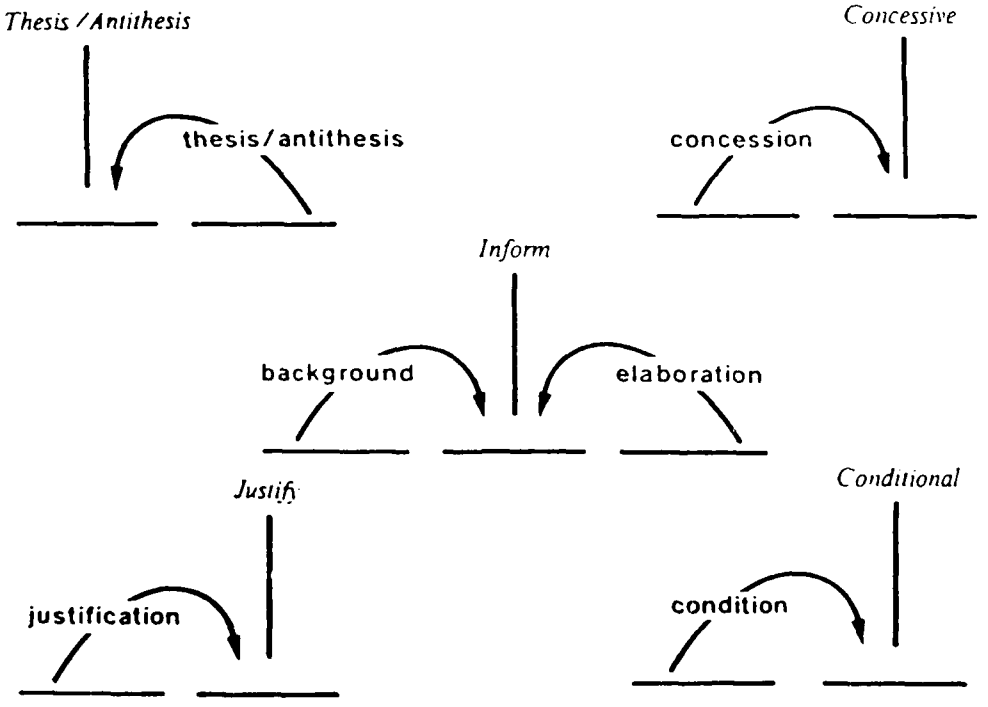



**Figure 4:** Other Schemas Used to Analyze the Political Text

[4] Although the unit begins "I don't believe that...," the claim here is really about whether this step is right for CCC. The evidence that follows in units 2 through 11 is about what benefits CCC, not about whether the author believes this claim or not. RST does not represent the indirectness of the form of the claim.

To illustrate the relations used here, we turn back to the text.

The *thesis/antithesis* relation connects units 11 and 12. 11: "But I don't think endorsing a specific nuclear freeze proposal is appropriate for CCC." 12: "We should limit our involvement..."

The *concessive* relation connects units 2 and 3: 2: "Tempting as it may be," 3: "we shouldn't embrace every popular issue that comes along."

Unit 8 is an *elaboration* for the Inform Schema; it lists instances, such as open government and campaign finance reform.

Unit 9 says "Let's be clear." This is in a *justification* relation to the argument that follows, in 10 through 12. It obtains permission to present a second argument, defending the same conclusion.

Finally, units 4 and 5 are in a *condition* relation 4 "When we do so " 5: "we use precious resources..."

Figure 5 shows the structure of the whole text.

## 2.1 Definition Mechanisms of RST

How is RST defined? How do claims about particular texts arise from an RST analysis of it? The theory is defined in terms of just three mechanisms: schemas, schema application conventions, and relation definitions.

Schemas are simply sets of relations. There is no schema-specific information beyond the identities of the relations that comprise the schema.

Schema application conventions are descriptions of what it means for a particular span of text to instantiate a schema. Its conventions are easy to state:

1. One schema is instantiated to describe the entire text.
2. Schemas are instantiated to describe the text spans produced in instantiating other schemas.
3. The schemas do not constrain the order of nucleus or satellites in the text span in which the schema is instantiated.
4. All satellites are optional.
5. At least one satellite must exist.
6. A relation that is part of a schema may be instantiated indefinitely many times in the instantiation of that schema.

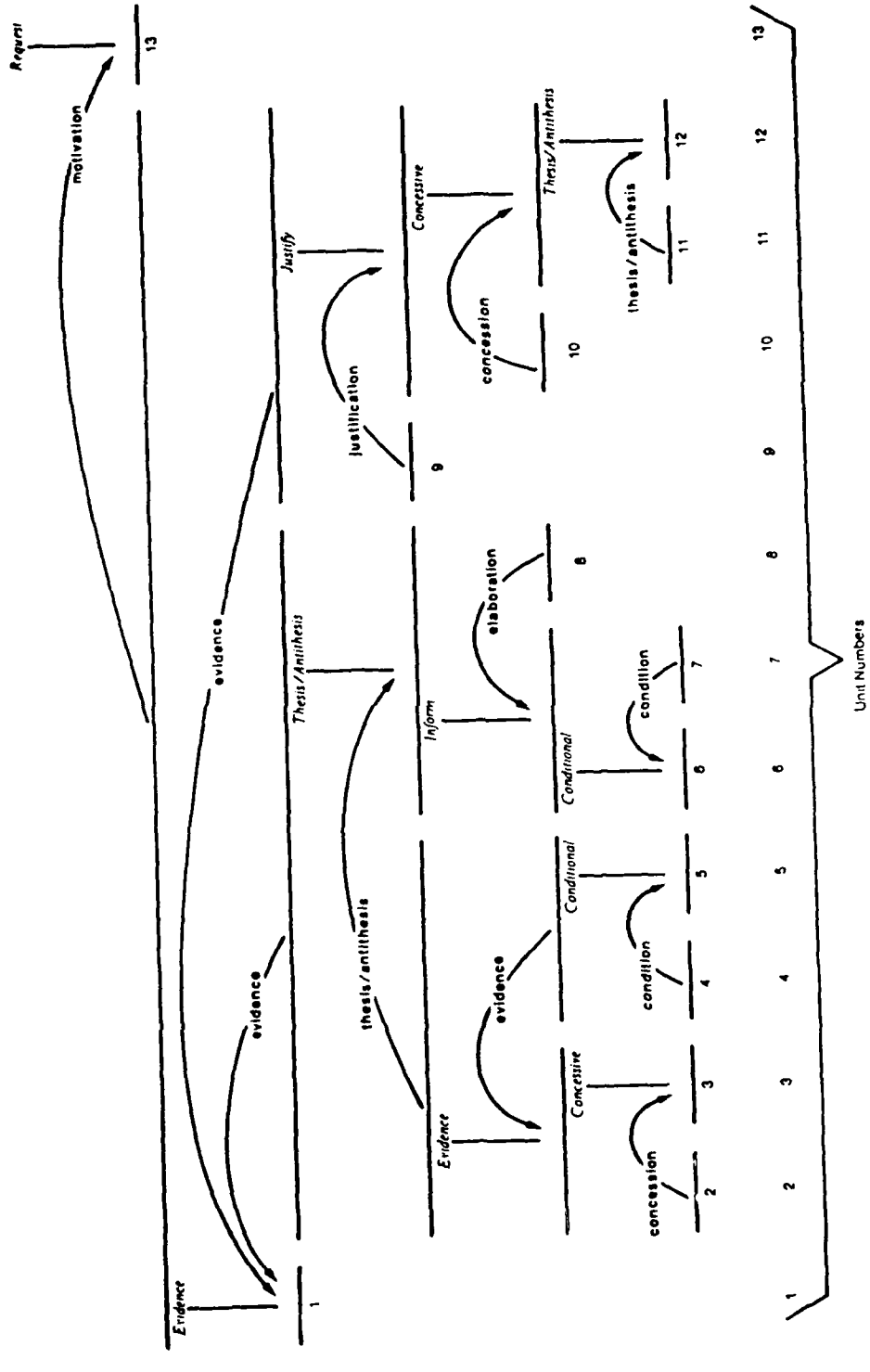



**Figure 5: Full Rhetorical Structure of the Political Text**

7. The nucleus and satellites do not necessarily correspond to a single uninterrupted text span.

8. The relation definition must be consistent with the spans of text related by the instantiation of the schema containing the relation.

It is possible for the conventions to apply to a text in more than one way, so that the text is rhetorically ambiguous.

A relation definition specifies three kinds of information:

1. A characterization of the nucleus.
2. A characterization of the satellite.
3. A characterization of what sorts of interactions between the conceptual span of the nucleus and the conceptual span of the satellite must be plausible.

To define, for example, the motivation relation, we would include at least the following.

1. The nucleus describes an action performable, but not yet performed, by the reader.
2. The satellite describes the action, the situation in which the action takes place, or the result of the action in ways that help the reader associate value assessments with the action.
3. The expected value assessments are positive, to lead the reader to want to perform the action.

The relational propositions do not arise independently of the relation definitions. Rather, finding that a relation definition holds is sufficient to establish the corresponding relational proposition. As readers recognize the functional relations of the parts, they are recognizing that the relation definitions hold. The content of the relational proposition is identified in this process. As a consequence, the definition scheme for RST requires no additional definitions in order to specify the relational proposition. In any particular case, the proposition can be derived from the way the relation definition fits.

We have found the relation definitions, useful in predicting other facts about the text, such as the kinds of conjunctions that will appear in certain places. We have analyzed a large number of texts, including thousands of clauses, in this way. We are confident that we can perform this analysis, with fairly high reliability, for virtually any small, written, multisentence monologue in general American culture, using only about 25 schemas.

Note that these rhetorical schemas are defined in terms of the functions of segments of text. The evidence relation applies when one segment supports another as evidence. Solutionhood applies when we can see one segment as identifying a

problem and another as a partial solution to that problem. These are not criteria of form; as one might expect, the relationship of these function categories to form categories is quite loose. The rhetorical structure of text, in these terms, is composed of function-specific units. The structure does not express categories of knowledge or form as much as it expresses the roles of specific parts in relation to the whole text, especially the role of each satellite relative to one particular, immediate portion of the text, the corresponding nucleus.

## 3 RST as an Account of Relational Propositions

The key observation for the purposes of this paper is:

***For every relation of the rhetorical structure of a text, a corresponding relational proposition is asserted.***

For solutionhood relations, the discourse structure asserts a solutionhood proposition. For evidence relations it asserts an evidence proposition, and so forth. Readers attribute the assertions to the text because they recognize the functional relations of the parts.

Now we can explain why relational propositions are coherence-producing.

First, RST structures always have the connectivity of trees. The schema application conventions guarantee this, because when a span is decomposed, each of the parts is further decomposed separately.

If a portion of the text is to be connected to the whole without being a non-sequitur, some relation must be established. A relation is established through implicit assertion of a relational proposition. Since the relations form a tree, denial of any one relational proposition is sufficient to separate the structure into two parts, thus destroying connectedness, a key attribute of coherence.

Now we can also explain why relational propositions are always present in coherent multisentence texts. In regarding the text as a single whole, readers impute rhetorical structure to it, necessarily positing relations between the parts; the relations give rise to assertion of relational propositions.

We can also see how to create more precise specifications of relational propositions. They can be developed from the relation definitions of RST. RST tells what sorts of propositions can be relational, gives the conditions under which relational propositions arise, and tells how to alter a text or a situation so that the asserted relational proposition is changed. Rather than simply searching texts for potential relational propositions, we can search rhetorical structures for the necessary relational propositions.

## 4 Uses of Rhetorical Structure Theory

Rhetorical Structure Theory provides an attractive basis for explaining relational propositions, although some details need development.

In addition, RST satisfies some of the attributes identified in section 2, above, as desirable for a descriptive theory of text organization. It is comprehensive enough to apply to many different kinds of text; it is functional, in that it explains what various portions of a text do for the writer. And it is scale insensitive, applying to a wide range of sizes of units, from simple clauses up to whole magazine articles.

However, RST still lacks two desired attributes: It needs for more detailed expression of each part, and it would be useful to develop a constructive counterpart to the descriptions, a way to select schemas and plan texts.

In addition to these attributes, we recognize other opportunities for and benefits of RST.

1. It gives a partial account of the distribution of interclausal and intersentential conjunctions.

2. It leads to new observations of text phenomena, including some related to nuclei, such as the most-favorable-reader hypothesis.

3. The advantages of a recursive theory are obtained for text structure.

Beyond the phenomena identified above, RST appears to be useful in accounting for other kinds of discourse phenomena. We have found no boundary for its uses; it is like trying to delimit the uses of a grammar. We have identified the following as particularly attractive applications:

- Thematization and text development
- Distributions of tense selections in text
- Lexical selection
- Patterned shifts of hypotheticality, identifiability, or conditionality
- Patterns of use of conjunctions
- Purposeful clause combining
- Distribution of topicalization markers
- Text ordering (under way)
- Relating coherence to cohesive devices

## 5 Summary

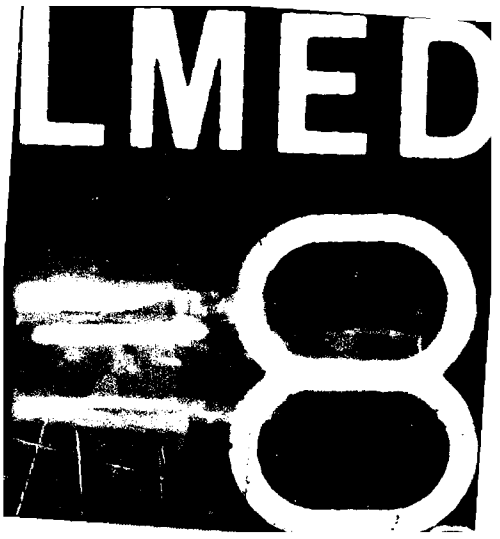
The assertion of relational propositions is a hitherto unexplained phenomenon. Rhetorical Structure Theory provides a way to explain such assertions in terms of discourse structure. In addition to explaining relational propositions,

Rhetorical Structure Theory can be used to explain other text characteristics as well, and it provides a way to address a wide range of discourse phenomena.

## References


[Mann 84] Mann, W. C., *Discourse Structures for Text Generation*, USC/Information Sciences Institute, Technical Report RR-84-127, February 1984. Also appeared in the proceedings of the 1984 Coling/ACL conference, July 1984.

[Mann & Thompson 83] Mann, W. C., and S. A. Thompson, *Relational Propositions in Discourse*, USC/Information Sciences Institute, Technical Report RR-83-115, July 1983. To appear in *Discourse Processes*.


LMED
8