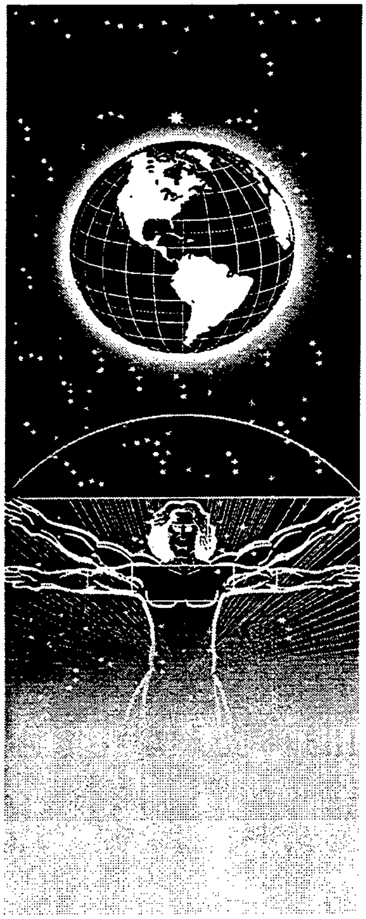
AFRL-HE-WP-TP-1998-0006



UNITED STATES AIR FORCE RESEARCH LABORATORY

Automating Maintenance Instructions Study

Norm Badler
Bonnie Webber
Martha Palmer
Juliet Bourne
Sonu Chopra
Matthew Stone
Hoa Trang Dang
Tonia Bleam

University of Pennsylvania
Philadelphia PA 19104-6389

19980805 014

February 1998

Final Report for the Period June 1997 to January 1998

Approved for public release; distribution is unlimited.

Human Effectiveness Directorate
Crew Survivability and Logistics Division
Sustainment Logistics Branch
2610 Seventh Street
Wright-Patterson AFB OH 45433-7901

NOTICES

DISCLAIMER

This Technical Paper is published as received and has not been edited by the Air Force Research Laboratory, Human Effectiveness Directorate.

TECHNICAL REVIEW AND APPROVAL

AFRL-HE-WP-TP-1998-0006

This paper has been reviewed by the Office of Public Affairs (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical paper has been reviewed and is approved for publication.

FOR THE COMMANDER

For THOMAS J. MOORE, Chief
Crew Survivability and Logistics Division
Air Force Research Laboratory

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
February 1998

3. REPORT TYPE AND DATES COVERED
Final - June 1997 - January 1998

4. TITLE AND SUBTITLE
Automating Maintenance Instructions Study

5. FUNDING NUMBERS
C - F41624-97-D-5002
PE - 62205F
PR - 1710
TA - D0
WU - 04

6. AUTHOR(S)
Norm Badler, Bonnie Webber, Martha Palmer, Juliet Bourne, Sonu Chopra, Matthew Stone, Hoa Trang Dang, Tonia Bleam

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Pennyslvania
Philadelphia PA 19104-6389

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory, Human Effectiveness Directorate
Crew Survivability and Logistics Division
Air Force Materiel Command
Sustainment Logistics Branch
Wright-Patterson AFB OH 45433-7604

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-HE-WP-TP-1998-0006

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
This report describes some methods underlying automation of maintenance instructions of the sort found in maintenance technical orders. The key problem under review here is the interface between the geometry of the device and the verbal description of the maintenance actions required for the human maintainer. Language generation underlies both the simulation of a maintenance procedure and its documentation in a technical order (or instruction). Coordination between language generation and task simulation is provided by a method called parameterized action representation. Examples of computational approaches to this automation approach are provided, and recommendations for research to extend and refine the techniques are provided.

14. SUBJECT TERMS
computational linguistics, natural language generation, human form modeling, maintenance technical orders, computer-aided design

15. NUMBER OF PAGES
48

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Standard Form 298 (Rev. 2-89) (EG)
Prescribed by ANSI Std. 239.18
Designed using Perform Pro, WHS/DIOR, Oct 94

THIS PAGE INTENTIONALLY LEFT BLANK

Preface

The work on the Automating Maintenance Instructions Study was conducted under Delivery Order 8 of the Logistics Technology Research Support (LTRS) program administered under the U.S. Air Force Contract Number F41624-97-5002.

The authors wish to thank Ed Boyle of the Air Force Research Laboratory and Stephen Deutsch of BBN technologies for contract organization and management.

Summary

This study of Automating Maintenance Instructions (AMI) focuses on the interface between the geometry of the device and the verbal description of the maintenance actions required for the human maintainer (currently Technical Orders). The interface issues are discussed in the context of requirements for geometric models and for the language (text) generation needed to accurately describe these maintenance actions. This report is organized into six main sections, two case studies, recommendations, a glossary of terms, and references. First we discuss the implications of object geometry on maintenance modeling and argue for the consideration of human task activities as an essential component of maintenance procedure planning and instructions. Then we introduce the language generation issues, including distinctions between state-space, kinematic, dynamic, and process control terms. We describe the lexical semantics that is necessary for the generation of precise and accurate verbal instructions. Since instructions will be executed sequentially, an important element of the instruction is specific information with respect to its completion or culmination, and culminating conditions are discussed in detail. The actual text of an instruction is created through processes of text generation and planning. The method by which the same planning process can be extended to include the consideration of a visual presentation of information as well is discussed, and the careful coordination that this would require. We then present the case studies involving a task where the presence of the human maintainer fixes a task ordering that is not determined solely from the geometry data. The animation study addresses collision detection and access requirements over the geometry. The language study looks at the same example from the sentence generation perspective and focuses on lexical choice and precise object description. Finally, we summarize our AMI recommendations.

Table of Contents

List of Figures

1 Introduction

"Here is the machine, isolated in time and in space from everything else in the universe. It has no relationship to you, you have no relationship to it..." [1]

The life-cycle of a complex machine such as an aircraft has a long maintenance "tail" where it must be operated and serviced by personnel other than the original manufacturing team. The job of the technical order author is to recast the "as built" design into discrete repair and replacement tasks that are undertaken periodically or as required by design changes, mission requirements, or equipment failure. The manuals that describe such activities are currently written by authors with some technical knowledge of the systems, possibly former maintainers themselves, but the task is both tedious and subtle. The instructions must be precise enough to be executed, easy to understand at the required maintainer education level, and ordered correctly to insure safety to crew and equipment.

Robert Pirsig's observation above arises in his criticism of technical maintenance manuals. The matter is relevant here. If we consider only the machine itself – its form and position – we will fail to take into account the fact that *maintenance is done by people.* The relationships between the machine and its operation and maintenance are not purely abstract; rather, we must understand how people effect the actions and how the system itself operates and responds to those actions. Accordingly, this study of Automating Maintenance Instructions (AMI) focuses on the interface between the geometry of the device and the human maintainer. The interface issues are discussed here in the context of requirements for geometric models and for the language (text) generation needed to accurately describe maintenance actions.

This report is organized into six main sections, two case studies, recommendations, a glossary of terms, and references. Section 2 discusses the implications of object geometry on maintenance modeling and argues for the consideration of human task activities as an essential component of maintenance procedure planning and instructions. Section 3 introduces the language generation issues, including distinctions between state-space, kinematic, dynamic, and process control terms. Section 4 introduces lexical semantics: how verbs and their linguistic context carry information about tasks and their execution. Since instructions are executed sequentially, it is necessary to understand or infer when an instruction is completed. The actual text of an instruction is created through processes of text generation and planning in Section 5. A system that combines visual and verbal outputs is discussed in Section 6. The case studies involve a task where the presence of the human maintainer fixes a task ordering that is not determinate solely from the geometry data. The animation study, Section 7, addresses collision detection and access requirements over the geometry. The language study, Section 8, looks at the same example from the sentence generation perspective. Finally, Section 9 summarizes our AMI recommendations.

2 CAD Models and Human Figures

The goal of this project is to assess technological approaches to the rapid production of correct procedure descriptions that an agent must follow to perform a specific maintenance task. Given the sorts of tasks an aircraft maintainer must perform, any computational system that aides in developing instructions must have an appropriate level of understanding of the affected systems. Moreover, as aircraft systems are serviced through the manual efforts of the maintainer, an understanding of the human presence and its limitations must also be factored into the instruction set. We address these two issues in this Section: extending geometric CAD models with the information necessary to manipulate them and developing an action representation compatible with and supportive of human modeling technologies.

2.1 The Object Removal Task

We will start with a brief review of one formalization of the object removal task in the Automated Maintenance Manual Production system (AMMP) as described by Hoffman, Keshavan, and Lankford[2]. Their system treats the problem essentially as a geometric search problem. (Interestingly enough, this framework is very similar to the generic case-based planning model[3].) In AMMP the disassembly of a set of components is based on known or computed motion freedoms. If the feasible motions (translations and rotations) are not known or given, they may have to be computed by geometry and/or functional agents. The nature of these agents (some heuristic, some mathematical) is not crucial here, but the results of their calculations are movement ``facts'' that can be used by the spatial planner.

Given that the movement freedom can be ascertained, part moves can be characterized through collision, escape, line up, and step aside actions. These ``steps'' are then used to compute candidate disassembly sequences. In addition to these formal moves, properties of the emergent subassemblies are computed to assess feasibility, including part stability and tool reachability. The stability checks the center of mass of a component against the convex hull of its extent. This appears to be a weak heuristic to check that the maintainer would not experience a sudden torque once the component is freed. This heuristic does not take into account the actual mass distribution of the assembly parts nor the actual maintainer strength (and reaction) capability. AMMP does a reachability check, but does not appear to utilize a human model, though tool access is considered – provided one supplies the tool.

In order to judge alternative disassembly solutions, AMMP uses a variety of ``ordering relations'' such as piece *count, remoteness, moved count*, and *cost.* These relations are used by the heuristic search planner with the goal of providing a minimum step procedure relative to the given cost functions.

AMMP is notable in that it attempts to work directly from the CAD data without operator notations. There are assumptions made about motion degrees of freedom, fasteners (as component contacts), object rigidity, and mass distribution for stability that may be more heuristic than actual. The essence of the planning process, however, could form the basis of a much more robust system if additional knowledge were encoded. Of course, selected operator (or ``intelligent software agent'') interventions—such as tagging fasteners—would also help speed up the planning process without burdening the TO author with all the mental spatial manipulations which AMMP can perform.

Although AMMP can reason over tools and their accessibility (and presumably, any necessary range of motion), it does not appear to consider human access. While a hand could presumably be defined as a tool, the flexibility of the hand in reaching confined spaces and then conforming to the grasped object geometry seems a capability well beyond the AMMP spatial planner. Moreover, AMMP does not attempt any deeper understanding of the functionality of the system it is disassembling, and therefore can fail to order steps that depend on such understanding, for example, to close a value on a pressurized line before removing a downstream component. The heuristic geometric algorithms in AMMP would nonetheless appear to be a good foundation for the automated geometric reasoning component of procedure step generation.

2.2 The Maintenance Problem from the CAD Perspective

For the AMI project more information is needed than just geometric shape of objects and the part or assembly hierarchy. We examine what sorts of information might be needed and how it could be specified or obtained semi-automatically. We will show how this added information is connected with the actions or processes involved in maintenance activities.

Component geometry data at one or more levels of geometric detail is usually provided by the manufacturer. A detailed level would encompass information needed for fabrication, engineering analysis, and so on. A coarser level of detail might suffice for a representative view drawing or maintenance access planning. The geometry itself may be represented as collections of planar polygons or curved surface patches (Boundary Representation) or as combinatorial volumes (Constructive Solid Geometry). Numerous software systems exist for the creation and manipulation of such geometric data for CAD applications.

Various schemes have been proposed for sharing geometry data across applications, vendors, or systems, such as IGES, PDES/STEP, CORBA, and VRML. The present state of database exchange through any of these systems is relatively primitive as there is negative pressure on vendors to provide their proprietary properties or functionality through common interfaces, hence such interchange formats tend to focus on the lowest common denominator across representations. Even large-scale industry efforts to organize interchange formats (as all the above list represent) has failed to make universal impact on database portability. The so-called *Product Data Model* tends to be user- or company-specific and little interpretation outside the vendor's suite of compatible tools can be expected.

Given this backdrop, often characterized as ``islands of automation,'' any expectation that CAD data alone will help TO generation appears tenuous. The major CAD data omissions include:

- The appropriate level of detail for maintenance manipulation. What detail is necessary for manufacturing may be excessive for maintenance planning. While *decimation* procedures exist for transforming detailed models into simpler (fewer polygon) models, there is as yet no established scheme for identifying and labeling the maintenance-relevant features in the process.
- Information on assembly-relevant features, including mating and manipulation features. What features are important for machining, engineering design, and structural integrity may be quite separate and different from features needed for part removal, hand grips, restraining devices, alignment indicators or keys, text labels, and contents.
- Function, state, and operation models. Information on the contents and operation of a system may be crucial to the safe and proper ordering of procedure steps.

Although some CAD databases contain some of these additional features, they are not widely recognized enough to have warranted significant industry expenditures on externally accessible formats. For example, PDES/STEP has an extensive ``form feature'' specification, but it is almost completely devoted to manufacturing features (holes, bosses, filets, etc.). Large-scale real-time simulation systems utilize multiple levels of detail for objects in order to help optimize transformation and display costs but do not require manipulation labels.

Recent CAD software products such as DesignWave from ComputerVision and SolidWorks from Bix go part way toward more integrated views of 3D modeling: they basically equate components and assemblies (thus allowing convenient hierarchic part structures) and offer underlying constraint engines that manage part shape in the design context. These and other PC-based CAD systems provide hooks for designer-supplied application data which, in the Microsoft Windows environment, can be easily copied to other applications such as spreadsheets, visualizers, or Word documents. Unfortunately, the features that are natively supported include manufacturing processes such as profiling, extruding, making holes, hollowing, etc. On the more positive side, the ability to name surfaces or features[4] such as ``bottom,'' ``side,'' and ``fixturing-area'' is a small but encouraging step toward better feature labels for AMI.

Outside the mainstream industrial CAD products, various research groups from both Artificial Intelligence and Computer Graphics have endeavored to design object representations that capture notions of object use and functionality. These are often termed ``knowledge bases'' to indicate that they contain the sort of information that a person might need to know in order to use, manipulate, operate, or otherwise reason about the object. Essentially, AMI requires a knowledge base plus suitable geometric CAD data. The knowledge needed goes beyond the CAD features available today. We must look to the human tasks themselves and determine what additional information and knowledge they require.

2.3 Beyond Geometry

It is important to understand why mere geometric reasoning – and even the addition of a human model – is not in itself sufficient for instruction generation. There are two crucial observations, already alluded to above:

- Form features are considered *geometric* constructions and constraints.
- Operations on assemblies consist of actions which must consider three aspects of manipulation: kinematics (translation and rotation), dynamics (rate, torque, and force), and process functions (contents and roles).

The conclusion we draw is that *geometric considerations yield only kinematic actions*. To obtain the full range of essential maintenance actions – *and hence to generate suitable instructions* – the dynamic and process relationships between components must be considered. Since form features are geometric, we *require features that relate to the dynamic and process properties* of the assembly. It is therefore not surprising that CAD modelers do not give us the data needed to promote AMI. The very use of the phrase *form features* as part of the product data model should trigger this realization. The features that we need should be given new names. For want of existing terms, we propose to use *manipulation features* and *function features*.

2.4 From Procedure Steps to Human Actions

Even if we have a precise mathematical description of the component and tool paths that would be needed for disassembly, that numerical data must be transformed into human readable and understandable instructions. While mechanics tells us that a rigid object may only be translated and rotated, the terms that people use to describe manipulation tasks are often strength- and effort-based and therefore far richer and expressive. So simply listing the ordered steps from a geometric disassembly planner does not by itself create instructions. What is needed is *a representation that maintains sufficient knowledge about the objects and processes involved to aid in selecting suitable textual terms that describe human actions.*

If human modeling systems are to be expected to execute the expanded set of actions required by dynamic and process models, then the appropriate movement generators and synchronization mechanisms should be built. We have experimented with strength-guided motion[5] and motion dynamics simulation[6]. Such tools can be used to judge task feasibility and human resource (strength and torque) requirements. Likewise, we have developed programming tools such as PaT-Nets to help synchronize human actions with external events such as might be produced by a process model. In some relevant experiments, we controlled the selection and timing of human actions based on events happening in a shop floor (assembly line) discrete simulation system that we developed. These could be extended to the maintenance domain where various diagnostic tests must be executed by multiple individuals and coordinated in both time and state: for example, testing cockpit controls that are linked to systems activated elsewhere in the aircraft requires coordinated tasks by two agents as well as a process model of the system states.

Maintenance actions described in Technical Orders have a structure which is parallel and hierarchic. One or more agents may be involved in a task, but even one agent may be doing two or more things simultaneously with her hands. Tasks are hierarchic in the sense that to perform one step a sequence of sub-steps must be accomplished, and so on until the recursion grounds out at elementary skills, hand motions, etc. We conjecture that one of the major difficulties in planning maintenance instructions is not just in the decomposition into the action hierarchy (as AMMP was doing) but the proper identification of parallel steps. These have various linguistic manifestations (as we will examine later) such as ``while'' and ``so that'' clauses. Often the actions will describe the movement and/or the result to be sensed or checked as a culminating condition. For the present discussion it is important to note only the structure implications of both parallel and hierarchic representations for procedures.

In a previous report[7] we proposed a *Parameterized Action Representation* (PAR) as a mechanism for storing action descriptions suitable for both simulation (animation) and language generation. The PAR thus serves as an ``interlingua'' for different input and output forms. In particular, PAR mediates actions in terms of kinematic, manipulation, dynamic, and process features. This means that we need to construct a lexicon of PAR instances for the objects and actions involved in maintenance procedures. While the initial cost of doing this may appear to be high, it is possible that case-based reasoning from pre-existing Technical Orders can be used to establish databases for new maintenance procedures. An alternative approach worth pursuing in the future is the ``just-in-time'' generation of (skeletal) PARs from pre-existing TO instructions. The expected advantage to this would be the focusing of processing resources on those aspects of the task description most salient to both the process steps and the maintainer actions. This hypothesis must be tested in practice, however, through Natural Language processing of existing TO's into PAR form.

First we will briefly describe the PAR for agents and objects. Then we will comment on the problem of obtaining suitably enriched CAD object representations for task animation.

2.5 Parameterized Action Representation (PAR)

PAR is a representation structure for procedures integrating expression in language, planning, and animation. The top-level type in the representation is the *parameterized* action; we call it ``parameterized'' because an action depends on its participants (agent and objects) for the details of how it to be accomplished. For instance, opening a door and opening a window will involve very different behaviors on the part of the agent. The subparts of a parameterized action can refer to particular aspects of the agent and objects as part of their meaning. The PAR is therefore intended to help capture agent processes (actions) as a function of the objects being manipulated. Clearly a feature-based object representation will be advantageous in allowing an action to be expressed in terms of object features no matter how those features are actually configured in the object geometry. The PAR is an object-oriented structure for process and action description and execution and consists of several data fields:

```
type parameterized action =
    (agent: agent representation;
    objects: sequence object representation;
    applicability conditions: sequence disjunctive-queries;
    culmination conditions: sequence disjunctive-sensor-queries;
    spatiotemporal: spatiotemporal specification;
    manner: manner specification;
    subactions: actions).
```

Figure 1 Parameterized Action Representation (PAR) structure

2.5.1 Agents

The human agent is the distinguishing feature between an action and a mere process. It specifies which agent is carrying out the process described in the rest of the representation. For AMI we assume that the agent refers to a human model of a maintainer.

PAR agent types and PAR object types are very similar in concept, except that the agent type has some extra fields which also describe the behaviors of the agent which would influence some of its actions.

```
type agent representation =
    (coordinate-system: site;
    state: state space;
    rel-dir: relative directions;
    special-dir: special directions;
    grasp-sites: sequence site;
    capabilities: sequence actions-and-applicability;
    nominals: sequence value-ranges).

type actions-and-applicability =
    (action: parameterized action;
    applicability: sequence disjunctive-queries).
```

Figure 2 Agent Representation structure

For each instance of the agent type, a list of actions that the agent is capable of performing is specified. The agents can also be considered to be capable of playing different roles. For each role, the agent performs different actions. So, instead of maintaining one long list of actions, we could group these actions under different roles. For example, the actions involved while driving a car, such as *grasp a steering wheel, sit with foot on the accelerator pedal*, etc., would be grouped under the ``car-driver'' role. Each of the listed actions is a primitive action. Unlike for the objects, each action is associated with a set of applicability conditions (test for reachability, etc) which check if the action can be performed by the agent. If not, another set of primitive actions is generated for that agent which have to be completed before the current action can be performed. Since agents have capabilities, different agents are easily represented.

Linking PAR to individualized agents defined by DEPTH or *Jack*[8] anthropometry software, for example, only requires that the graphical agent be able to execute the primitive actions with any appropriate parameters. Primitive actions available in *Jack*, for example, include *walk to, reach for, look at, grasp*, and *lift*. Additional primitive actions may be established by manually constructing (authoring) motions or by motion capture. While direct motion playback is sometimes possible, it is far more likely that the motion has parameters and will have to be executed in a spatial context different from that of its origin. In other words, the parameters may define local adjustments to a primitive movement: the final orientation for a walk, the target point in space for a reach, the rung spacing for ladder-climbing, and so on. Another important class of movement parameters is the agent's body anthropometry: movements created for one agent usually will not do the same thing on another. Parameterizing arbitrary movements across different agents remains a research topic, primarily because there is no obvious way to automatically determine which components of the movement are fixed by the environment and which by the body's own structure.

2.5.2 Objects

The PAR object type is defined explicitly for a complete representation of a physical object. Each object in the environment is an instance of this type.

```
type object representation =
    (reference-coordinate-system: site;
    state: state space;
    rel-dir: sequence relative direction;
    special-dir: sequence special direction;
    grasp-sites: sequence site;
    actions: sequence parameterized action).

type site =
    (position: real vector;
    orientation: real vector).

type state space =
    (position: real vector;
    velocity: real vector;
    acceleration: real vector;
    force: real vector;
    torque: real vector).

type relative direction =
    (name: (front, back, left, along, inside);
    value: real vector).

type special direction =
    (name: string;
    value: real vector).
```

Figure 3 Object Representation structure

The state field of an object describes a set of constraints on the object which leave it in a default state. The object continues in this state until a new set of constraints are imposed on the object by an action which causes a change in state. The other important fields are the reference coordinate frame, a list of grasp sites, and directions defined with respect to the object. Some, but not all, of these may be expected as part of a CAD product data model. For example, it is likely that the relative direction information can be obtained directly from the CAD data but it is unlikely that grasp sites would be. The benefit of the PAR is that we can specify *why* we need this additional object information – it is needed for the understanding of maintenance actions.

For each instance of the object type, a set of actions is defined. Each of these actions can be further described as a group of one or more actions. Also, the objects can be represented hierarchically. This allows us to describe actions for a class of objects rather than for every object. The actions are defined at the highest possible level in the object tree. So the action field in an instance of the object type could point to a description or to the parent. None of these actions are likely to be obtained from a CAD model. The information, however, should not be difficult to define and need be done only when a new object class is generated. Automatically generating actions from CAD geometry (and necessary PAR object data) is an interesting future research project. A possible approach to this problem is presented in Section 2.7.

2.6 *Maintainer Accessibility, Interference, and Performance*

Accessibility and interference checking define a crucial role for human models. It is not sufficient to check only that a tool can be inserted and that there is clearance for the tool actuation (such as a sufficient rotation range for a wrench). There must be quantitative evidence that the maintainer can actually reach the tool or her hand to the necessary position and perform the manipulation actions. The preparatory (insertion) and extraction phase of a maintenance step may force both additional steps to remove obstacles that do not directly interfere with the target part but that restrict body access. For example, a nut at the end of a long channel may simply be out of reach for the maintainer even with the typical arsenal of tool extenders. Likewise part extraction can be confounded by obstacles outside the immediate assembly (such as an access door that is too small for the part to pass through) or maintainer limitations (such as inadequate strength or hand width to safely grasp and extract the part). Moreover, all these actions must be feasible for all maintainers across expected size and strength variations. The use of human modeling simulation tools as such as DEPTH and *Jack* along with maintenance access solid generation[9] is essential for such accessibility checking.

There are several complex issues remaining in establishing human modeling inputs to advance automation of technical instructions.

- Human modeling systems need better interference-avoidance path planning, in particular, effective algorithms for reach planning in complex environments. While the TO author might be better experienced to analyze a reach and remove task, there is nothing inherently impossible in automating the geometric part of the planning process[10]; geometric searches may just be computationally slow. (Some promising new randomized (probabilistic) techniques may be very helpful in the high-dimensional configuration spaces encountered in maintenance access situations[11].) The main challenge is in establishing the maintainer's effective strength in a particular (contorted) posture. A secondary challenge is validating the access plan over the likely range of maintainer anthropometry – developing geometric planners which work with size intervals may be an untouched problem..
- The user may have to use experience to associate the proper tool with the process step. This should be obtained from part feature knowledge. Also, the AMI user may have to decide whether a second assisting hand access (from the same or another maintainer) is needed. Data on strength requirements and safety margins can help determine whether such instructions should be considered.
- Object-specific tools and grasps are not always specified in Technical Orders. It is apparently left to the maintainer to establish the pragmatics of the task based on training or prior experience. Better object feature notations will help, but ultimately the intricate hand movements of a maintainer will not be modeled so that generate clearances should be considered in the design phase.
- Since the design is fixed by the time instruction steps are produced, it cannot be the function of the T.O. author to debug the design. What the T.O. author should do is warn the maintainer of possible hazards by generating text that focuses on safe handling by noting avoidance of sharp parts, pinch points, greased surfaces, delicate subparts, and so on. These cautions and warnings may be developed for general classes of situations and inserted in the T.O. as required. The challenge is to distinguish *practice* (and thus the choice of words in the actual instruction) from general cautions and warnings. While a case-based approach might help, we know of no study that has yet established differentiation criteria.

2.7 *System Level Understanding*

We have already argued that maintenance planning is not simply a geometry problem: Disassembly or assembly of components requires knowledge of geometry, structure, function and action:

- The geometry and maintenance features necessary for manual handling.
- The hierarchic structure and connection constraints of mechanical components.
- Component contents, such as hydraulic fluid, fuel, and electricity, and consequent function, such as pump, reservoir, conduit, valve, switch, etc.
- Human actions that carry out the part motions, and which are sensitive to accessibility, interference, and adequate strength.

Restating this another way, instruction planning is *the creation of a correct ordering for component removal which converts the geometric data, maintenance features, and structural hierarchy into steps that respect content (function) integrity or safety, and allow successful maintainer access.* The influence of functional constraints are crucial, since they are directly related to action term (verb) choice, cautions, and warnings relevant to system maintenance. We recommend that extensions to the PAR in this area be pursued in a subsequent study.

An approach to obtaining the necessary PAR data for each of these aspects of maintenance planning is through *software agents*. Rather than try to compute a complete representation of all possible object feature data, it seems more reasonable to work on an ``as-needed'' basis. Software agents can be launched on raw CAD data to ascertain necessary manipulation and maybe some function features. The technology to do this is not impossible, but it is clear that such agents must be designed to consider human (maintainer) capabilities as well as structure, geometry, and function. Agents ask for user assistance or evaluation as needed while they process.

- Polygon decimation to reduce geometric detail.
- Spatial analysis and planning to determine motion freedom, as in the AMMP system, and maintenance access solids.
- Feature analysis for edges, grasp sites, and stability regions[12] [13].
- Tool selection based on fastener data (as is done in DEPTH).
- Schematic diagram representations correlated with geometric components via existing CASE tool databases or engineering simulations.

(The usefulness of the last one is noted in another AMI report[14]: integrated technology (mechanical, electrical, hydraulic propulsion) systems are common in the aircraft environment.) Of course, these agents must be developed, designed, and coded within an overall software agent architecture. The information they gather in the service of the PAR database can underlie any user interface for T.O. automation.

3 Language Issues

Natural Language Generation (NLG) is a support technology for human-computer dialogue in particular and for any communication in general. Applications of NLG include text summarization, explanation in complex decision support, instruction, and consistent multi-lingual communication.

Work in Natural Language generation is continually attacking harder and harder problems, whose solutions are ever more useful. One of these involves generating effective *instructions* from rich models of actions and of the situations in which they are being performed. This requires producing clear, concise and effective descriptions of entities and actions—descriptions that go beyond what current technology is able to produce. Work in this area, however, will complement and enhance the general body of techniques available for Natural Language generation.

The four main language-related issues involved in generating maintenance instructions are:

- Identifying the features needed in a task specification adequate for producing all relevant aspects of Technical Orders (T.O.);
- Identifying the features of the language used in T.O.s;

- Building and organizing the lexical resources needed for generating T.O.s;
- Constructing the NL generation procedures needed to produce T.O.s.

(There are also general system-related issues such as the problem of how users can collaborate with a system in authoring instructions, including editing automatically generated text and having that reflected in subsequent interactions, and the problem of how generated text could be made to conform to standards of vocabulary and usage.)

3.1 Generating Action Descriptions

There are three main factors determining the language-related needs of automated maintenance instructions, the first being the *genre* of the text in which it occurs:

Narrative, explanation and instruction each has a different purpose and this affects what is conveyed about actions. *Narratives* are intended to describe what has happened, usually in connection with agents who undertake the actions and/or experience their effects. The reader of a narrative therefore only needs to know as much about an action as will allow them to understand the agents' behavior and responses. *Explanations* involving actions are intended to support an agent's decision-making. The recipient of an explanation therefore needs to know all and only those characteristics of an action that might affect their choice (e.g., how long the action takes to perform, its likelihood of success, etc.). *Instructions*, on the other hand, are intended to enable agents to perform an action themselves and to do so correctly. The recipient of an instruction therefore needs to know everything about an action that may be relevant to correct performance. This shapes, in part, the kind of communicative goals that must be satisfied in the action descriptions that are generated generate.

A second feature affecting the descriptive features of an action description is the complexity of the action's conceptualization. This ranges from a simple *state-space* conceptualization, to conceptualizations in terms of *kinematics*, *dynamics* and *process control.*

3.1.1 Action Descriptions in State-Space Terms.

The same action can be conceptualized in different ways. The simplest way of viewing an action is just in terms of its input-output characteristics. These can be called *state-space* action descriptions, since they can be modeled with a state-space representation such as STRIPS operators[15] or situation calculus functions[16]. Such action descriptions simply indicate (either explicitly or implicitly) what must be true prior to the action and what will hold afterwards. The action itself is viewed as atomic: its internal characteristics over time are not considered. For example, the following instructions conceptualize the actions of positioning and releasing in state-space terms, referring only to the goal state and the atomic action needed to bring it about:

a. Position engine feed switch to off.

b. Release fuel hot switch.

The verbs that occur in these action descriptions are typically simple verbs indicating simple changes-of-state such as turn off, turn on, start, stop, switch on, switch off, etc. As illustrated in the example, a verb indicating causation of change-of-location, position, can also be used. These would all be described as achievements, indicating a moment in time that signals the occurrence of a new state. The assumption is that the focus of the description is on the new state, rather than the process that achieved it.

3.1.2 Action Descriptions in Kinematic Terms.

Where more of an action's characteristics over time are relevant to its correct performance and intended outcome, a state-space conceptualization and representation may be inadequate.

What may be required is a conceptualization and representation of an action in terms of its kinematic properties—properties of movement, including path, goal, object orientation during path, object orientation at goal, manner of movement, and temporal bounds. For example,

c. Shove the armature assembly into the hole of the gear case with the fan end down.
(*goal, orientation during path*)

d. Place the field case in the fixture so that the slot faces you.
(*goal, orientation at goal*)

e. Slowly open adapter assembly valve.
(*manner of motion*)

f.. Support position shall be maintained until bolts have been torqued.
(*temporal bounds*)

As already noted, it is not the action itself that is either state-space or kinematic, just its specific conceptualization and representation.

While an action may usually be conceived of in state-space terms, when its kinematic properties are relevant to its performance, it will be described in kinematic terms. For example,

g. When power switch is positioned to on, only ready light should come on. If either of other lights come on, reset switch shall be momentarily depressed.
(*temporal bounds*)

3.1.3 Action Descriptions in Dynamic Terms.

A kinematic conceptualization and representation may itself be inadequate. Instead, what may be required may be a conceptualization and representation in terms of dynamic properties (properties involving force). But forces themselves are not visible and people find it hard to monitor them directly. Rather they are monitored in terms of the effects they have on other things, and these effects are what are commonly specified in instructions. Moreover, the difficulty in monitoring forces also makes it easier to perform an action incorrectly. Thus, when forces are involved, effective action descriptions often include indications of

- perceivable target and path conditions that an agent can and must monitor for so as to adjust force accordingly, and
- the consequences of those conditions not holding, so the agent knows what action to take if they don't.

h. Engage bolts. Tighten as required to clamp up and fair in cover to match adjacent structure with no looseness in joint.
(*target conditions*)

i. Evenly torque four nuts to 110-140 inch-pounds.
(*path conditions, target conditions*)

3.1.4 Action Descriptions in Process Control Terms.

Finally, even a dynamic conceptualization and representation may not suffice. Instead, what may be needed is a conceptualization and representation in terms of the control they exert over processes that can act as independent forces, whose behavior in response to the agent's action the agent must monitor. Describing such actions for the purpose of enabling an agent to perform them correctly often then requires connecting the agent's actions with the processes they effect either intentionally or contingently. This may involve indicating process conditions that must be monitored for (both intended conditions and ones indicating problems) and actions to be taken in each case. Below, example j merely indicates the presence of an independent process. Example k indicates the intended condition (brought about by an independent process) that must be monitored for.

j. Loosen pressure adapter assembly nut and allow nitrogen in service line to escape.

k. Monitor fuel indicator until indicator reads 150-400 pounds in each reservoir.

Note that from the perspective of choosing verbs to use in realizing action descriptions, one particular class may be more natural to use to describe an action at one level of complexity than at another, as discussed in Section 4.4.

Aspectually, a verb may be inherently momentary or durative, and it may or may not have particular consequences (changes of state) associated with it. As shown in[17], these two dimensions determine the different types of actions, which are shown in the following table along with verbs that express each type of action:

	momentary	durative
+conseq	**achievement**	**accomplishment**
	release	*position*
–conseq	**point**	**activity**
	hit	*push*

Figure 4 Aspectual characteristics of verbs

An additional factor affecting the descriptive complexity of an action description is its relationships to other actions. Such relations can be exploited to generate efficient descriptions. We have already shown[18 19 20] how *purposive* relations between actions (*generation* and *enablement*) can be ``overloaded'' in an action description, so that besides conveying purpose, a purpose clause can also convey important features of the action to be performed.

But *similarity* relations between actions can also be exploited— for example, to take advantage of what the listener is presumed to already be familiar with:

> Grimy lustres can be washed in the same way as other glass objects, **but** either remove the metal parts first, or dry them very carefully afterwards.
>
> Use neat washing-up liquid to remove any stubborn patches, then rinse off and dry carefully. Plastic laminates can be wiped down in the same way **but** should be rinsed well to avoid streaking.

In the above examples, actions are described in terms of components added to an action procedure that the listener is presumed already familiar with. In other examples, actions are described in terms of modifying the manner of another action.

> Tape the unglued section of the joint, **but** not too tightly.
>
> Push firmly, **but** don't force the drill to cut too fast.

3.2 Previous NL Generation Work

To date, work in Natural Language generation involving action descriptions has been done for the purpose of generating *instructions* and has viewed actions solely in *state-space* terms. Even Dale's seminal PhD thesis on instruction generation[21], which took a recipe for making butter bean soup as its model, explicitly *excluded* from its consideration all features of actions that could not be described in state-space terms (italicized text below):

> Soak the butter beans, then drain and rinse them. Peel and chop the onion and potato; scrape and chop the carrots; slice the celery. Melt the butter in a large saucepan and add the vegetables. Sauté them *for 7-8 minutes, but don't let them brown,* then add the butter beans, water or stock, the milk and the bouquet garni. Simmer *gently, with the lid half on the saucepan, for about 1 ¼ hours, or until the butter beans are tender.* Remove the

herbs, and liquidize the soup, stir in the cream, and add the sea salt, freshly ground black pepper and nutmeg *to taste*. Reheat the soup, *but don't let it boil. Serve each bowl sprinkled with croutons.*

The instructions generated consisted of a sequence of what appear to be state-state descriptions:

> ... Slice the celery. Melt the butter. Add the vegetables. Saute them. Add the butter beans, the stock and the milk. Simmer. Liquidize the soup ...

However, even these are not all truly state-space descriptions, since ``saute'' and ``simmer'' are activities over time and have no intrinsic culmination. To actually do what is intended, their termination conditions must be conveyed as well.

While for many applications outside the realm of maintenance instructions (e.g. instructions filling out forms or for using consumer software), state-space descriptions may suffice[22 23 24 25], for generating maintenance instructions, one must consider a range of issues involved in realizing more complex conceptualizations.

4 Lexical Semantics

In this section we discuss the ways in which words can be formed into sentences in order to precisely convey the types of actions that are necessary to perform maintenance activities. Initially we only consider linguistics issues, and do not discuss the coordination of these sentences with visual output until Section 6.

4.1 Verb Classes

In recent linguistic literature, authors such as Levin and St. Dizier have made attempts to classify verbs in terms of their semantic properties[24]. The goals of these efforts have been to identify semantic factors which influence and correlate with syntactic behavior. This has resulted in the identification of useful components of meaning, which are on the one hand linguistically encoded in structures such as the lexical entries of verbs or grammatical constructions, but have on the other hand great relevance for the representation of actions in the world. These components include:

- exertion of force: requires a magnitude of force which in turn can affect speed and distance or change of location
- directed motion: requires a trajectory
- contact: requires respective location points
- change of location of an object: requires a path for the object

4.2 Action Hierarchies

Verbs can be represented in a lattice that allows semantically similar verbs, such as all motion verbs, to be closely associated with each other, with a higher level node that captures the properties these verbs have in common. Many of the actions appearing in F-16 corpus[39] can be categorized as change-of-state verbs or verbs of exertion of force (see Figure 5). Change-of-state verbs can involve either simple discrete change (e.g., turn on generator set) or continuous change such as change of orientation or location (motion). Most of the motion verbs in the corpus describe manner of motion controlled by an external agent (e.g., slide and rotate). In contrast to simple change-of-state verbs, which are inherently bounded or telic (the endpoint of the event is specified as part of the meaning of the verb), these manner of motion verbs are unbounded or atelic and require a path prepositional phrase or some other adjunct to specify the culmination of the action. For example, slide is inherently atelic, but with the adjunction of a path prepositional phrase with an endpoint, slide sleeve onto air tube, the culmination of the action becomes clear. This is particularly important when the action descriptions are instructions, since the agents following the instructions must know when to stop performing the action.

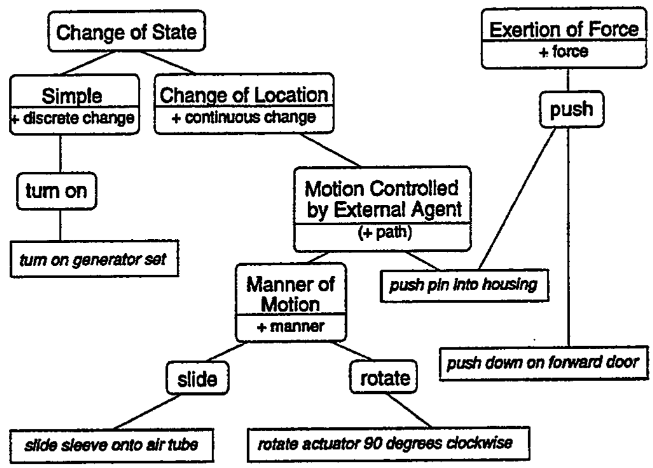


Figure 5 Actions in hierarchies inherit semantic features.

4.3 Sense Extensions

Beth Levin has defined verb classes for English that are distinguished partly by the underlying semantic components, but also by the syntactic frames in which the verbs can occur.[26 27] Levin verb classes are based on the ability of a verb to occur or not occur in pairs of syntactic frames that are in some sense meaning preserving, such *as John climbed the mountain, John climbed up the mountain.* The fundamental assumption is that the syntactic frames are a direct reflection of the underlying semantics. The sets of syntactic frames associated with a particular Levin class are supposed to reflect underlying semantic components that constrain allowable arguments. For example, *the sleeve slides easily on the elbow* indicates two arguments, the sleeve, the object in motion, and the path along which it is moving, the elbow. A controlling agent can be added, *John slid the sleeve easily on the elbow*, creating a causative form of the verb. This is not true of all motion verbs. *Mary came into the room* has two arguments, Mary, the object in motion, and a description of her path, which ends when she is in the room. However, a controlling agent cannot be added, * *John came Mary into the room* is not acceptable. The causative form of inherently directed motion verbs such as *come, leave, exit, enter, et cetera*, is not allowed.

Many of the Levin verb classes have overlapping members, and these members highlight the shared semantic components of the classes to which they belong. The verbs that are triple listed in the *carry, split* and *push/pull* classes are *push, pull, tug, yank, kick, shove.* In its base sense, as a member of the *push/pull* class such as *push* is a verb of **exerting force**, which leads to the possibility of **motion,** *John pushed the filing cabinet.* We do not know whether or not the filing cabinet moved, but it is possible, in fact probable. In this base sense *push* can also appear in the conative alternation which adds the preposition *at* to the transitive form, *John pushed at the filing cabinet.* This emphasizes the **force** semantic component and the ability to express a recognizable "attempted" action, where any result that might be associated with the verb (e.g., motion) is not necessarily achieved. As a *carry* verb the possibility of **motion** is made a certainty by the addition of a path prepositional phrase, *John pushed the box across the room.* In fact, the motion is considered to be **accompanied motion**, in that the object and the agent traverse the same path.

The critical point is that a *push/pull* verb's meaning can be extended to either "attempted" action where motion of the object is ruled out, or causation of motion where motion of the object is necessary. These two extensions cannot co-occur – they are mutually exclusive, so we cannot say * *John pushed at the box across the room.* The *carry* verbs that are not also in the *push/pull* class, such as *carry, tote* and *tow*, are more "pure" examples of the *carry* class and always imply the achievement of causation of motion. Thus they can never take the conative alternation, * *John toted at the bag.* The subset of *carry* verbs that can take the conative are only those verbs that are also listed in the *push/pull* class.[28] [29]

Note that when *push* is used as a pure verb of **exertion of force** (e.g., *push down on forward door*), it is atelic — no endpoint or bound is specified, and it is not clear when the pushing should be stopped. This information about when to stop performing the action must come from other sources (purpose clauses, "until" clauses, etc.). However, when a path prepositional phrase is added, or adjoined, as in *push pin into housing,* the bound is specified and no additional information is needed.

Finally, push can also be extended to a change-of-state reading by the adjunction of the adverb *apart, John pushed the boxes apart.*

In summary, certain syntactic frames indicate the adjunction of prepositional phrases or adverbs that provide a regular extension of meaning to the core sense of many verbs. In the same way that we associate the directed motion feature with path prepositional phrases for manner of motion verbs (and other classes, such as sound emission), we can also associate a change of state verb feature with the adverb, *apart,* and its adjunction onto *split* verbs and *push/pull* verbs (e.g., *John pulled the components apart*). In addition, the conative regularly indicates the performance of an action but not necessarily the achievement of the result which is the goal of that action. The inherent semantic components of a verb suggest which extensions of meaning are possible, and the extensions themselves may be mutually exclusive.

4.4 Action Hierarchies and Complexity of Action Descriptions

As described in Section 3.1, the complexity of an action description may range from simple state-space actions to richer conceptualizations in terms of kinematic or dynamic models. It may be more natural to use a particular class of verbs to describe an action at one level of complexity than another. Simple change-of-state verbs easily describe actions in state-space terms *(turn on generator set);* motion verbs readily describe kinematic aspects of an action (*rotate actuator 90 degrees clockwise*); and verbs of exerting force lend themselves to descriptions of the dynamics of an action *(pull gently on coupling nut).* Verb classes define the semantic components entailed and allowed by a verb, which affects the complexity of the actions that the verb can describe. However, as described in the previous section, the basic meaning of a verb can be extended in a variety of ways, and can thus be coerced into a more complex action description. Therefore, the complexity of action description is determined as much by the adjuncts of a verb (its usage in a sentence) as by the verb itself.

Thus, the same verb, *pull,* can be used to describe:

1. state-space changes, with the adjunction of *apart,* as in *pull the components apart*
2. kinematic actions, with the adjunction of a path prepositional phrase, as in *pull sleeve into housing*
3. dynamic actions, with the adjunction of a manner adverb, as in *pull gently on coupling nut*

The adjunct *apart* in the first example changes *pull,* which is inherently an **activity** like *push,* into an **accomplishment** with a consequent state in which the object is now "apart".

In generating maintenance instructions automatically, it is critical for us to understand precisely the import of the verbs and their syntactic frames in order to ensure that the instructions can be followed and will achieve the desired result. This can be illustrated by examining the examples from Section 3.1 in more detail, such as example c., *Shove the armature assembly into the hole of the gear case with the fan end down.* Our verb classification lists *shove* as a *push/pull* verb, emphasizing its dynamic aspects. However, as mentioned above, the meaning of these verbs can readily be extended to include directed-motion through the adjunction of a path prepositional phrase which puts them in the *carry* class as well. In this example, that phrase is *into the hole of the gear case* and it indicates the goal of the path of motion of the *armature assembly.* The emphasis in this sentence is on the **motion,** or **change-of-location,** indicated by the *shoving* action. However, there must be some necessary **exertion of force** involved, or *slide* or *place* would have been chosen instead of *shove.* The final location of the armature assembly should be in the hole of the gear case.

Place is in the same verb class as *position,* and they both normally indicate the **causation** of a **change-of-location.** However, in *Position engine feed switch to off,* that **change-of-location** is insignificant, and the emphasis is on the new state of the *switch:* **off.** In *Place the field case in the fixture so that the slot faces you,* we have the explicit locational prepositional phrase, *in the fixture,* so the change-of-location of the *field case* becomes salient.

Finally, in *Slowly open adapter assembly case,* the verb *is open,* which is classed as a change-of-state verb – the new state of the *adapter assembly valve* will be **open**. However, *opening* and *closing* objects is a complicated process that is highly dependent on the characteristics of the object involved. For this reason, in a particular application domain, in addition to classifying *open* as change-of-state, we also rely on object-oriented descriptions of open and close states for all of the domain objects which include descriptions of the actions required to achieve these states. An important element in our sentence is the adverb, *slowly,* which describes the manner in which the valve opening activity must be performed. Our object-oriented action descriptions have to be parameterized to allow this type of context-dependent modification.

In all of these examples the element of time and change over time represents a significant departure from previous state-space action descriptions. The modeling of spatio-temporal characteristics of actions is a critical factor in our ability to handle these types of examples.

4.5 Tasks as Sequences of Complex Actions

We have focussed thus far on detailed analyses of individual actions such as rotations, slides, pushes and pulls. Actual maintenance tasks are much more complicated, and comprise several actions being performed sequentially or in parallel. PARs can capture temporal and causal relations between individual actions in the same way that PAT-Nets can, and can provide a hierarchical representation of a complete task. For instance, the most common task in the data we have examined is a removal task, as portrayed in . In 119 out of 800 cases, a removal task first involves an opening action in order to obtain access, and in all cases it also involves a removal of a specific object such as a bolt or a support. In a very few cases it involves purging actions or rotations, and it quite often involves disconnections before the removal can be accomplished. It is important for preconditions and post-conditions for each of these actions to be listed explicitly, and this is often information that is missing from the text, and cannot be automatically inferred by a natural language system. For instance, removing nuts and bolts that are behind a closed panel door cannot be accomplished without first opening that door. However, even though these actions are often listed sequentially, there is nothing in the instructions that explicitly ties the opening action to the removal action as a precondition. This type of causal linking between certain actions has to be manually inserted.

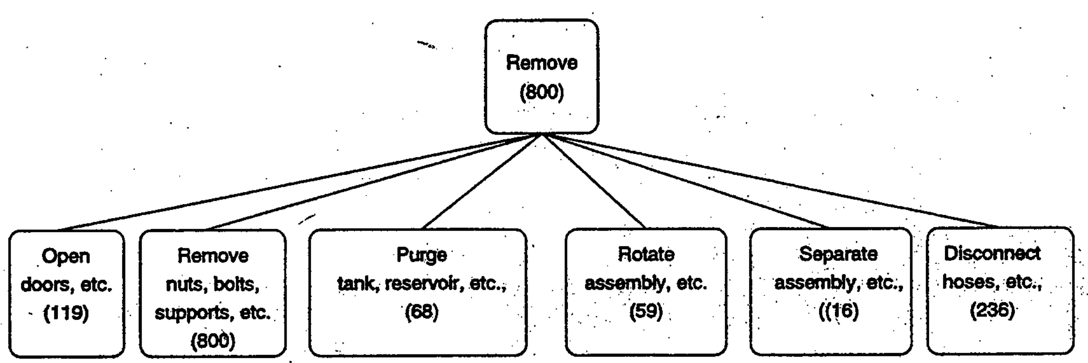


Figure 6 Task Breakdown of Removal Procedure

4.6 *Creating a Procedure Library*

As discussed above, both the objects and the actions involved in the maintenance procedures can be represented as hierarchies. This relevance of these types of hierarchies to this domain has been well established by a joint British/French project for interactive instruction definition.[30] In designing a system that will allow for flexible application of existing procedures to new objects and new situations, it is essential that these hierarchies and the generalizations available at each level be represented explicitly. This includes the representation of the preconditions and post-conditions that allow the system to reason about optimal sequencing of actions, especially of actions that are being put together sequentially for the first time. Although existing technical order data, and the specific procedure cases that BBN can derive, provide a rich resource of typical actions, it does not provide an explicit representation of the causal and temporal links that are necessary for this type of reasoning. A system designed to adapt existing maintenance procedures to new equipment would require a rich procedure library of objects, actions, and tasks (action sequences) that allows for the inheritance of properties, preconditions, and post-conditions from more general object and action types. Creating such a procedure library will require extensive manual knowledge modeling in addition to the extraction of existing task sequences and typical actions from available resources such as technical orders.

The procedure library that we envision as a prerequisite for the task of automating maintenance procedures is given in Figure 7. It would provide a hierarchical representation of PARs for simple actions, complex actions, and sequences of actions, as well as all of the objects that are possible participants. The entire library would be cross-indexed, which would allow all objects associated with an action to be retrieved by means of the action itself, or conversely all actions associated with a particular object to be retrieved by the object. In addition to the existing technical orders, CAD models, as discussed in Section 2, would also provide a rich resource for object information. Our goal would not be to duplicate this information, but to provide appropriate links to it. Our PAR hierarchy would link the linguistic generalizations we have discussed previously with their action visualizations, and provide a seamless integration of animation and natural language. It would include information about the argument structure (participants) of each action and the process modeling requirements.

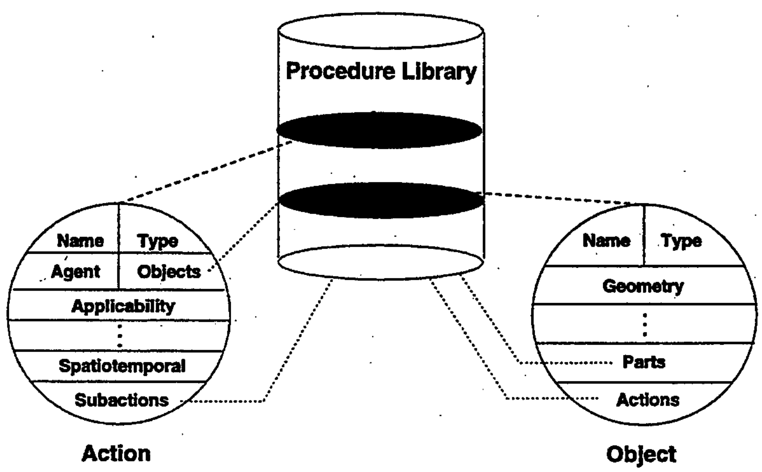


Figure 7 Procedure Library: Actions and Objects

4.7 *A Grammar for Automating Maintenance Instructions*

Any text generation system requires a grammar. We have chosen a lexicalized grammar which combines information about words and information about their grammatical functions into a single structure - a tree. Our formalism is Lexicalized Tree-Adjoining Grammars (LTAGs) which is a tree rewriting system as described in Joshi 1975, Schabes, Abeille, and Joshi, 1988 and Schabes, 1990.[31 32 33] A lexicalized grammar offers the semantic preciseness of a semantic grammar without losing the generality inherent in syntactic description. By coupling our lexicalized grammar with verb class information represented as features, each verb can state precisely its syntactic characteristics, its semantic constraints, and any relevant inheritance information, either domain dependent or domain independent.[1]

The primitive elements of LTAGs are called elementary trees and are of two types: initial trees and auxiliary trees. The minimal, non-recursive linguistic structures of a language, such as a verb and its arguments, are captured by initial trees. Recursive structures of a language, such as prepositional phrases, are represented by auxiliary trees.

Elementary trees are combined by the operations of substitution and adjunction. Every tree is associated with a lexical item (or set of items) of the language, called the **anchor** of the tree. The tree represents the domain over which the lexical item can directly specify syntactic constraints, such as subject-verb number agreement, or semantic constraints, such as selectional restrictions, all of which are implemented as features. These features must unify at the end of parsing in order for a sentence to be grammatical. Alternative syntactic realizations of a lexical item are grouped together into *tree families*, and the semantic constraints automatically apply to the same arguments in the alternative trees.

There are critical benefits to lexical semantics that are provided by the extended domain of locality of the lexicalized trees. Each lexical entry corresponds to a tree that will be used to build the parse of the sentence that that item appears in. If the lexical item is a verb, the corresponding tree is a skeleton for an entire sentence with the verb already present, anchoring the tree as a terminal symbol. The other parts of the sentence, the noun phrases that occur as subject and object, will be substituted in at appropriate places in the skeleton tree in the course of the derivation. In this way the tree captures the predicate argument structure of the lexical item that is its anchor, such as a verb like *disconnect*, and can express specific semantic class constraints on the arguments, such as the subject being an **animate agent**. Features can be used to indicate that *disconnect* belongs to the *change-of-state* verb class, linking the lexical item to the action hierarchy.

4.7.1 Stylistic Characteristics of Instructions

Existing technical orders provide a useful guideline for appropriate instruction content, and stylistic and vocabulary considerations. A lexico-syntactic analysis of the verbs that occur in task order descriptions for F16 corpus reveals regularities and linguistic preferences in the descriptions of actions and instructions, which are typical of the genre. Characteristics include the subcategorization frames for verbs, with corpus-based frequency counts, the selectional restrictions on the associated verb arguments, and class membership in a domain-specific verb lattice. A distributional analysis of the corpus provides counts for different types of actions (e.g., 12 instances of *turn on/off*, 20 instances of *turn*-OBJECT-DIRECTION), as well as different manners of expressing termination/culmination of actions using purpose clauses, *rotate elbow to provide clearance around valve*, and *until* clauses, *rotate until packing is exposed*. In addition, domain-specific preferences of action descriptions can be recognized (e.g., *slide* always occurs with a directional adverb or path prepositional phrase, even though such adjuncts are not generally required in other domains).

Domain-specific characteristics of action descriptions must be combined with general grammatical principles governing verb behavior (such as allowable arguments and subcategorization frames) to generate new action descriptions that are not copied verbatim from legacy corpora, but that still conform to the stylistic requirements of the domain. As discussed in the previous section, verb classes can efficiently capture syntactic regularities useful in generating action descriptions. Semantic features of each lexical item organize the lexicon, differentiate verbs (useful in lexical choice), and constrain allowable constructions of the verbs in action descriptions.

[1] BBN's semantic grammar could be directly mapped to a lexicalized grammar representation.

4.7.2 Analyzing Example Technical Orders

We have provided detailed lexical entries for 10 verbs in the F16 corpus: *connect, disconnect, lift, pull, push, rotate, slide, turn, turn off, turn on.* These verbs split into two classes: **(i) motion** verbs; and **(ii) change of state** verbs. **Change of state** verbs describe an action that is intrinsically bounded - the endpoint coincides with the occurrence of the change of state, and are considered telic, as discussed above. Many **motion** verbs, on the other hand, do not inherently specify a termination of the motion, but simply a description of the manner of motion, so they are considered unbounded or atelic. Therefore it is much more likely for the motion verbs to co-occur with adverbs (*clockwise, downward, forward),* and prepositional phrases which specify a direction or path of **motion**, *(around, into, onto, over)*, as well as an **endpoint** - a locational goal of the motion such as *to the other side.* In addition to path endpoints, motion verbs can also occur with modifiers such as *until* clauses, **durative** *for*-prepositional phrases, *for 5 minutes*, and adverbial **amount** noun phrases (*three quarters of a mile, half of a turn*) which offer alternative endpoint specifications.

To indicate the verb class memberships which impose these restrictions, verbs in the lexicon were marked with the features **endpt** + or **endpt** - and **motion+** or **motion-**. Path/goal prepositions and adverbs were marked with a feature **motion+** to indicate that they should adjoin onto verb trees that belong to the **motion** verb class, and are marked with the same feature. In contrast, *until* was marked to only adjoin to VPs with the feature specification: *endpt* -, meaning that the verb was lacking an endpoint which could be supplied by the *until* phrase. The adjunction of the *until* phrase or a *to* phrase would result in the verb phrase feature being changed to **endpoint+**, which would prevent the adjunction of another **endpoint** phrase.[2] The feature **motion** is used to distinguish motion verbs from non-motion verbs in order to constrain what kinds of modifiers can adjoin. Path, direction, and locational goal prepositional phrases and adverbs can only occur with motion verbs.[3]

Likewise, the feature **endpt** is used to characterize inherent telicity (whether an endpoint is specified as part of the meaning of the verb). Termination conditions, such as until clauses can only be adjoined to atelic verb phrases. *Rotate* is marked **motion+** and **endpoint-**, since it is a manner of motion verb with no inherent culminating condition, as demonstrated by Figure 8. *Disconnect*, in the same figure, has the opposite feature values, **endpoint+** and **motion-**, since as a **change-of-state** verb it has an inherent culminating condition, the new state, but does not indicate motion. In Figure 8, the NP (noun phrase) object node is a substitution site. By substituting an NP like *assembly* into that site, we generate the instruction *rotate assembly.* As discussed previously, motion verbs like *rotate* can have direction modifiers like *downward.* In order to constrain which PPs (prepositional phrases) and ADVs (adverbials) can adjoin to the **motion +** verbs, the ADVs and PPs are also marked with the **motion+** feature at the VP (verb phrase) adjunction site as shown in Figure 9.

---

[2] Obviously, telicity can be built up by various elements in the verb phrase and is not just determined by the properties of the verb alone. For example, inherently telic verbs can be made atelic (or given an iterative reading) by having a plural (unbounded) object. However, no examples of unbounded plural objects were present in the corpus examined, so this was not handled by the features.

[3] Or verbs in related classes such as sound emission, which can be coerced into motion verbs.

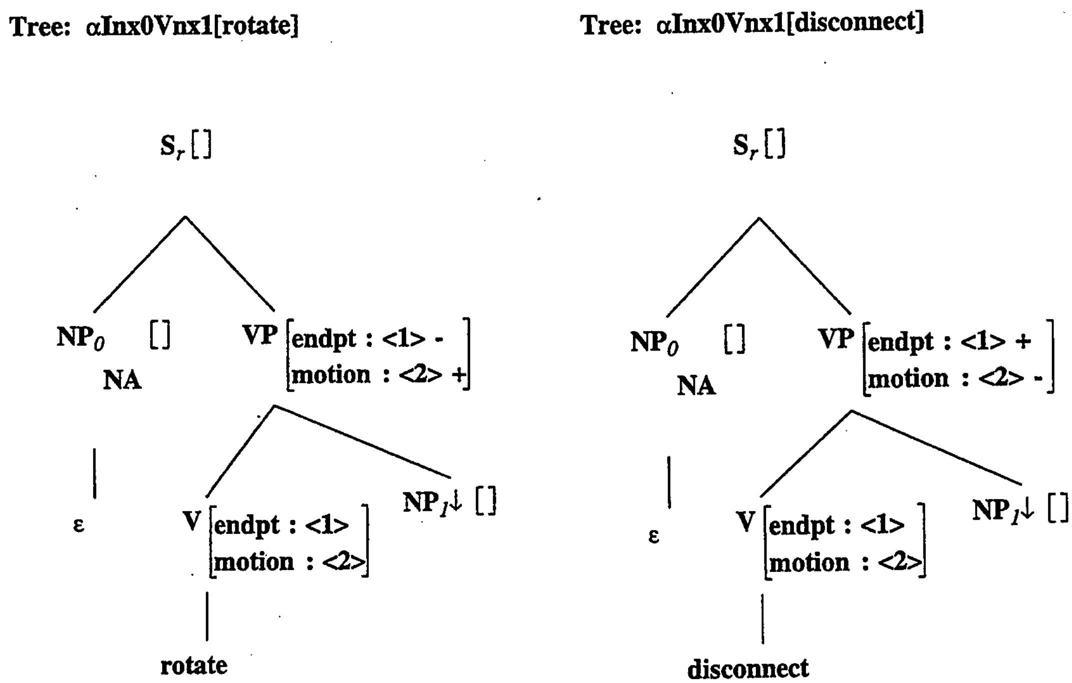


Figure 8 Transitive elementary trees for *rotate* and *disconnect*

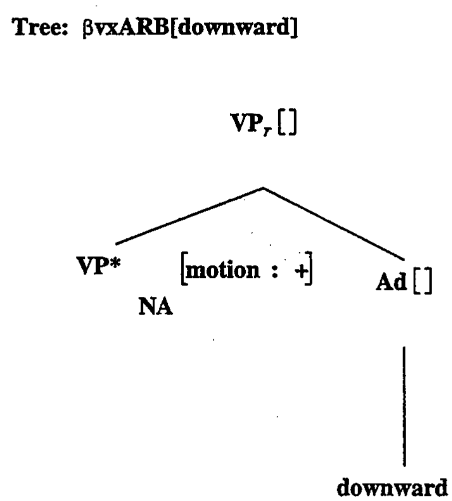


Figure 9 Auxiliary tree for *downward*

The adverbial auxiliary tree can adjoin into the VP node of the *rotate* verb elementary tree, resulting in the derived tree in Figure 10. Since the **endpoint** feature of this structure is still -, an *until* phrase could also be added, which would change that feature to endpoint+. The disconnect tree does not allow the adjunction of either a path/direction PP or an *until/to* PP, since it is motion- and already **endpoint+.**

Tree: substitution-G130459

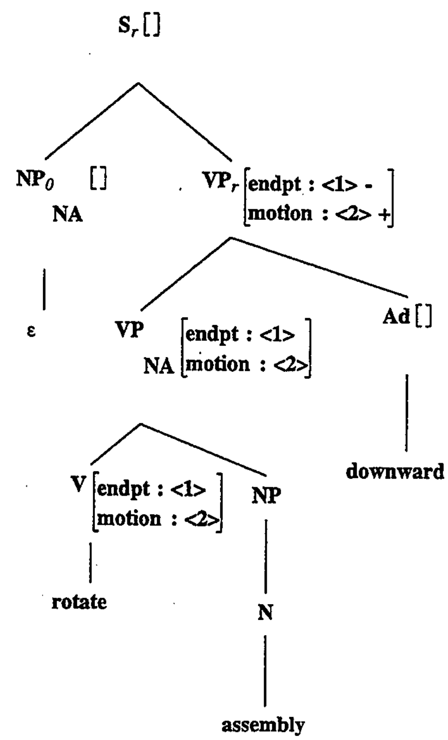


Figure 10 Derived tree for *rotate assembly downward*

4.8 Expressing Action Termination

An important goal of the generation of natural language instructions is to describe the actions in instructions fully and accurately so that they can be carried out correctly from the description. This goal is particularly important to the generation of written instructions where ``speaker" and ``hearer" are separated spatially *and* temporally. In the case of such written instructions, the hearer does not have the opportunity to ask questions to clear up any doubts about the action to be performed and the speaker likewise does not get any feedback from the hearer about the success of the instructions. Therefore, attention must be paid to the adequacy of the instructions generated to be sure that they can be carried out correctly. However, attention must also be paid to the efficiency of the instructions. That is, all the necessary information should be included in the instructions but in an efficient, non-redundant manner. Understanding how information about an action is expressed, which constructions (linguistic structures) are used for which purposes, etc, is essential to generating instructions that describe actions both adequately and efficiently.

There is a growing body of research relevant to generating action descriptions. Researchers have characterized the problem of *lexical choice*, choosing the words (particularly the verb) to describe an action. Lexical choice depends on having lexical semantics (described in the previous section) for all of the possible words and constructions that could be chosen. Finding the best match between the semantics of words and what is to be conveyed is a well-known problem.[34 35] Another problem that has received attention is that of the generation of *referring expressions*.[21 54] Referring expressions are descriptions of objects that successfully refer to (pick out) the intended objects. The description of a particular object can be different at different times in a set of instructions, depending on the other objects from which it needs to be distinguished. The work on referring expressions by Dale and Reiter[56] is the generally accepted method of generating object descriptions and is incorporated into the SPUD generation system (see Section 5.3). Because expressions of *purpose* are common in action descriptions, they have also been a topic of research. Di Eugenio[18] has looked at the sort of (limited) inferences that are used in interpreting instructions which contain purpose clauses. Her work shows how purpose clauses can constrain or modify the performance of actions -- inferences that, in generation, would allow features of an action description to be conveyed implicitly. Work by Vander Linden and others[36 37 38] explores the range of expressions of purpose and other *rhetorical relations* (functional relations that hold between clauses and serve communicative goals) and has begun to formalize the situations in which each type of expression is used. Through the analysis of naturally-occurring instructional text, they have found that the action, the participants, the type of rhetorical relation, and the syntactic options available all influence the choice of expression. These issues, lexical choice, referring expressions, and rhetorical relations, all are important to generating instructions, however one important, but unexplored issue is that of expressing the *termination* of an action, or when an action will or should stop.

Termination information is critical for carrying out instructions correctly. As noted in Section 3.1, actions have inherent aspectual type and an action's type can convey termination information. For instance, *accomplishment* and *achievement*} actions, such as *removing* and *breaking*, have inherent *culmination*, which is termination plus a significant change of state. Since these actions have their termination as part of their meaning, a person performing them does not need to be told when to stop doing the actions. However, some actions, such as *rotating*, do not have inherent termination information. These actions, called *activities*, need to have termination information provided along with them to form an adequate action description for an instruction. Without termination information, an instruction involving an activity will not tell the hearer when he is supposed to stop doing the action. The termination of an action can be provided *explicitly*, in the instruction describing the action or in following instructions, or *implicitly*, in the interaction of the action with other actions in the instructions. However it is provided, termination is important information to have for the performance of actions, especially those without an inherent end.

With termination information representing a vital part of instructions, understanding how termination information is expressed becomes an important part of generating adequate instructions. Existing collections of maintenance instructions can provide information about naturally-occurring expressions of termination information. Analyzing the choices of expression made in naturally-occurring instructions and the contexts in which those choices are made allows the similar choices to be made when generating similar instructions. Thus, examination of naturally-occurring maintenance instructions manifests the importance of termination information as well as its means of expression.

4.8.1 Characteristics of Maintenance Instructions

The analysis of naturally-occurring instructions was carried out on sets of simple step-by-step maintenance and repair instructions. The main corpora are a collection of Technical Orders for the maintenance of F-16 aircraft and the *Reader's Digest New Complete Do-It-Yourself Manual*.[39 40] The *Reader's Digest Manual* is included to augment the limited number of expressions and contexts found in the F-16 instructions. In both, only the numbered step-by-step parts of the corpora have been considered rather than the longer, more complicated text, especially in notes, cautions, and warnings. These step-by-step instructions are fairly simple and straightforward and are accompanied by illustrations to show the objects involved and perhaps the desired end configurations. Restricting the instructions to the simpler step-by-step type provides a manageable collection of contexts and choices of expressions.

Many actions in the domain represented by the corpora are kinematic, that is, they are *viewed* as involving motion over time (see Section 3.1.2). Having kinematic, as opposed to state-space, actions means that some actions will not have intrinsic ends and thus will need explicit termination information. State-space actions, actions that are viewed as atomic changes in state such as *switch*, also occur in the domain but are not as interesting since they have intrinsic ends. The domain also has dynamic actions, actions viewed as involving forces. They are relatively rare but, like kinematic actions, many will not have intrinsic ends, so they are interesting in terms of expressing the termination information. The important point is that a significant number of the actions in the domain are viewed in kinematic terms, as opposed to state-space, and thus provide an interesting set of action descriptions which have expressions of termination information.

4.8.2 Constructions for Expressing Termination

Every language has a limited number of ways to convey information about actions and therefore action termination. Information about an action is realized, or expressed, by many different linguistic sources, usually different parts of a sentence. For example, the basic action itself is usually expressed through the verb. As mentioned above, actions are of different types and this is reflected in the verbs that express actions. For instance, the verb *remove* is considered an *accomplishment* verb, which means that it has an inherent culmination. However, the type (and thus termination) of an action is determined by *all* of the action information and thus linguistic expressions for each part of the action information can convey information about the type of the action as well. In addition to the verb, the arguments to the verb and additional phrases such as purpose clauses and temporal clauses (e.g., "until") reflect the type of the entire action. Interactions among these linguistic expressions also affect the type of action expressed and thus must be considered when producing an action description. The variety of realizations of termination, though limited, information still provides several choices for expressing the termination of an action, all with different purposes and implications in different contexts. Choices fall into one of the following four groups:

- **Predicate-argument structure** is the verb and its required arguments. Termination conveyed by predicate-argument structure results from interaction of the verb's inherent aspectual features and the features of its arguments. Here's a simple example:

  Remove access panel 3428.

- **Optional arguments of the verb**, such as prepositional phrases for paths or locations, adverbial phrases for direction or manner, *et cetera*, can also give termination information. For example:

  Rotate aerial refueling control ***to full counterclockwise (off) position.***

- **Additional clauses** such as *until* and *when* clauses, purpose clauses (including purposive *and* clauses), free adjuncts, *et cetera*, can provide the termination of an action. For instance:

  Slide valve aft ***and remove.***
  Depress bleed valve sufficiently ***to obtain stream of fluid flow.***
  Bleed ***until fluid stream is free of air.***

- **Interaction between action descriptions**, that is, whether a generation or enablement relationship exists between two actions, whether one action is done for the purpose of another, *et cetera*, can give the termination of an action. This group covers those sources of termination information that are not expressed linguistically but rather are assumed to be inferred by the hearer.

Optional arguments and additional clauses provide a range of constructions for the termination information of an action description. These constructions can be used together in the same sentence and thus, for any action, termination information can be provided by multiple linguistic sources. For example:

> *NOTE:* ***To remove actuator,*** *it will be necessary to lift actuator* ***slightly*** *and rotate actuator* ***90 degrees clockwise until sufficient clearance is obtained to disengage actuator splines.***

All of the constructions discussed above can be found in maintenance instructions, including Technical Orders (from which the above example was obtained). The context, which includes the action and situation in which it is to be performed, will determine the appropriateness of a particular construction. By analyzing different sources of naturally-occurring instructions, general correlations between expressions of termination information and the contexts in which they are used can be found and incorporated into a system for generating effective instructions.

5 Text Generation and Planning

5.1 Overview

In existing Natural Language generation systems, text generation is divided into three main stages: text planning, sentence planning, and surface realization[41], shown in Figure 5.1. This section briefly describes these stages.

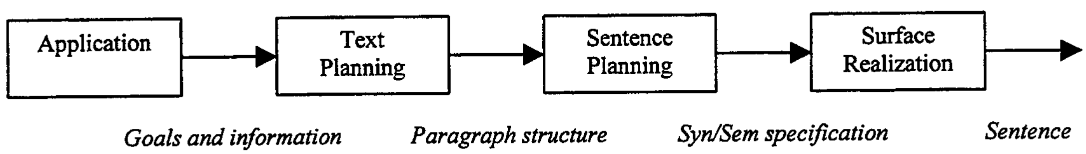


Figure 11 The stages of language generation

In an interactive system that produces Technical Orders, performing these tasks requires tackling standard issues of Natural Language generation, such as organizing content into a coherent rhetorical structure and devising appropriate descriptions for objects and actions. But it also brings up issues of its own. These include the problem of how users could collaborate with the system on writing instructions, or could modify automatically generated text, and the problem of how generated text could be made to conform to standards of vocabulary and usage. We show here that both general and task-specific principles should constrain the design of a Technical Order text generation system.

5.2 Text Planning

Natural Language systems need to generate *coherent* text. It is not enough to decide on a collection of facts and string a description of those facts together. The facts must be organized so as to signal the causal, logical, and intentional relationships among them. One important signal of the relationships among facts is the order in which those facts are presented. Another is the explicit inclusion of special connectives, like *however* or *in particular*, at appropriate places in the text.

Using the following example, Hovy[42] provides convincing evidence of the importance of linking information together in the right order and with the right connectives. This discourse uses an inappropriate order and is missing key connectives; it is very difficult to understand:

> The system performs the enhancement. Before that, the system resolves conflicts. First, the system asks the user to tell it the characteristic of the program to be enhanced. The system applies transformations to the program. It confirms the enhancement with the user. It scans the program in order to find opportunities to apply transformations to the program.

The discourse below conveys exactly the same propositions, yet is relatively clear:

> The system asks the user to tell it the characteristic of the program to be enhanced. Then the system applies transformations to the program. In particular, the system scans the

program in order to find opportunities to apply transformations to the program. Then the system resolves conflicts. It confirms the enhancement with the user. Finally, it performs the enhancement.

It presents the information in a more appropriate order, and uses connectives such as *then, in particular* and *finally* to reinforce that order.

If it is to generate coherent text automatically, a system needs a precise description of what features make discourses like the first awkward and discourses like the second coherent. One obvious difference is that the second discourse, unlike the first, describes the events in the order in which they occur. This might suggest that a generation system could produce a coherent text simply by following a few simple maxims, such as "describe events in the order they happen".

There are good reasons for a more abstract treatment of coherence in discourse, however. First, maxims have exceptions. For example, some discourses are perfectly coherent even though they introduce events out of order.

Open the access panel, but be sure the device is unplugged first.

These exceptions are often easy to explain intuitively. Here, for example, we understand the instruction as coherent because the first clause presents the *main* action in the instruction, and the second clause describes an *auxiliary* action. To elaborate concepts like *main* and *auxiliary* for a computational model, we must be explicit about what the speaker is trying to *do* with the text. That is, we regard *opening* as the *main* action because the clause describing it most directly contributes to the intentions the speaker has in producing the discourse in the first place.

A second reason for a more abstract treatment is that a small list of general maxims seems unlikely to cover all of the decisions that a generation system must make. For example, in the second *enhancement* discourse, the temporal order of actions does not completely determine the order of sentences. The second sentence describes a process that repeats at a high level; the next sentences describe the repeated events that make up this process—these events overlap with the high-level event. (Such ambiguities are particularly likely in maintenance instructions because, as noted in Section 4.4, maintenance actions have a parallel, hierarchical structure.) Again, we can easily motivate a speaker's choice in such exceptional cases, provided we can make reference to what the speaker is trying to *do* in the text. Here, for example, since the main point of the discourse is to describe the overall action of the system, it makes sense to put the high-level description of that action first.

Because of such considerations, researchers have argued that the *intentional structure*[43] of discourse is the key to explaining what makes discourse coherent. Intentional structure formalizes the intuitive idea that sentences in discourse fit together to serve some main point that the speaker has. In the formalism, discourses are broken up into nested blocks of contiguous material, called *segments*, on the basis of how the material contributes to the speaker's (or writer's) intentions for presenting information. Thus, the first step in generating a coherent discourse is to make sure that adjacent information can be grouped into segments with a uniform contribution. Each segment is associated with a *discourse segment purpose* which describes this contribution and which the speaker expects the hearer to recognize as part of understanding the discourse. The second step in generating a coherent discourse is thus to make sure that the hearer can easily recognize the purpose of each segment. If order does not suffice, the system should insert cue words, like *finally* or *in particular* above, to help this recognition: the cue words will explicitly indicate the intentional relationships between segments. (A formal theory allows us to describe this effect to an automatic system; we can say that *finally* marks the concluding step in achieving the intentions of the current discourse segment, while *in particular* indicates that a new segment is beginning, whose purpose is to provide auxiliary information about a process or generalization which the speaker has introduced earlier and which their principal intention is to describe[44].)

There are two approaches, *schema-* and *plan*-based, that bring this idea to bear in Natural Language generation. Early work on text planning, pioneered at Penn by McKeown[45], used *schemata*, which represent naturally occurring patterns of discourse. For example, McKeown's schemata reflected common patterns of describing objects—in particular, naval vessels. One such schema laid out the description of a concept in terms of relating it to a more general concept, naming its parts, and listing its attributes. Schemata in McKeown's **text** system were implemented using an existing formalism for reasoning about linguistic structure, *augmented transition networks* (ATNs)[46], which had been developed to describe how words in sentences fit together. ATNs work by specifying sequences of actions to be taken to recognize or produce a linguistic structure; actions include a set of basic actions, and recursive actions that invoke ATNs to recognize or produce embedded constituents. In **text**, the basic actions are queries that retrieve particular kinds of facts from a database of general knowledge, for inclusion in the discourse; recursive actions indicate the order in which this information is to be obtained and presented in the discourse (according to an outline like the sequence *general concept, parts, attributes* given above).

The plan-based approach has been taken in much subsequent work. This work began from the observation that schema-based text planners cannot reason about the structure, content and goals of the texts they produce[47][48][49]. Such reasoning is very important for systems that must be able to answer follow-up questions, or to provide an alternative discourse, in case the user is not satisfied with or doesn't understand the previous explanation. These systems, which include those by Hovy, Moore and Paris, and Wahlster *et al.* [50], use planning operators that construct a discourse piece by piece, by keeping track of the intentions the system has and applying rules to include information to achieve those intentions. For example, in describing an object, one rule might say that the intention to describe is achieved when the hearer understands what category the object is and what special properties the object has, while other rules might spell out the motivation for the remaining content in McKeown's schemata. These rules could be applied to generate similar texts to McKeown's—but now, after a clarification question, the rule-based text structure could be analyzed to discover which goal failed and why. Given this goal and the reason for its failure, the system would apply alternative rules to answer the question while achieving the original descriptive goal.

As it commonly happens, there is a trade-off between the two approaches. Schemata are less powerful, but are easier to write than plan operators, and plan-based text generators (which are only in the prototype stage) are generally less efficient than text generators that use schemata[51]. On the other hand, plan-based approaches more directly implement the underlying theory of discourse, and therefore derive much more of the flexibility, generality and validity of the theory. Despite these differences, the two approaches are nevertheless related: you could imagine compiling plan operators in fixed combinations to derive schemata—special text plans that indicate how plan operators would be applied in normal or common situations.

In technical order generation, an important consideration in the evaluation of this trade-off is the need for users to make modifications to text when they feel it is unclear, awkward or redundant. (It is unreasonable to expect that an automatic system will always—or even *ever*—produce clear, polished, concise instructions.) To facilitate such modification, the rules and data structures used to generate text must be accessible and understandable to users. This suggests that a plan-based text generator may be more appropriate for Technical Order generation. In representing the intentional structure of text, plan-based generators can provide a record of why particular content was considered for presentation at a particular point in the text, why one option was judged better than its alternatives, and what the content contributed to the overall discourse. This record provides a natural representation for the technical writer to interact with. For example, the writer may often be able to correct infelicitous choices by the generator simply by selecting what the generator thought was the next-best alternative. In more complicated cases, the writer can update the specifications for planning operators to correct the reasons for the mistake and thereby improve future document generation.

5.3 Sentence Planning

Text planning selects and organizes the propositions, events and states that should be described in a discourse. However, text planning is typically performed at an abstract level that passes over a host of important decisions about the discourse—including, decisions about what words to use in the discourse and decisions about how to connect those words together into grammatical sentences. In Natural Language Generation, the process of making such decisions is called *realization.* This process generally involves two phases. The first phase converts the text plan from a conceptual representation (in terms of propositions, events and states) into a grammatical representation (in terms of words and abstract structural relationships). This phase is called sentence planning[52]; in this section, we observe that the phase of sentence planning in Techical Order generation will itself require sophisticated reasoning. The second phase, *surface realization*, processes a representation of grammatical relationships to obtain a string of words, in the form and order that signals those relationships in the target language. We describe surface realization in the next section.

The key function of sentence planning is to select and adapt linguistic forms so that they are suitable for the local context in which they are to appear. One important issue in sentence planning is the content of definite noun phrases. Definite noun phrases need to refer uniquely to an object and thereby allow a hearer to identify the object quickly and naturally. In contrast, the internal representations of objects in computer systems are often inscrutable symbolic names, and rarely record explicitly an appropriate description of the object. Good descriptions must therefore be *constructed* as part of sentence planning. Research that addresses this problem includes[53, 54, 55, 56].

The content needed to refer uniquely to an object varies greatly, according to the *salience* or attentional availability of the object in context, as well as the salience of *distractor* objects with similar properties. In the best case, the object is the most salient entity of the appropriate type, and it can be described using a pronoun—with almost no content at all. On the other hand, if the object has not been mentioned in discourse at all, it may be necessary to provide a ``complete description'' of the object, including detailed information about its size, color, location and type. In the range of cases in between these extremes, the system must construct a reduced description, which should contain enough information to uniquely identify the object but should also be brief, so that the discourse remains clear and concise.

Title:
/tmp/AMI/alldoors.ps
Creator:
XV Version 3.10a Rev: 12/29/94 - by John Bradley
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Title:
/tmp/AMI/somedoors.ps
Creator:
XV Version 3.10a Rev: 12/29/94 - by John Bradley
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 12 Access panel configurations for "Open the top panel (just left of center)"

We can illustrate this variation by considering how to describe the truck pictured in these three illustrations, which can be configured with a variety of different access panel arrangements.

Title:
/tmp/AMI/doorhandles1.ps
Creator:
XV Version 3.10a Rev: 12/29/94 - by John Bradley
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 13 Access panel configuration for "Open the top left panel"

In the first setup, the truck has a three by six array of panels; to identify one of them to open, it was necessary to say ``open the top panel just left of center.'' In another configuration, the same panel was selected from a very sparse array of three by two panels; to identify it, it was necessary to say only ``open the top left panel.'' Finally, in a medium array of three by three panels, this same panel was now identified in the instruction ``open the top center panel.'' As described in Section 6, visual cues such as arrows or highlighting can provide alternative or supplementary methods of identifying a particular panel. The use of both visual and verbal methods of description has to be carefully coordinated. Another advantage of a plan-based approach to text planning is that, since it provides a presentation of the intentional structure of text with a record of why particular content was considered for presentation at a particular point in the text, why one option was judged better than its alternatives, and what the content contributed to the overall discourse, it can easily coordinate decisions about verbal descriptions with decisions about visual descriptions.

The **SPUD** sentence planner tackles such alternatives as a central part of sentence planning. The main loop of SPUD incrementally extends its partial sentence plan by an available operator, along the classic lines of AI planners. The innovation in SPUD is to combine a formal grammar that determines which operators apply, and a model of the interpretation of the partial sentence that determines how well the partial sentence fulfills SPUD's communicative goals. Each operator corresponds to a combination of a lexical item and a structural description of a syntactically correct way to use that lexical item (this is made possible by a grammar formalism based explicitly on building large trees out of smaller primitive component trees: the Lexicalized Tree-Adjoining Grammar formalism). Each operator is evaluated by coupling it with an associated semantics and pragmatics. The *semantics* is a representation of what it means to use that operator; the *pragmatics* is a representation of what circumstances are required for the operator to sound natural or appropriate. SPUD uses these two representations to determine how tightly the operator fits with the context, how much it helps the hearer identify the objects that the sentence refers to, and how much it contributes to the overall message of a sentence.

As required by the task, SPUD constructs descriptions to identify actions as well as objects. As with objects, actions must be distinguished from other possible actions. For example, if a rack has only one open configuration, you can say: ``Open the rack.'' This is the only possible type of action in the domain that fits this description. If, however, a rack has several open configurations, you may have to say: ``open the rack to the full and upright locked position.'' Now there are many types of action that would count as ``opening the rack'' and so (just like any other object) more information is required to allow the hearer to identify which of these possibilities is intended. Again, knowing that several open configurations are available is an important cue for indicating that a visual image should accompany the verbal description.

To generate natural descriptions of objects and actions, SPUD also calculates the interactions between descriptions in a sentence. Taken together, the rack sentences provide an example of why this is needed. In the rack sentences, the description of the object of the action contributes to identifying several features of the action. In particular, once a sentence identifies which rack is being opened, properties of that rack should be taken into account as necessary to infer how the opening should occur. If you have identified a rack which clearly can only open to one position, it may be distracting or confusing to explicitly indicate how far to open it.

Another way descriptions in sentences interact is that specifying a kind of action to perform can help identify the objects the action should apply to. For example, suppose a truck like that in the figures has two racks—one on either side of the truck—but only one of these racks has a hose. Then by saying ``Remove the hose from the rack'' you can refer successfully to the rack with a hose in it. The action helps identify the right rack in this case: only from there can a hose be removed. For other actions, this description will not suffice (e.g. ``shut the rack'')—if you intend to refer to one of the two racks, it will be necessary to identify the racks by location or by association with the hose, or by visual cues.

We can see the expression of termination conditions (discussed in Section 4.8) as a special case of interaction among descriptions in a sentence. When to stop is one of the things every instruction must describe. However, not every sentence includes an explicit *until* clause that indicates the termination point. As more information is included in a sentence, it becomes likely that the hearer will be able to identify the endpoint even without such a clause. This is just what we saw in Section 4.7.2; a specified path or goal allows the hearer to identify when to stop, and hence no overt description of when to stop is needed. SPUD is well-suited to modeling this kind of interaction.

Another important issue is to use controlled and consistent syntax and vocabulary. In Technical Orders, words and constructions must be used with consistent precise meanings in instruction text. For example, technical order specifications dictate that *shall* is to be used (in warnings/cautions) to describe correct procedure. NL generation systems must be designed to use such specifications as part of determining which words (and syntactic constructions) they can use. SPUD expects its representation of grammar to indicate exactly such semantic and pragmatic conditions on when words and constructions are to be used; formulating these conditions appropriately will guarantee that SPUD's output conforms to specification. The corpus characteristics of existing technical orders, as described in Section 4.7.1, are an important resource for naturally occurring language that complies with these specifications.

5.4 Surface Realization

The final stage of generation is surface realization. This is a purely linguistic level, which takes choices about words and syntactic structures made during sentence planning, and constructs a sentence using them. We illustrate this with a simple example: *a dog barks*. For this sentence, a sentence planner computes only that the sentence involves the subject *dog* and main verb *bark*, with an indefinite determiner on *dog*. The syntax of the language indicates that *dog* must precede *bark*. The morphology of the language (the rules governing which forms of a word can express which different grammatical functions) indicates that *bark* must be inflected to *barks* to match the subject *dog*. Finally, the phonology of the language (the rules governing how different sounds are used in different contexts in the language) indicates that the indefinite determiner takes the form *a* before *dog* because *dog* begins with a consonant. As this example suggests, surface realization relies on basic linguistic and computational linguistic research, and can be thought of as describing the reverse of processes performed in Natural Language Analysis[57 58 59].

6 Automating Maintenance Instructions

We have discussed two different approaches to sentence generation, the schema approach, which we recommended in our previous report, and the planning operator approach, which we are currently recommending. Schema approaches, based on predetermined patterns of information, are distinguished by being easier to build and having faster execution than the operator approach, but by also being less flexible and not allowing interactive modification. Although planning operators, which reason explicitly about intentions, are harder to implement and validate, they are more suitable for providing details, answering follow-up questions, and interactive modification. There are two important considerations that have affected our choice of approach and which lead us to currently favor the planning approach. The goal of allowing users to interactively modify maintenance procedures and the possibility of integrating the display of visual information with text both require a more flexible approach.

6.1 A User Interface for Planning Procedures

We envision an environment in which an instruction author can interact freely with a system that has knowledge of typical maintenance procedures and methods for modifying them, as pictured in Figure 14. This system will allow for varied methods of data presentation, ranging from animations to 2-D drawings to text, and to all combinations of these. The author will be allowed to view maintenance activity simulations, and to edit or revise such activities. The system will be able to automatically produce text corresponding to these procedures, and the author will also be able to edit and revise the text. The system will store edited and revised versions for future access, along with notes about preconditions and post-conditions as well as pros and cons that the author considers relevant. The Task Transfer process is the method by which we will link existing data sources such as CAD, CASE and LSAR with the objects in our Procedure Library (see Section 4.6), so that we can be constantly enriching it as new data becomes available. Our core planning process would be operator-based planning described in more detail below, and it would attempt to apply existing procedures to new situations. Planning agents would analyze these new situations to offer "pros" and "cons" with respect to the suitability of the existing procedure in this situation, and the author would be able to modify these comments as desired. The final choice of procedure application would be up to the author, who might choose to define new actions or procedures as potential subtasks. The resulting procedure, and any new actions, would all be stored as new PARs in the Procedure Library for future use.

This same interface would also be suitable for the initial creation of the Procedure Library, in particular for enhancing the skeletal action descriptions that can be derived from the existing technical orders.

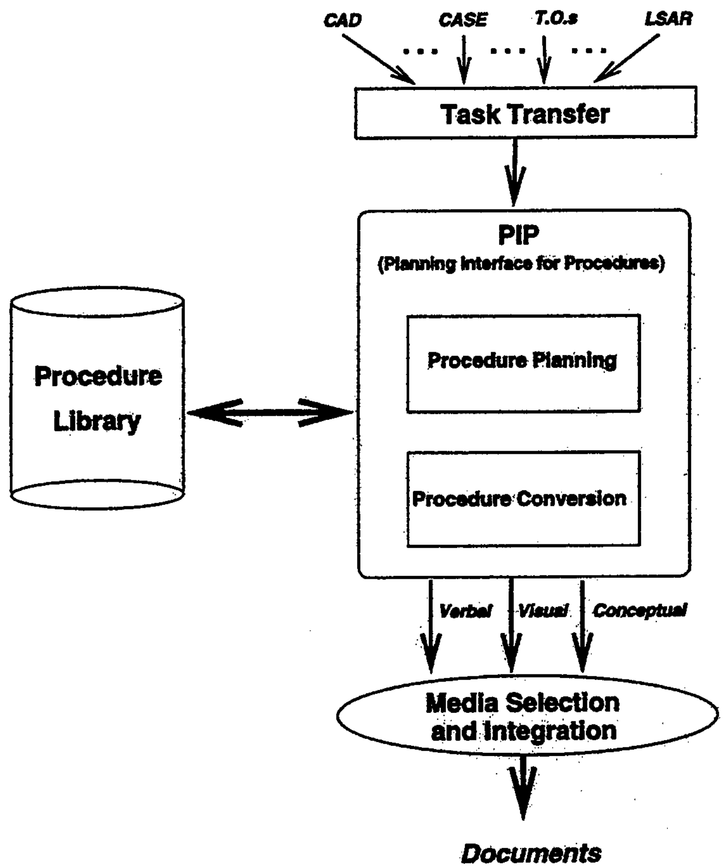


Figure 14 Overall Architecture

6.2 Coordinating Visual and Verbal Information

In the mixed media approach of combining visual and verbal descriptions, each task has to be evaluated individually to determine the advantages and disadvantages of the different explanatory elements, and their displays have to be carefully coordinated. A planning based approach will facilitate a much wider range of presentation choices (customizing to user training levels, ``hypertext'' mode allowing more detailed explanations, animated displays of activities, *et cetera*), which will allow the underlying maintenance procedure representations to be used in a variety of ways.

The existing implementation that has the most to offer us in with respect to planning based approaches to generation is the WIP system, developed at the German Research Center for Artificial Intelligence, DFKI, by Wahlster, André, Finkler, Profitlich, and Rist.[50] They have demonstrated that the generation of a multi-modal presentation can be considered an incremental planning process aimed at the achievement of a given communicative goal. Their system allows the generation of alternative presentations of the same content taking into consideration various contextual factors. In this way, the choices for graphics generation influence the production of text and vice versa. In order to achieve this fluid integration of graphics and text, it is necessary to extend the definition of natural language concepts such as speech acts, anaphora, and rhetorical relations, so that they can be defined with respect to graphical information as well. Since our task breakdown provides an already determined structure, we only need to consider a subset of the communicative goals considered by WIP. However, their system was only designed to accommodate static graphical displays, so their simple state-based action representation would have to be extended to include spatio-temporal and manner information for the description of kinematic and dynamic events.

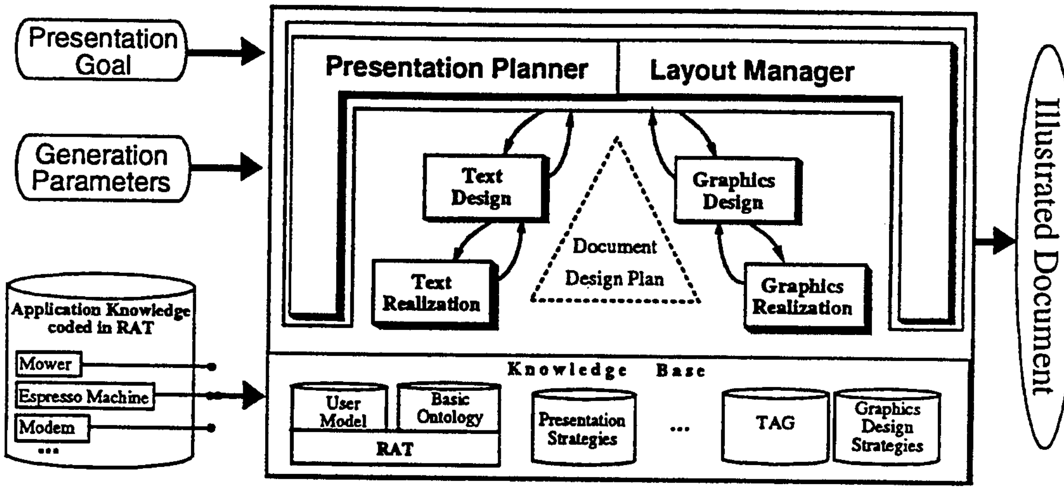


Figure 15 The WIP planner for coordination of visual and verbal presentation

There are many areas of compatibility between the WIP approach and ours. As indicated in Figure 15, they rely on a Tree-Adjoining Grammar formalism for the lexico-syntactic representation as do we, and they are strongly committed to incremental processing, as are we. The presentation planner takes a domain plan, in our case it would be a maintenance task, as the goal to be presented, and performs a reasoning process to determine the optimal coordination of its visual and verbal presentation. This reasoning process relies on atomic action representations that consist of a set of action parameters and associated pre- and post-conditions, which are quite similar to our PARs.

In addition to extending WIP's action representation to include kinematic actions, we would need to develop domain-specific diagram "wizards" that would be experts in making decisions about:

- visual presentation context (none, single image, image sequence, animation)
- diagram type (cutaway, exploded, sequence)
- diagram layout
- caption, label and arrow placement
- viewpoint selection (camera view, assembly view)
- proper tool use

Since both the visual and verbal display decisions would be organized around the same set of communicative goals, we would also have to develop heuristics and metrics for comparing which style is most effective in satisfying the goals. Examples of the kinds of tradeoffs that need to be considered were given in Section 5. The more we can predetermine the type of information that is best displayed visually or verbally, the simpler our planning process can be. For example, we will need to identify rules for determining when diagrams must be included, and for correlating task steps with progressive illustrations. A particular challenge will be the interleaving of multi-person tasks into linear documents.

7 Human Modeling Case Study

Our case study allows users to manipulate and view the geometry of an F16 Internal Fuel Tank Vent. A pop up menu appears with a list of allowable actions: *open elbow coupling, slide sleeve on elbow, rotate elbow, disconnect pressure sense tube, open pressurization tube coupling, slide sleeve on pressurization tube* and *disconnect pressurization tube.* Some actions will succeed while others will fail due to collisions between our animated human figure's arm and the geometry or due to connections between geometry segments. For example, if the user attempts to disconnect the pressure sense tube without first creating clearance, the arm will collide with the elbow geometry.

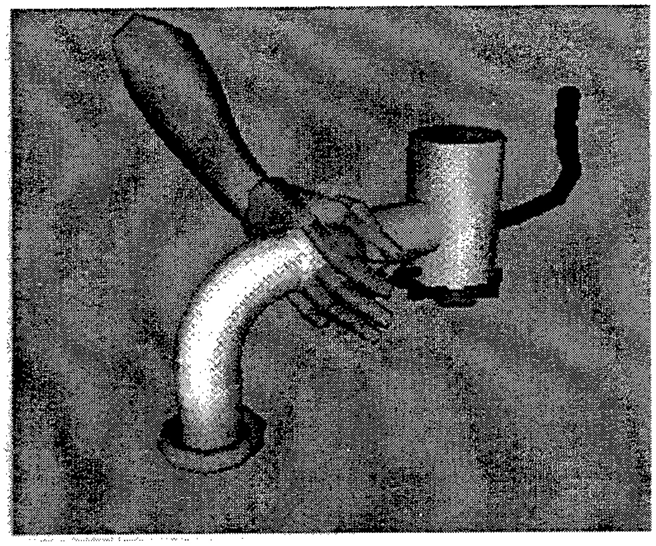


Figure 16 Collision as result of attempted action

An illustration of the attempted action and its result appears in the preceding figure. Whenever the arm collides with the fuel tank geometry, the colliding segments are temporarily highlighted.

Essentially, the user finds a feasible ordering of actions by trying various menu actions. In the example discussed previously, the user must first create clearance around the elbow in order to reach the pressure sense tube. If the user tries to rotate the elbow without first detaching the elbow coupling, he will receive an error message indicating that the elbow coupling is still attached and must be removed. The series of actions to provide clearance to the pressure sense tube are illustrated in the following figures.

First, the user must detach the elbow coupling:

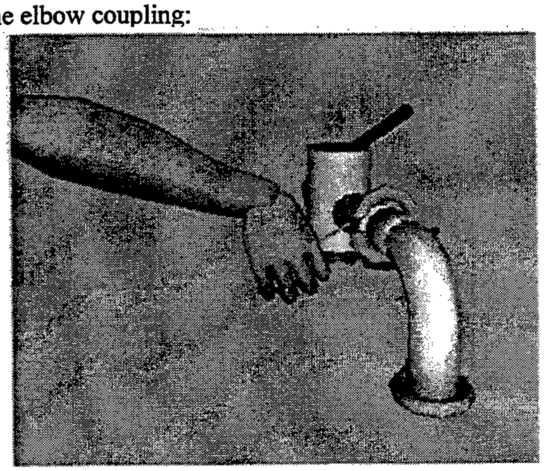


Figure 17 Detaching the elbow coupling

Then, the user should slide the coupling sleeve on the elbow.

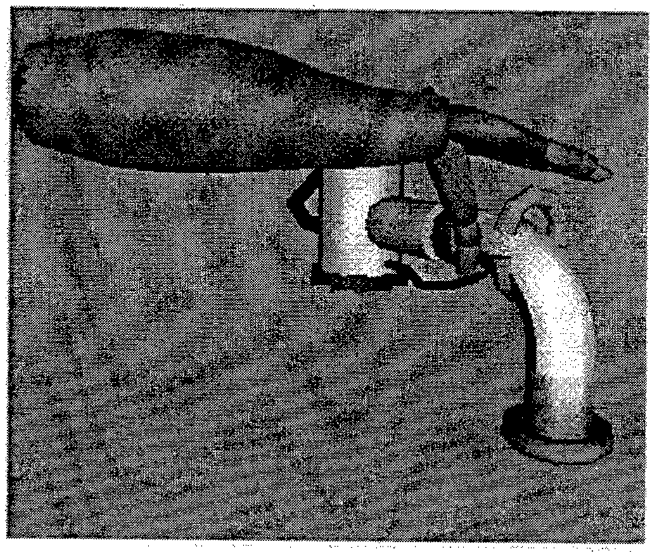


Figure 18 Sliding the coupling sleeve on the elbow

Finally, the user can successfully rotate the elbow:

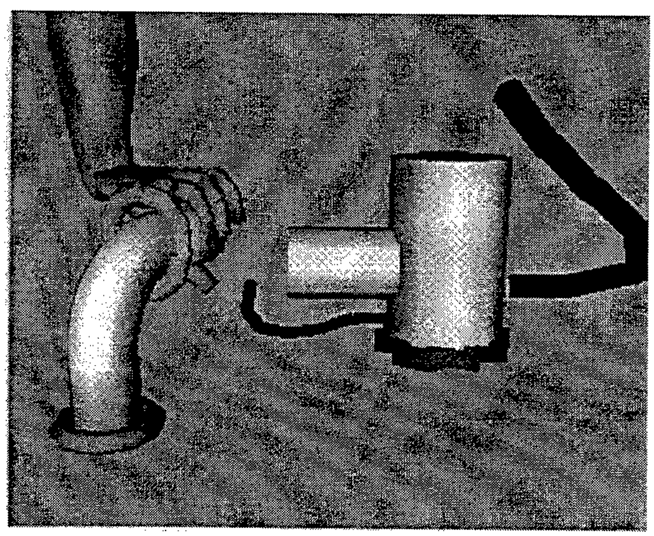


Figure 19 Rotating the elbow

When the geometry reflects collision free access to the pressure sense tube, the menu action *disconnect pressure sense tube* will succeed. An example of the successful reach and disconnect action is illustrated in the following figure.

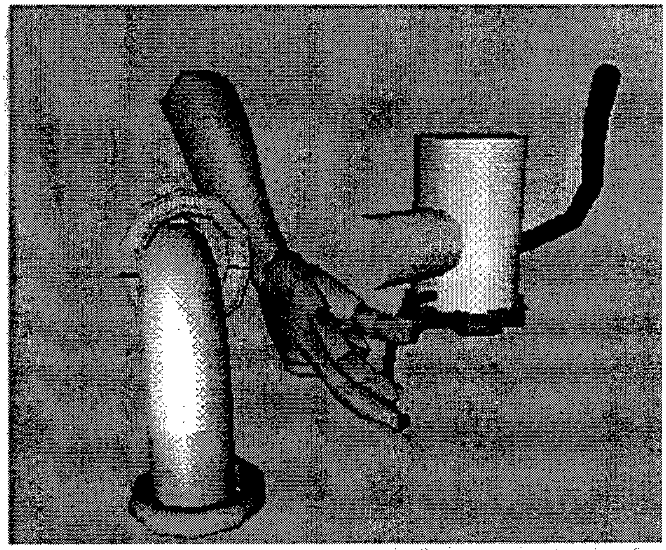


Figure 20 Successful reach and disconnect action

8 Generation of Case Study Instructions

We have simplified the original instructions (instructions 3 through 6 in Technical Order 2-14-1: "Removal of internal fuel tank vent and pressurization valve"), to avoid problems of ambiguity associated with the conjunction *and*. Ambiguity arises in action descriptions joined by *and* as to whether they are meant to convey a sequence of actions or a single complex action. All of the actions here are to be done in sequence, thus breaking sentences such as "Remove coupling and slide sleeve on elbow" into two sentences is acceptable. The modified instructions are as follows:

1. Remove the elbow coupling.
2. Slide the sleeve on the elbow.
3. Rotate the elbow to provide access around the valve.
4. Disconnect the pressure sense tube.
5. Remove the pressure sense tube.
6. Remove the pressurization tube coupling.
7. Slide the sleeve on the pressurization tube.

Figure 21 Modified case study instructions

In order to generate these instructions with SPUD, we need information about the objects in the world, about the lexical items, and about the actions to be described.

Object information specifies each object in the world in terms of its type, its properties and its relationships to other objects in the world. (This information is already needed for the animation of actions.) For instance, information about the pressure sense tube (`pstube`) includes the following assertions:

```
tube(pstube).
concerns(pstube, pressure_sense).
connected(r1, pstube, valve).
location(r1, pstube, place(around, valve)).
```

Figure 22 Example object information: *pressure sense tube*

This states that pstube is a tube, that it deals with sensing pressure, that it is connected at time r1 to the valve, and that at time r1 it is located around the valve. Information of this sort must be included for all the objects that could be acted upon by an agent. This information is used by SPUD when it determines which lexical items to use in describing actions, given the state of the world.

Lexical information specifies the syntax, semantics, and pragmatics of each lexical item (word or construction). A lexical item's syntax is specified by trees (described in Section 4.7) in which the lexical item can appear. Its semantics consists of a set of propositions conveying aspects of its meaning. Its pragmatics specifies the contexts in which it is appropriate to use the lexical item. For instance, lexical information for *disconnect* consists of the following (the sVLnp tree corresponds to the tree in Figure 8):

```
name = disconnect
semantics =
       { agent(Action, Agent), during(Time, Action),
         connected(Time, Object, Other-Object),
         ?NewTime (result(Action, NewTime), free(NewTime, Object))}
pragmatics = { true }
trees = { sVLnp(S,R,E,A,O,F), iVLnp(S,R,E,A,O,F) }
```

Figure 23 Example lexical information: *disconnect*

This states that the lexical item *disconnect* appears in the trees named sVLnp and iVLnp and is appropriate to use in any pragmatic context. (The tree structures for sVLnp and iVLnp have to be provided to SPUD as well.) The semantic propositions convey that at the beginning of the action the object of the action is connected to some other object and after the action the object is no longer connected. SPUD uses information such as this to can choose the lexical items and constructions that most effectively to describe an action.

The particular actions that underlie the instructions are also specified in SPUD. Action *instances* (which are derived directly from PARs) record who the agent is, whether the action involves motion, what its path is, *et cetera.* For instance, the action instance leading to the fourth instruction ("*Disconnect the pressure sense tube*") looks like:

```
agent(action4, you).
during(r4, action4).
present(r4).
result(action4, r5).
free(r5, pstube).
```

Figure 24 Action instance example: *Disconnect the pressure sense tube*

This states that the agent of action4 is the hearer (you), that action4 takes place during the time period r4, which is the present time, that the result of action4 is the time period r5, and that in the resulting time period, the pressure sense tube is free. Matching this information, as well as information about the state of the world, against the semantics of lexical items, SPUD can choose which lexical item best describes this particular action.

Given world and domain knowledge (objects, relationships, *et cetera*), information associated with particular lexical items, and syntactic trees used by the lexical items, SPUD can then find the best lexical items to describe action instances and use the syntactic trees to generate grammatical sentences describing the actions. This flow of information is represented pictorially in Figure 25.

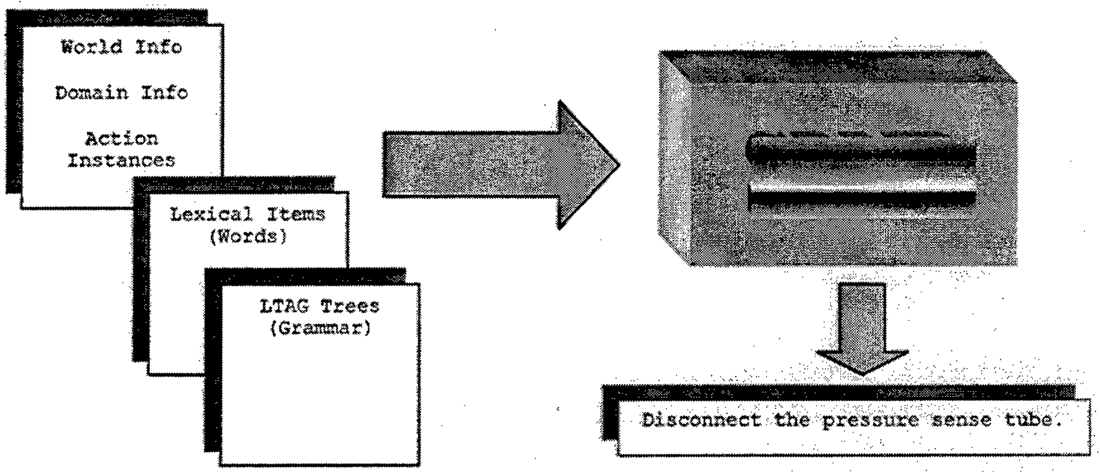


Figure 25 Information flow in SPUD

We have generated the set of the case study instructions (listed above) using SPUD. What makes SPUD a promising method of instruction generation is that subtle changes to the world information or the action information can lead SPUD to choose different lexical items. For instance, SPUD can use the same action information to generate "Disconnect the pressure sense tube" or "Remove the pressure sense tube" depending on whether or not the pressure sense tube is connected to something. Removing constraints on action instances can also lead SPUD to make different lexical and syntactic choices. For example, omitting the fact that an object will be free after an action can change an instruction which uses the verb *remove* into one which uses *place* instead. Changes to the state of the world or the richness of an underlying action specification affect the instructions that are generated by SPUD, making SPUD particularly suited to the automation of maintenance instructions.

9 Recommendations

We envision an environment in which an instruction author can interact freely with a system that has knowledge of typical maintenance procedures and methods for modifying them. This system, described in more detail in Section 6, will allow for varied methods of data presentation, ranging from animations to 2-D drawings to text. The author will be allowed to view maintenance activity simulations, and to edit or revise such activities. The system will be able to automatically produce text corresponding to these procedures, and the author will also be able to edit and revise the text. The system will store edited and revised versions for future access, along with notes about preconditions and post-conditions, as well as pros and cons that the author considers relevant. Extensions to our current implementation which have the highest priority with respect to moving towards this type of system are given below:

- Enriched CAD representations will not naturally (nor quickly) arise from the standards community nor from the CAD vendors; AMI must lead in establishing data requirements.
- Parameterized Action Representation (PAR) provides the key to information about the semantics of procedures, their object-specific, kinematic, dynamic, and process characteristics, and human performance in context. We recommend that extensions to the PAR into non-geometric objects be pursued in a subsequent study.
- Human modeling and task planners are needed to fully model the complexity of human-machine interactions for maintenance.

- The variety of actions and situations needs to be expanded, both in terms of modeling them and in terms of the necessary lexical resources.
- The list of actions should include more actions involving dynamics as well as a wider range of kinematics. Situations should include those that involve independent processes. Corresponding lexical resources should be developed to encode the syntactic constructions needed to realize the expanded variety of actions and situations. We recommend a lexicalized grammar approach, which can build on BBN's semantic grammar approach, because of its greater flexibility and portability.
- The generation system needs to be extended in several ways. First, it should be able to use a verb classification indexing of the LTAG trees, which would support a flexible mapping between communicative goals and lexical realization. Second, it should be able to generate descriptions of actions that extend over multiple clauses. Finally, it needs to be scaled up to handle the large grammars and the large number of communicative goals that will be needed to generate the full range of maintenance instructions.
- The state-space action representation of the WIP planning interface for the coordination of visual and verbal information needs to be extended to include kinematic and dynamic actions. The interface needs to be extended to allow for revision of existing procedures and input of new procedures. The task breakdown provides a suitable structure for guiding the planning process, simplifying somewhat the decision process.

The major research challenges involve the modeling of dynamic actions and the development of causal models. Dynamic actions need to be modeled in terms of the exertion of force and its impact on the state of world, a complex problem that has yet to be fully explored. Causal models need to be developed in order to model actions that involve process control situations, where independent processes and forces are at work. Causal models are especially important for the generation of the cautions and warnings that appear in maintenance instructions.

10 Glossary

accomplishment—a type of action that has a preparatory activity and a culmination; e.g. ``remove''

action description—the presentation or account of an action in a Natural Language text

action instance—an instantiation of an action representation for a particular action

action representation—data structure to hold information about actions

activity—a type of action that has no inherent termination; e.g. ``walk''

adjective—a modifier of a noun, *the **red** coat, the **smooth** surface.*

adjunct/argument—an optional verb argument such as a temporal or path prepositional phrase

adjunction—the operation of combining an optional auxialiary phrase, such as a prepositional phrase or a relative clause, with an initial elementary tree

adverb—a modifier of a verb, *he pulled **gently** on the housing.*

atelic—an unbounded action, without an inherent endpoint

boundary representation—a family of methods for representing objects by covering their surfaces (boundaries) with planar polygons or curved surface patches.

bounded—an action having an having an endpoint

CAD—Computer-Aided Design: software systems used to create 2- and 3-dimensional computer models and drawings of physical objects.

CASE—Computer-Aided System Engineering: software systems used to create and simulate schematic or process models of physical systems (electrical, structural, hydraulic, etc.).

causative—a verb form with an explicit external agent causing an action, as in *The general marched the soldiers around the field.*

coerce—the application of a verb to an argument it would not normally take which requires an adjustment of either the meaning of the verb or the argument, as in *The teapot sang happily on the stove.*

conative—a construction that inserts an at preposition in between the verb and the object of a transitive verb in order to indicate an attempted but not necessarily completed action; ex., *John shoved at the filing cabinet with all his strength, but couldn't budge it.*

construction—a linguistic structure, such as a prepositional phrase.

Constructive Solid Geometry— a method for representing 3-dimensional objects out of volumes which may be combined by union (``glueing''), intersection (the area common to both simultaneously), and difference (``material removal'').

CORBA—Common Object Request Broker Architecture: an emerging standard for sharing object-oriented data models across applications and systems.

culmination—the endpoint of an action or event which has a change of state associated with it.

decimation—the process of reducing the polygon count in a boundary representation model as that it is more compact in storage and faster to display or manipulate.

dynamic—involving forces or torques.

enablement relationship—a relationship between actions; one action enables another action if doing the former (enabling) action allows the doing of the latter (enabled) action, and the latter could not be successfully initiated and performed with some enabling action (e.g. unlocking the door enables opening it)

feature-structure—a data structure made up of has feature-value (property) pairs (e.g. <action-type = activity>)

form feature—an term used to describe some named aspect of shape in a geometric model, such as a hole, filet, profile, etc.

function feature— part of an object that must be identified for understanding how the object functions or fits into a larger process, e.g., intake port, fuel container, conduit, etc.

generation relationship—a relationship between actions; one action generates another action if doing the generating action also does the generated action, and no other action is needed for the generated action to have been performed (e.g. unscrewing the lid generates opening the jar)

hearer model—representation of the hearer (what the hearer knows about, etc.).

hierarchic plan—a task sequence whose members are themselves tasks, and so on.

IGES—Initial Graphics Exchange Specification: a standard format for exchanging geometric shape data, usually in boundary representation form, between CAD systems.

inchoative—denoting the beginning of an action, state, or event.

intransitive—a verb with one argument, *the wheel rotates*.

kinematic—involving motion.

lexical choice—deciding which words are the most appropriate to describe an action, event, state, or object.

lexical item or entry—a word in the lexicon and information about it.

Lexicalized TAG—Tree Adjoining Grammar in which each (elementary) tree is anchored by at least one lexical item.

maintenance access solids—the solid representation of the path and volume occupied over time by an object as it is extricated and removed from its attachment position.

manipulation feature—part of an object that must be identified for proper grasping and manipulation by an agent.

PAR—Parameterized Action Representation used to define objects, actions, and their connection with human abilities; they are parameterized so that they can be used in various contexts in general rather than having a specific procedure for each conceivable case.

PDES/STEP—Product Data Exchange using STEP: STandard for the Exchange of Product model data; an ISO standard for geometric and manufacturing data exchange between CAD and certain types of application programs.

pragmatic context—the communicative goals and world knowledge that hold in the current situation

predicate-argument structure—a verb and its required arguments

purpose clause—clause which expresses the purpose of the action described in the main clause of a sentence; in formal syntax, a purpose clause is an infinitival ``to'' clause attached to a verb phrase.

realization—syntactic expression of semantic or pragmatic information

referring expression—linguistic form that is used to pick out an entity in the world; an anaphoric referring expression is an abbreviated linguistic form which depends on contextual information in order to pick out an entity.

semantics—having to do with the meaning of a word or string of words.

semantic context—what meanings exist in the current situation (e.g. the semantics of the current action)

spatial planner—a system that reasons with or derives various measurements from geometric models.

state-space—a way of viewing actions primarily in terms of state changes they bring about

substitution—the combination of two elementary trees, one of which has a substitution site labelled with node N, and the other of which is the elementary tree for N.

syntactic—having to do with grammatical structure

telic—having an endpoint.

termination—the endpoint of an action or event

transitive—a verb with two arguments, *John pushed the filing cabinet.*

Tree Adjoining Grammar—grammar formalism based on the combination of trees which represent syntactic information. TAGs are more ``powerful'' than phrase-structure grammars—ie., ``weakly context sensitive'' rather than ``context free''—so they can provide an efficient account of a wider range of linguistic constructions.

unbounded—an action without an inherent endpoint, *the wheel rotates.*

VRML—the Virtual Reality Modeling Language that is designed to transfer 3D models and animations across the WWW.

world knowledge—common or shared knowledge about the world

world model—representation of the world (objects and their relations to each other, etc.)

11 References

[1] R. M. Pirsig. *Zen and the Art of Motorcycle Maintenance.*

[2] R. Hoffman, H. Keshavan, and J. Lankford. AMMP: An automated maintenance manual production system. Proc. IEEE Conf. On AI for Applications, March 1994.

[3] S. Deutsch, et al., AMI Final Report to AF/HRG, WPAFB, BBN Technologies, 1998.

[4] T. Laakko, M. Mäntylä, and J. Opas. User-defined features in EXTDesign++, ASME Computers in Engineering Conf., 1996.

[5] P. Lee, N. Badler, S. Wei, and J. Zhao. Strength guided motion. *Computer Graphics* 24(4), pp. 253-262, 1990.

[6] E. Kokkevis, D. Metaxas and N. Badler. User-controlled physics-based animation for articulated figures. Computer Animation 1996.

[7] N. Badler, B. Webber, M. Palmer, T. Noma, M. Stone, J. Rosenzweig, S. Chopra, K. Stanley, J. Bourne, and B. Di Eugenio. Final report to Air Force HRGA regarding feasibility of natural language text generation from task networks for use in automatic generation of Technical Orders from DEPTH simulations. Technical report, CIS, University of Pennsylvania, 1997.

[8] N. Badler, C. Phillips, and B. Webber. *Simulating Humans: Computer Graphics Animation and Control.* Oxford University Press, New York, NY, 1993.

[9] H. Chang and T. Y. Li. Assembly maintainability using motion planning. Proc. of IEEE International Conference on Robotics and Automation, May 1995.

[10] N. Badler, R. Bindiganavale, J. Granieri, S. Wei, and X. Zhao. Posture interpolation with collision avoidance. Computer Animation '94, Geneva, Switzerland, IEEE Computer Society Press, Los Alamitos, CA, pp. 13-20, 1994.

[11] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high dimensional configuration spaces. *IEEE Trans. On Robotics and Automation* 12(4), pp. 566-580, 1996.

[12] C. Bard, C. Laugier, C. Milési-Bellier, J. Troccaz, B. Riggs, and G. Vercelli. Achieving dextrous grasping by integrating planning and vision-based sensing. *Robotics Research* 14(5), pp. 445-464, October 1995.

[13] H. Rijpkema and M. Girard. Computer animation of hands and grasping. *Computer Graphics*, 25(4), pp. 339–348, July 1991.

[14] McDonnell Douglas Aerospace. Automated authoring of T.O. maintenance data for IETM applications. St. Louis, MO, 1996.

[15] R. Fikes anad N. Nilsson. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, Volume 2, pp. 189-208, 1971.

[16] J. McCarthy and P. Hayes. Some philosophical problems from the standpoint of Artificial Intelligence. In B. Meltzer and D. Michie (eds.), *Machine Intelligence*, Volume 4, Edinburgh University Press, pp. 473-502, 1969.

[17] M. Moens and M. Steedman. Temporal ontology and temporal reference. *Computational Linguistics* 14, pp. 15-38, 1988

[18] B. Di Eugenio. Understanding Natural Language Instructions: A Computational Approach to Purpose Clauses. PhD thesis, University of Pennsylvania, 1993.

[19] B. Di Eugenio. An action representation formalism to interpret Natural Language instructions. *Computational Intelligence*, Volume 14, Number 1, 1998.

[20] B. Di Eugenio and B. Webber. Pragmatic Overloading in Natural Language Instructions. *International Journal of Expert Systems* 9(2), pp. 53-84, 1996.

[21] R. Dale. Generating Referring Expressions: Constructing Descriptions in a Domain of Objects and Processes. ACL-MIT Press Series in Natural Language Processing. MIT Press, 1992.

[22] J. Delin, A. Hartley, C. Paris, D. Scott, and K. Vander Linden. Expressing Procedural Relationships in Multilingual Instructions. In Proceedings of the 7th International Workshop on Natural Language Generation, Kennebunkport, Maine, pp. 61-70, 1994.

[23] A. Hartley and C. Paris. Two sources of control over the generation of software instructions. In Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics, June 1996.

[24] C. Paris, K. Vander Linden, M. Fischer, A. Hartley, L. Permberton, R. Power, and D. Scott. A Support Tool for Writing Multilingual Instructions. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-95), Monteal, Canada, pp. 1398-1404, August 1995.

[25] D. Scott. NLG Tools to Support Technical Authors and Translators. Unpublished Information Technology Research Institute (ITRI) report. 1996.

[26] B. Levin. *English Verb Classes and Alternation, A Preliminary Investigation.* The University of Chicago Press, 1993.

[27] P. Saint-Dizier. Constructing verb semantic classes in French: Methods and associated semantic representations. In Proceedings of the International Conference on Computational Linguistics, Copenhagen, Denmark, 1996.

[28] M. Palmer, J. Rosenzweig, H. Dang, and F. Xia. Capturing syntactic/semantic generalizations in a lexicalized grammar. Presentation in working session of Semantic Tagging Workshop, ANLP-97, 1997.

[29] H. Dang, J. Rosenzweig, and M. Palmer. Associating semantic components with Levin classes. In Proceedings of Interlinga Workshop, MTSUMMIT97, San Diego, CA, October 1997.

[30] Ghostwriter.

[31] A. Joshi, L. Levy, and M. Takahashi. Tree adjunct grammars. *Journal of Computer and System Sciences,* 1975.

[32] Y. Schabes, A. Abeille, and A. Joshi. Parsing strategies with ``lexicalized'' grammars: Application to Tree Ad joining Grammars. In Proceedings of the 12th International Conference on Computational Linguistics (COLING'88), Budapest, Hungary, August 1988.

[33] Y. Schabes. Mathematical and Computational Aspects of Lexicalized Grammars. PhD thesis, Computer Science Department, University of Pennsylvania, 1990.

[34] M. Elhadad. Using Argumentation to Control Lexical Choice: A Functional Unification Implementation. PhD Thesis, Columbia University, 1992.

[35] M. Elhadad, K. McKeown, and J. Robin. Floating Constraints in Lexical Choice. *Computational Linguistics*, 23(2): 195--239, 1997.

[36] K. Vander Linden. Speaking of Actions: Choosing Rhetorical Status and Grammatical Form in Instructional Text Generation. PhD Thesis, University of Colorado, 1993.

[37] K. Vander Linden and J. H. Martin. Expressing Rhetorical Relations in Instructional Text: A Case Study of the Purpose Relation. Computational Linguistics, 21(1): 29--57, 1995.

[38] L. Kosseim and G. Lapalme. Choosing Rhetorical Relations in Instructional Texts: The Case of Effects and Guidances. In Proceedings of the 5th European Workshop on Natural Language Generation, May 1995.

[39] USAF. Organizational Maintenance Job Guide (Fuel System Distribution, USAF Series F16C/D Aircraft). 1988.

[40] Reader's Digest. *Reader's Digest New Complete Do-It-Yourself Manual.* 1991.

[41] E. Reiter. Has a consensus NL generation architecture appeared, and is it psycholinguistically plausible? In Seventh International Workshop on Natural Language Generation, pp. 163--170, June 1994.

[42] E. Hovy. Planning coherent multisentential text. In ACL, pp. 163--169, 1988.

[43] B. Grosz and C. Sidner. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12:175--204, 1986.

[44] R. Cohen. Analyzing the structure of argumentative discourse. *Computational Linguistics*, 13(1-2), pp. 11 -24, 1987.

[45] K. McKeown. *Text generation. Using discourse strategies and focus constraints to generate natural language text.* Cambridge University Press, 1985.

[46] W.A. Woods. What's in a link: Foundations for Semantic Networks. In D. G. Bobrow and A. M. Collins (eds), *Representation and Understanding: Studies in Cognitive Science*, Academic Press, New York, 1975.

[47] J. Moore and C. Paris. Planning text for advisory dialogues: Capturing intentional and rhetorical information. *Computational Linguistics*, 19(4), pp. 651--695, 1993.

[48] A. Cawsey. *Explanation and Interaction: The Computer Generation of Explanatory Dialogues.* MIT Press, 1992.

[49] J. Moore. *Participating in Explanatory Dialogues*. MIT Press, 1995.

[50] W. Wahlster, E. André, S. Bandyopadhyay, W. Graf, and T. Rist. WIP: The coordinated generation of multimodal presentations from a common representation. In Oliviero Stock, John Slack, and Andrew Ortony, editors, *Computational Theories of Communication and their Applications*. Berlin: Springer Verlag, 1991.

[51] J. Lester and B. Porter. Developing and empirically evaluating robust explanation generators: the KNIGHT experiments, 1995.

[52] R. Kittredge, T. Korelsky, and O. Rambow. On the need for domain communication knowledge. *Computational Intelligence*, 7(4), pp. 305--314, 1991.

[53] D. Appelt. Planning English Sentences. Cambridge University Press, Cambridge, England, 1985.

[54] R. Dale and N. Haddock. Content determination in the generation of referring expressions. *Computational Intelligence*, 7(4), pp. 252--265, 1991.

[55] E. Reiter. A new model of lexical choice for nouns. *Computational Intelligence,* 7(4), pp. 240--251, 1991.

[56] E. Reiter and R. Dale. A fast algorithm for the generation of referring expressions. In Proceedings of COLING, pp. 232--238, 1992.

[57] S. Shieber, G. van Noord, F. Pereira, and R. Moore. Semantic-head-driven generation. *Computational Linguistics*, pp. 16:30--42, 1990.

[58] G. Yang, K. McCoy, and K. Vijay-Shanker. From functional specification to syntactic structures: systemic grammar and tree-adjoining grammar. *Computational Intelligence*, 7(4), pp. 207--219, 1991.

[59] S. Prevost and M. Steedman. Generating contextually appropriate intonation. In Proceedings of the Sixth Conference of the European Chapter of ACL, pages 332--340, Utrecht, 1993.

THIS PAGE INTENTIONALLY LEFT BLANK