AD-A254 795

RL-TR-92-57
In-House Report
April 1992

(2)

PROCEEDINGS OF JOINT RL/AFOSR WORKSHOP ON INTELLIGENT INFORMATION SYSTEMS

Raymond A. Liuzzi and Abraham Waksman

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700

RL-TR-92-57 has been reviewed and is approved for publication.

APPROVED:

SAMUEL A. DINITTO, JR., Chief
Software Technology Division

FOR THE COMMANDER:

JOHN A. GRANIERO
Chief Scientist for C3

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	April 1992	In-House

4. TITLE AND SUBTITLE
PROCEEDINGS OF JOINT RL/AFOSR WORKSHOP ON INTELLIGENT INFORMATION SYSTEMS

5. FUNDING NUMBERS
PE - 62702F
PR - 5581
TA - 27
WU - TK

6. AUTHOR(S)
Raymond A. Liuzzi, Abraham Waksman

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Rome Laboratory (C3CA)
Griffiss AFB NY 13441-5700

AFOSR
Bolling AFB WA 20332

8. PERFORMING ORGANIZATION REPORT NUMBER
RL-TR-92-57

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Rome Laboratory (C3CA)
Griffiss AFB NY 13441-5700

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
Rome Laboratory Project Engineer: Raymond Liuzzi/C3CA(315) 330-3577

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

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The RL/AFOSR Intelligence Information System Workshop (IIS) brought together researchers in the field of Artificial Intelligence and Data Bases. Represented were various government agencies, leading university groups and government contractors. The workshop agenda included an integrated set of tutorial presentations, individual research thrusts and panel discussions, general agreement was reached that Intelligent Information Systems will play a major role in future C3I application areas.

14. SUBJECT TERMS
Computers, data base, Artificial Intelligence

15. NUMBER OF PAGES
404

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	U/L

TABLE OF CONTENTS

CHAPTER 1

AGENDA

ATTENDEE LIST

APPENDIXES

A: "KNOWLEDGE-BASE SYSTEMS: A TASK ORIENTED OVERVIEW
B. CHANDRASEKAVAN, OHIO STATE UNIVERSITY

B "TIMINGISSUES IN REAL-TIME INFORMATION SYSTEMS"
JANE W.S. LIU, UNIVERSITY OF ILLINOIS

C. "RESEARCH IN DEDUCTIVE DATABASES"
JACK MINKER, UNIVERSITY OF MARYLAND

D. "INTELLIGENT DATABASES FOR PLANNING AND SCHEDULING"
TIM FINN, UNIVERSITY OF MARYLAND, BALTIMORE COUNTY

E. "EXTRACTING RULES FROM SOFTWARE FOR KNOWLEDGE-BASES"
NOAH S. PRYWES, UNIVERSITY OF PENNSYLVANIA

F. "HETEROGENOUS KNOWLEDGE BASES"
GIO WIEDERHOLD, DARPA

G. "SIMS: SERVICES AND INFORMATION MANAGEMENT FOR DECISION SUPPORT", YIGAL ARENS, UNIVERSITY OF SOUTHERN CALIFORNIA

H. "CASUAL PROCESS MODELING"
B. CHANDRASEKAVAN, OHIO STATE UNIVERSITY

I "COOPERATIVE ANSWERING: THEORY AND PRACTICE"
JACK MNKER, UNIVERSITY OF MARYLAND

J. "COBASE: A COOPERATIVE DISTRIBUTED DATABASE"
WESLEY CHU, UNIVERSITY OF CALIFORNIA, LOS ANGELES

K. "MONTOTONE DATABASE"
JANE W.S. LIU, UNIVERSITY OF ILLINOIS, URBANA IL

L. "TEMPORAL REASONING FOR REAL APPLICATIONS"
MARK BODDY, HONEYWELL SRC

M. "PROGRAMING PARALLEL ARCHITECTURES FOR KNOWLEDGE-BASED APPLICATIONS", CHRISTINE MONTGOMERY, LANGUAGE SYSTEMS, INC, JEAN-LUC GAUDIOT, UNIVERSITY OF CALIFORNIA

Chapter I

1991 AFOSR/RL Workshop on Intelligent Information Systems (IIS)

The IIS workshop brought together researchers in the field of Artificial Intelligence and Data Bases from a number of various sources. Represented were various government agencies that included Air Force, Army, Navy, Darpa, and NASA, several university research groups were represented, and finally government contractors were represented.

The workshop agenda included a tightly integrated set of tutorial presentations, individual research thrusts in the IIS area, and panel discussions. The AFOSR research program in the area was represented by several by a series of talks that concentrated on an integrated research program in robust, cooperative, and uncertain knowledge-bases. Rome Labs cooperative program with Darpa in knowledge-base planning and scheduling was presented to the research group assembled and provided an application scenario where many of the research thrusts could applied.

Those attended agreed that the workshop was extremely helpful in understanding Air Force problems in the information area. The details of the Darpa/RL Planning and Scheduling Initiative defined specific data/knowledge problems in the large application areas and focused on possible research areas.

A lively two days of interchange was culminated by a panel summarizing the workshop and recommending future directions.

The conclusions reached included:

> unanimous support that the workshop should be held again.
>
> need exists to establish a consortium aimed at developing a rich knowledge base to be made available for community test and evaluation.
>
> ways to apply technology of Artificial Intelligence and intelligent data bases must be explored and expanded.

Possible areas of research include:

mediating databases
cooperating databases
approximate, incomplete, uncertain computation
buggy databases
reasoning about information
performance, storing information issues
new information language
high performance AI/DB paradigms
general purpose problem solvers

Other notions regarding the workshop included:

A panel should immediately immediately follow each topical area to maximize discussion of the area and gain specific recommendations.

Expand list of invites.

AGENDA

*************************** FINAL PROGRAM *******************************

INTELLIGENT INFORMATION SYSTEM WORKSHOP

The following is the program for the Workshop on Intelligent Information Systems, jointly sponsored by AFOSR and Rome Laboratory. The workshop will be held at Rome Labs, Building 3 C3 Conference Room 1, Griffiss Air Force Base, New York, on October 22,23 1991.

Tuesday October 22, 1991

7:15 - 8:00	Registration*: Front Lobby Building 3, Griffiss AFB, NY
8:00 - 8:15	Welcome (AFOSR/RL) - Dr. Fred Diamond - Rome Labs
8:15 - 9:45	"Two Decades of Knowledge-Based Systems: A Task-Oriented Overview", B. Chandrasekaran, Ohio State University
9:45 - 10:00	Break
10:00 - 11:30	"Timing Issues in Real-Time Information Systems", Jane W.S. Liu, University of Illinois
11:30 - 12:45	Working Lunch: "Overview of DARPA/RL Knowledge-Based Planning and Scheduling Initiative" Donald Roberts, Rome Laboratory
12:45 - 2:15	"Cooperative and Informative Answers for Deductive Databases", Jack Minker, University of Maryland
2:15 - 2:30	Break
2:30 - 4:00	"Probabilistic Models of Retrieval", Bruce Croft, University of Massachusetts
4:00 - 4:15	Break
4:15 - 4:45	"General Approach to Schema Merging" Peter Buneman, University of Pennsylvania
6:00 - 7:00	Social Hour - Reception Quality Inn Red Coach Restaurant Cornor S. James Street and Erie Blvd. Rome, New York

Wednesday Oct 23

8:00 - 8:30	"Intelligent Databases for Planning and Scheduling" Tim Finin, Unisys Corporation
8:30 - 9:00	"Extracting Rules from Software for Knowledge Bases" Noah Prywes, U. of Pennsylvania
9:00 - 9:30	"Heterogenous Knowledge Bases", Gio Wiederhold - Darpa

9:30 - 9:45	Break	
9:45 - 10:15	"Services and Information Management for Decision Systems (SIMS)"	Yigal Arens, University of Southern California
10:15 - 10:45	"Ariel Database Rule System"	Eric Hanson, WL/Wright State Univ.
10:45 - 11:15	"Knowledge Organization"	B. Chandrasekaran, Ohio State Univ.
11:15 - 11:45	"Intelligent Data Bases"	Jack Minker, Univ. of Maryland
11:45 - 1:00	Working Lunch "COBASE: Cooperative Distributed Database"	Wesley Chu, UCLA
1:00 - 1:30	"Retrieval with Partial Information"	Bruce Croft, Massachusetts Univ.
1:30 - 2:00	"Monotone Databases"	Jane Liu, Univ. of Illinois
2:00 - 2:30	"Representing and Reasoning About Temporal Information"	Mark Boddy, Honeywell Infomation Systems
2:30 - 3:00	"Highly Programmable Architectures for Knowledge Base and Natural Language Processing"	Christine Montgomery, Language Systems Inc., Jean-Luc Gaudiot, Univ. of Southern California
3:00 - 3:30	" Data Base Reasoning in Engineering Design"	Katia P. Sycara, Carnegie Mellon University
3:30 3:45	Break	
3:45 - 4:15	Panel Discussion "Future Directions of Intelligent Information Systems" (All Participants Provide 2-4 Minute Position Statement)	
4:15 - 4:45	Wrapup	

* There will be a minimal charge for coffee and refreshments - $3.00

Also, there will be a $5.00 charge, for hors d'oeuvres, for those attending social hour with pay as you go bar.

Lodging Information:

Beeches-Paul Revere Lodge
Route 26
Rome, New York
315-337-1775

Consort Horizon Airport Inn
Oneida County Airport
Oriskany, New York
315-736-3377

Quality INN
Erie Blvd. & S. James Street
Rome, New York
315-336-4300

Radisson Hotel Utica Center
Genesee Street
Utica, New York
315-797-8010

ATTENDEE LIST FOR 1991 AFOSR/RL

INTELLIGENT INFORMATION SYSTEM WORKSHOP

Bill Baker
WL/AAA-1
Wright Patterson AFB,
Ohio 45433

Yale Smith
173 Proctor Blvd.
Utica, New York 13501
(315) 735-9107

Fred Diamond
RL/CA
Griffiss AFB, NY 13441

Nort Fowler
RL/C3CA
Griffiss AFB, NY 13441

Robert Ruberti
RL/C3CA
Griffiss AFB, NY 13441

Donald Roberts
RL/C3CA
Griffiss AFB, NY 13441

Dr. Som Karamchetty
AMCLD-PM
LABCOM
2800 Powder Mill Rd.
Adelphi, MD 20783-1145
skaramch@alexandria-emhl.army.mil

Raymond Liuzzi
RL/C3CA
Griffiss AFB, NY 13441
liuzzir@lonex.rl.af.mil

Jennifer Skidmore
RL/C3CA
Griffiss AFB, NY 13441

Michael McHale

Prof. Jack Minker
University of Maryland
CIS Dept.
College Park, MD 20742
minker@cs.umd.edu

Prof Jane Liu
University of Illinois
CIS Dept.
1304 West Springfield Ave
Urbana, IL 61801
janeliu@a.cs.uiuc.edu

Prof. Bruce Croft
University of Massachusetts
Amherst, MA 01003

Prof. Chandrasekaran
Ohio State University
Columbus, Ohio 43210
chandra@cis.ohio-state.edu

Prof. Noah Prywes
University of Pennsylvania
Philadelphia, PA 19104

Eric N. Hanson
Wright State
513-259-1397
ehanson@valhalla.cs.wright.edu

Ms Leah Wong
Code 441
Naval Ocean System Center
San Diego, CA 92152-5000
wong@cod.nosc.mil
(619)-553-4127

Mark Boddy
Honeywell SRC
3660 Technology Drive
P.O. Box 1361
Minneapolis, MN 55440
boddy@src.honeywell.com

Tim Finin
Univ of Maryland, Balt County
Baltimore, MD 21228
(310)-455-3522, 3969 (fax)
finin@cs.umbc.edu

Yigal Arens

RL/C3CA
Griffiss AFB, NY 13441

Prof. Peter Buneman
Dep. of CIS
University of Pennsylvania
Philadelphia, PA 19104

Prof. Susan Davis
Dep. of CIS
University of Pennsylvania
Philadelphia, PA 19104-6389

Patrick McCabe
RL/IRDD
Griffiss AFB, NY 13441

Dr Katia Sycara
CMU, The Robotic Institute
Pittsburgh, PA 15213
isl1.ri.cmu.edu

Robert Stumberger
Language Systems Inc.
6269 Variel Avenue, Suite F
Woodland Hills, CA 91367

Louis Hoebel
RL/C3CA
Griffiss AFB, NY 13441

Craig Knoblock
USC/ISI
4676 Admiralty Way
Marina Del Rey, CA 90292

Stuart Shapiro
SUNY at Buffalo
Dept. of Computer Science
Buffalo, New York 14260

USC/ISI
4676 Admiralty Way
Marina DelRey, CA 90292
arens@isi.edu

Gio Wiederhold
DARPA/CSTO
3701 North Fairfax Drive
Arlington, VA 22203-1714
gio@darpa.mil

Ted Kral
BBN Systems and Technologies
4015 Hancock Street
San Diego, CA 92110
TKRAL@sd-vax.bbn.com

LTC. Robert Powell
ONR Div. of Engr. Services
800 N. Quincy Street
Arlington, VA 22017
(703)-696-4407
powell@itd.nrl.navy.mil

Cristine Montgomery
Language Systems Inc.
6269 Variel Avenue, Suite F
Woodland Hills, CA 91367
chris@priam.usc.edu

Bruce Berra
Syracuse University
121 Link Hall
Dept. of Elect. and Comp. Eng.
Syracuse, NY 13244
berra@cut.syr.edu

Abe Waksman
AFOSR
Bolling AFB,
D.C. 20332
waksman@isi.edu
(202)-767-5028
DSN 297-5028

Wesley Chu
UCLA
Los Angeles, CA
chu@cs.ucla.edu

shapiro@cs.buffalo.edu

Mi' McNeil
BB. Systems and Technologies
4051 Hancock Street
San Diego, CA 92110
mmcneil@sd-vax.bbn.com

Prof. Jon Doyle
MIT
Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139
doyle@zermatt.lcs.mit.edu

John F. Lemmer
CTX Incorporated
200 Liberty Plaza
Rome,
New York 13440
jlemmer@npac.syr.edu

Jim Jacobs
ISX Corporation
4: Park Terrace Drive
Westlake Village, CA 91361
jjacobs@isx.com
(818) 706-2020

Lee D. Erman
Cimflex Teknowledge Corp.
Knowledge System Division
1810 Embarcadero Road
P.O. Box 10119
Palo Alto, CA 94303
(415)-424-0500 ext 422
lerman@teknowledge.com

Alan Lazzara
Sterling Corporation
Beeches Technical Campus
Rome, New York 13440
lazzara@nova.npac.syr.edu

Christopher Owens
Univ of Chicago
owens@gargoyle.uchicago.edu

Nancy Roberts
RL/C3CA
Griffiss AFB, NY 13441

John Crowter
RL/C3CA
Griffiss AFB, NY 1:441

Jean-Luc Gaudiot
USC
EEB-336
Los Angeles, CA 90089-2562
gaudiot@usc.edu
(231) 740-4484

Ed Walker
BBN Systems
ewalker@bbn.com

Ed Huff
Ames Research Center
Moffit Field
819 Cathedral
Sunnyvale, CA 94087
(415)-604-4870

APPENDIX A

APPENDIX A

Knowledge-Based Systems: A Task-Oriented Overview

B. Chandrasekaran
Laboratory for AI Research
The Ohio State University
Columbus, OH 43210

Email: chandra@cis.ohio-state.edu

Goals for the Talk

1. **TYPES OF INFORMATION PROCESSING PROBLEMS AND HOW APPROPRIATE AI IS FOR THEIR SOLUTIONS**
2. **TYPES OF AI ARCHITECTURES THAT ARE UNDER DISCUSSION IN AI**
3. **TASKS, METHODS AND KNOWLEDGE**
4. **ARCHITECTURES FOR KNOWLEDGE-BASED SYSTEMS**

INFORMATION PROCESSING PROBLEM TYPES AND AI

1. **TYPES OF INFORMATION PROCESSING PROBLEMS AND HOW APPROPRIATE AI IS FOR THEIR SOLUTIONS**
2. **TYPES OF AI ARCHITECTURES THAT ARE UNDER DISCUSSION IN AI**
3. **TASKS, METHODS AND KNOWLEDGE**
4. **ARCHITECTURES FOR KNOWLEDGE-BASED SYSTEMS**

Laboratory for AI Research, The Ohio State University

Types of IP Problems and Appropriateness of AI

Information Processing Task and Computer Programs

Task Types:

Type 1

- Closed From Algorithms Exist
- Information Needed for Running the Algorithms Available in the Domain
- Time and Space Requirements are Tractable

E.G.:

FEM, Multiplication Routines, Sorting Programs

Laboratory for AI Research, The Ohio State University

Type 2 Problems: Approriate for AI Techniques

Type 2

* **Algorithms of Type 1 are not Applicable**

  Because, e.g., They are not Known,
  Information Needed not Available,
  or the Algorithms are not tractable,
  (Takes Too Long, e.g.)

* **But in the Domains of interests, There Exist Human Problem Sovers (Experts) who Routinely Solve Versions of the Problem Well Enough**
* **Qualitative, in General**
* **Humans May Do it by a *Recognition Process* or a *Deliberative Process***
  * **e.g.: Diagnosis, Planning, Design in specific domains**

Type 3 Problems

Type 3

* **Not Types 1 or 2 but a Solution May be Possible with AI Methods of Search.**

  (Heuristic Search in Large Spaces.)
  ---- Traveling Salesman Problem, Search for Patterns in Large Databases

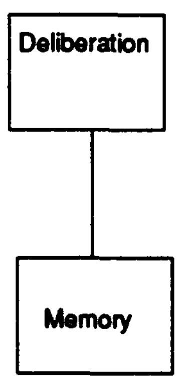
Will Concentrate in this Talk on Type 2 Tasks as Appropriate for Expert System Approach

DELIBERATIVE VS RECOGNITION ARCHITECTURES

1. **TYPES OF INFORMATION PROCESSING PROBLEMS AND HOW APPROPRIATE AI IS FOR THEIR SOLUTIONS**
2. **TYPES OF AI ARCHITECTURES THAT ARE UNDER DISCUSSION IN AI**
3. **TASKS, METHODS AND KNOWLEDGE**
4. **ARCHITECTURES FOR KNOWLEDGE-BASED SYSTEMS**

Laboratory for AI Research, The Ohio State University

Metaphors for Problem Solving in AI



Reasoning

- Knowledge
- Rules of Inference

Problem Solving

- Initial state
- Goal state
- Operators
- Search

- Networks
- Frames
- Cases

- Connectionist nets

Laboratory for AI Research, The Ohio State University

Metaphors for Problem Solving in AI: Contd

Deliberation: Explicit operation of manipulating knowledge to produce new states of knowledge

* Logic -- Operation on predicates and clauses by means of Inference Rules (normally soundness is a major concern).

* Problem solving as search in a State Space, where each state corresponds to a knowledge state, and operators change states, goal is to achieve knowledge corresponding to goal state.

Duality between the logic view and the search view

-- Need for search in the logic view as well -- many inferences can be made, few of which lead to the goal
-- State expansion operators in the state space view can be deemed to be a kind of inference rule.

Mycin and R1 as Search Systems

All knowledge systems have to have knowledge to

* Set up alternatives (Elements of the initiial state)

* Generate new problem states (inference rules in logic, operators in GPS)

* Control Search.

The Diagnostic Part of Mycin:

Initial State --- Observations, but no knowledge of causative organism

Goal state -- Knowledge of causative organism

Problem space: The hierarchy of organisms

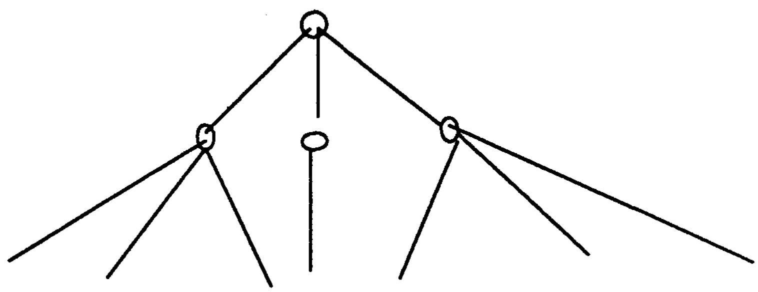
Operators for Statte Expansion : "Subclass of"

Search control knowledge: heuristics about what to consider, when to prune space of hypotheses, etc.

R1 as Problem Space Searcher

R1 as a Configurer:

Initial state : A list of components, some connceted
Goal state : All components connected appropriately
Operators: Various connection operators
Search control knowledge : Local configuration tests



Non-deliberative phenomena

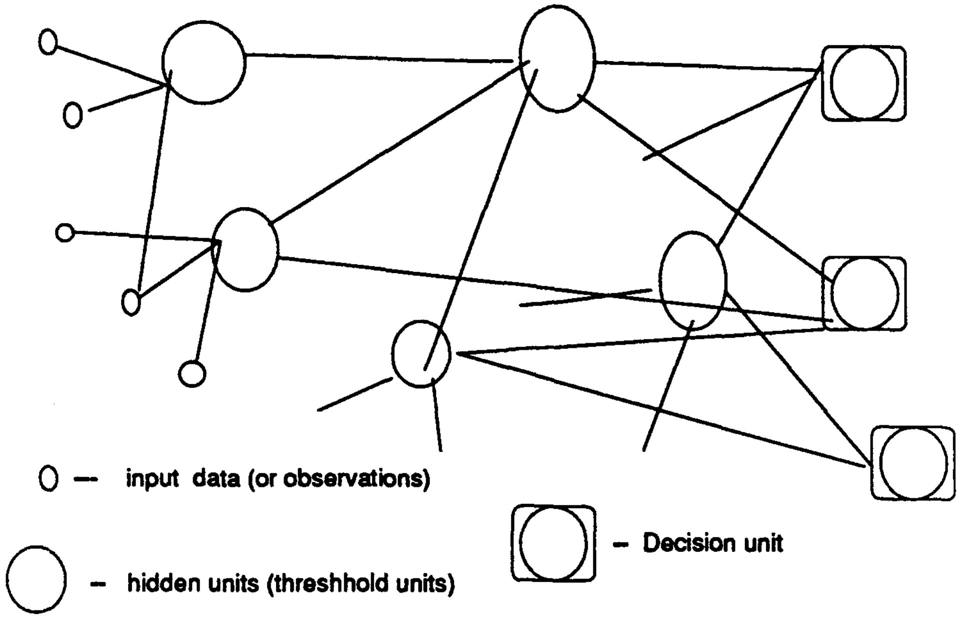
Memory : "One-shot" phenomena : behavior of interest is produced as "one mental cycle" in humans

-- matching, retrieval, classification, recognition common task-level descriptions

--- Case-based reasoning, connectionist (neural) networks

--- Symbolic versus "continuous" mathematics representations

A connectionist Network



Laboratory for AI Research, The Ohio State University

Connectionist Networks -- Continued

This style of network can be seen as doing evidence combination at multiple levels (hidden units correspond to intermediate abstractions)

The input data can be observations of a mechanical system, the hidden units can be intermediate hypothesis , and malfunctioons can be decision units.

Quite independent of the connectionist nature of the architecture, we can say that the system is combining evidence from groups of observations to form intermediate hypotheses, which in turn are combined to evaluate evidence for various malfunction hypotheses.

The Connectionist system then does diagnosis by matching and evidence abstraction.

Laboratory for AI Research, The Ohio State University

Case-Based Reasoning

Problem solving as Retrieving a "similar" problem's solution from memory and "tweaking" it.

Some sort of associative matching, based on indexes to the cases in memory

Tweaking still involves deliberative search-like activity

-- one still has to consider a number of possible modifications, select some, possibly backtrack, etc.

-- Thus memory does a "best match" retrieving, while deliberation does additional problem solving

-- We need task-based theories of indexing

 -- how do we index cases? What is the connection between kinds of tasks and the kinds of indices? Some work has been done in this connection on planning (Hammond), diagnosis (Kolodner), and design (Goel).

TASKS

1. TYPES OF INFORMATION PROCESSING PROBLEMS AND HOW APPROPRIATE AI IS FOR THEIR SOLUTIONS
2. TYPES OF AI ARCHITECTURES THAT ARE UNDER DISCUSSION IN AI
3. **TASKS, METHODS AND KNOWLEDGE**
4. ARCHITECTURES FOR KNOWLEDGE-BASED SYSTEMS

Knowledge Level VS Symbol Level in AI

* Newell's *Knowledge Level Vs Symbol Level* Distinction

* Too much of the discussion in Knowledge-Based System is at the Symbol Level, i.e., at the level of implementation details that obscure the structure of the task that is being solved.

* The language of tasks, and what it takes to accomplish them (methods) is useful to make sure that the mechanisms fit the requirements of the task.

Phenomena at the Symbol Level

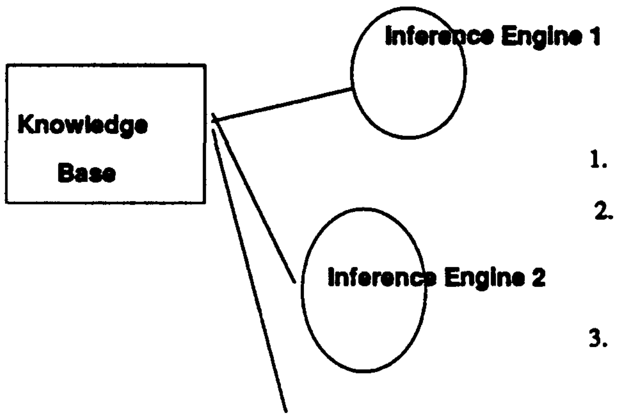
* Rules
* Frames
* Connectionist networks
* ATMS
* Nexpert
* Kee
* Problem Spaces
* ....

*All of these make some degree of commitments to an implementation formalism

*They all have a place, but after the structure of the task is delineated.

*Often, how to organize knowledgeand what kinds of knowledge are needed are obscured by an overemphasis on the symbol level

Problems with Traditional Picture of KBS's

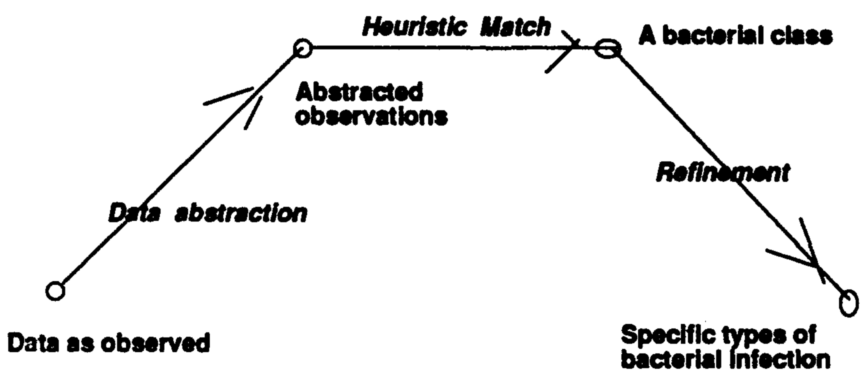


1. Knowledge separated from use
2. No theory of how the task Constrains the type of knowledge in the KB and the type of task-level control needed
3. Often, task-level control is embedded in the rules and invisible. Instead of control "emerging ," it is simply hidden and obscured..

Laboratory for AI Research, The Ohio State University

Mycin as Heuristic Classifier

Clancey analyzed the diagnostic part of Mycin and showed that it was performing a type of problem solving called **Heuristic Classification.**



* The Rule system is merely an implementation medium for the above inference structure.

* This structure was implicit and hidden in the rules -- had to be brought out for explanation.

Laboratory for AI Research, The Ohio State University

R1 As a Subtasker

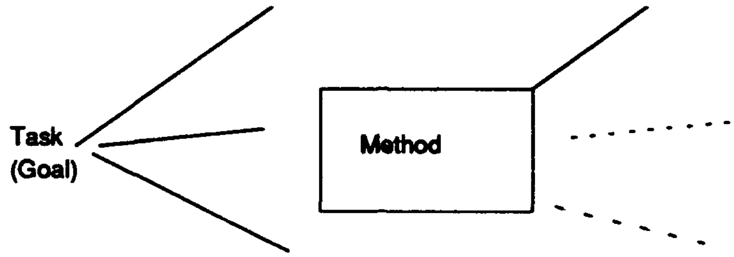
Similarly, the R1 System decomposed the design problem into
a series of subtasks.

The subtask structure was implemented in OPS-5, but was implicit.

This led McDermott and associates to study how to make
use the constraints of the task for structuring the problem solving
activity and also use those constraints for knowledge acquisition.
He launched a study of the *roles* of knowledge in various tasks.

Compiled and Deep Models

Tasks, Methods and Knowledge



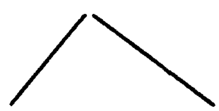
Method: Characterized by *Operators* operating on *Objects*
Plus *knowledge (Declarative or embedded) about how to organize operator application to satisfy goal*

Different methods may call for different types of knowledge

Examples of Methods

Examples:

1. Multiply two multi-digit numbers



"Logarithmic" Method ....

Extract Logarithms of input numbers

Add the two logarithms

Extract anti-logarithm

Operators underlined

Objects: Input numbers

Various results of operator application

Methods Set Up Subtasks

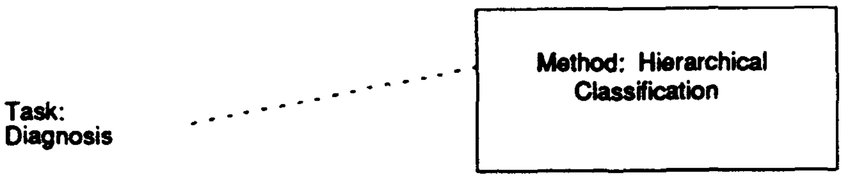
<u>Subtasks</u>: Conditions for operator application may not be satisfied

1. Operator not primitive

   e.g. "Extract logarithm" may require setting up as subtask

2. Objects needed may be missing

   e.g.



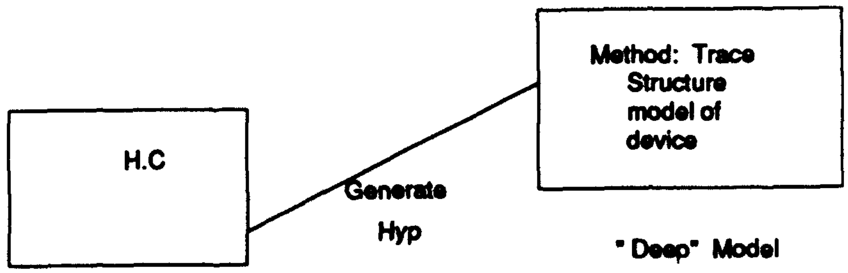
Operators: Establish Hypothesis
Refine Hypothesis

Objects: Diagnostic Hypotheses

Some of the objects may be missing, or operator preconditions may not be satisfied

"Deep" Models Used to Generate Knowledge Needed for Objects and Operators of a Method

If these objects not directly available, set up subtask of generation of diagnostic hypotheses



"Deep" Model also useful for operator application: "Establish" Hypothesis (simulation of device).

*POINT: Depth of knowledge is relatively to a method for a task.
It is not an absolute notion.

Types of Methods

Types of methods

: *Well-defined "closed-form" algorithms (tractable)*

: *So-Called "Problem-solving" methods*

Simon's Classification of Problem-solving Methods in AI

-*Problem-Space Search Methods*

-*"Reasoning" Methods*

- *Parallel, constraint-satisfaction Methods*

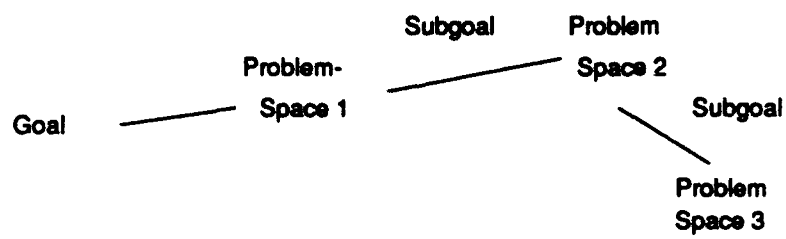
Newell : Algorithms Special Cases of Search Methods, where there are no choice points, the selections have already been made when the algorithm was composed.

* No finite, mutually excl. set of methods

Laboratory for AI Research, The Ohio State University

A Method Can Be a Problem Space: The HC and MDX Examples

In Newell's Problem-Space Framework, each problem-space set up for a goal is a method. A method can also be a name given to a sequence of pre-compiled problem-spaces.



Difft Methods call for Dif. Types of Knowledge.

Heuristic Classification (Clancey); MDX (Chandrasekaran, et al)

PS1: Space of Hypotheses

PS2: Space of Credibility Assignments

PS3: Space for Data Abstraction

Laboratory for AI Research, The Ohio State University

The Design Task

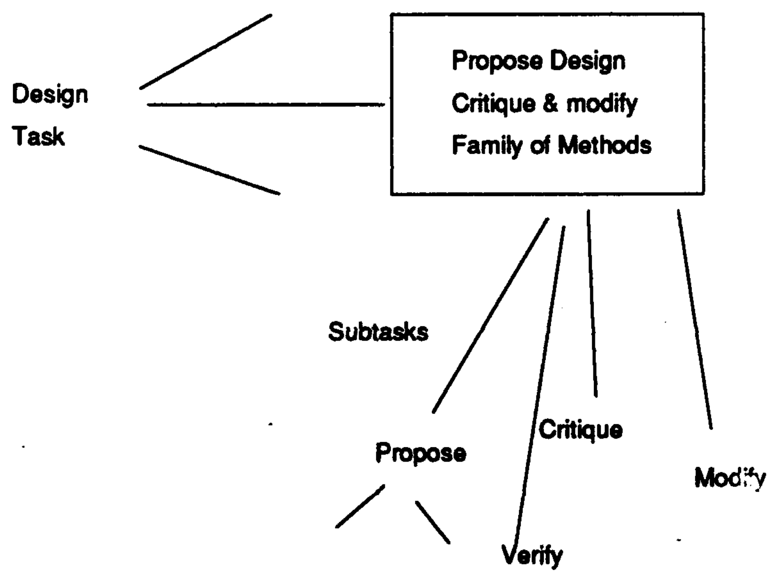
Complete specifications of a set of "primitive" components and their relations ("connections") so as to meet set of functions or goals while satisfying a number of specified constraints.

-Domain-independent character of Design
(planning, programming, mechanical design, ...)

-- Design not unitary process
-- Repertoire of strategies each using specific forms of knowledge & inferences
-- Differences in design process in different domains function of available knowledge

Laboratory for AI Research, The Ohio State University

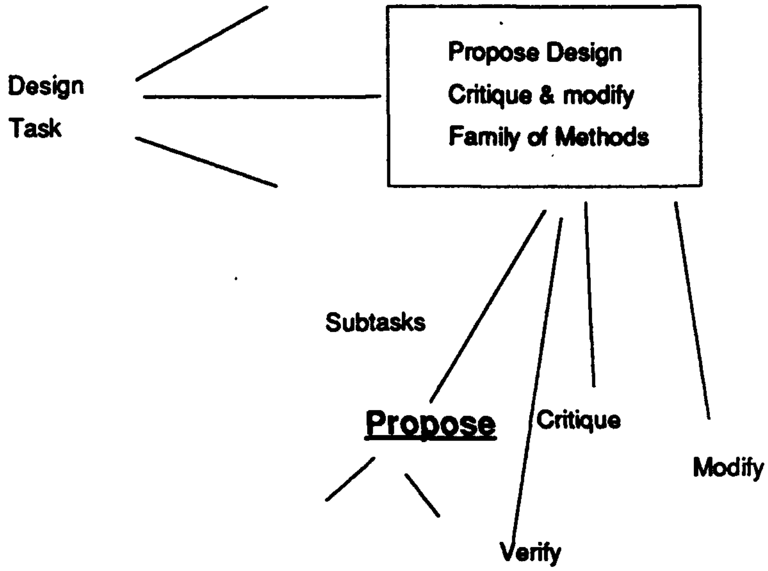
The PVCM Method Family



Numerous Ways in which subtasks can be set up & invoked & combined

Laboratory for AI Research, The Ohio State University

The PVCM Method Family- Propose Methods



Laboratory for AI Research, The Ohio State University

Methods for Proposing Design Choices

* Design Problem Decomposition/Solution composition
  - Design plans a special case
* Retrieval of cases from memory for similar problems
* Families of Optimization/Constraint satisfaction methods

Laboratory for AI Research, The Ohio State University

Decomposition Methods

Knowledge Needed

D ---------> D1, D2, ... Dn

Di's "smaller" problems (smaller search spaces)

- If Alternate Decompositions possible, then search in decomposition problem space takes place

- Repeated applications result in design hierarchies, no search

Subtasks

* Translating from specification of D to specifications

* Composing subproblem solutions into problem solutions

Laboratory for AI Research, The Ohio State University

Decomposition, Continued

Information Needed:

-- How goals/constraints of D get translated into corresponding ones for Di's.

-- How to glue solutions back

-- May be given as part of decomposition knowledge or as auxiliary problem solving

-- CRITTER System (Kelly) is a domain-specific simulation facility for generating constraints for Di's & gluing them back

Laboratory for AI Research, The Ohio State University

Task decomposition does not specify control in detail

Subproblem constraint generation may involve alternating
between design proposal & constraint generation

- propose & revise method (McDermott and Marcus)
- progressive deepening (Steier's work on algorithm synthesis)

In configuration tasks, subproblem solution already given,
problem dominated by specification generation
& solution composition

Design Plans

-- Sequence of design actions

-- Precompiled partial solution

-- Recursive application of plans

-- Incorporates *decomposition* knowledge

-- Indexed by functions or components, multiple plans possible

-- Inference: instantiate & expand (e.g. Noah)

-- Dependencies bet. parts of plan may be discovered or pre-compiled

Ubiquity of Idea of Plans in AI

Miller, Galanter & Pribram: Almost all knowledge is plans

C. Rich : Programmer's Apprentice

Soloway and Johnson: Programming expertise is accumulation of plans

Perry Miller, et al., Plans for Therapy Planning and Critiquing

Schank & Abelson, Plans for Almost Everything

Friedland, Stefik : Molgen -- Planning of Genetics Experiments

Brown & Chandrasekaran: Design expertise as design plans

Mittal: Design expertise as design plans

Design Cases For Design Proposal

METHOD 3 FOR DESIGN PROPOSING:

-- Cases (Sussman, Schank)

"All Design is Redesign"
"Critique & Modify Almost Correct Designs"

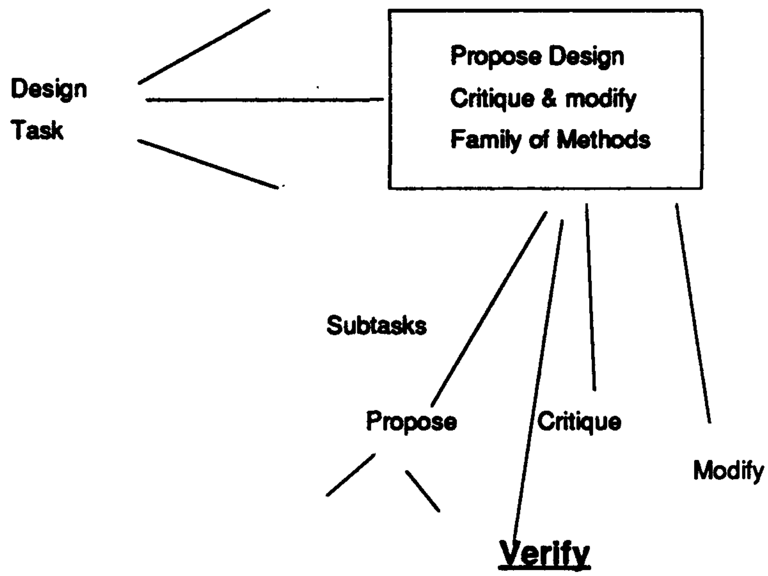
Important Issues:

-- Matching: goal priority, similarity

-- Critiquing
------ Simulation
------ Constraint Analysis (Dependency Analysis)

-- Modifying
-- Another design problem

The PVCM Method Family- Verify Methods



Laboratory for AI Research, The Ohio State University

Methods for Verification

Methods for Verification

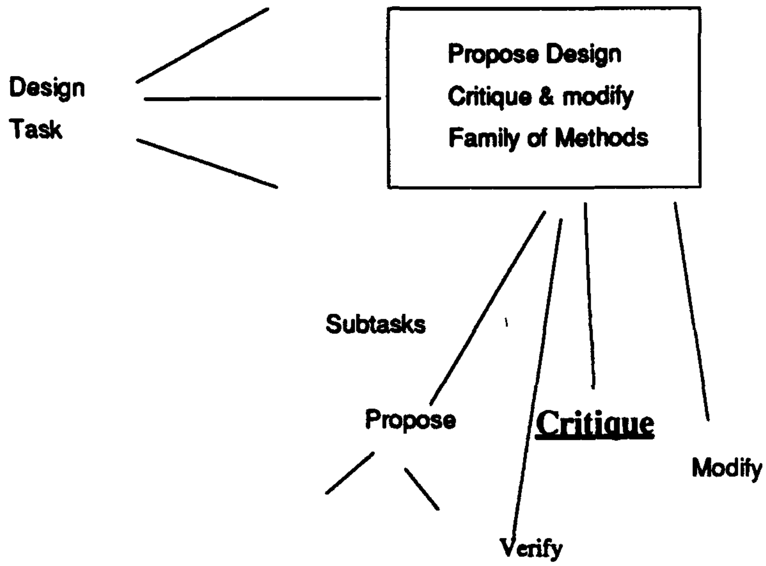
* Direct Calculation

  (Finite Element, e.g.)

* Simulation
  - Quantitative
  - Qualitative Simulation
  - Visual (Perceptual) Simulations

Laboratory for AI Research, The Ohio State University

The PVCM Method Family- - Critique Methods



Laboratory for AI Research, The Ohio State University

Critiquing Methods

Methods for Critiquing:

- A generalized version of the diagnostic problem

* Dependency Analysis (Sussman)

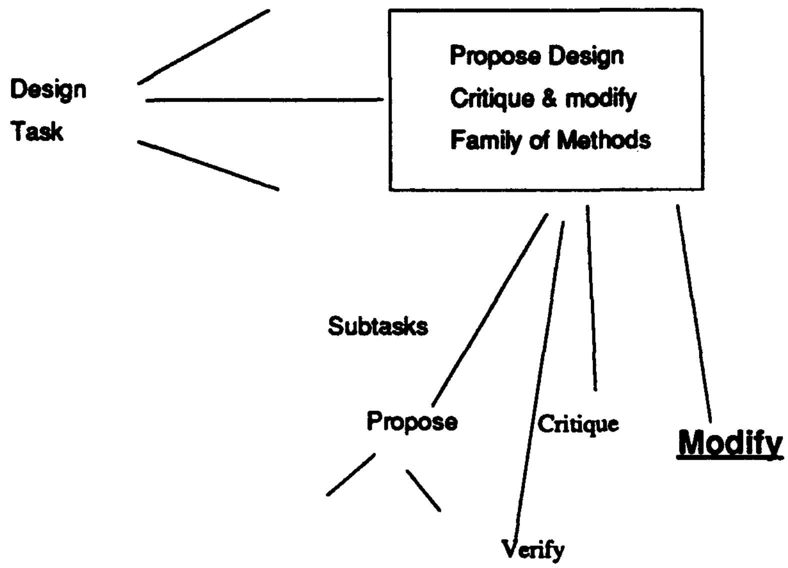
* Functional Analysis of Proposed Design (Goel)

- If design proposal is endowed with causal indices that explicitly indicate relation bet structure & function, substructures for modification can be identified

.... ?

Laboratory for AI Research, The Ohio State University

The PVCM Method Family- - Modification Methods



Laboratory for AI Research, The Ohio State University

Modification Methods

Modification

* Means-ends like methods

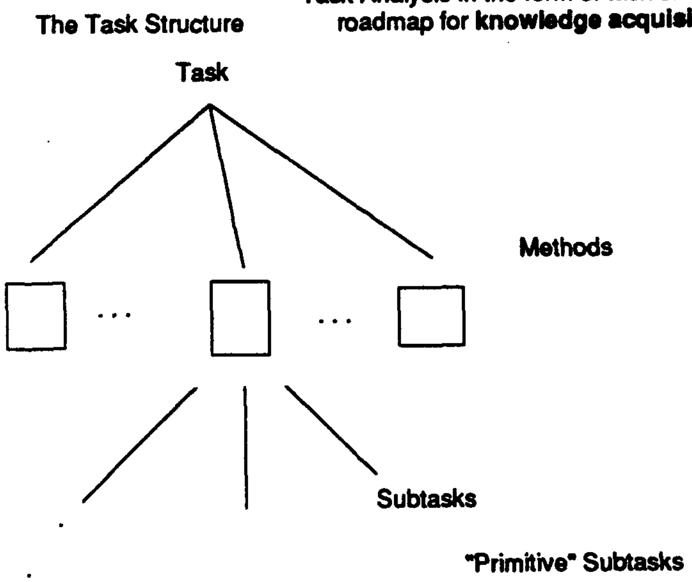
* hill-climbing for parametric (problems)

* If criticsm identified missing functions
  for which separate components should be added,
  modification can be posed as a new design problem

Laboratory for AI Research, The Ohio State University

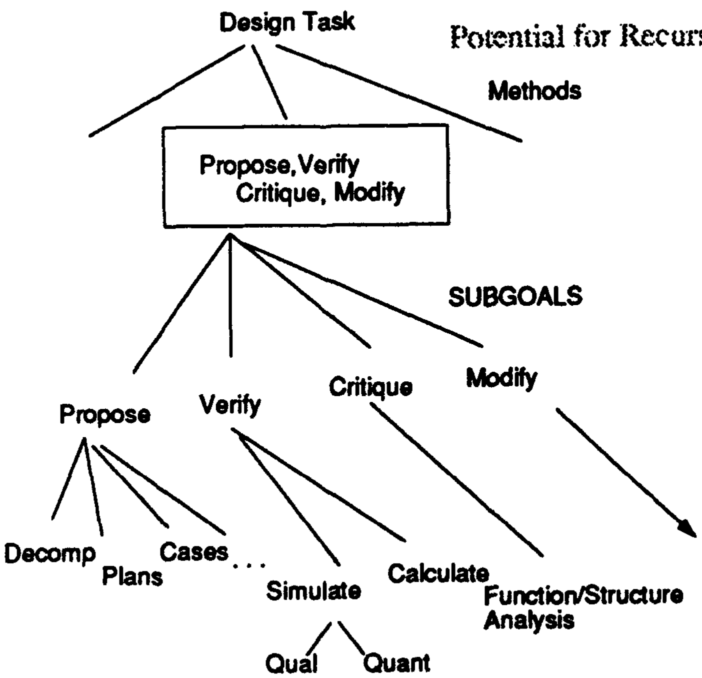
The Task Structure As An Organizing Idea

Task Analysis in the form of task structure provides clear roadmap for **knowledge acquisition**



Laboratory for AI Research, The Ohio State University

Potential for Recursion And Reuse



Laboratory for AI Research, The Ohio State University

Criteria for Choice of Methods

- Properties of Solution (Qualitative/Quantitative), degree of accuracy..)

- Propeties of Solution Process (Computational properties, ...)

- Availability of Knowledge in the Domain,
  to Support Method, or Knowledge
  to Generate Method-Specific Knowledge

- No one ideal method for design
- Not NORMATIVE, rather language
  to help us describe domains (Clancey: Knowledge Bases as Domain Models)

Laboratory for AI Research, The Ohio State University

Additional Properties of the Task Structure View

- Explains how real human expertise comes about in a tractable way : Not by access to a UNITARY algorithm for a complex task, by accumulation of knowledge to decompose tasks, and a repertoire of methods which match tasks and the domain environment

- Integration of different types of methods:
  Qualitative and quantitative

Laboratory for AI Research, The Ohio State University

KBS ARCHITECTURES

1. TYPES OF INFORMATION PROCESSING PROBLEMS AND HOW APPROPRIATE AI IS FOR THEIR SOLUTIONS
2. TYPES OF AI ARCHITECTURES THAT ARE UNDER DISCUSSION IN AI
3. TASKS, METHODS AND KNOWLEDGE
4. **ARCHITECTURES FOR KNOWLEDGE-BASED SYSTEMS**

Implications for An Architecture for Problem Solving

1. Method specific shells, pre-combined (GT's, TSA's)

2. Method-specific shells, to be dynamically chosen & combined (TIP work of Punch, BB1, ..)

3. Methods may be too much like a "big-switch"
   - -- Use methods as specification of a set of goals, but instead of scheduling a method as a unit, use a finer-grained goal scheduling architecture
   - - Implement the search-like methods in "Soar"-like architecture
   - - Todd Johnson's work on implementing GT's in SOAR

ARCHITECTURAL IMPLICATIONS
1. Method-specific shells, pre-combined (GT's, TSA's)

MORE

MORE is a tool for acquiring and using diagnostic knowledge.

Authors: Gary Kahn, Steve Nowlan and John McDermott
Carnegie-Mellon University

Forerunner: MUD expert system

Task Specification:

Given observations about symptoms, determine the hypotheses.

Kind of Knowledge:

Hypotheses, Symptoms caused by hypotheses, Tests for symptoms, and Conditions affecting the strength of links between entities.
Measure of belief and disbelief indicate link strength (-10 to +10) which can vary depending on conditions.

Control: Evaluate hypotheses by treating weights like EMYCIN certainty factors.

Special Features:

MORE actively elicits knowledge

CSRL

CSRL (Conceptual Structures Representation Language) is a tool for hierarchical classification, which is searching for hypotheses in a classification hierarchy.

Authors: Tom Bylander and Sanjay Mittal

The Ohio State University and Xerox Parc

Forerunner: MDX expert system

Task Specification:
Find the categories or hypotheses that apply to the situation being analyzed.

Kind of Knowledge: Hypotheses and associations between patterns of data and confidence in hypothesis.

Organization of Knowledge:

- Hypotheses are organized as a classification tree.

Control:

- Top-down multiple path search of the classification hierarchy (establish-refine).

- When a hypotheses is invoked, its knowledge groups are used to determine its confidence value.

Laboratory for AI Research, The Ohio State University

HYPER

HYPER (Hypothesis Matcher) is a tool for hypothesis matching, which determines the fit of a hypothesis to a situation.

Authors: Todd R. Johnson, Jack W..Smith & Tom Bylander

The Ohio State University

Forerunner: CSRL tool

Task specification:
Given a set of data, determining the confidence value of some hypothesis.

Kind of Knowledge:

Decision tables, called knowledge groups, that associate patterns of input with confidence values.

Knowledge Organization:

- A hierarchical organization of knowledge groups.
- Each knowledge group specifies how selected data and the results of its children are mapped to a confidence value.
- The root knowledge group makes the overall decision.

Laboratory for AI Research, The Ohio State University

IDABLE

IDABLE (Intelligent Data Base Language) is a tool for knowledge-directed information passing, which is inferring attributes of a data concept from conceptually related data.

Forerunner: PATREC subsystem of MDX expert system.

Author: Jon Sticklen, The Ohio State University

Task Specification:

Given a datum to retrieve, infer it from a conceptually related datum.

Kind of Knowledge:

- Data concepts, their attributes and valid values, and relations between data concepts.
- Inferences for mapping one datum to another (along some relationship).
- Temporal relationships between data.

Knowledge Organization:

- Data concepts are organized in a type hierarchy.
- Inferences are attached to the attribute to be inferred.
- Case data is temporally organized under events and temporal relationships to events.

Laboratory for AI Research, The Ohio State University

PEIRCE

PEIRCE is a tool for abductive assembly, which is finding the best explanation for a set of data.

Authors: William F. Punch Michael C. Tanner & John Josephson
The Ohio State University

Forerunner: RED expert system

Task Specification:

Given a set of data to explain and a set of hypotheses to explain the data, find the best composite hypothesis that explains the data.

Kind of Knowledge:

Plausability of hypotheses and the data that each hypothesis explains.

Knowledge Organization:

Assemblers indicate what subtask to do next.

Laboratory for AI Research, The Ohio State University

HERACLES

HERACLES (Heuristic Classification Shell) is a tool for performing diagnosis by heuristic classification.

Author: William J. Clancey, Stanford University

Forerunners: MYCIN expert syste,NEOMYCIN expert system

Task Specification:

Given a set of data, determine a classification; HERACLES is conceptualized as a diagnosis tool.

Kind of Knowledge:

Findings, Hypotheses, and Relations between them (generalizations, taxonomic, causal, associational).

Knowledge Organization:

- Hypotheses are organized as a classification hierarchy.
- EMYCIN production rules for determining confidence in hypostheses.
- Causal structures, Generalization structures for findings, etc.

Laboratory for AI Research, The Ohio State University

HERACLES, CONTD

Control:

A structured collection of task procedures (each task is implemented by metarules) including:

- Generate-questions (probe for additional information to suggest new hypotheses),
- Group-and-differentiate (focus on a general process consistent with hypotheses),
- Explore-and-refine (top-down search from some hypothesis in the classification hierarchy),
- Process-finding (trigger hypotheses that explain a finding).

Special Features:

HERACLES is intended to be a complete diagnosis tool.

Laboratory for AI Research, The Ohio State University

SALT

SALT is a tool for constructing expert systems that perform constraint-satisfaction tasks.

Author: Sandra Marcus and John McDermott

Carnegie-Mellon University

Task Specification: "Propose-and-Refine"

Given some parameters with specified values specified, assign values to all parameters satisfying constraints.

King of Knowledge:

- Procedures for calculating values of parameters
- Constraints on parameters
- Fixes for failed constraints

Knowledge Organization:

- Procedures form an acyclic graph
- A fix revises a parameter used to calculate the incorrect parameter

SALT, CONTD

Control:

- Parameter values are calculated as soon as possible, i.e., when a procedure's inputs are available and its preconditions satisfied.

- Constraints may be checked either as soon as a parameter is calculated, or after all parameters are calculated.

- If a constraint fails, alternative fixes are applied to previously calculated parameters until the constraint is satisfied.

-All other values depending on the fixed parameter are recalculated.

Special Features:

SALT provides a knowledge acquisition facility for interviewing experts and helping them analyze the knowledge base.

Generic Tasks

* BUILDING BLOCKS FOR COMPLEX KNOWLEDGE-BASED PROBLEM SOLVING TASKS
* EXAMPLES :
  * HIERARCHICAL CLASSIFICATION
  * SYMBOLIC HYPOTHESIS ASSESSMENT
  * ROUTINE DESIGN BY HIERARCHICAL PLAN SELECTION
  * FUNCTIONAL SIMULATION
  * ABDUCTIVE ASSEMBLY
  * DATA ABSTRACTION
* EACH IS A "GENERIC" TASK, WITH CHARACTERISTIC KNOWLEDGE AND STRATEGIES

Laboratory for AI Research, The Ohio State University

Generic Tasks, CONTD

COMPLEX TASKS SUCH AS DIAGNOSIS CAN BE PERFORMED BY DECOMPOSING IT INTO GENERIC TASKS AND THEIR RESULTS COMPOSED

DIAGNOSIS AS ABDUCTIVE TASK

— ABDUCTION BY DECOMPOSITION INTO HYPOTHESIS GENERATION AND *ABDUCTIVE ASSEMBLY*

— HYPOTHESIS GENERATION BY *HIERARCHICAL CLASSIFICATION* OVER A MALFUNCTION HIERARCHY

— HIERARCHICAL CLASSIFICATION USING *HYPOTHESIS ASSESSMENT* AS A SUBTASK

....

Laboratory for AI Research, The Ohio State University

GT TOOLSET CHARACTERISTICS

• Highly modular knowledge structures, functionally organized. Based on cooperating communities of specialized agents with embedded knowledge.

• Tools are work together for building problem solvers that integrate different stypes of reasoning.

• Explanation facilities are coupled to problem-solving architecture. The course of the problem solving makes sense.

• Problem solvers can call upon external software, and can be embedded.

ARCHITECTURAL IMPLICATIONS
2. Method-specific shells, to be dynamically chosen & combined (TIP work of Punch, BB1, ..)

Laboratory for AI Research, The Ohio State University

TIPS AND BB1

- CENTRAL IDEA IS A MECHANISM THAT CAN INVOKE DIFFERENT METHODS THAT ARE RELEVANT FOR THE CURRENT GOAL, EVALUATE THEM FOR APPROPRIATENESS FOR THE CURRENT SITUATION, AND CHOOSE THE "BEST" METHOD
- THIS IS DONE RECURSIVELY, METHODS CAN BE ABANDONED AND OTHER METHODS CHOSEN

Laboratory for AI Research, The Ohio State University

ARCHITECTURAL IMPLICATIONS
3. use a finer-grained goal scheduling architecture

Laboratory for AI Research, The Ohio State University

Soar

Soar (Newell, et al.)

* The Knowledge Base is used to set up a Problem space in response to a goal
* Operators have to be selected & applied to search the problem space.
* Impasses can occur at any stage (such as operator missing, pre conditions not satisfied, ...)
* Subgoals to resolve such impasses can be set up recursively.
* uniform treatment as search in problem-space
* goals can be mixed from different methods to achieve fine-grained control structure
* Generic Tasks in Soar (Todd Johnson, 91)

Laboratory for AI Research, The Ohio State University

Relation of Tasks/Methods Viewpoint to Traditional GT Work

1. GT's (so far) are fixed task-method-subtask combinations of very generic utility.

2. New approach retains basic insight of GT's: close connection bet method, knowledge & inference, but generalizes ideas

   - Bridge between flexible GP architectures & methods that take advantage of task structure

Laboratory for AI Research, The Ohio State University

Libraries of Tasks, Methods and Knowledge

What is really going on is that there is an international collection of researchers developing precisely this: a library of tasks and methods for many important complex tasks, such as diagnosis and design.

Reusable knowledge modules, indexed by domain, method and knowledge will soon become available

Task: Diagnosis —— "Optimal" METHOD

(de Kleer, Reiter)

"INTRACTABLE"

Domain Knowledge Modeling.

FORMAL COMPLEXITY ANALYSIS OF METHODS

APPENDIX B

TIMING ISSUES IN

REAL-TIME INFORMATION SYSTEMS

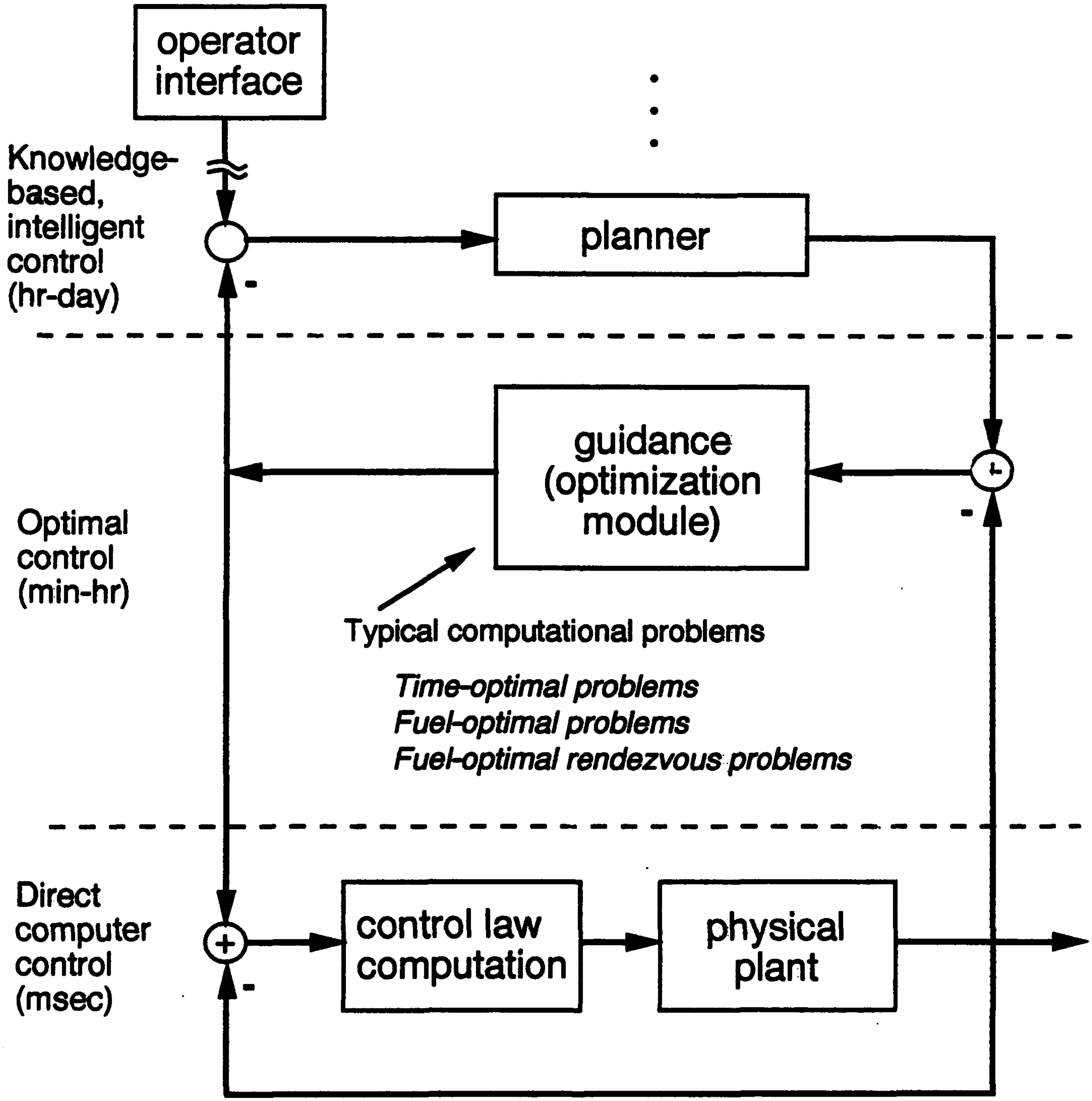
Jane W. S. Liu,
Department of Computer Science
University of Illinois
Urbana, Illinois 61801

- Typical hard real-time systems and their timing constraints
- Challenges in maintaining timing constraints
- Traditional approaches to hard real-time scheduling and resource management
- Recent progresses and unsolved problems

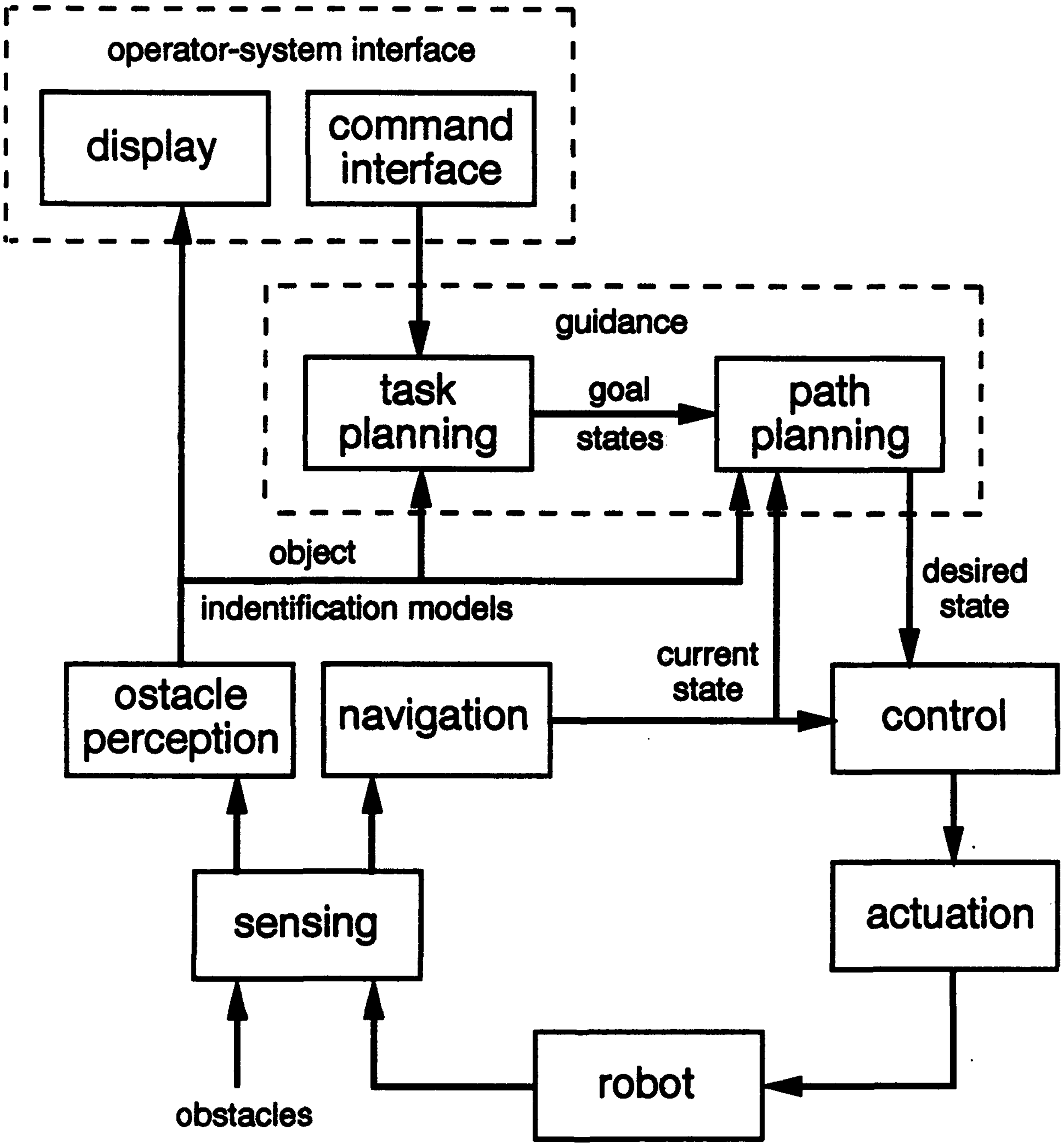
TERMINOLOGY

- *Job* or *task*:
  - — A unit of work (a granule of computation, or data transmission, etc.)
- *Hard real-time task*:
  - — A task whose failure to complete in time is considered to be a fatal error.
- *Soft real-time, essential task*:
  - — A task whose result becomes less and less useful after its deadline
  - — Examples: display update, operator commands and non-critical monitoring.
- *Soft real-time, non-essential task*
  - — A task that may be aborted after its deadline.
  - — Examples: connection establishment, input/output requests.

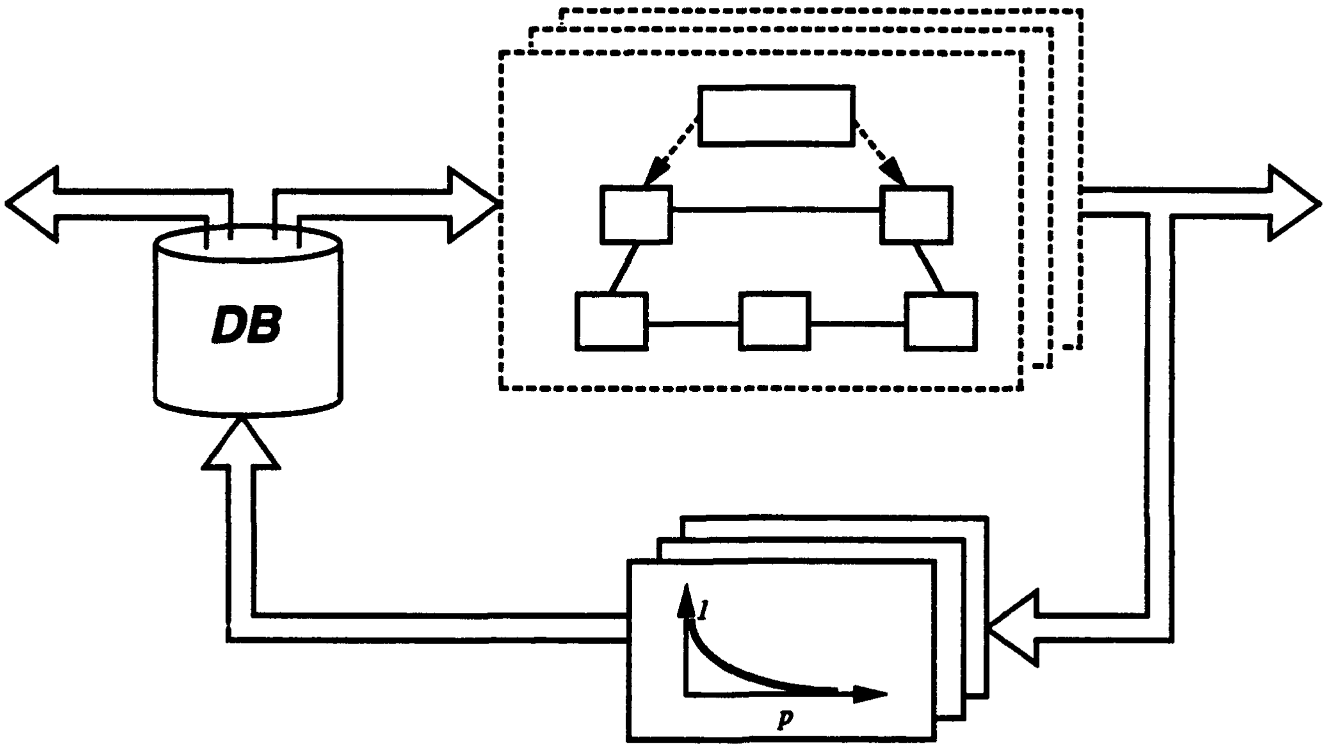
Control Hierarchy



An Example



operator-system interface
display
command interface
guidance
task planning
goal
states
path planning
object
indentification models
desired state
current state
ostacle perception
navigation
control
sensing
actuation
robot
obstacles

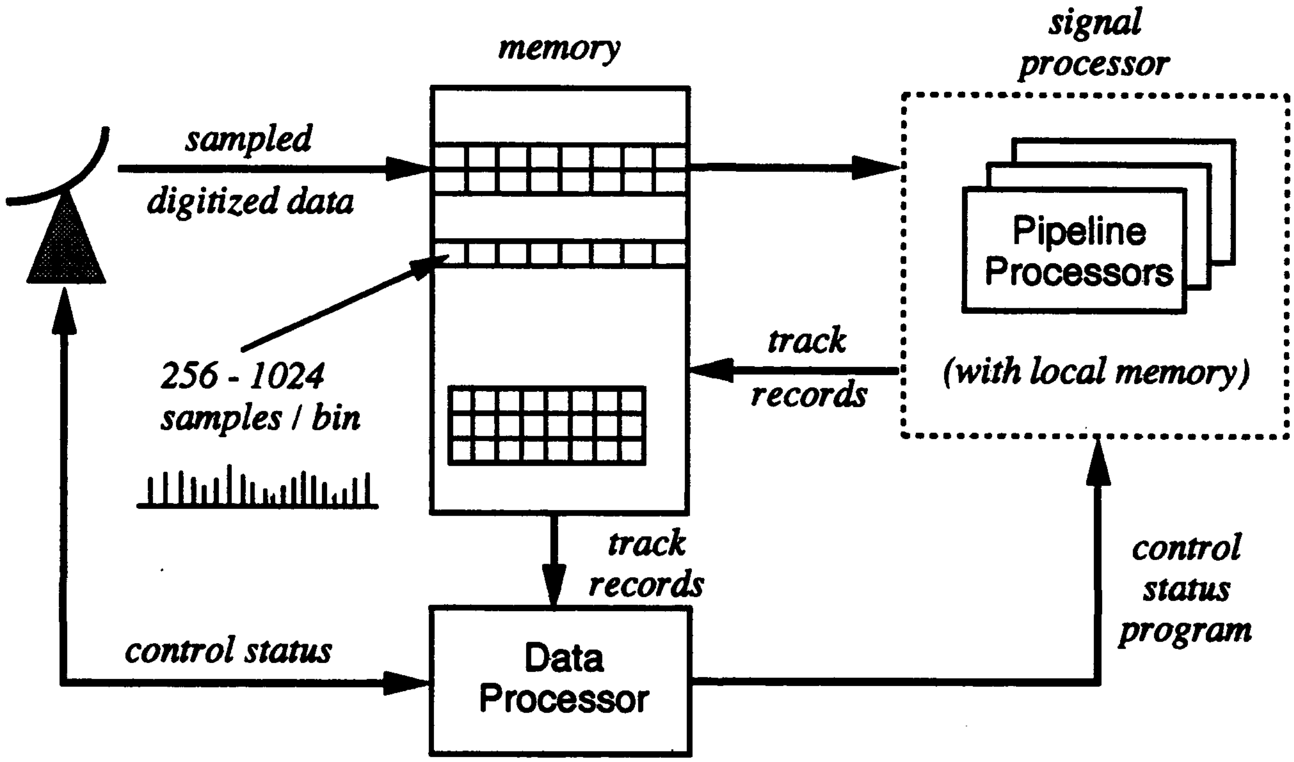


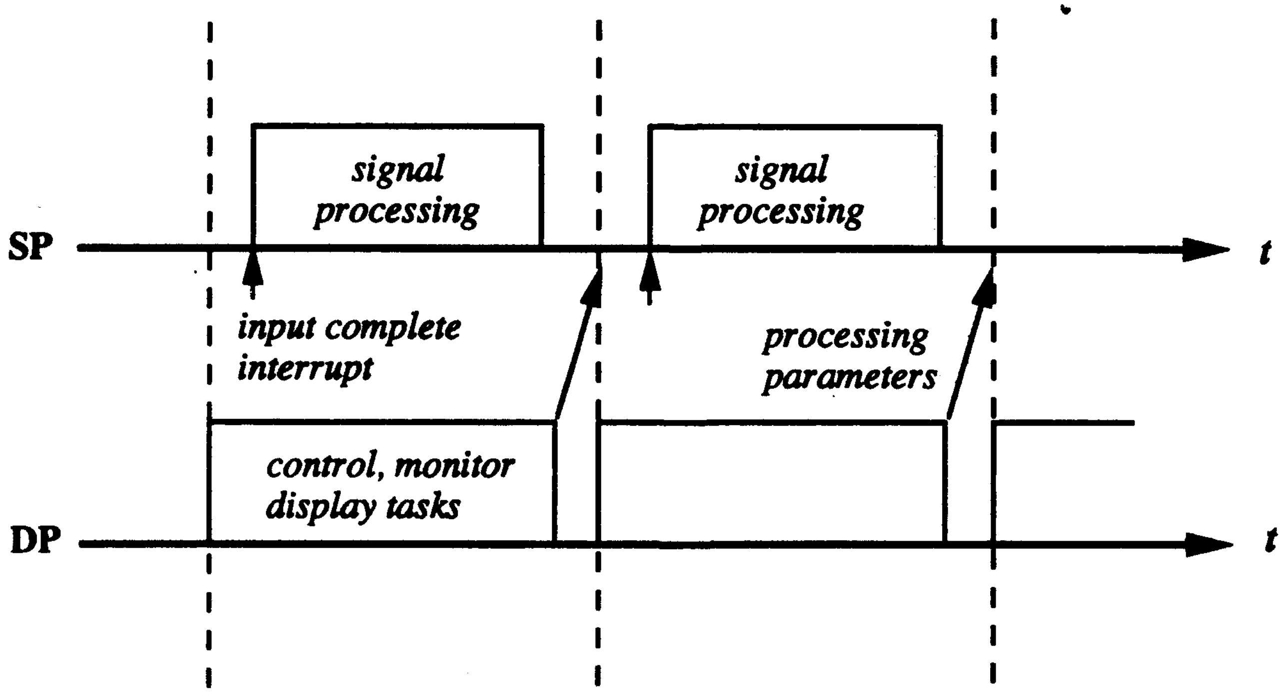
- Rigid timing constraints and exact computations — hard real-time

- Flexible timing constraints and exact computations — soft real-time

- Rigid timing constraints but flexible computations — imprecise computations

Typical radar processing system





The Traditional Periodic Job Model

(*The simplest scheduling problem*)

- The work to be scheduled on each processor is a set of jobs:

  — Each job is a periodic sequence of requests for the same computation, called tasks.

  — The period of a job is the length of the interval between arrivals of two consecutive tasks.

  — The execution time of the tasks in each job is given.

  — The tasks are to be scheduled preemptively.

  — Each task must be completed before the next task becomes ready.

- The problem:

  Find a feasible schedule in which every task completes before its deadline.

A cyclic executive is a program that deterministically interleaves the execution of periodic tasks on a single processor.

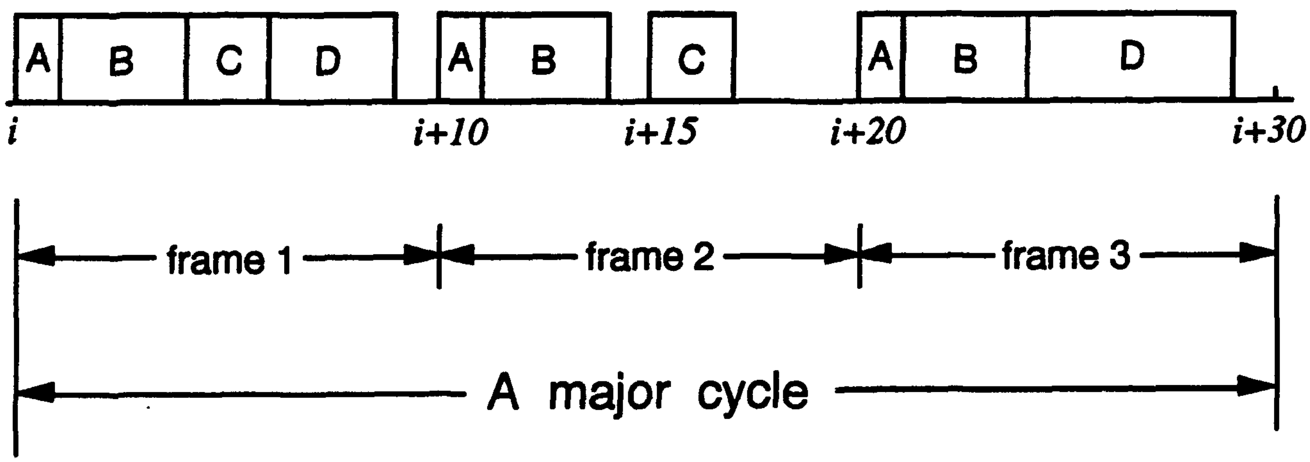
Example

A = (10, 1)

B = (10, 3)

C = (15, 2)

D = (30, 8) => (30, 3) (30, 5)

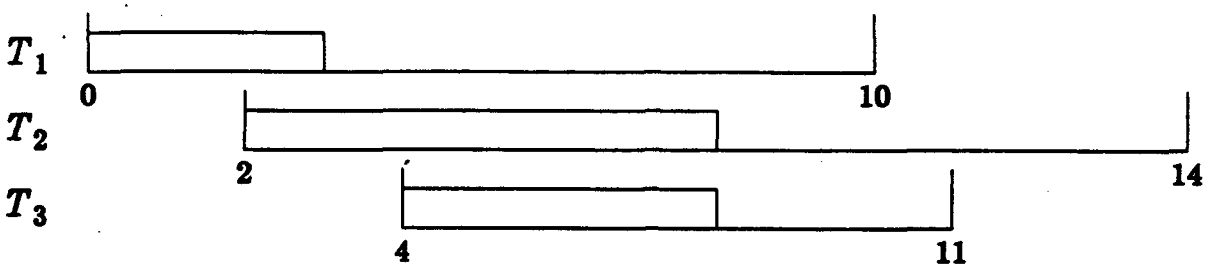


Priority-Driven Algorithms

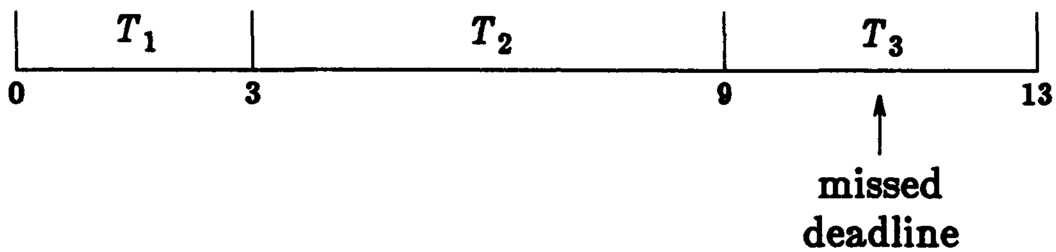
- They are scheduling algorithms that never leave the processor(s) idle intentionally,

- Such an algorithm can be implemented as follows:
  — Assign priorities to tasks.
  — If preemptive, scheduling decisions are made
    - when any task becomes ready,
    - when any task completes,
    - when the task priorities change
  — At each decision time, schedule and execute the ready task with the highest priority.

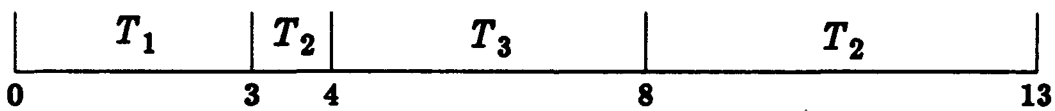
- A priority-driven algorithm is
  — *static* if priorities are assigned to jobs once and for all, and is
  — *dynamic*, if priorities of tasks may change.



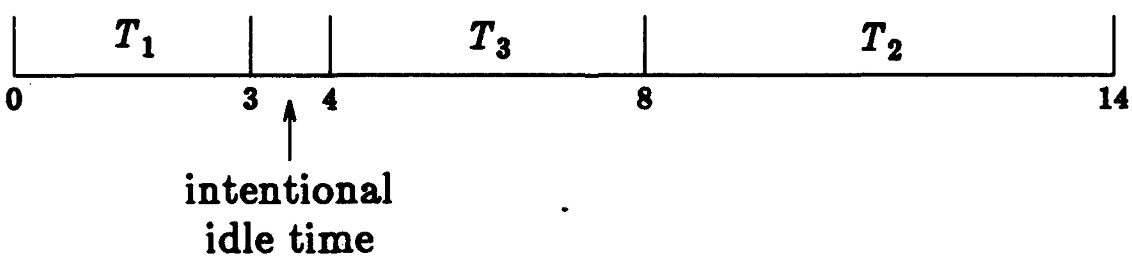
Non–preemptive EDF, FIFO



Preemptive EDF



Non–preemptive, not priority–driven

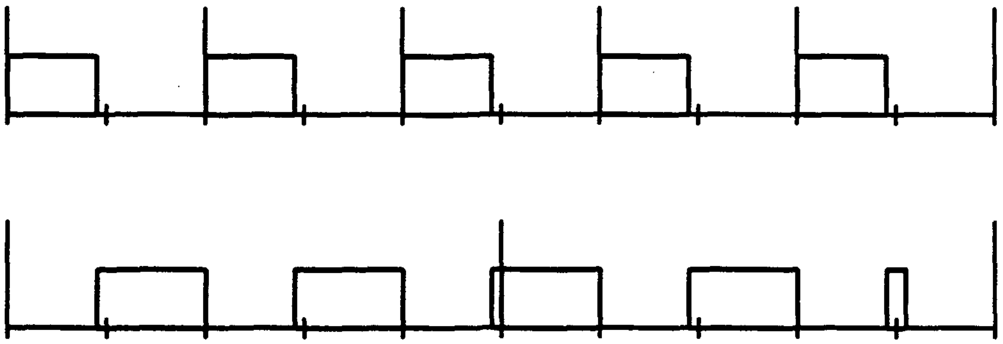


Two Classical Algorithms

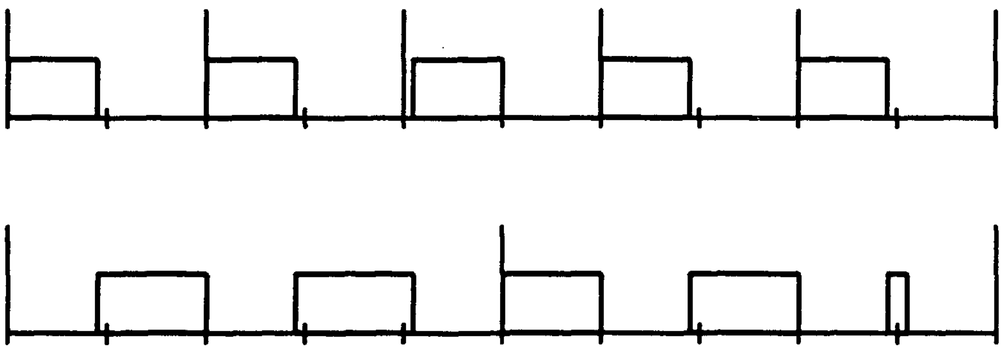
- *Rate monotone*
  - — higher priorities are assigned to jobs with shorter periods
  - — optimal among all static priority-driven algorithms
- *Earliest deadline first*
  - — the highest priority is assigned to a task with the earliest deadline.
  - — optimal among all dynamic priority-driven algorithm

An example: schedule (2, 0.9) (5, 2.3) on one processor

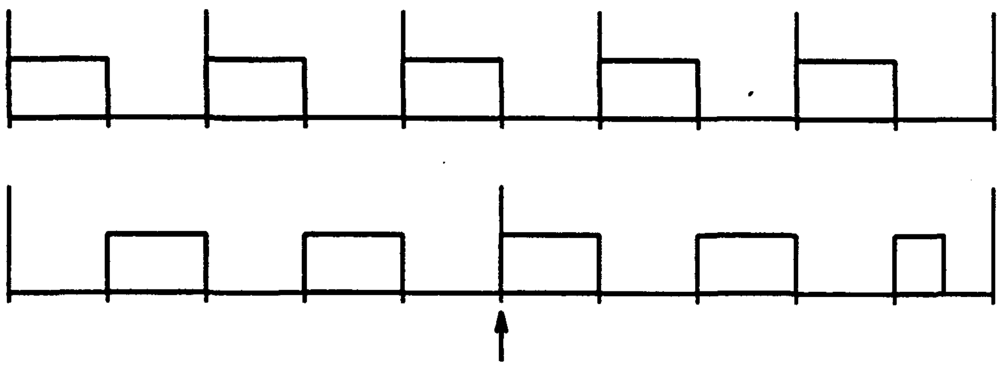
- **Rate monotone**



- **Earliest deadline first**

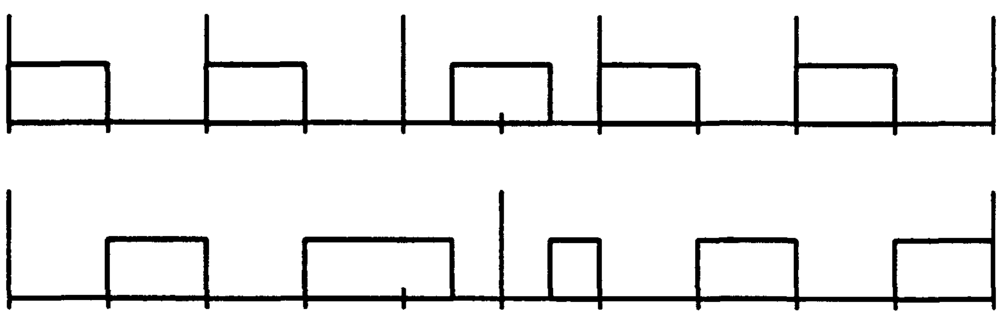


- Schedule (2, 1) and (5, 2.5) by Rate monotone



missed deadline

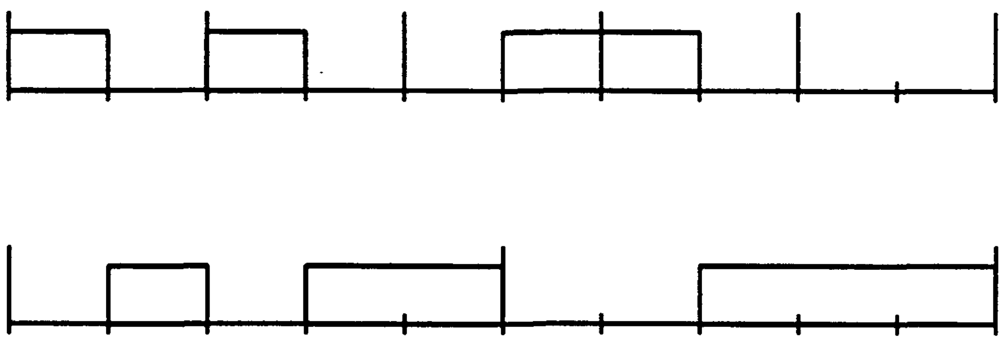
- According to Earliest deadline first



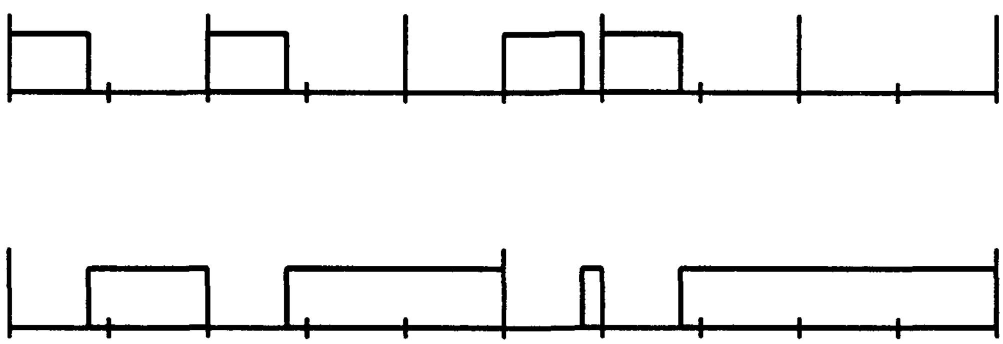
- **Schedulability Criteria**
  - — The total utilization U of a set of job = the fraction of time it keeps a processor busy.
  - — The set is schedulable using
    - the earliest deadline first if $U \leq 1$,
    - the rate monotone if $U \leq 0.82 — 0.63$.

Undesired behavior of the earliest deadline first algorithm

- Schedule (2, 1) and (5, 3) with $U = 1.1$

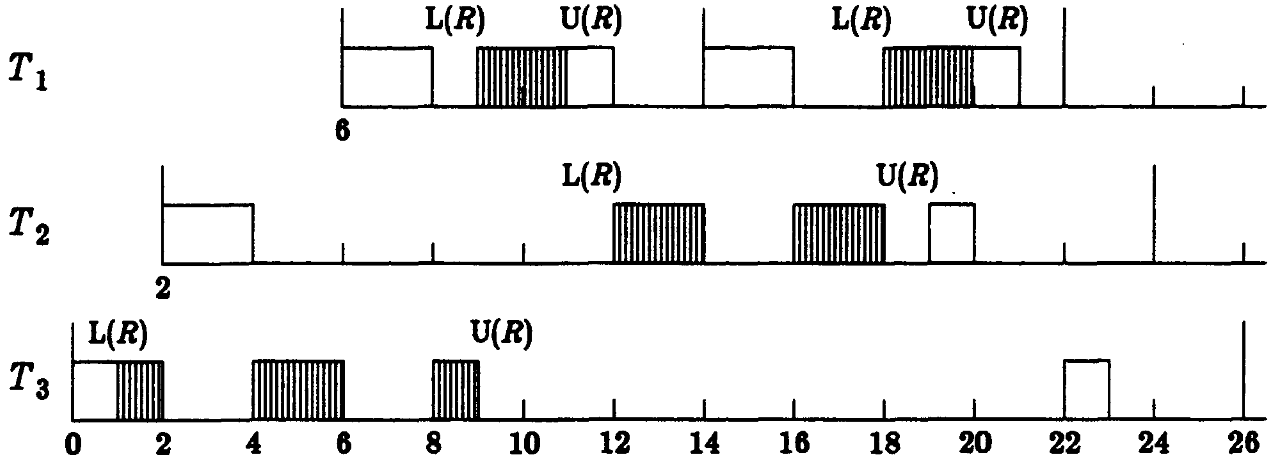


- Schedule (2, 0.8) and (5, 3.5) with $U = 1.1$

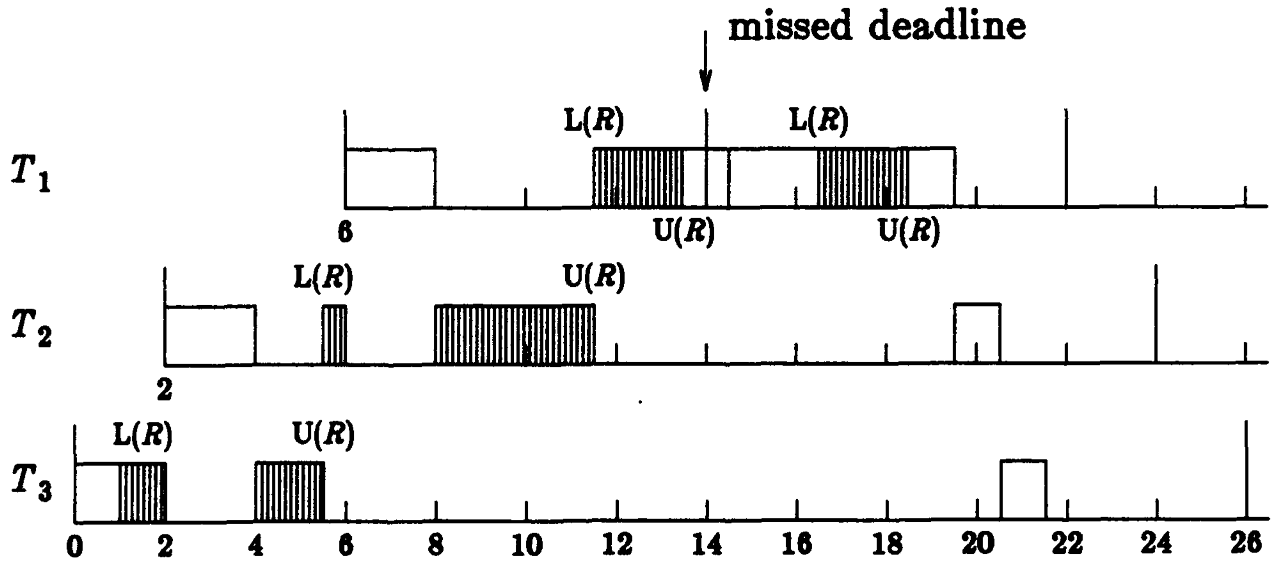


As U increases which deadlines will be missed cannot be predicted.

Scheduling anomaly



$T_1 = (8, 5)$, $T_2 = (22, 7)$, $T_3 = (26, 6)$



$T_1 = (8, 5)$, $T_2 = (22, 7)$, $T_3 = (26, 4.5)$

We have methods to predict such behavior

State of the Art in
Scheduling and Resource Management

- *To support time-critical applications, scheduling algorithms should*
  - guarantee enforcement of timing constraints
  - have predictable behavior under dynamic loads
  - make effective use of system resources

- We *have* optimal and good heuristic algorithms for traditional applications: tasks to be scheduled are
  - periodic and have bounded processing times, or
  - aperiodic with bounded interarrival times.

- We *need* good heuristics for scheduling
  - tasks that have time-varying and data-dependent processing times and resource requirements,
  - tasks with end-to-end timing constraints on multiprocessor and distributed systems, and
  - tasks that interact frequently.

- *We need integrated strategies*

Two Challenging Problems in

Intelligent, Time-Critical Information Systems

- How to keep information temporally consistent
- How to deal with unpredictability in time and resource requirements

Temporal Consistency

A set of data objects gives the state of the real world.

- *Relative temporal consistency*:

  The data objects represent a snapshot.

- *Absolute temporal consistency*:

  The snapshot is sufficiently current.

Data Objects

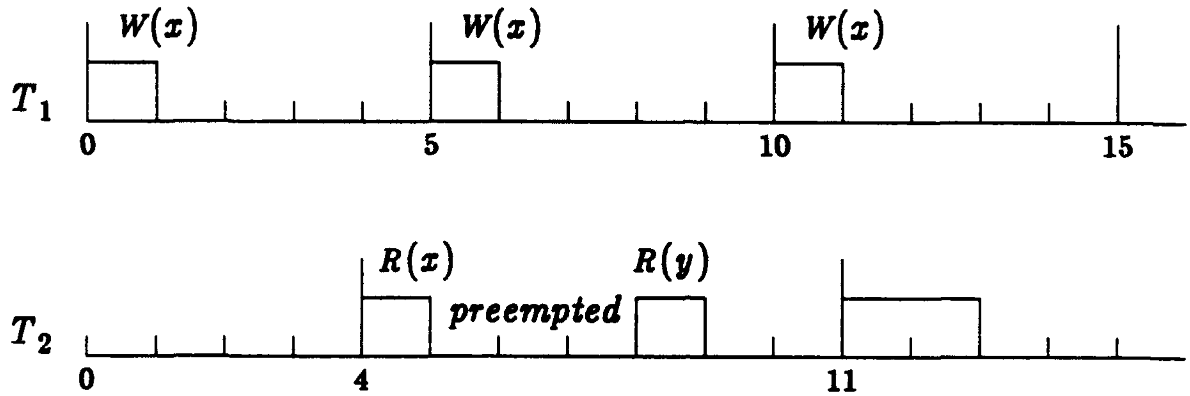
- *Images* which store sampled values of real-world objects

  In the ith period after image x was last updated, age of $x = i + 1$.

- *Derived objects* whose values are computed

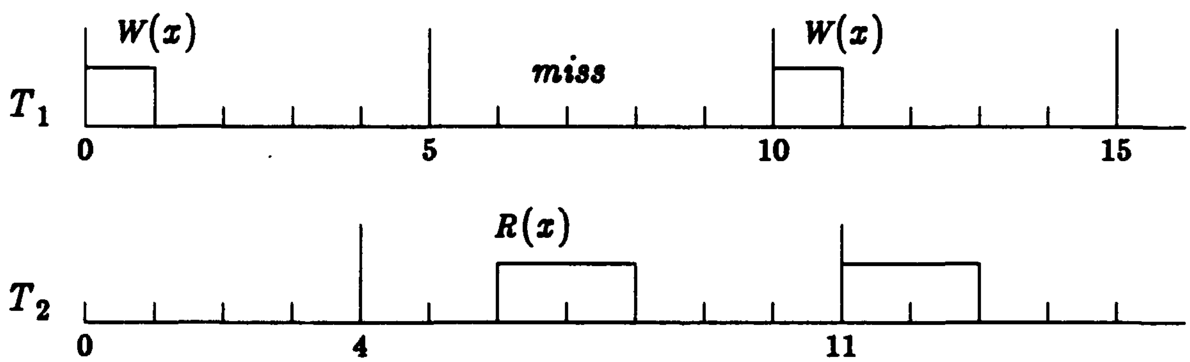
  If the derived object y is computed from $\{x_i, y_j\}$, age of y = max { ages of x_i and y_j }

Relative Temporal Consistency



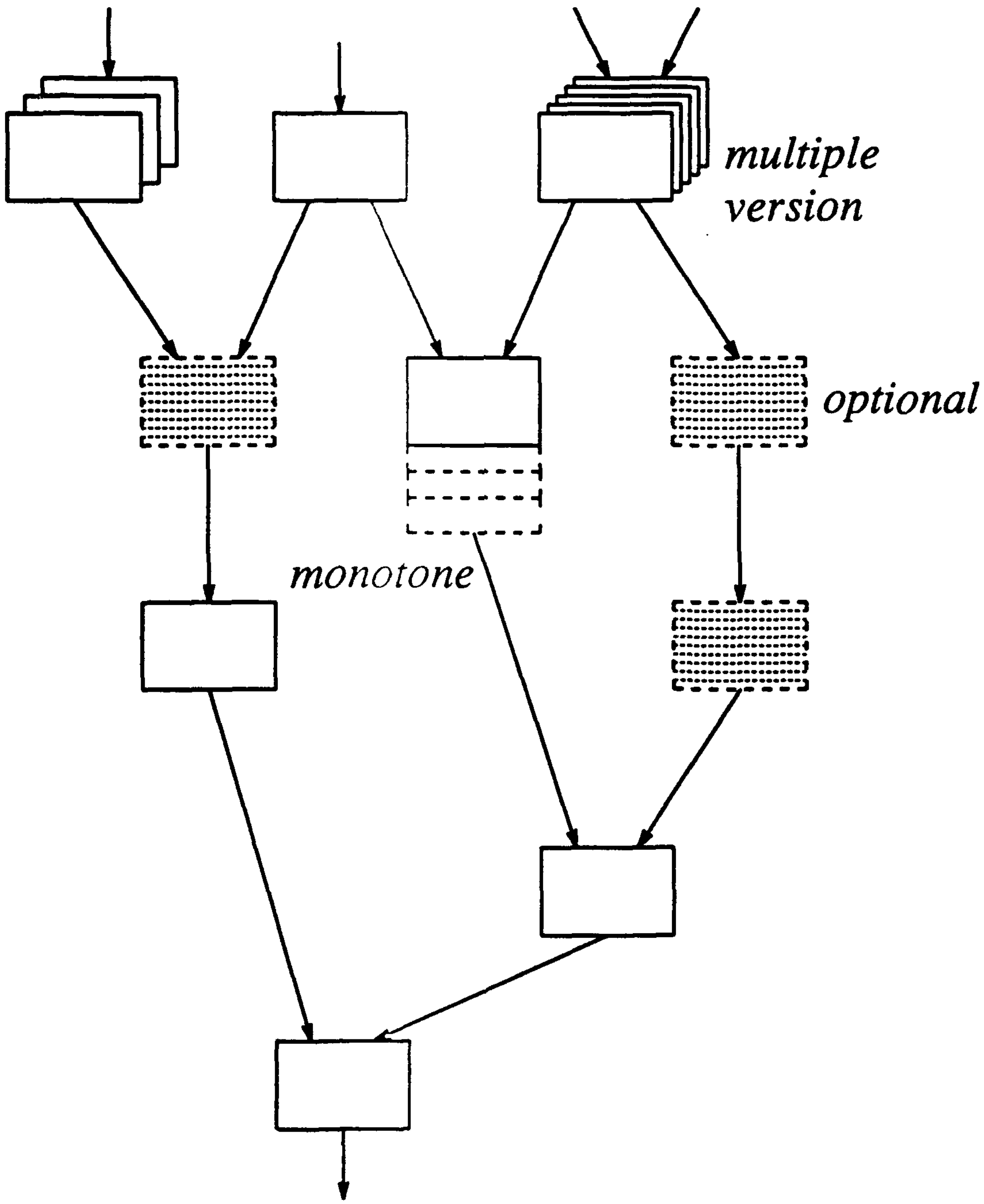
- The values of x and y read by T_2 may be temporally dispersed.

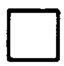
Absolute Temporal Consistency



- The values of x read by T_2 may be obsolete.

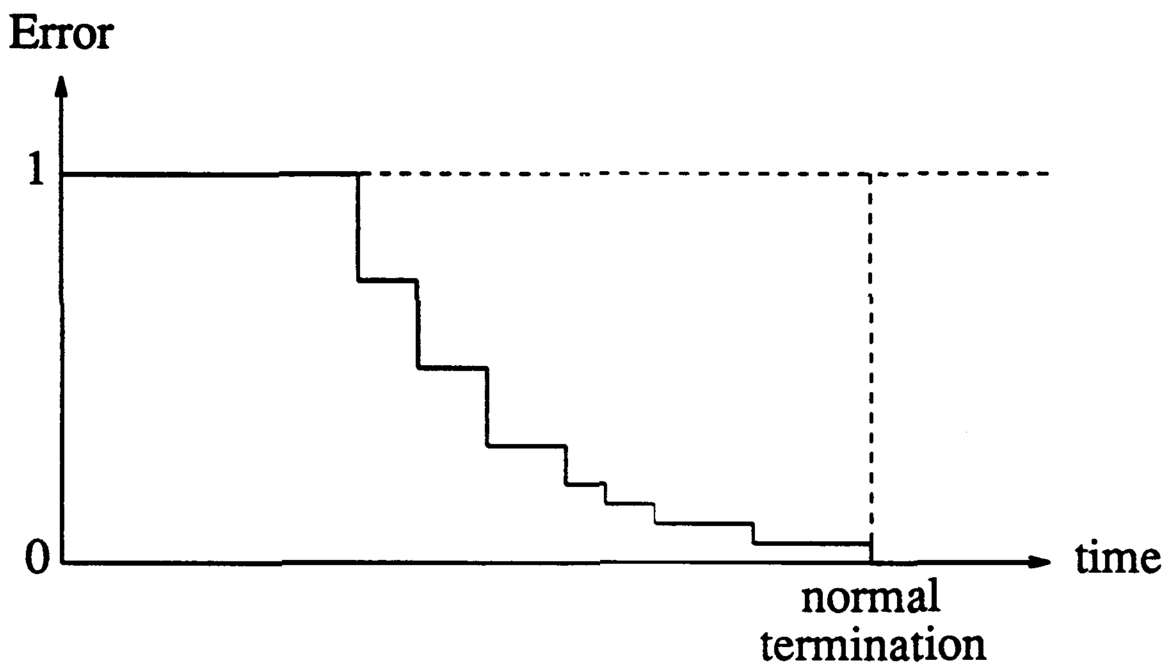
The Imprecise Computation Model

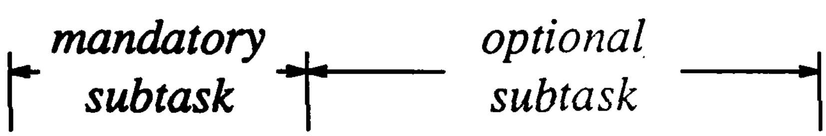


 A task, a granule of computation

→ Dependency between tasks

Monotone Imprecise Computations





error recovery mandatory ←|→ *error recovery optional*

Precise Results Vs Imprecise Results

The called procedure

```
newton(xguess, xdistance, acceptable_distance, desired_distance)
    {
                    /* mandatory part */

        repeat {
            xnew = xguess - f(xguess)/fprime(xguess);
            xdistance = xnew - xguess;
            xguess = xnew;
        } while (abs(xdistance) > acceptable_distance);
        return_imprecise_result (xnew, xdistance);

                    /* optional part */

        repeat {
            xnew = xguess - f(xguess)/fprime(xguess);
            xdistance = xnew - xguess;
            return_imprecise_result (xnew, xdistance);
            xguess = xnew;
        } while (abs(xdistance) > desired_distance);
        return_precise_result (xnew, xdistance);
    }
```

The calling procedure

```
{
    ..........
    guess = initial_value;
    imprecise_result (newton, handle);
    newton(guess, distance, acceptable_range, desired_range);
    if (distance ≤ desired_range)
            /* a precise result returned */
    else if (distance ≤ acceptable_range)
            /* an acceptable imprecise result returned */
    else    /* the returned result not usable */
    ..........
}
```

Suitable Methods and Applications

- Iterative algorithms, accumulation problems
- Statistical methods, e.g. Monte Carlo method
- Approximate relational algebra query computation
- Successive doubling method for computing FFTs
- Generation of images from holograms
- Transmission of compressed digitized voice
- Transmission of facsimile images

Application in the Database Domain

- Work in progress
  - — Monotone query processing when the database contains complete information and queries are precise.
- Future work
  - — Monotone query processing in the presence of imprecise or incomplete information
  - — Imprecise updates

Where can flight United 941 land?

The Exact Relation

Location	Appro. Dist.
Chi-O'Hare, IL	350
Manchester, NH	250
Peoria, IL	400
Syracuse, NY	100

An Approximate Relation

Location	Appro. Dist.	
Chi-O'Hare, IL	350	*Certain tuples*
Peoria, IL	400	
Paris, IL	380	*Possible tuples*
Hell, MI	330	
Manchester, NH	250	
Berlin, MA	300	
Syracuse, NY	100	

Location	Approx. Dist.
Chi-O'Hare, IL	350
Peoria, IL	400
Syracuse, NY	100
Paris, IL	380
Hell, MI	330
Manchester, NH	250
Berlin, MA	300

Location	Approx. Dist.
Chi-O'Hare, IL	300
Peoria, IL	400
Paris, IL	380
Manchester, NH	250
Berlin, MA	300
Syracuse, NY	100

Location	Appro. Dist.
Chi-O'Hare, IL	300
Peoria, IL	450
Paris, IL	380
Hell, MI	330
Manchester, NH	250
Berlin, MA	300
Syracuse, NY	100

Elements of the Approximate Relational Model

- A set of all approximate relations of any standard relation — approximate answers to a relational query
- A partial order relation over the set for comparing them
- Monotone approximate relational algebra operations
- A monotone query processing algorithm for returning monotonically improving answers

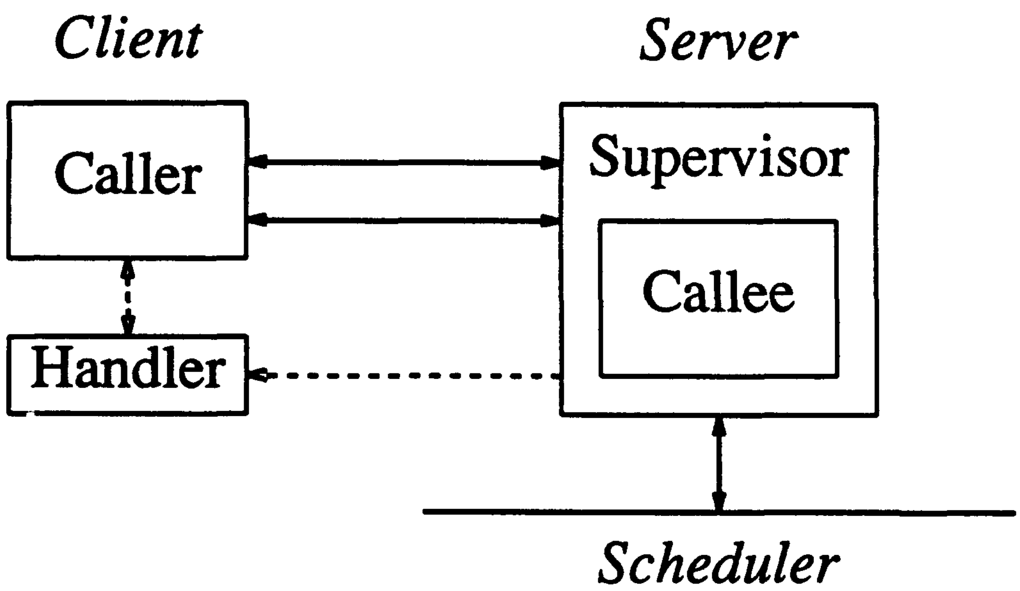
APPROXIMATE

(*A Prototype Monotone Query Processor*)

- Supports processing of relational algebra queries on relational database systems
- Uses an objected-oriented approach to implementation:
  - — relies on an object-oriented view for semantic support in identification of initial approximations
  - — avoids operations on possible tuples
  - — provides lazy evaluation of possible classes upon user interrupt or faults
- Improves data availability and query processing time predictability

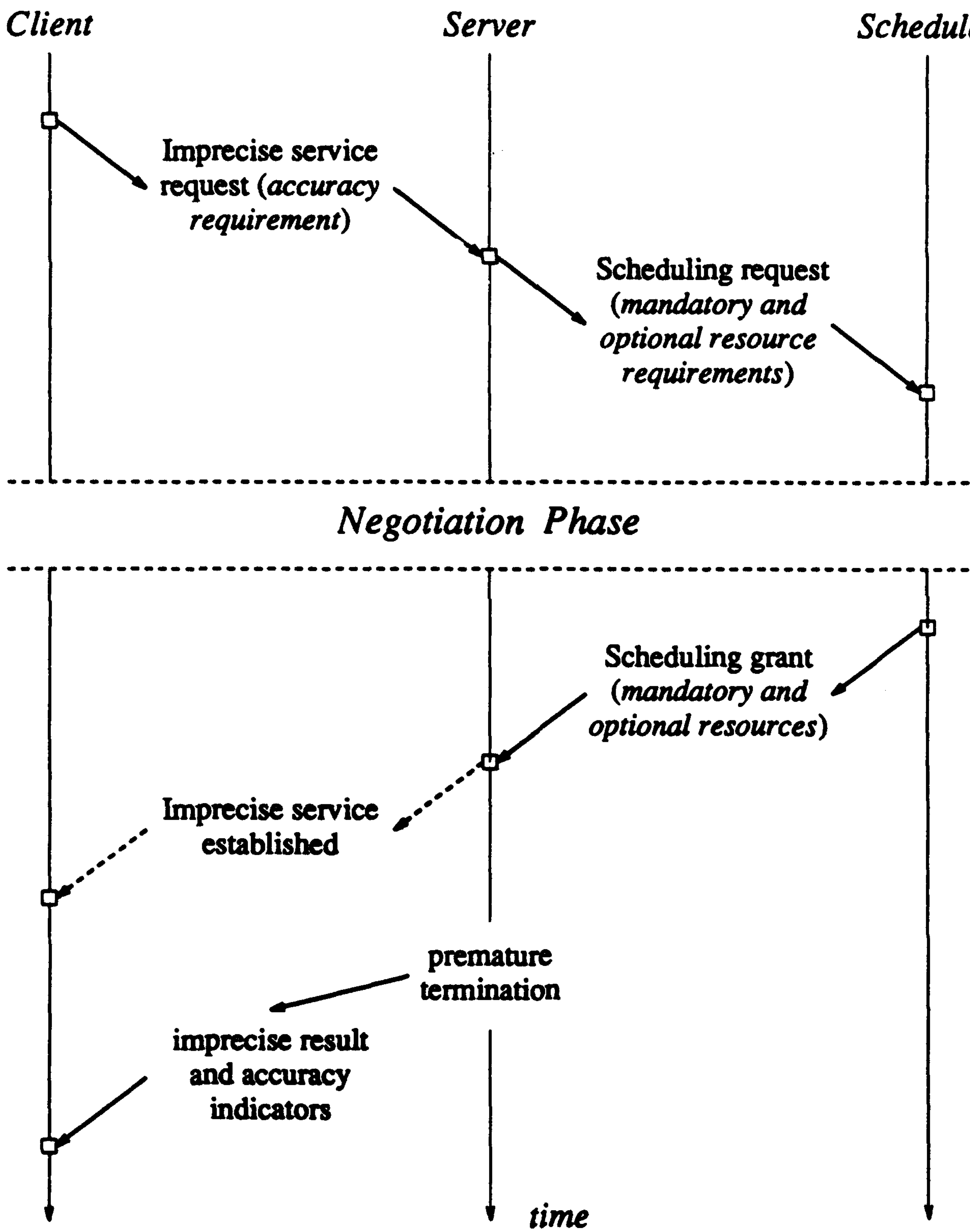
Elements of An Architectural Framework

- Process Structure



- *Result Saving Protocol* — for the provision of monotonically improving imprecise results
- *Imprecise Service Establishment Protocol* — governing the interactions among the application tasks, as well as among the application tasks and the underlying support system
- *Imprecision Management Policies and Mechanisms* — to ensure the correct usage of imprecise results

Imprecise Service Establishment



Periodic Job Models

- Precise periodic job set $\{J_k\}$

  — J_k is specified by (p_k, τ_k)

  p_k = period length.

  τ_k = execution time

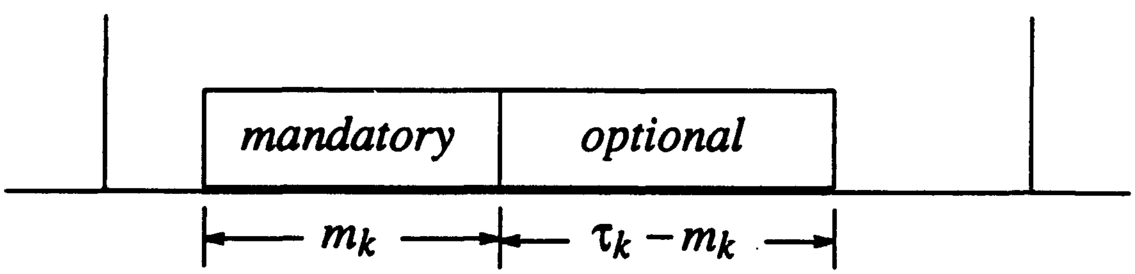
  ready times and deadlines

- Imprecise periodic job sets $\{M_k\}$ and $\{O_k\}$

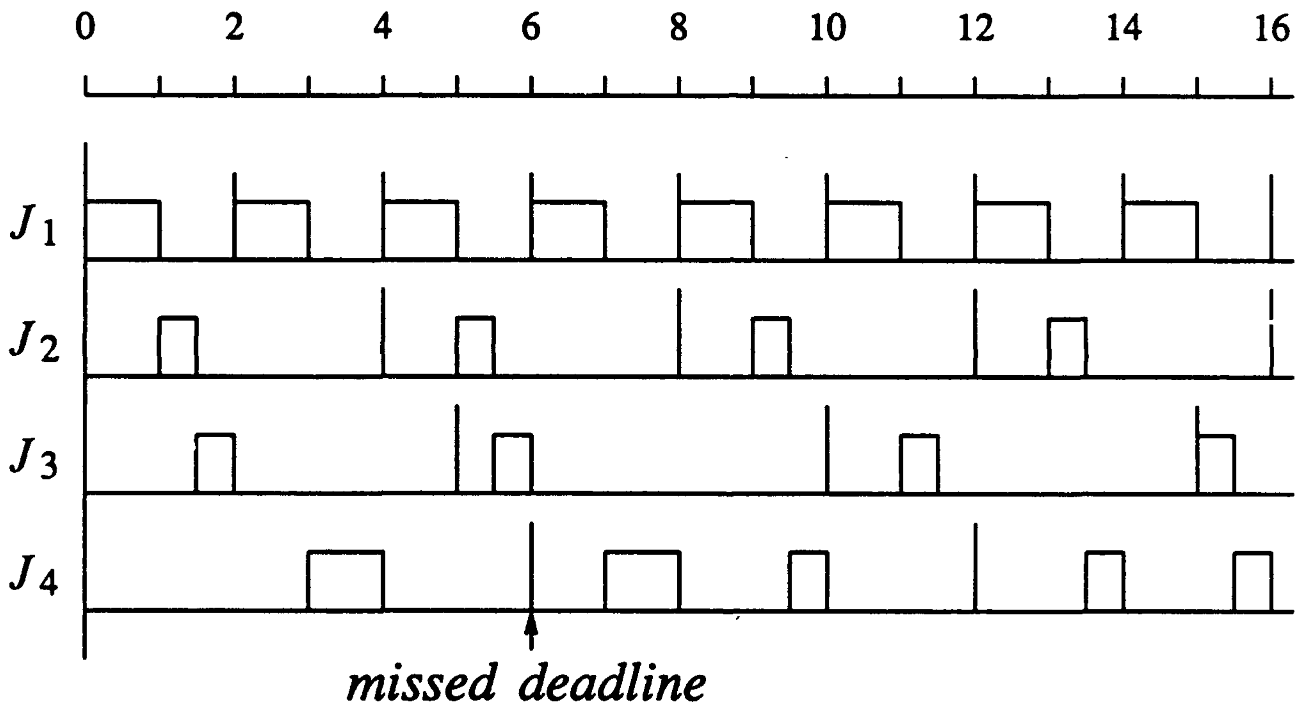
  — m_k = minimum execution time

  — Given $J_k = (p_k, \tau_k)$ there is

  - a mandatory job $M_k = (p_k, m_k)$
  - an optional job $O_k = (p_k, \tau_k - m_k)$

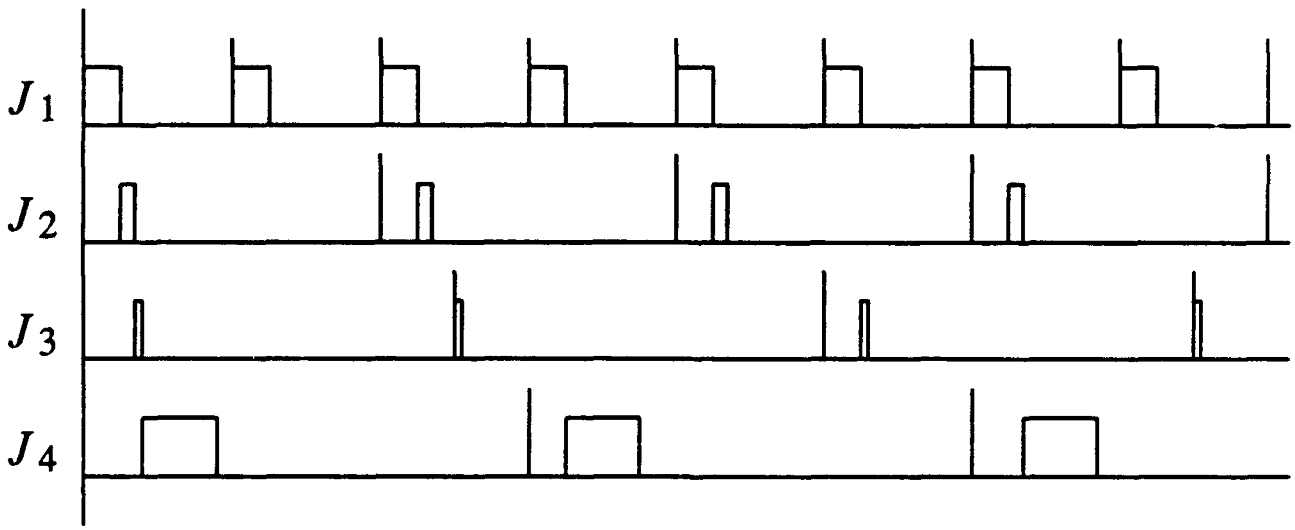


$\{J_k\}$: (2, 1) (4, 0.5) (5, 0.5) (6, 1.5)

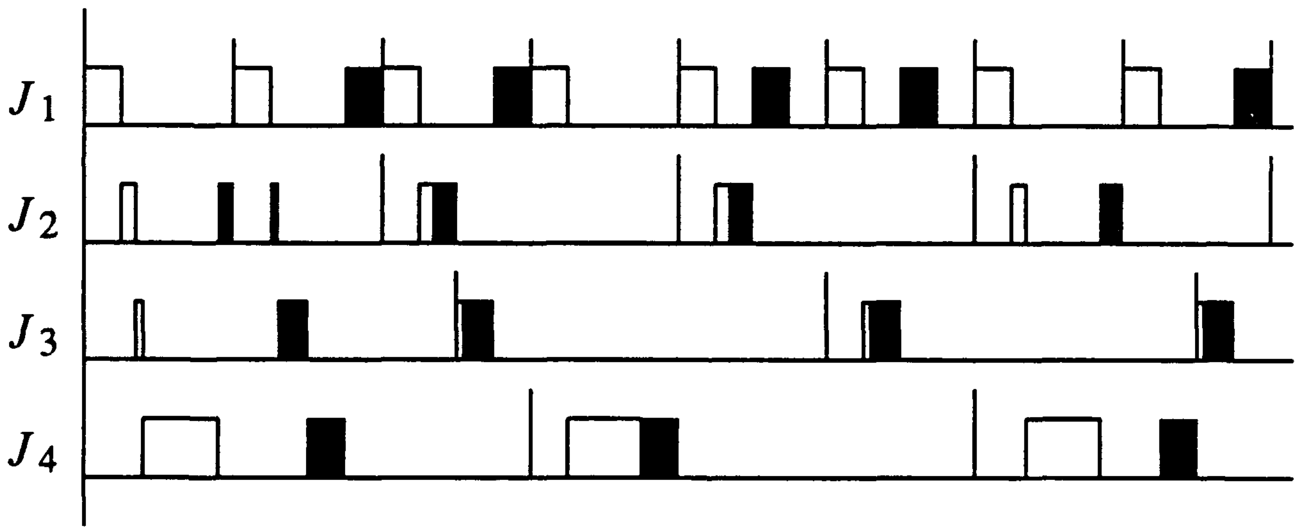


A traditional rate-monotone schedule

$\{J_k\}$:	(2, 1)	(4, 0.5)	(5, 0.5)	(6, 1.5)
$\{m_k\}$:	0.5	0.2	0.1	1.0
$\{M_k\}$:	(2, 0.5)	(4, 0.2)	(5, 0.1)	(6, 1.0)
$\{O_k\}$:	(2, 0.5)	(4, 0.3)	(5, 0.4)	(6, 0.5)



A schedule for the mandatory set



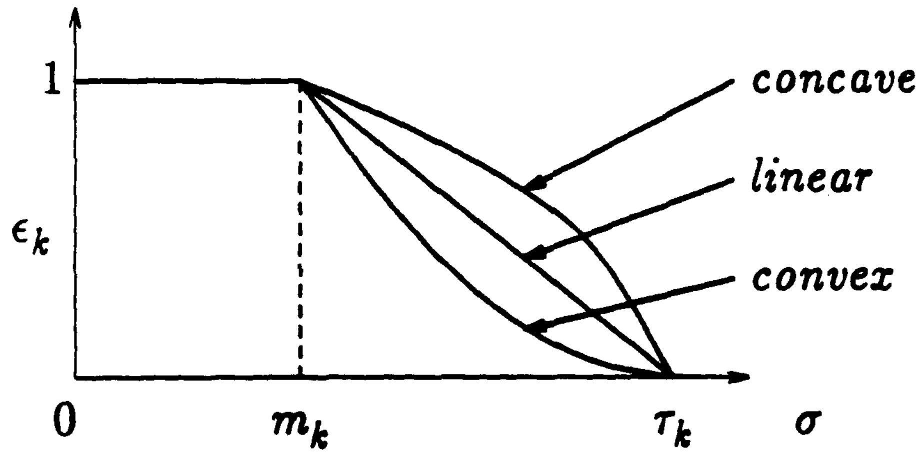
and the optional set

Types of Jobs

- Error non-cumulative jobs
  - — optional parts need not be completed
  - — the average effect of error is observable
  - — example: image enhancement
- Error cumulative jobs
  - — optional part must be completed occasionally
  - — errors in different periods have cumulative effects
  - — example: target tracking

Performance Measure for Scheduling Error–noncumulative Jobs

- Error Functions



- **Average error of job J_K over p periods**

  p = least common multiple of all periods

- **Average error over K jobs in J**

$$E = \frac{1}{K} \sum_{k=1}^{K} E_k$$

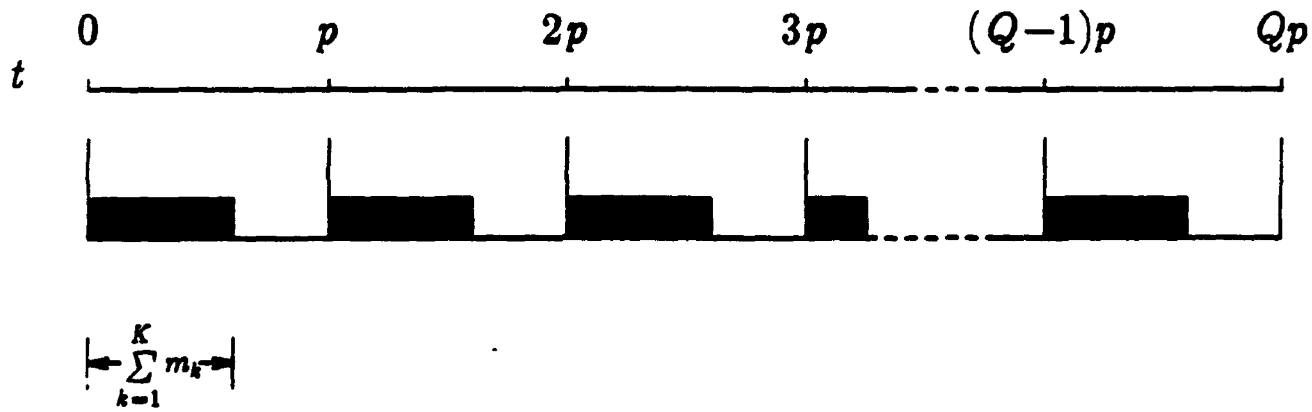
Relative Merits of Different Algorithms

$U \leq 1$		$U > 1$
Classical scheduling algorithms:		Our scheduling algorithms:
Earliest–deadline	***concave error***	Most–attained–time
Least–slack–time	***linear error***	Least–utilization
Rate–monotone	***convex error***	Least–attained–time

Scheduling Type C Jobs

(The Simplest Case)

- Jobs in **J** has the same period p
- One task in every job must complete in Q periods
  — *Cumulation rate* $= Q$
- After the tasks in **M** are assigned, we have



Must pack K pieces of lengths $\tau_k - m_k$ in

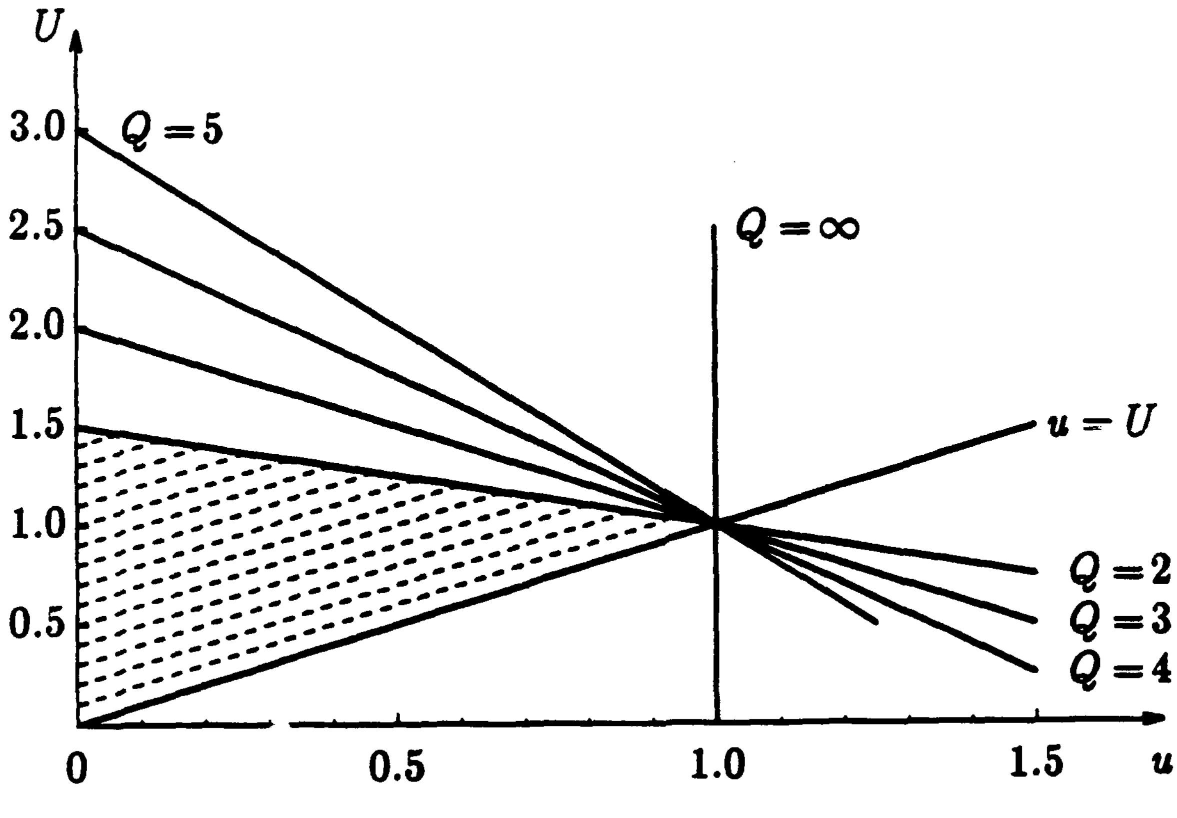
Q bins of size $p - \sum_{k=1}^{K} m_k$

Schedulability Criterion

(*for the length monotone algorithms*)

A set of jobs with repetition period p, cumulation rate Q, total utilization factor U, and minimal utilization factor u is schedulable if

$$\frac{Q-1}{Q+1}\,u + \frac{2}{Q+1}\,U \leq 1$$

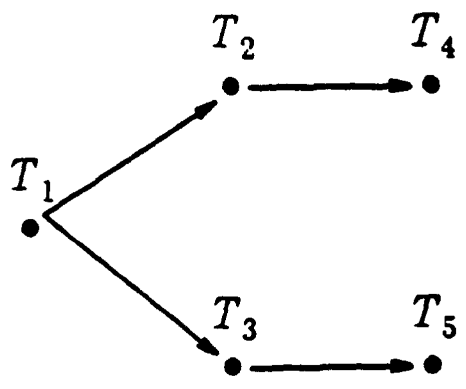


Scheduling Aperiodic Jobs That Allow Imprecise Results

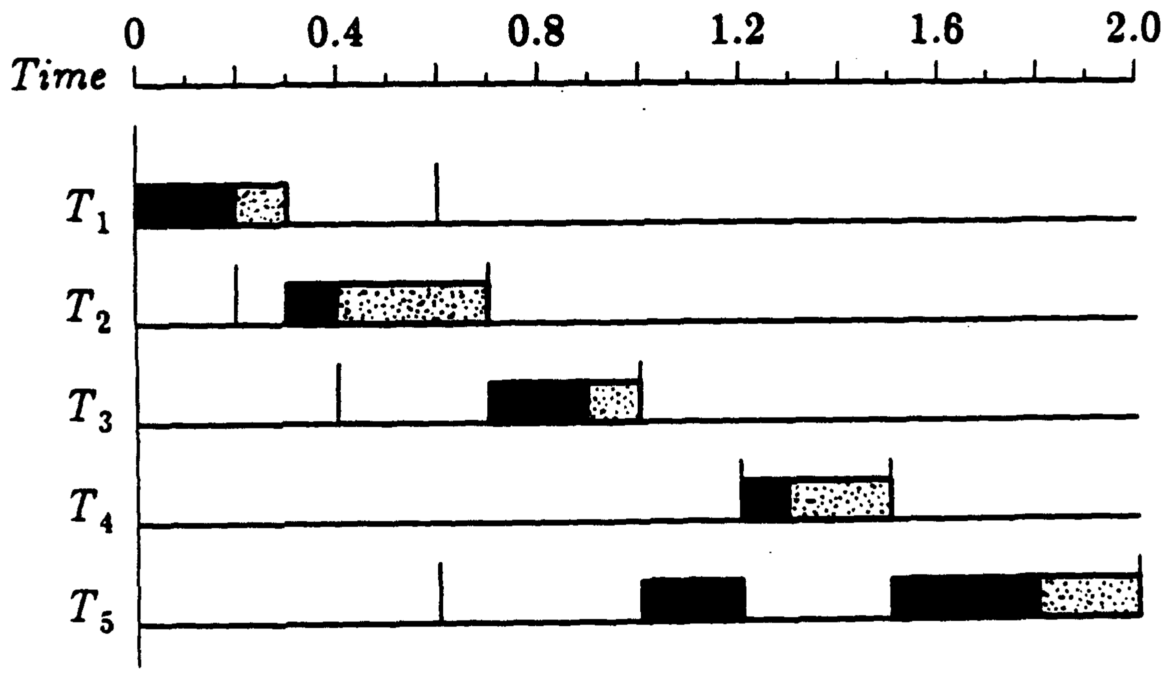
- Given a dependent tasks set $\{ T_k \}$
  - Each task T_k is characterized by
    - ready time
    - deadline
    - execution time
  - dependencies between the tasks

- Each task is decomposed into
  - a mandatory subtask
  - an optional subtask

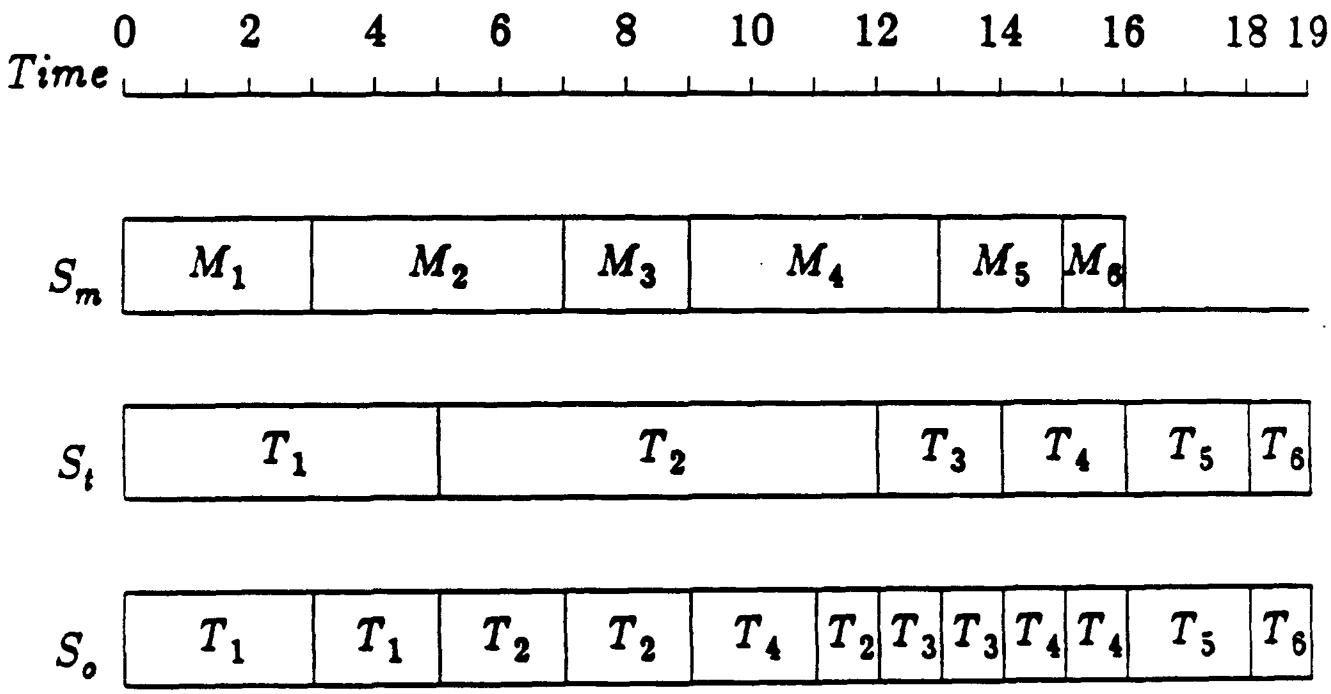
An Example: Scheduling to Minimize Error



	r_i	d_i	τ_i	m_i	o_i
T_1	0.0	0.6	0.4	0.2	0.2
T_2	0.2	0.7	0.4	0.1	0.3
T_3	0.4	1.0	0.5	0.2	0.3
T_4	1.2	1.5	0.3	0.1	0.2
T_5	0.6	2.0	0.8	0.5	0.3



	r_i	d_i	τ_i	m_i	o_i
T_1	0	7	5	3	2
T_2	3	12	7	4	3
T_3	2	14	6	2	4
T_4	5	16	6	4	2
T_5	5	18	3	2	1
T_6	10	19	4	1	3



Available Building Blocks of Imprecise Computation Systems

- Algorithms for scheduling aperiodic tasks with ready times, deadlines and precedence constraints on uniprocessors
  - to minimize the total error
  - to minimize the maximum error
  - to minimize the mean flow time
  - to minimize the number of discarded optional subtasks

- Algorithms for scheduling aperiodic independent tasks with ready times and deadlines
  - to minimize the total error on multiprogrammed multiprocessor systems
  - to minimize the total error when parallelized

- Algorithms for scheduling periodic tasks, that are
  - error-noncumulative, to minimize average error
  - error-cumulative, to meet all deadlines

- Task assignment algorithms for replicated imprecise tasks

- 2-level queuing policies for optimal tradeoff between average response time and result quality

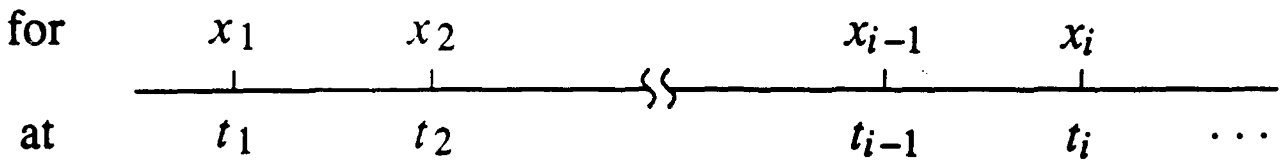
- A monotone query processor for relational algebra queries

Using Imprecise Computation Technique

- to increase availability
- to reduce the need for error recovery
- to simplify recovery actions
- to support semantical approaches to reduction of checkpoint sizes
- to reduce the overhead of replication

Problems in Different Application Domains

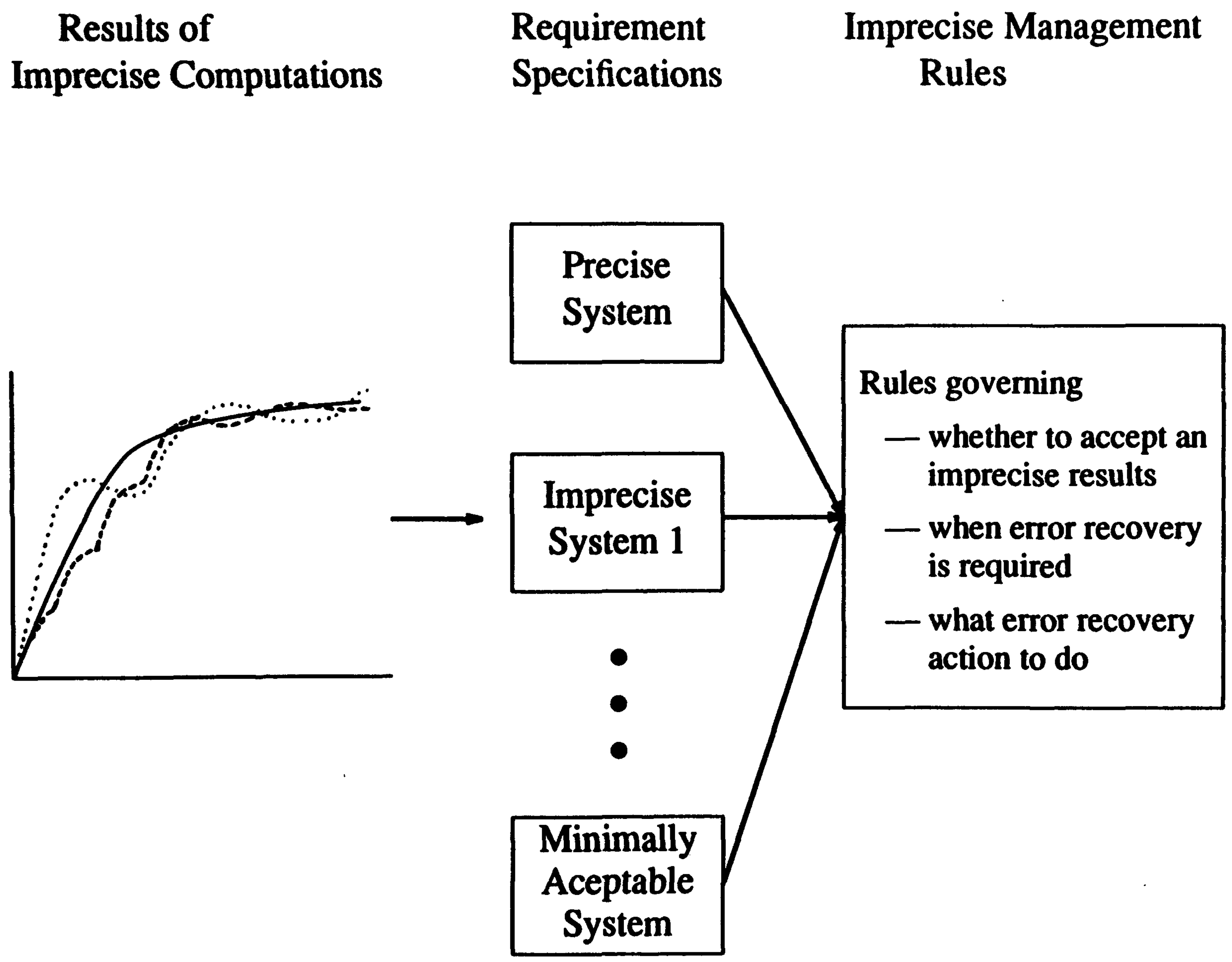
Solve $f(x,t)=0$



- *Error-noncumulative*:
  - Type-1 problems:
    - x_i is independent of $x_{i-1}, x_{i-2}, \cdots$
    - Examples include signal processing, still-image transmissions, database queries, etc.
- *Error-cumulative*:
  - Type-2 problems:
    - $x_i \approx x_{i-1}$, but is independent of $x_{i-2}, x_{i-3}, \cdots$
    - Examples include Newton-like iterative algorithms, traditional feedback control, tracking, etc.
  - Type-3 problems:
    - x_i is dependent on $x_{i-1}, x_{i-2}, \cdots$
    - Examples include real-time simulation and non-linear control

Types of Imprecision Management Policies

Types of Application Domain	Decision Support Information or Recovery Actions		
	Type-1	Type-2	Type-3
Acceptance of x_i is based on	x_i	x_i and x_{i-1}	$x_i, x_{i-1}, x_{i-2}, \cdots$
Recovery action depends on	x_i	x_i and x_{i-1}	$x_i, x_{i-1}, x_{i-2}, \cdots$
Alternative recovery actions:	refine x_i	refine x_i, or precompute x_{i+1}	refine x_i, or precompute x_{i+1}, or recompute $x_{i-1}, x_{i-2}, \cdots$



Results of Imprecise Computations
Requirement Specifications
Imprecise Management Rules
Precise System
Imprecise System 1
Minimally Aceptable System
Rules governing
— whether to accept an imprecise results
— when error recovery is required
— what error recovery action to do

Current and Future Work

- Development and evaluation of the needed architectural elements for the efficient generation and correct usage of imprecise results for fault tolerance

- Development of useful approximation semantics and monotone algorithms in key application domains

- Development of imprecision management policies for application domains including
  - — presentative numerical computations
  - — compressed voice and video data transmissions
  - — signal processing
  - — feedback control, feedforward control, declarative control, etc.
  - — database queries and updates

SUMMARY

- The desired characteristics of time-critical systems is *PREDICTABILITY*, not speed, or fairness, etc.:
  - — Under a normal load, all hard real-time tasks meet their timing constraints.
  - — Under overload conditions, failures in meeting timing constraints occur in predictable manner

- We need application-directed scheduling and resource management:
  - — Scheduling mechanisms should allow different policies.
  - — Resolution of resource conflicts can be under the explicit direction of the applications.

- Applications must be designed so that either
  - — they have deterministic behavior, or
  - — they allow trade off between time and result quality.

APPENDIX C

Research in Deductive Databases at the University of Maryland

by

Jack Minker

Department of Computer Science
and
Institute for Advanced Computer Studies

Current Research in Deductive Databases

- Disjunctive Logic programs and Deductive databases
- Combining deductive databases
- Cooperative Answers
- Null values in deductive databases
- View updates for deductive databases

Deductive databases – Background

A deductive database is a function free Horn logic program.

(Definite) Logic Programs

$$A \leftarrow B_1, \ldots, B_n$$

Example

$$\begin{array}{l} \text{stealth} \leftarrow \text{radar}, \text{faint} \\ \text{plane} \leftarrow \text{radar}, \text{clear} \\ \text{radar} \\ \text{clear} \end{array}$$

Meaning

$$\{\text{radar}, \text{clear}, \text{plane}, \text{stealth}\}$$

$$\{\text{radar}, \text{clear}, \text{plane}\}^*$$

...Background

Use of negation

$$A \leftarrow B_1, \ldots, B_n, \ not\ D_1, \ldots, \ not\ D_m$$

Stratified Logic Programs

$$\frac{\begin{array}{l}\text{stealth} \leftarrow \text{radar, faint}\\ \text{radar}\end{array}}{\text{plane} \leftarrow \text{radar, } \mathit{not} \text{ faint}}$$

Normal Logic Programs

stealth ← radar, faint, *not* obstructed
obstructed ← radar, faint, *not* stealth
lookdown ← obstructed
lookdown ← stealthradar
faint

Summary of Definite Semantics

Semantics	Theory	Reference
Positive Consequences		
Fixpoint	T_P	vEK76
Model	Least Model	vEK76
Procedure	SLD	Hill74
Negation		
Theory	CWA	Rei78
Rule	NAF	Cla78
Procedure	SLDNF	Cla78
Stratified Programs		
Fixpoint	T_P	ABW88
Model	Standard	ABW88
	Perfect	Prz88
Normal Programs		
	Well-Founded	
Fixpoint	I^{∞}	VRS88b
Model	$M_{WF}(P)$	VRS88b
Procedure	SLS	Ross89b/Przy89b
	General Well-Founded	
Fixpoint	I^{E}	BLM89b
Model	M_P^E	BLM89b
Procedure	SLIS	BLM89a

Recursion and Bottom-Up Computation

Top-Down produces answers one by one.

Bottom-Up produces all answers at once.

Computational engine

Hierarchical programs: Relational algebra.

Recursive programs: Compute *least fixpoint* of relational equations.

Optimization: *Magic Sets* restrict the computation to what is needed for the query.

Disjunctive Deductive Databases

A disjunctive deductive database consists of clauses of the form:

$$A_1 \vee \cdots \vee A_k \leftarrow B_1, \ldots, B_n, \textit{ not } D_1, \ldots, \textit{ not } D_m$$

Example

$\text{stealth}(X) \vee \text{obstructed}(X) \leftarrow \text{radar}(X), \text{faint}(X)$
$\text{plane}(X) \leftarrow \text{radar}(X), \textit{ not } \text{faint}(X)$
$\text{radar}(3)$
$\text{radar}(1) \vee \text{radar}(2)$
$\text{faint}(1) \vee \text{radar}(2)$

Meaning:

$\{\text{radar}(3), \text{radar}(2), \text{plane}(3), \text{plane}(2)\}$
$\{\text{radar}(3), \text{radar}(1), \text{faint}(1), \text{plane}(3), \text{stealth}(1)\}$
$\{\text{radar}(3), \text{radar}(1), \text{faint}(1), \text{plane}(3), \text{obstructed}(1)\}$

Queries and Answers

Query	Answers
$\text{radar}(X)$	$\text{radar}(3)$ $\text{radar}(1) \vee \text{radar}(2)$
$\text{faint}(X) \vee \text{plane}(Y)$	$\text{plane}(3)$ $\text{faint}(1) \vee \text{plane}(2)$

Disjunctive Semantics

Semantics	Horn		Disjunctive	
	Theory	Reference	Theory	Reference
Positive Consequences				
Fixpoint	T_P	vEK76	T_P^I	MR90
Model	Least Model	vEK76	Min. Model	Min82
			Model-State	LMR89
Procedure	SLD	Hill74	SLO	LMR89
Negation				
Theory	CWA	Rei78	GCWA	Min82,MR90
			WGCWA	Ross87, RLM87
Rule	NAF	Cla78	SN	MR88
			NAFFD	RLM87
Procedure	SLDNF	Cla78	SLONF	MRL89
Stratified Programs				
Fixpoint	T_P	ABW88	T_P^C	MR89
			T_P^I	Ross87,MR89
Model	Standard	ABW88	Stable State	MR89
	Perfect	Prz88	Perfect	Prz88
Normal Programs				
	Well-Founded		Strong/Weak Well-F/Stationary	
Fixpoint	I^∞	VRS88b		
Model	$M_{WF}(P)$	VRS88b	$M_{WF}^{S/W}(P)$	Ross89a
			M_P	Przy90
Procedure	SLS	Ross89b/Przy89b		
	General Well-Founded		General Disjunctive Well-Founded	
Fixpoint	I^E	BLM89b	S^{ED}	BLM90
Model	M_P^E	BLM89b	MS_P^{ED}	BLM90
Procedure	SLIS	BLM89a	SLIS	Prz90

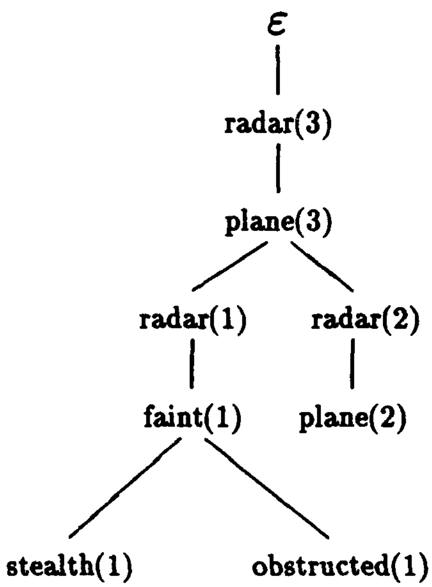
Disjunctive Bottom-Up Computation

Problem How to represent the minimal models?

Minimal models

$\{radar(3), radar(2), plane(3), plane(2)\}$
$\{radar(3), radar(1), faint(1), plane(3), stealth(1)\}$
$\{radar(3), radar(1), faint(1), plane(3), obstructed(1)\}$

Model-tree



Disjunctive Bottom-Up Computation

Problem How to represent the minimal models?

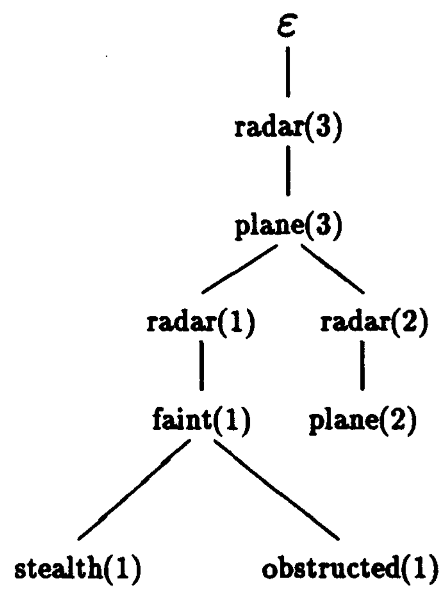
Minimal models

{radar(3), radar(2), plane(3), plane(2)}
{radar(3), radar(1), faint(1), plane(3), stealth(1)}
{radar(3), radar(1), faint(1), plane(3), obstructed(1)}

State Model

{ radar(3),
plane(3),
radar(1) ∨ radar(2),
radar(1) ∨ plane(2),
faint(1) ∨ radar(2),
faint(1) ∨ plane(2),
stealth(1) ∨ obstructed(1) ∨ radar(1),
stealth(1) ∨ obstructed(1) ∨ plane(2) }

Model-tree



Future Work on Disjunctive Deductive Databases

- Algorithms for Stratified and Normal DDDB.

- Magic Sets, or other optimization techniques.

- Algorithms for updating relations and views in a DDDB.

Combining Databases

Problem: Given a set IC of *integrity constraints* and databases DB_1 and DB_2 each one consistent with IC and such that $DB_1 \cup DB_2$ is inconsistent wrt IC.
How can consistency be restored?

Example:

$$\left.\begin{array}{l} DB_1 = \{a; c\} \\ \\ DB_2 = \{b; d \leftarrow a\} \end{array}\right\} \{a; b; c; d \leftarrow a\}$$

$$IC = \{\leftarrow b \wedge d\}$$

Results on Combining

Create a disjunctive database:

$$DB_1 + DB_2 = \{ \begin{array}{l} d \leftarrow a \\ c \\ a \vee b \end{array} \}$$

$$IC = \{ \leftarrow b \wedge d \}$$

$DB_1 + DB_2$ is maximaly consistent with respect to IC.

All minimal models satisfy the IC:

$$\{c, a, d\}$$
$$\{c, b\}$$

The Algorithm combines definite stratified deductive databases.

Future Work on Combining Databases

- Algorithms for First Order theories and Disjunctive databases.

- Theories with defaults
  - Auto-epistemic,
  - Stable Clases,
  - etc.

- Prioritizing information.

- Aplication to view updates.

APPENDIX D

AFOSR/Rome Laboratory
Intelligent Information Workshop

Intelligent Databases for Planning and Scheduling

Rebecca Davis, Rich Fritzson, Robin McEntire
Don McKay, Jon Pastor, and Barry Silk
Unisys Center for Advanced Information Technology

Tim Finin
Univerity of Maryland, Baltimore County

Stu Shapiro and Hans Chalupsky
SUNY Buffalo

October, 1991

Overview

- Focus
- IDI - Intelligent Database Interface
- LIM - Loom Interface Module
- KQML - Knowledge Query and Manipulation Language
- Conclusions

Focus

⇒ Integrating AI systems and Conventional Databases.

⇒ Integrating Knowledge Representation Languages with Relational Databases.

⇒ Interoperability technology to support distributed intelligent systems.

⇒ Technology provider for the DARPA/Rome Planning Initiative for military transportation.

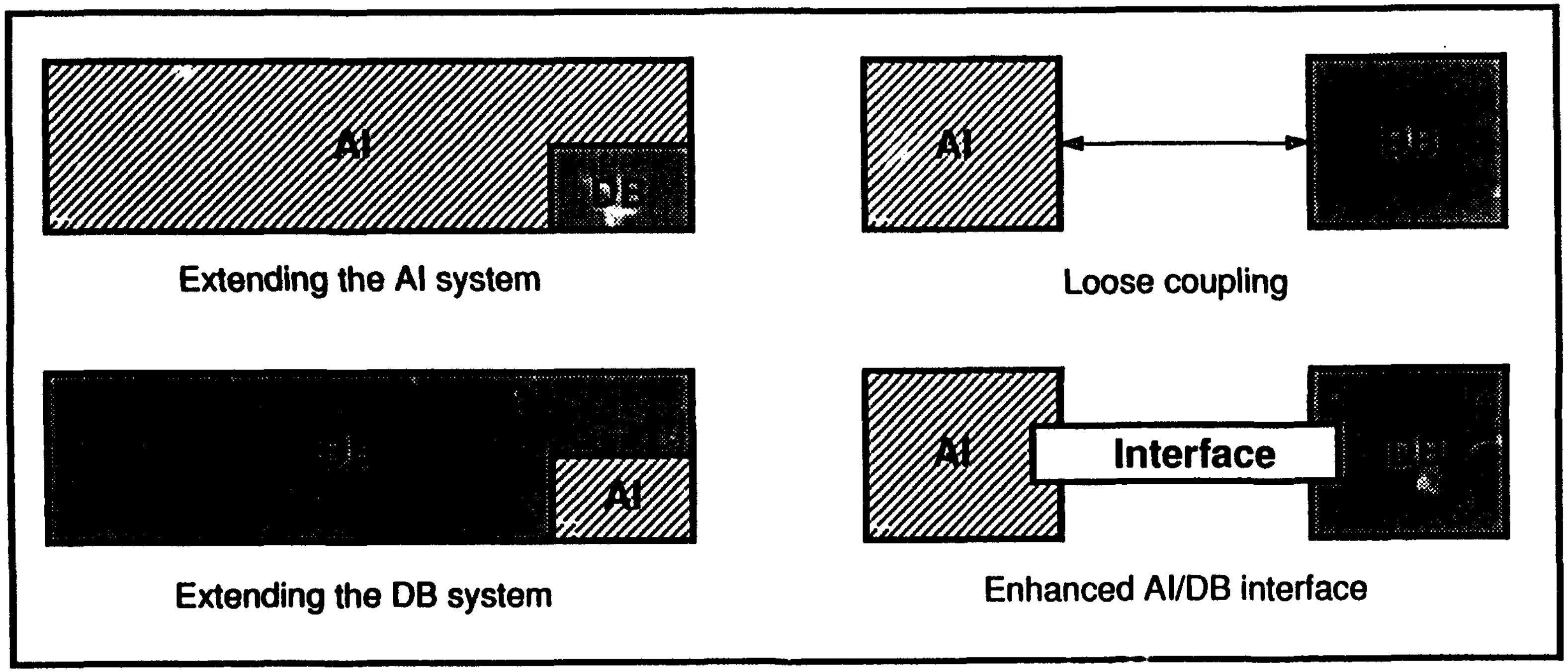
Our Logo

AI/DB Integration Approaches

Intelligent applications will require access to information in databases.

- **Corporate database.**
  The information is already stored and maintained in conventional databases.
- **Database as glue.**
  The DBMS is already being used as the persistent store for a distributed system.
- **Database as workhorse.**
  Very large collections of information may need database technology to for efficient storage and manipulation. We should avoid reinventing it and, if possible, reimplementing it.
- **Databases for sharing.**
  Sharing knowledge bases raises issues of privacy, concurrency, etc. which are addressed and partially solved by database systems.
- **Database as persistent store**
  DBMS technology provides a way to store objects persistently with high reliability.
- **Databases as frontman**
  Customers may feel more comfortable interacting with a DBMS rather than an *"Artificial Intelligence"* system.

AI/DB Integration Approaches



UNISYS Center for Advanced Information Technology

Extending the AI System with DBMS capabilities

- DBMS capabilities *ad hoc* and limited
- *Reimplement/reinvent* DBMS technology
- Lack of access to existing databases

Extending the DBMS System with AI capabilities

- AI capabilities *ad hoc* and limited
- *Reimplement/reinvent* AI technology
- Lack of access to AI applications (and existing databases)

Loose Coupling integrating existing AI and DBMS applications

- Weak access to DBMS
- Weak integration of data with knowledge representation structures
- Data mismatch, e.g., elational data models vs. frames

Enhanced AI/DB Interface

- Existing AI systems and existing DBs and DBMSs
- Transparent access to data stored in DBMS
- Performance achieved through Cache-based interface or server architecture.

Unisys Intelligent Database Interface

- Provides an interface between a logic-oriented query language and SQL databases.
- In use at Unisys and RL/DARPA Prototyping Environment for Knowledge-Based Planning Initiative (soon).
- Intelligent Cache architecture for performance.
- Common Lisp CLOS implementation.
- Server mode.

IDI Architecture — Features

The IDI supports several features which simplify its use in an AI system

- Connections to a DBMS are managed transparently
- Connections to a given database are opened upon demand
- Database schema information loaded automatically
- Logic-based query language (IDIL)
- Programmatic interface
- Results of queries to a DBMS are cached

IDI Architecture — Making Connections

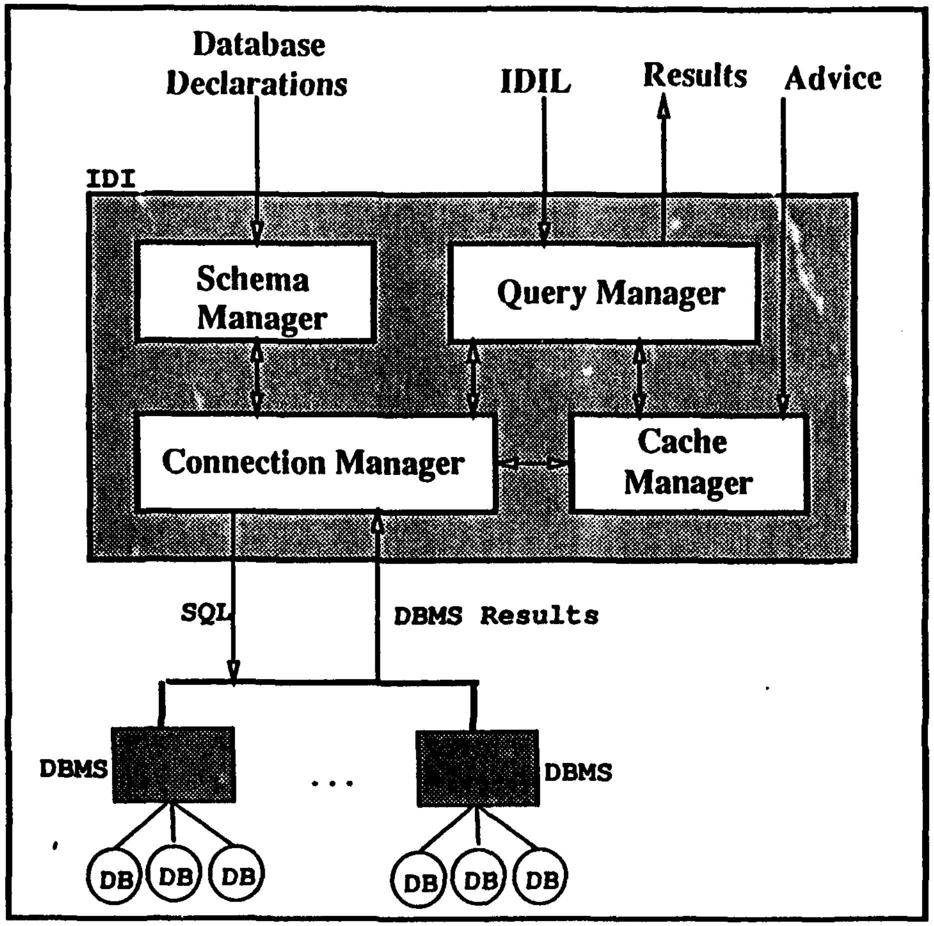
Issue: balance cost of creating DBMS connections & processing query results.

- Few queries with large results; only fraction of results used
  - Stream-based tuple at a time
  - Requires separate connections for each query
- Many queries with small results; most or all results used
  - Single DBMS connection for multiple queries
  - Requires entire result to be processed for each query

The IDI Approach:

- Open DBMS connections upon demand
- Stream-based result processing
- *Database stream* implementation

IDI Architecture — Components



IDI Architecture — Components

- **Schema Manager**

  Reads, processes and stores schema information for each open database

- **DBMS Connection Manager**

  Opens, closes and manages connections to each open database

- **Query Manager**

  Translates IDIL queries to SQL and returns a tuple generator

- **Cache Manager**

  Identifies results for IDIL queries in the cache, caches results, replaces cache elements and accepts *advice*

Architecture — Cache Manager

Uses of Advice:

- Pre-fetching - which relations and when?
- Results caching - which results to be saved?
- Query generalization - which queries should be generalized?
- Replacement strategy - which relations to remove?

Sources of Advice:

- Part of the AI application
- Automatically generated by analysis of the AI system
- Automatically generated by statistical analysis

Architecture — Cache Manager

Cache Validation

- Data copy problem
- Cache validation solutions
  - Only cache non-volatile information
  - Cache between scheduled updates
  - Heuristic caching
  - Snoopy cache
- Stale knowledge problem

IDIL Query Language Examples

Get supplier names for suppliers who do not supply part p2.

```
((ans _Sname)
 <-
 (supplier _Sno _Sname _Status _City)
 (not (supplier_part _Sno "p2" _Qty)))
```

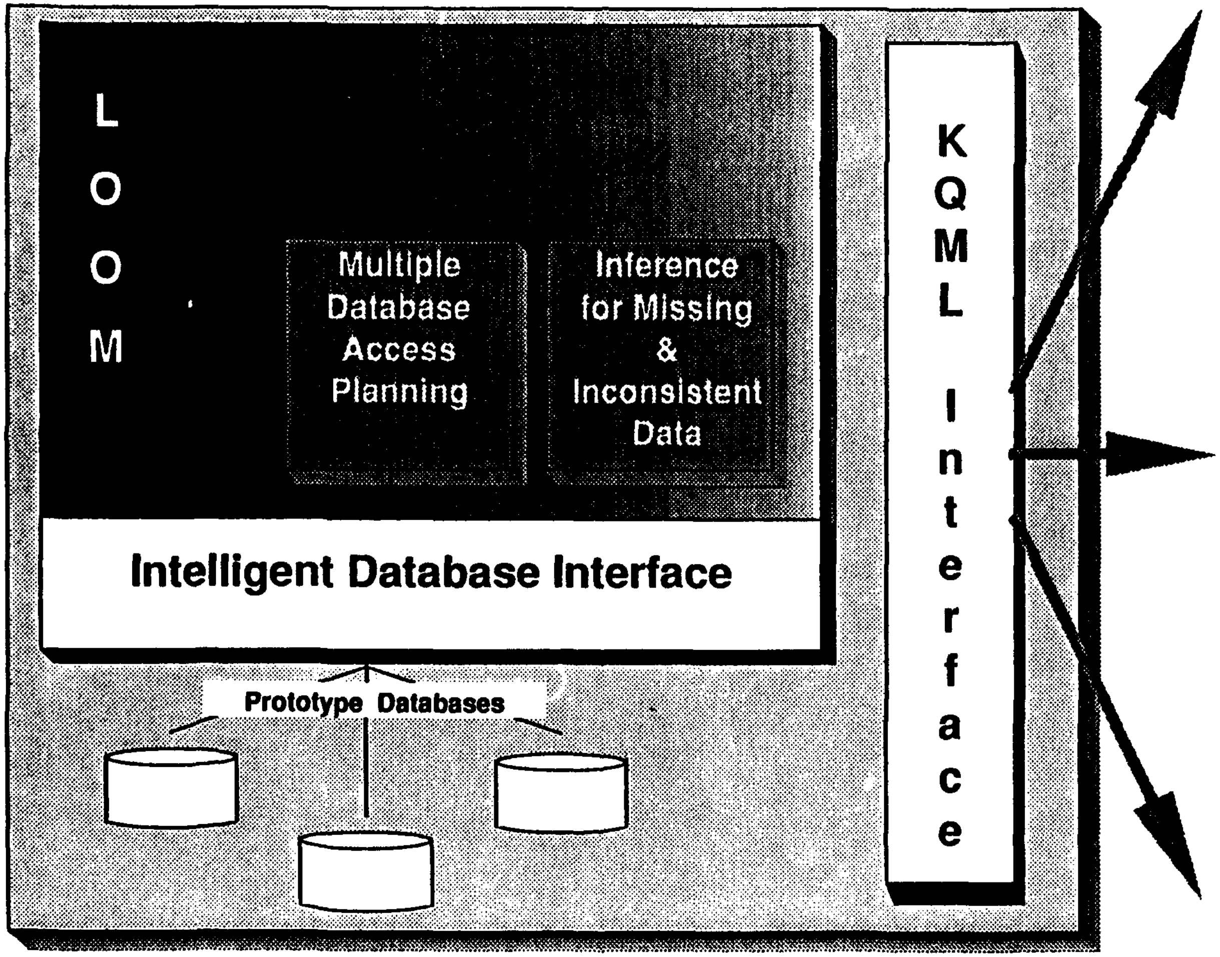
Get supplier names and quantity supplied for suppliers that supply more than 300 units of part p2.

```
((ans _Sname _Qty)
 <-
 (supplier _Sno _Sname _Status _City)
 (supplier_part _Sno "p2" _Qty)
 (> _Qty 300))
```

Integrating LOOM andthe Intelligent Database Interface

- Provides an interface between LOOM and SQL databases.
- Under active development at Unisys.
- Intended for use in the RL/DARPA Prototyping Environment for Knowledge-Based Planning Initiative.
- Transparent interface integrating LOOM A-Box with databases.
- Extension of IDI.
- Common Lisp, CLOS and LOOM implementation.

Loom Interface to the IDI



UNISYS Center for Advanced Information Technology

LIM Architecture — Features

- LOOM model for describing the DB and the DB-KB mapping.
- Extension of LOOM retrieval query language.
- Instantiation of LOOM instances on demand and from cache.
- Lightweight LOOM objects.
- LOOM-specific cache advice.
- Dynamic KB/DB interface.

LIM Overview — Background

Based on:

- View-Object Model (Wiederhold, Barsalou, et al)

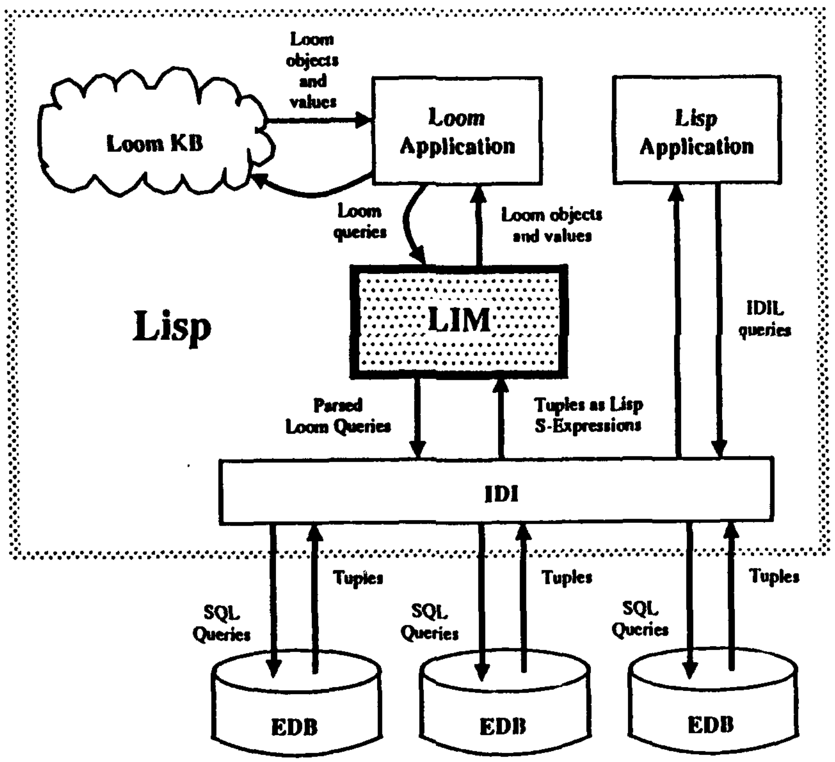
  An extended E/R model used to map from application objects to relational schema supporting efficient algorithms for determining updatability.
- Unisys CYC KB to DB interface (Overton, Pastor, et al)

  Supports object-oriented access from CYC biomedical knowledge base to GenBank database.

LIM:

- Uses LOOM knowledge representation system for rich semantic model.
- *Goal:* Supports transparent access to databases for LOOM application programmer.
- *Goal:* Supports definition of complex application objects based on semantic mapping KB assuring database retrievals and updates.

LIM — System Overview



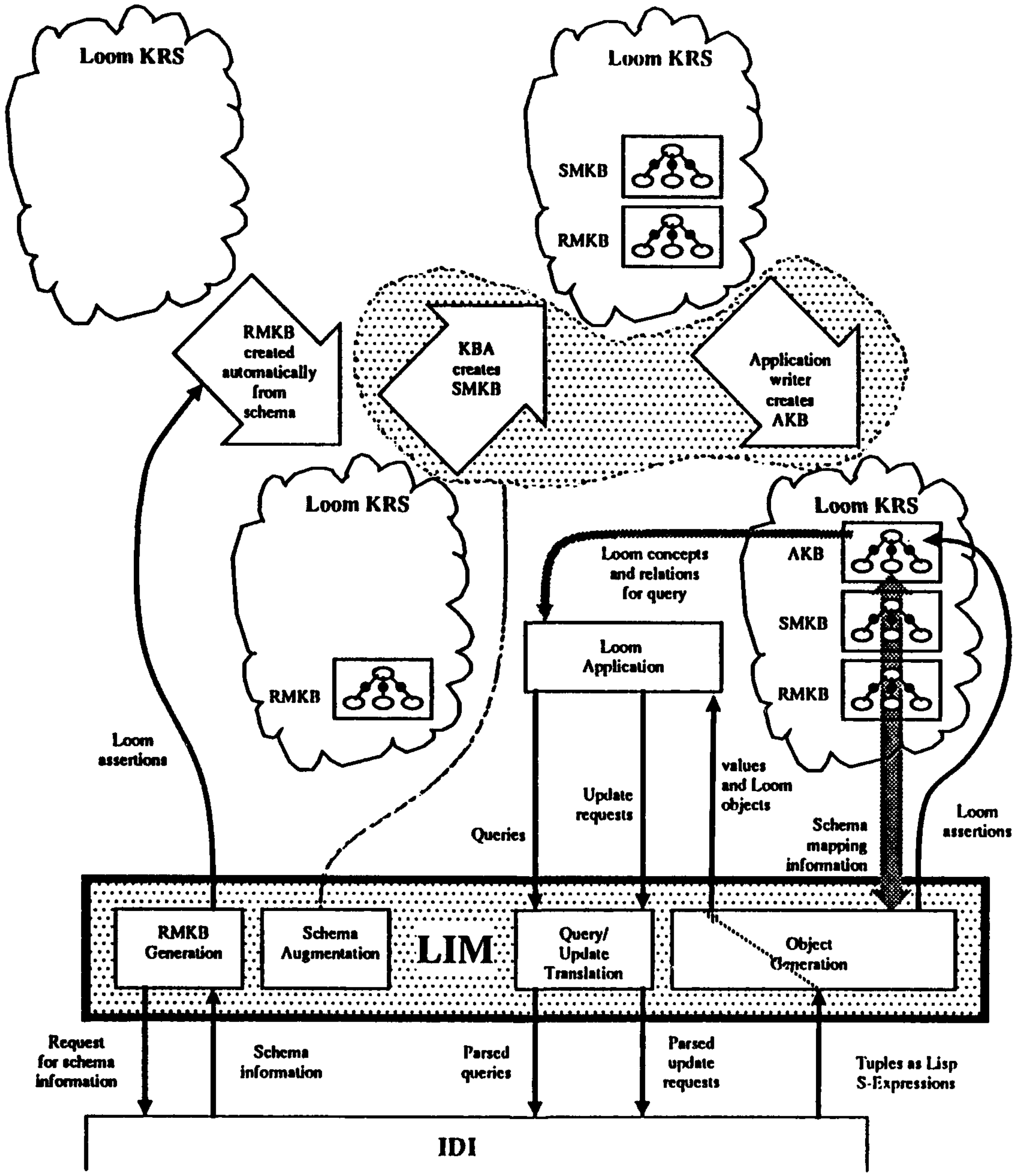
UNISYS Center for Advanced Information Technology

LIM Overview — Operation

- LIM creates Raw Mapping Knowledge Base (RMKB) from database schema.

  One concept per table and one relation per column with mapping information stored on relations
- Knowledge Base Administrator (analogous to DBA) creates Semantic Mapping Knowledge Base (SMKB) by adding semantic information to schema.
- Application developer defines application-level concepts in Application Knowledge Base (AKB).
- Application program accesses database transparently via retrievals in Loom query language referencing concepts in the AKB.
- Queries requesting values do not cause creation of any LOOM instances; queries requesting objects cause creation of LOOM instances.

LIM Operation — Overview



LIM Operation – Details

An application program uses the Loom a-box query language to state queries.

- LIM determines what kind of results are required (i.e., simple values or Loom objects),
- Follows inference paths among the layers of the KB to determine the DB source for each requested query component,
- Generates an SQL query and passes it to the IDI for submission to the DB,
- Receives the returned tuples and performs any instantiation and restructuring required, and
- Returns the results to the application.

LIM Operation – Details *Continued*

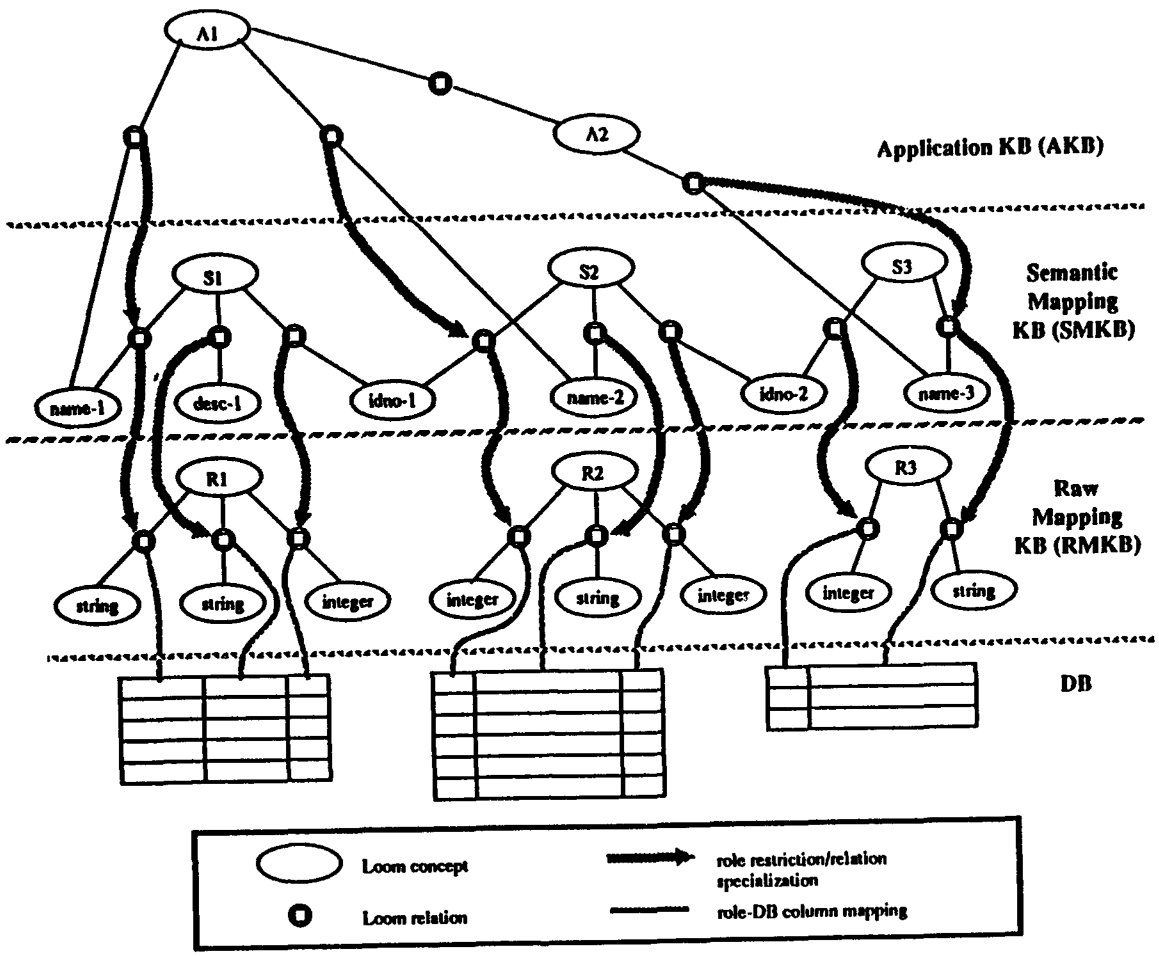
The LOOM application developer does not need to know

- Which of the requested items are resident in the KB and which must be retrieved from the DB,
- In what DB, on which host, DB-resident items are stored,
- Names, types, or other information about the DB structures in which DB-resident items are stored, or
- anything about DBMS access (e.g., SQL)

LOOM objects are only created if specifically requested by the application:

- if values (corresponding to DB column values) are requested, these are returned
- only when Loom instances are requested are instances created.

Knowledge Base Architecture



UNISYS Center for Advanced Information Technology

Creating Mapping and Application KBs

LIM asks the IDI to read the schema, then build a literal representation of it:

- DB column →Loom relation
  for which is asserted the name of the column and the table in which it appears; the name of the relation representing column C in table T is T.C
- DB table →Loom concept
  for which is asserted the DB in which the table resides; the name of the concept representing table T is T.
- DB data types →Loom role restrictions
  For each column C in a table T with DB data type D, a role restriction of the form *(:the C D)* is stated for the concept representing T; this effectively states that the concept T has exactly one C whose type is D.

Result in the RMKB – *Raw Mapping KB* which is isomorphic to the DB schema.

Creating Mapping and Application KBs *Continued*

The KBA uses domain knowledge to introduce domain semantics into schema model by:

- Defining concepts representing semantic types to be used as value (type) restrictions
- Defining concepts specializing the concepts in the RMKB, specializing the role restrictions on these concepts using the semantic types.

Results in SMKB – *Semantic Mapping KB.*

Roles corresponding to columns over which joins are meaningful will have their values restricted to the same semantic type.

SMKB is isomorphic to the DB schema, but has sufficient semantics added to permit the LIM to deduce permissible joins.

Creating Mapping and Application KBs *Continued*

An application writer or KBA uses additional domain knowledge to create Loom concepts (db-views) whose structure more closely approximates the conceptual structures in the domain:

- Columns may be projected out from one or more tables.
- Hierarchical structures may be defined.

During the creation of db-views, LIM interacts with the user to assure that the specified concept is retrievable (all tables from which columns are projected can be joined) and updatable (if required).

Results in the AKB – *Application KB.*

No longer isomorphic to the DB, and may be arbitrarily complex.

LIM — User Interaction

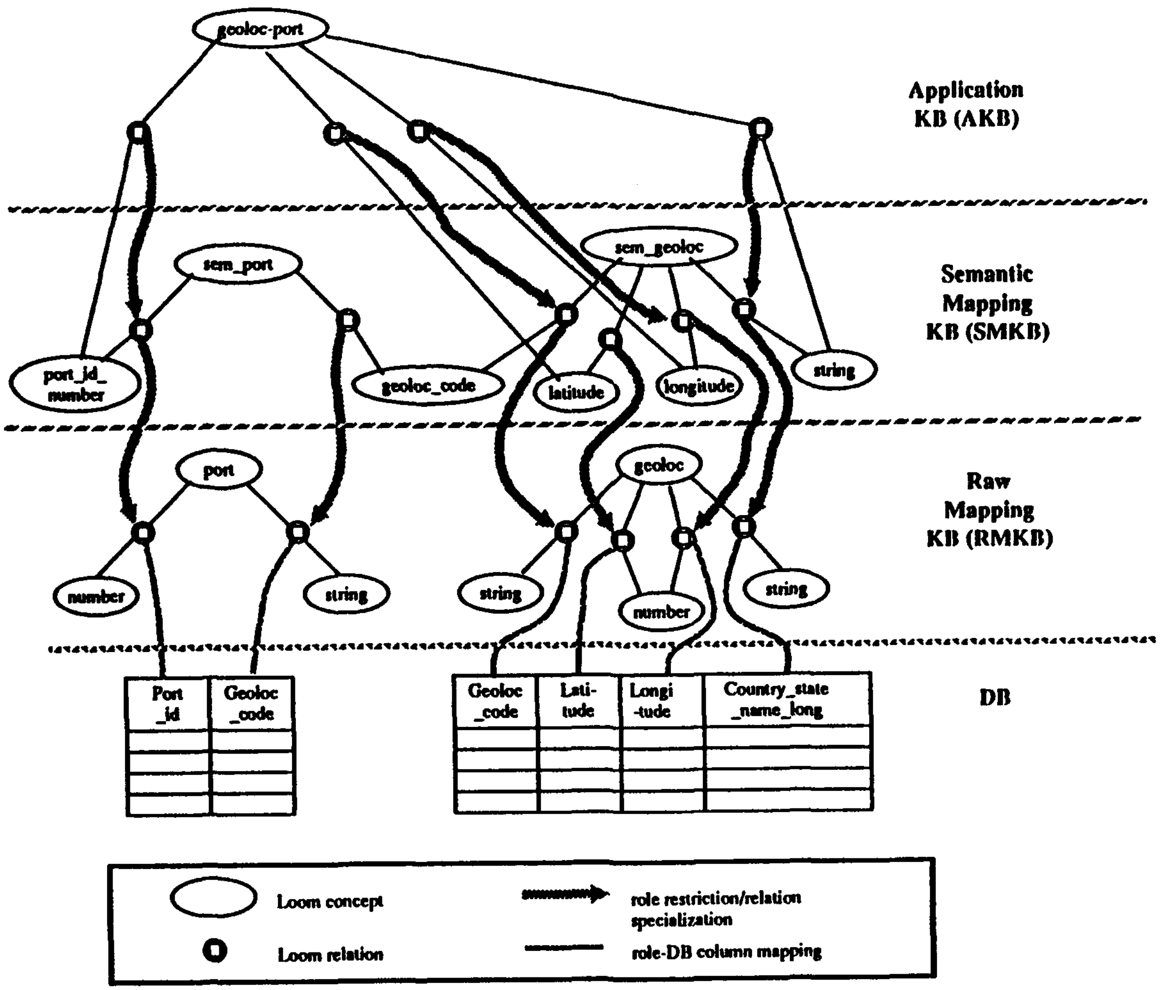
Transparency at definition time: when new concepts are defined, the application developer:

- must specify joins
- must be retrievable
- must state whether updatable
- may have to add information to assure updatability

Transparency at query/update time: tight integration of DB access with normal LOOM retrieval, same syntax (at minimum)

- DB is viewed as extension of LOOM'S A-Box
- location of information irrelevant to user
- lazy instantiation (and light-weight objects)

Example of LIM Use – Sample Knowledge Base



UNISYS Center for Advanced Information Technology

Example of LIM Use – RMKB

Raw Mapping KB concept definition: generated automatically from DB schema

```
(defconcept Geoloc
  :is (:and Db-Concept
            (:the Geoloc.Gsa_County_Code String)
            (:the Geoloc.Gsa_City_Code String)
            (:the Geoloc.Gsa_State_Code String)
            (:the Geoloc.Civil_Aviation_Code String)
            (:the Geoloc.Record_Owner_Uic String)
            (:the Geoloc.Prime_Geoloc_Code String)
            (:the Geoloc.Logistic_Planning_Code String)
            (:the Geoloc.Longitude String)
            (:the Geoloc.Latitude String)
            (:the Geoloc.Province_Name String)
            (:the Geoloc.Province_Code String)
            (:the Geoloc.Country_State_Long_Name String)
            (:the Geoloc.Country_State_Short_Name String)
            (:the Geoloc.Country_State_Code String)
            (:the Geoloc.Installation_Type_Code String)
            (:the Geoloc.Location_Name String)
            (:the Geoloc.Geoloc_Code String))
  :annotations (Thing))
```

Example of LIM Use – SMKB

Semantic Mapping KB concept definition: created by KBA using domain semantics

```
(defconcept Sem_Geoloc
  :is-primitive (:and Geoloc
           (:the Geoloc.Gsa_County_Code County_Code)
           (:the Geoloc.Gsa_City_Code City_Code)
           (:the Geoloc.Gsa_State_Code State_Code)
           (:the Geoloc.Civil_Aviation_Code Civil_Aviation_Code)
           (:the Geoloc.Record_Owner_Uic Unit_Identifier_Code)
           (:the Geoloc.Prime_Geoloc_Code Geoloc_Code)
           (:the Geoloc.Logistic_Planning_Code Logistics_Planning_Code)
           (:the Geoloc.Longitude Longitude)
           (:the Geoloc.Latitude Latitude)
           (:the Geoloc.Province_Name String)
           (:the Geoloc.Province_Code Loc_Code)
           (:the Geoloc.Country_State_Long_Name String)
           (:the Geoloc.Country_State_Short_Name String)
           (:the Geoloc.Country_State_Code Loc_Code)
           (:the Geoloc.Installation_Type_Code Installation_Type_Code)
           (:the Geoloc.Location_Name String)
           (:the Geoloc.Geoloc_Code Geoloc_Code))
  :annotations (Thing))
```

Example of LIM Use – AKB

Application concept: created by KBA or application designers

```
(defconcept geoloc-port
    :is-primitive
        (:and
            Thing
            (:the g-latitude Latitude)
            (:the g-longitude Longitude)
            (:the g-name String)
            (:the g-portno Port_Id_Number)))

(defrelation g-latitude  :is-primitive Geoloc.Latitude)
(defrelation g-longitude :is-primitive Geoloc.Longitude)
(defrelation g-name      :is-primitive Geoloc.Country_State_Name_Long)
(defrelation g-portno    :is-primitive Ports.Port_Number)
```

Example of LIM Use – Queries

Loom DB query: *Find ports in Tunisia, reporting location and port-number.*

```
(db-retrieve (?lat ?long ?portno)
        (geoloc-port ?g-p)
        (g-name ?g-p "Tunisia")
        (g-latitude ?g-p ?lat)
        (g-longitude ?g-p ?longitude)
        (g-portno ?g-p ?portno))
```

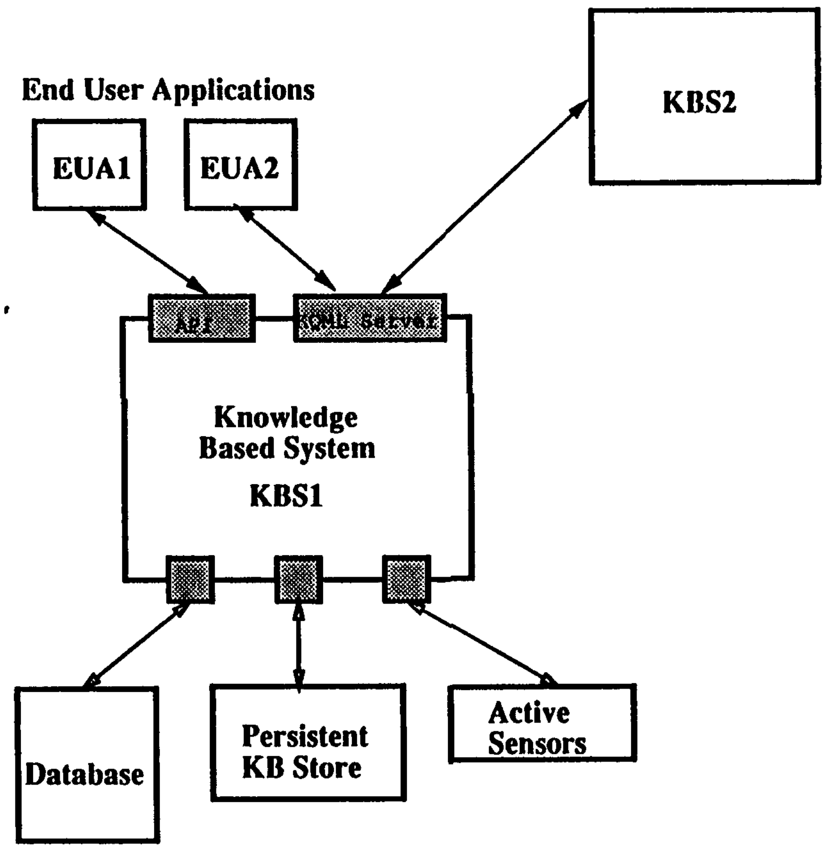
Loom DB query: *Find geoloc-ports in Tunisia.*

```
(db-retrieve (?g-p)
        (geoloc-port ?g-p)
        (g-name ?g-p "Tunisia"))
```

Knowledge Sharing – Integrating AI Systems

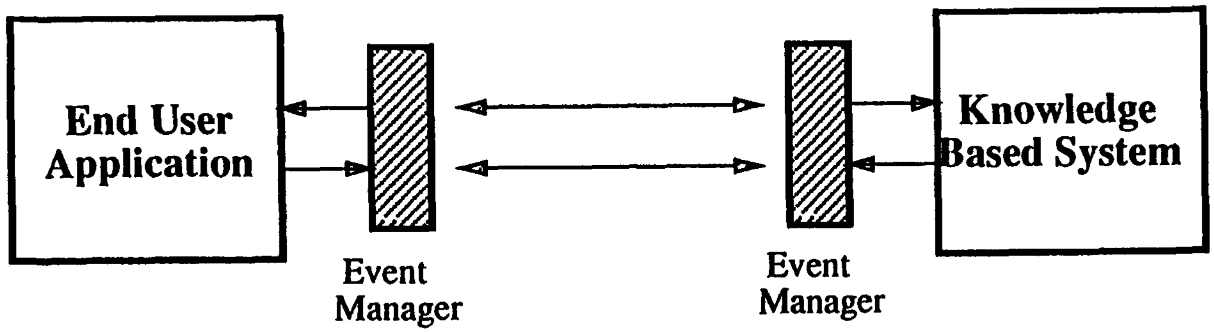
- DARPA sponsored *Knowledge Sharing Effort* goals:
  - Bigger, Better, Cheaper systems.
  - Knowledge sharing and knowledge reuse.
  - Integrating Intelligent Systems with other complex software systems — issues of interoperability, client/server architectures, open systems, standards, etc.
- "Run-time" sharing between multiple intelligent agents.
- Requires communication conventions or protocols
- Specialized *Communication Facilitators* help application developers.

Overview of Interfaces



UNISYS Center for Advanced Information Technology

Event Driven Communication Architecture



A loose coupling between components which can support a number of asynchronous communication paradigms or protocols.

- Communication with a KB potentially richer than with a DB (e.g. need for callbacks, approximate or partial answers, etc.)
- Use a *speech act* model for possible "locutions" and their responses.
- E.g.: A *query* can result in a embedded *query/response* dialog.
- E.g.: A *query* can result in a sequence of *assertions* in response.

The KQML Interface

- **Knowledge Query and Manipulation Language**
- "The SQL of Knowledge Based Systems" ?
- A high-level, portable alternative to the system dependent API.

⇒ Based on a *protocol stack*, supporting event-driven asynchronous communication paradigm.

⇒ Content layer can be expressed in the *Interlingua* or some other KR language.

Basic KQML Functions

Basic TELL and ASK level functions:

```
(assert <s>)       (about <obj>)     (new-kb)
(query <s>)        (load <kb>)       (current-kb <kb>)
(retract <s>)      (clear)
(eval <expr>)      (write <kb>)
(define <s>)       ...
```

Where < *s* > is a sentence in the content language (e.g. *Interlingua*).

Optional arguments used to control the reasoning and behavior of the KRS and what is "returned".

Interlingua

Designed as an *interchange format* rather than a language.

Based on *first order logic* with (possible) extensions.

A company is a kind of business entity with at least one officer. Known companies include XYZ_CORP and ABC_CORP. All of a company's officers are people, subsidiaries of companies are themselves companies, and for any given company C and any given subsidiary S of C, C is an owner of S.

```
(define_prim_relation COMPANY (?c)
    (and (BUSINESS_ENTITY ?c)  (at_least 1 (OFFICER ?c))))

(COMPANY XYZ_CORP)

(COMPANY ABC_CORP)

(forall (COMPANY ?c)
    (and (all (OFFICER ?c) (PERSON))
         (all (SUBSIDIARY ?c) (COMPANY))
         (forall ?s (implies (SUBSIDIARY ?c ?s)
                             (OWNED_BY ?s ?c)))))
```

Optional Arguments

- There are many suggestions for the *optional arguments* which vary in complexity and how well they are "understood".
- Not all will be supported by or relevant to all KRSs.

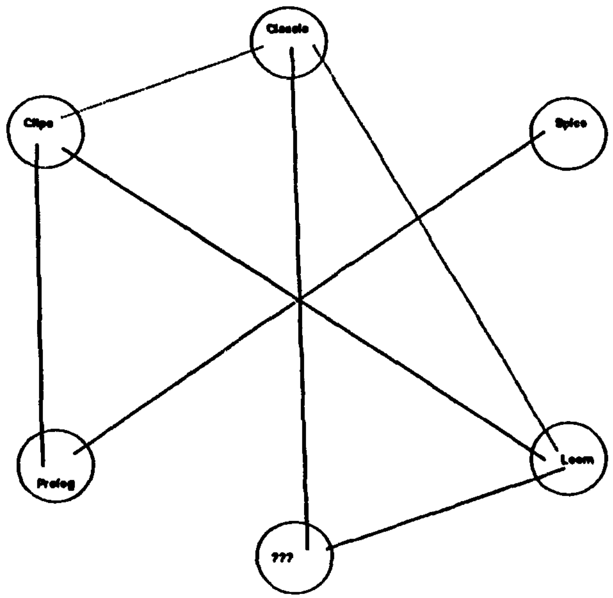
- VOLUME !∞— Control or inquire into the expected volumes in terms of bytes of the IK representation. A bound will packetize the IK result into multiple segments for execution.
- SIZE !∞— Control or inquire into the expected number of IK *statements*. A bound on *requests* will *similarly cause that multiple* executions may be needed to retrieve all of the IK.
- *TIMECOST !∞— Set a bound or inquire into the temporal delay before any IK results will arrive.*
- SCOPE !all — This parameter is intended to control or inquire about the scope of the theory represented by the IK referenced. Its specific operation will depend greatly on the higher-order facilities of the Interlingua.
- ORDER !unordered — Specify or inquire into the ordering applied to the IKs being obtained.
- (!NO) META ONLY — Return (ASK) only the schema of the IK specified, without any actual IK values. This implementation will depend greatly on the capability of the Interlingua and the KBS, and should return any information that is not strictly first-order logic.
- (!NO) TYPE — Only return (ASK) the type hierarchy or lattice of the IK specified, not its values.
- (NO) !SUBSUMPTION — Return (ASK) all subsumed predicates (, or eliminate subsumed predicates from result (ASK) or KBR (TELL))
- (NO)!DUPLICATES — Indicate that duplicates are or are not acceptable in the result (ASK) or in the KBR (TELL).
- (NO)!CONTRADICTIONS — Indicate that contradiction are or are not acceptable in the result (ASK) or in the KBR (TELL).
- (!NO)SUBSUMPTION — Indicate the (no) check is to be made if IK statements subsume each other in the result (ASK) or in the KBR (TELL).
- ...

Communicating Agents – Current Practice

Not only do systems operate and communicate in different languages, but they follow different protocols for communicating.

⇒ Communication is hard and each application has to deal with it independently.

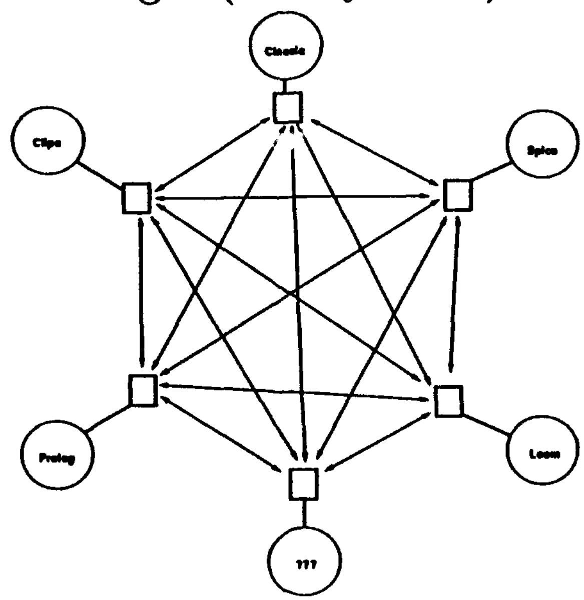
⇒ Having a multitude of protocols makes a hard problem worse.



UNISYS Center for Advanced Information Technology

Communication Facilitators

- Facilitators speak one Protocol (not one language) among themselves.
- Communication between processes is handled by a pair of facilitators.
- Each application system only has to know how to communicate with a facilitator to transmit messages (in any KRL) to another system.

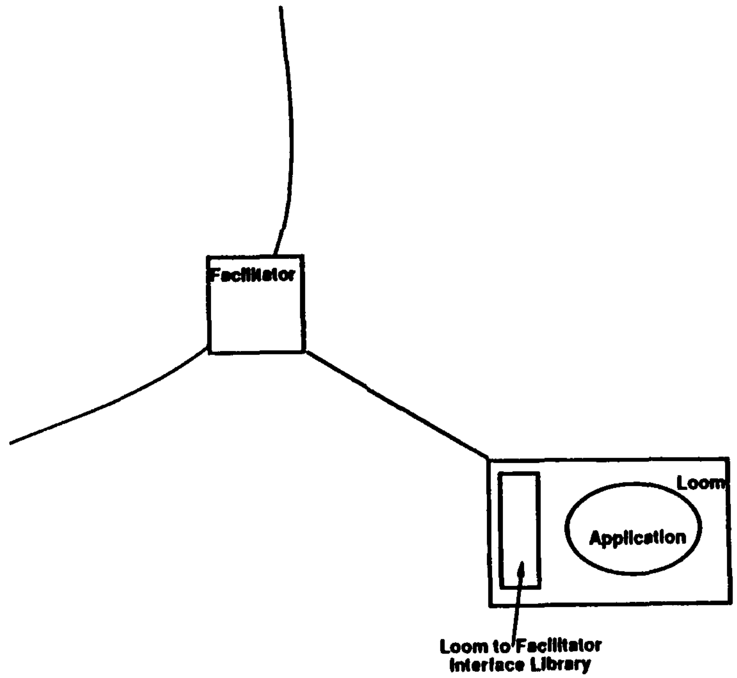


UNISYS Center for Advanced Information Technology

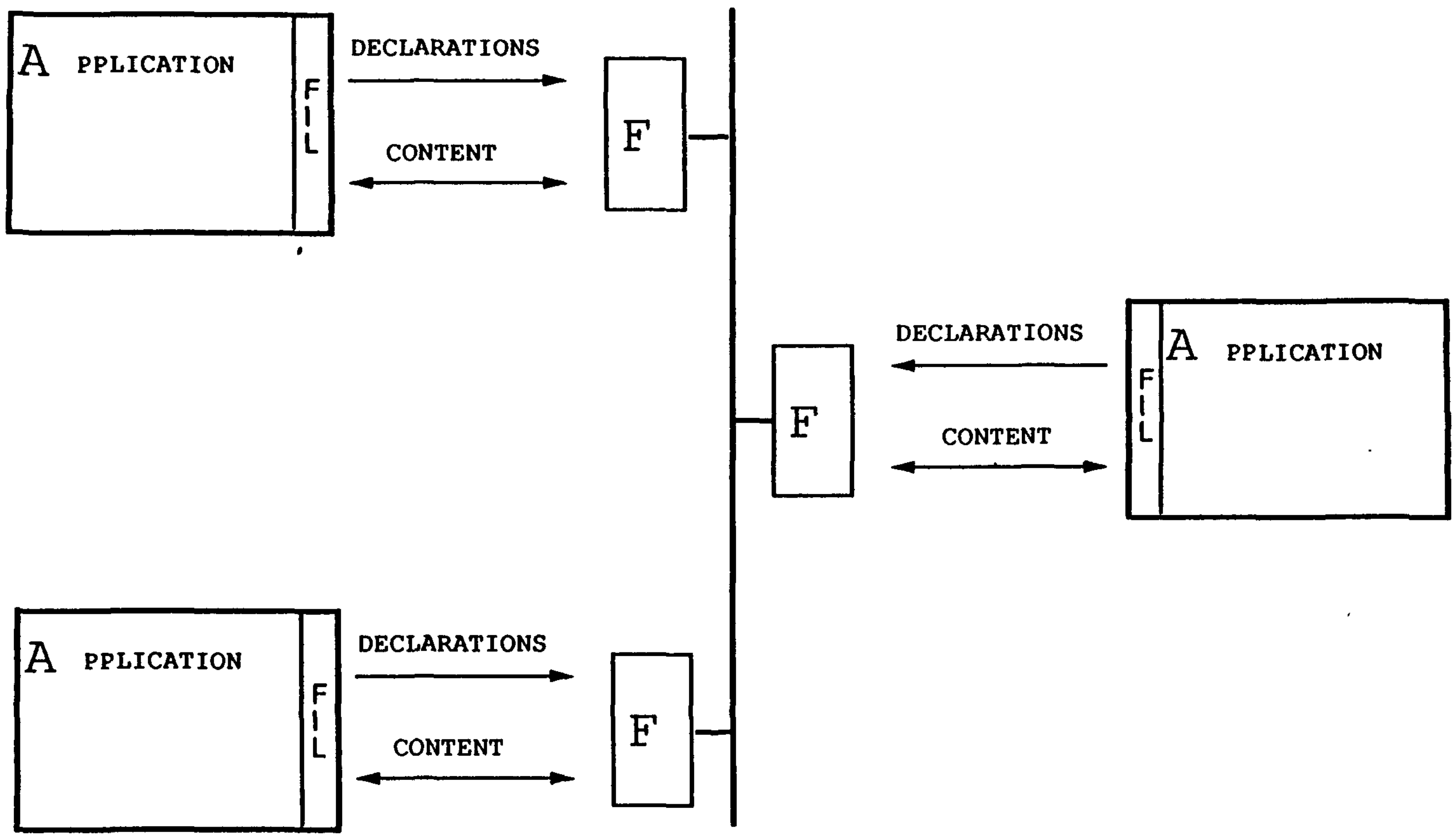
Facilitator — Application

An application is implemented within a "system" (e.g., LOOM).

A system specific library handles the interface to the facilitator.

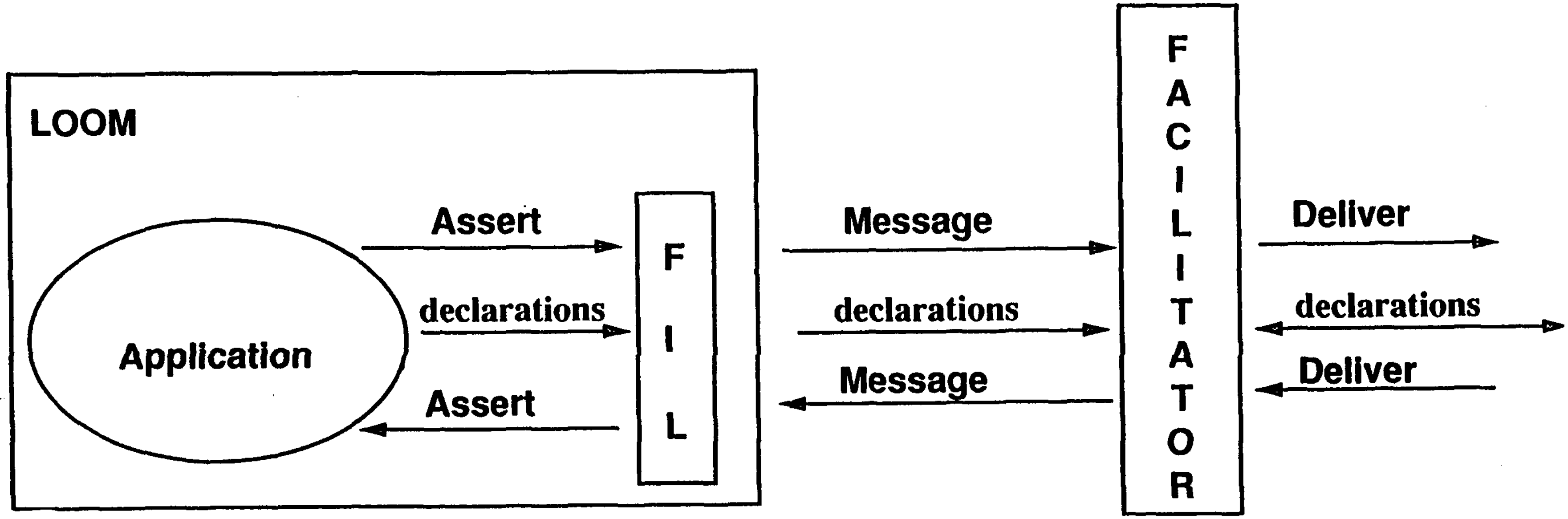


A Network of Communicating Processes



UNISYS Center for Advanced Information Technology

Protocol Organization



- Applications send/receive statements in a KRL of various standard types (e.g. assertions, queries, definitions, ...)
- The Facilitator Interface Library (FIL) packages these as standard "messages" to be handled and routed by the facilitator network.
- Facilitators are completely independent of the systems they service. They can interface to any type of KR system. The FIL connects a particular system (not application) to a facilitator.

UNISYS Center for Advanced Information Technology

Applications send messages to Facilitators

A message describes the statement being transmitted:

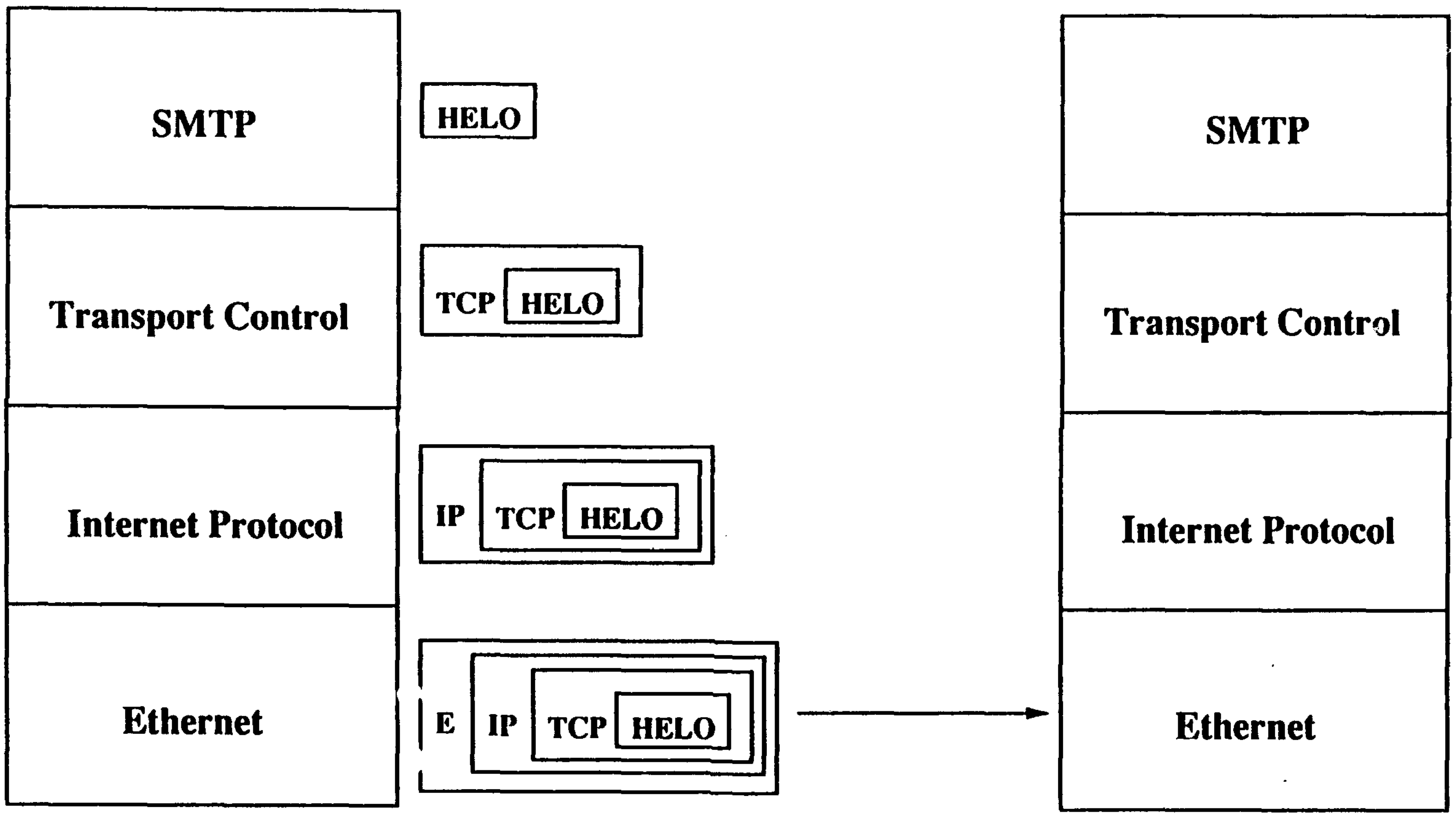
- **Type:** assertion, query, definition, etc.
- **Protocol:** specifies inference, type of responses desired, etc.
- **Topic:** description in some expressive language (NL?)
- **Language:** e.g., Interlingua, LOOM, Prolog, etc.
- **From:** sender identification.
- **To:** recipient description –
  - unique identifier of the recipient.
  - a description of the potential recipients (all systems interested in ..., all systems which know about ...)
  - destination may be implicit; inferred by facilitator using topic, language, type and knowledge of other facilitators.
- etc.

UNISYS Center for Advanced Information Technology

Facilitators traffic in packages

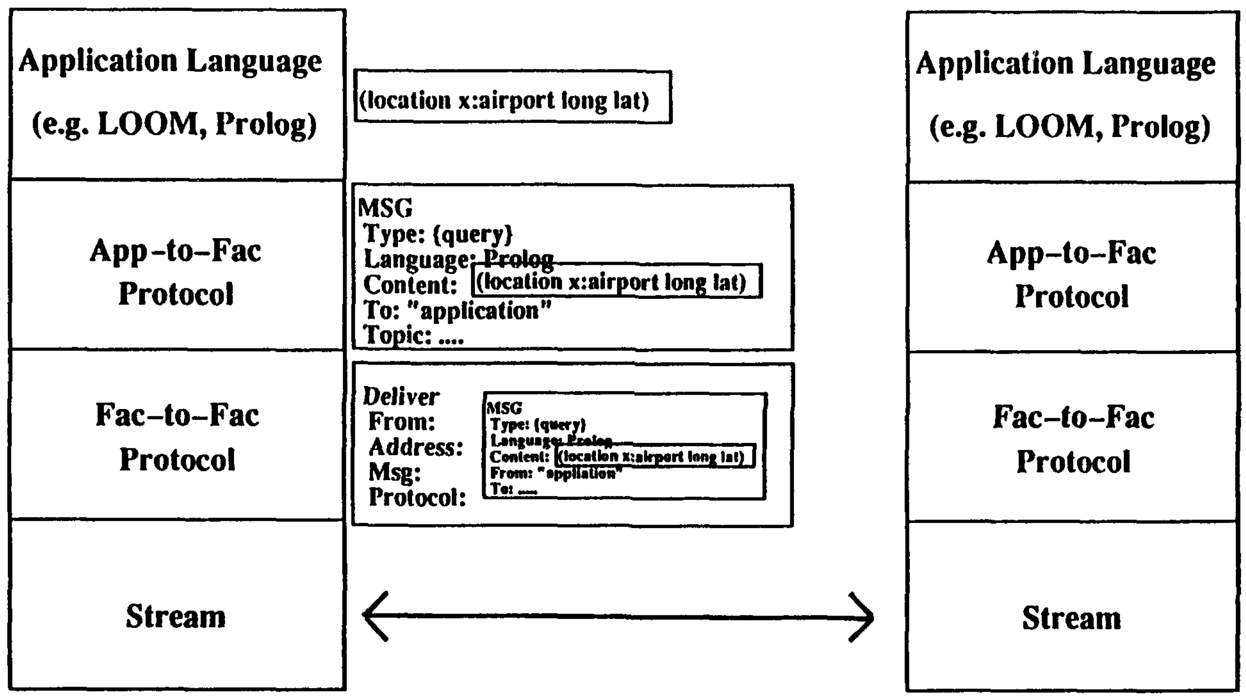
```
(DELIVER
 :FROM        faciltator-128-126-7-41
 :TO          application-128-126-7-41-2000
 :PROTOCOL    drop-ship
 :MESSAGE     (MESSAGE
               :TYPE        assertion
               :PROTOCOL    (inform-of-contradictions
                             unique)
               :LANGUAGE    typed-prolog
               :TOPIC       ("airports, location of"
                             "USTRANSCOM locations")
               :ONTOLOGY    (geoloc ustranscom)
               :CONTENT     (location bwi (124.12 38.36))))
```

Simplified Protocol Stack



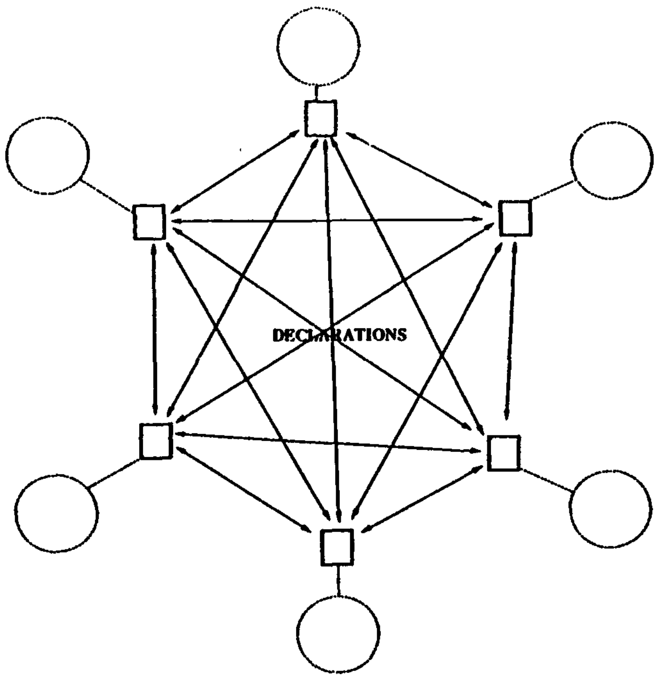
UNISYS Center for Advanced Information Technology

Knowledge Protocol Stack



UNISYS Center for Advanced Information Technology

Facilitator – Facilitator Talk



- *Facilitators exchange descriptions of what their applications are looking for and can provide.*
- *These declarations are not transmitted to applications, but are used to build databases describing the resources and consumers on the network.*

UNISYS Center for Advanced Information Technology

Application to Facilitator Declarations

These application system dependent patterns describe types of statements which the FIL should assist with. E.g.

```
(DCL
 :FROM       application-128-126-7-41-2000
 :DIRECTION  import
 :MESSAGE    (MESSSAGE
               :TYPE      query
               :LANGUAGE  typed-prolog
               :TOPIC     ("airports, location of"
                           "USTRANSCOM locations")
               :ONTOLOGY  (geoloc ustranscom)
               :CONTENT   (location ?x:airport ?y:location))
```

This declaration tells the FIL that the application is willing to accept queries about the location of airports, provided they are presented in "typed prolog" syntax using the given ontologies.

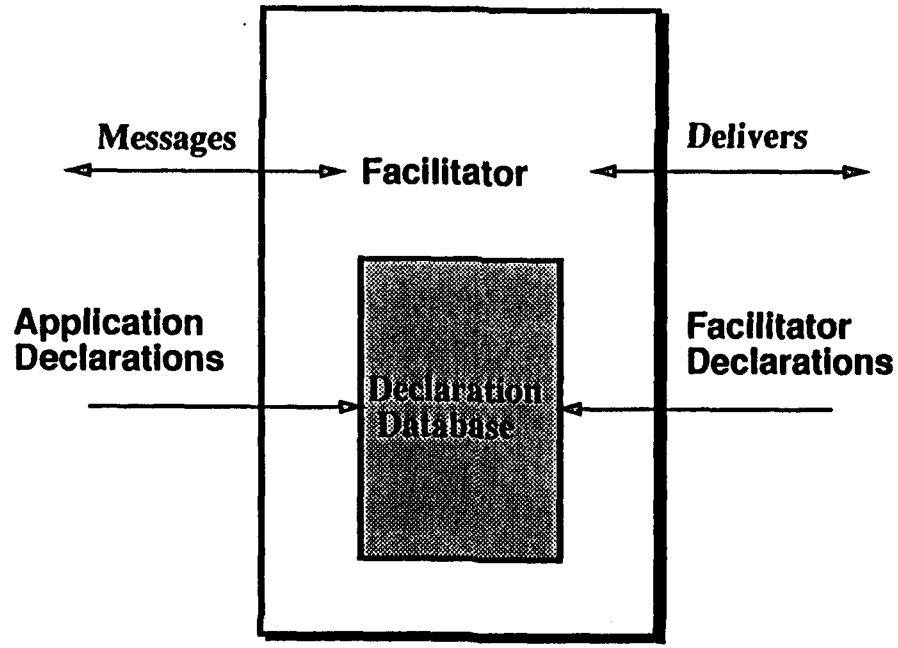
Facilitator to Facilitator Declarations

The facilitator uses a query of this type to advertise the availability of the service:

```
(DCL
 :FROM        facilatator-128-126-7-41
 :TO           (faciltator-128-*  faciltator-130-91-6-5)
 :DIRECTION   import
 :PROTOCOL    drop-and-pick-up
 :MESSAGE     (MESSAGE
               :LANGUAGE   typed-prolog
               :ONTOLOGY   (geoloc ustranscom)
               ...))
```

Possible Implementations of a network database

- local copies.
- centralized.
- replicated database.
- distributed (as in the internet nameserver).



UNISYS Center for Advanced Information Technology

Conclusions

- Strong general need to integrate intelligent systems with conventional software systems.
- Immediate requirement for high-performance integration of intelligent systems and database management systems.
- Strong general need to develop standard interfaces for distributed intelligent systems (e.g. KQML).

APPENDIX E

APPENDIX E

Extracting Rules from Software for Knowledge Bases

Noah S. Prywes, Insup Lee, and Jee-In Kim
Department of Computer and Information Science,
University of Pennsylvania, Philadelphia, PA 19104 *

October 23, 1991

Prepared for the Intelligent Information System Workshop at Rome, NY
on 22-23 Oct. 1991.

Abstract

Many widely used programs contain valuable knowledge of algorithms, processes and methods which could be employed usefully in expert systems. The basic notion is to extract such knowledge from reliable programs, translate it into expert system rules and enter the generated rules into an expert system knowledge base. These programs can provide a reliable and consistent set of rules that search a problem space systematically and assure termination of execution. The expert system can then be employed to provide a user with expertise on the respective algorithms, processes and methods. A visual programming tool can be provided to employ the expert system in testing of the programs.

For example, a program to solve simultaneous equations is used in a wide variety of systems, e.g., for allocation of transportation resources. It can be translated into a set of rules that embody the solution method. The rules can then be loaded into the knowledge base of an expert system. A user of the expert system may input a problem and have the expert system determine the scheduling and allocation of resources, as well as "explain" the decisions by referencing the rules that it has used.

In this manner, it will be possible to automatically tap an immense source of knowledge embedded in existing and future programs to enrich the knowledge base of expert systems.

The translation of a program into expert system rules consists first of translating the procedural language program into an equational (functional) language. The advantages of an equational language are in the arbitrary ordering of equations and no side-effects [Prywes,90]. The equations are translated, one by one, into respective rule definitions used by an expert system. The execution model of an equational language can be graphically portrayed by a data flow or Petri-net diagram. It is similar to the Rete algorithm [Forgy,82] used to schedule execution (pattern matching) of expert system rules.

The paper examines the feasibility of this approach. The approach is demonstrated in translating a MODEL equational language program [Szymanski&Prywes,88] for solving linear simultaneous equations into rules for the CLIPS expert system [Giarratano,89]. The paper then explores use of the expert system and visual programming to test the rules extracted from a program.

*Supported by Contract AFOSR-90-0335A from AFOSR

Contents

1 INTRODUCTION 3

2 THE EXAMPLE 5

3 THE GRAPHICS ENVIRONMENT FOR VISUALIZATION OF EQUATIONAL LANGUAGE 5

4 TRANSLATION OF EQUATIONS INTO RULES 12

5 VISUAL PROGRAM TESTING USING AN EXPERT SYSTEM 12

6 IMPLEMENTATION APPROACH 12

INTRODUCTION

'he underlying notion in this paper is that there exist several computer programming paradigms — each rith its own advantages, and that the translation among these paradigms can benefit each other. The paper roposes combining these translations with a powerful graphics facility for visual programming [Kim,91].

'he paper addresses especially translations among three programming paradigms and their respective anguages, as follows:

(1). **Procedural Languages:** This paradigm is based on the computation model of the sequential computer [Gelernter,90]. It is by far the most developed and popular paradigm with extensive existing libraries of programs.

(2). **Equational Languages:** This paradigm is based on stating a program as a set of equations [Szymanski,91]. The underlying computation model is a Petri-net graph. Computation proceeds as soon as inputs become available. This paradigm is advantageous (a) where a program reliability is important and (b) for parallelizing the computation.

(3). **Rule-Based Languages:** In this paradigm, a program can be developed progressively accumulating rules in a knowledge base. The computation model is based on the Rete pattern matching algorithm [Forgy,82]. This approach is advantageous for (a) accumulating rules in a knowledge base, (b) prototyping, and (c) testing [Giarratano,89]

[he benefits of translation among these paradigms are as follows: The main interest in this paper is in leveloping a methodology for rapidly populating the rules in knowledge bases. There are numerous useful ılgorithms programmed in procedural languages that would be very useful in expert systems. They can)e translated into sets of rules that can be readily incorporated in knowledge bases.

Equally important is that expert systems can offer a superior testing environment for procedural programs;)ecause of their man-machine interface and capability to "explain" the rules used in a computation.

[he approach proposed in this paper is illustrated in Figure 1. The visual programming, translation, ınd testing are embedded in a graphics and repository environment that incorporates visualization and)perations on programs. Programs in respective paradigms are stored in the repository. They can be)perated upon to perform translations. They can be visualized graphically to make them easier to be ınderstood. A user can request a translation from one paradigm to another and benefit from the advantages)f the latter paradigm.

[he focus in this paper is on the translation between Equational and Rule-Based languages. The translation 'rom Procedural to Equational languages has been described in a previous report [Prywes,90].

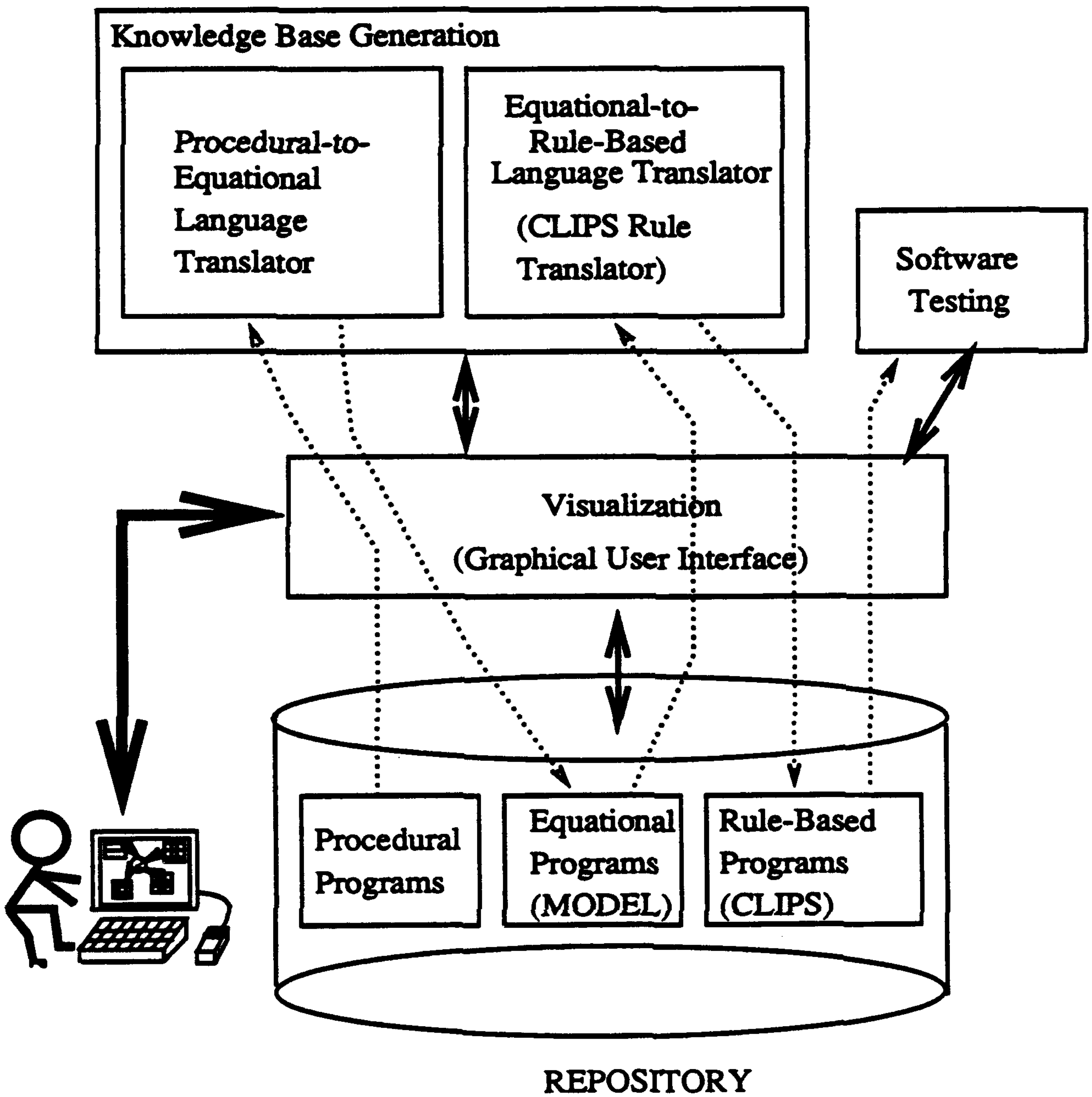
3ecause of limits of space in this paper, the approach is presented through an example. The Gauss Elimination algorithm is used to solve simultaneous linear equations. Solution of simultaneous equations s widely needed in expert systems. Translation of this algorithm to a rule-based language is used here as ın example. The algorithm is described in Section 2.

Section 3 describes the visualization of algorithms in the equational language, MODEL [Szymanski&Prywes,88] ınd the graphic operations for composition and understanding of algorithms.

[he translation from the MODEL equational language into the CLIPS rule-based language [Giarratano,89] s addressed in Section 4.

Section 5 discusses the use of visualization and the CLIPS expert system for program testing.

Figure 1: **Extracting Expertise from Programs, Converting It into Rules, and Accumulating the Rules into a Knowledge Base.**



The paper concludes with the highlights of the implementation plan for a prototype of the environment.

2 THE EXAMPLE

The function of the Gauss Elimination algorithm, used as an example in this paper, is illustrated in Figure 2. The figure illustrates the input and output as sets of simultaneous linear equations. The illustration of testing in Section 5 will utilize the shown set of four equations. The algorithm manipulates the input matrix of coefficients and right hand side constants through successive matrices until it obtains the matrix with the lower left hand triangle of zeroes. The solution for the variables can then be readily obtained.

3 THE GRAPHICS ENVIRONMENT FOR VISUALIZATION C EQUATIONAL LANGUAGE

The visualization of an Equational language program is illustrated in this section. It consists of visualizing separately:

(1). **Header:** a context-like diagram of the algorithm depicting its inputs and outputs (Figure 3).

(2). **Data structure:** depicting the tree-like structure of data with special icons for one, two, and three dimensional array objects. Arrays are the basic data structure in the MODEL Equational language (Figure 4).

(3). **Equations:** depicting a data flow or Petri-net like graph of equations and array variables called "array graph". This graph portrays the dependencies among variables and equations. The subscript expressions of array elements are also shown for each dependency edge (Figure 5 and Figure 6).

Each of this parts consists of a pair of displays — graphics and textual. The two displays are coordinated and consistent. Each graph is called a "view". It can be composed, edited, and visualized through a "window" using the DECdesign methodology [DEDdesign,90]. The palette at the bottom of each window shows the icons for objects in the graph. A circle denotes an equation. A rectangle denotes an array. Shadow arrays denote control structures such as a size of a dimension.

The graphics and textual views are checked to verify the consistency among them. There are also algorithms to check and add details so that the equation can be fully evaluated. This requires through statically checking equations and data chains for causality, for existence of variables and sizes for all dimensions, and for data type compatibility.

Figures 3-6 portray the graphics and texts for the Gauss Elimination example. The figures especially show how the input matrix (`m_i(i,j)`) is translated into a series of matrices that form a cube (`a(i,j,k)`) with associated arrays: `q(i,k)` denotes the non-zero elements and `p(k)` denotes the positions of pivoting elements. Finally, the output (`m_o(i,j)`) is presented. The internal consistency and completeness of the equations can also be checked visually by tracing the dependency edges between variables, equations, and the subscript expressions shown for each edge.

Figure 2: Illustration of Inputs and Outputs to Gauss Elimination.

Input Linear Equations

m_i(1,1)*x1 + m_i(1,2)*x2 + m_i(1,3)*x3 + m_i(1,4)*x4 = m_i(1,5)

m_i(2,1)*x1 + m_i(2,2)*x2 + m_i(2,3)*x3 + m_i(2,4)*x4 = m_i(2,5)

m_i(3,1)*x1 + m_i(3,2)*x2 + m_i(3,3)*x3 + m_i(3,4)*x4 = m_i(3,5)

m_i(4,1)*x1 + m_i(4,2)*x2 + m_i(4,3)*x3 + m_i(4,4)*x4 = m_i(4,5)

↓

Gauss Elimination Algorithm

↓

Output Linear Equations

m_o(1,1)*x1 + m_o(1,2)*x2 + m_o(1,3)*x3 + m_o(1,4)*x4 = m_o(1,5)

m_o(2,2)*x2 + m_o(2,3)*x3 + m_o(2,4)*x4 = m_o(2,5)

m_o(3,3)*x3 + m_o(3,4)*x4 = m_o(3,5)

m_o(4,4)*x4 = m_o(4,5)

igure 3: **Header.**

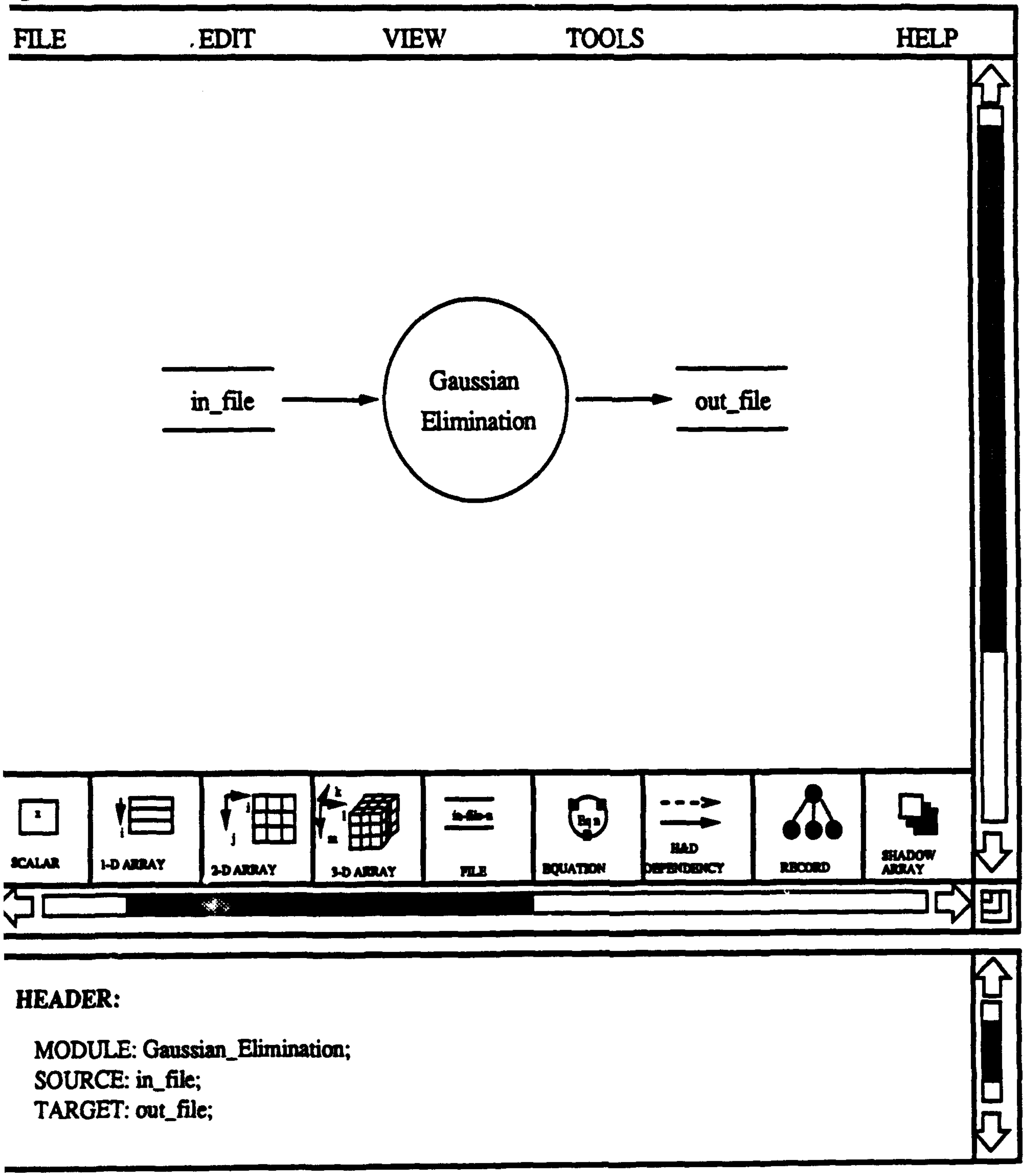
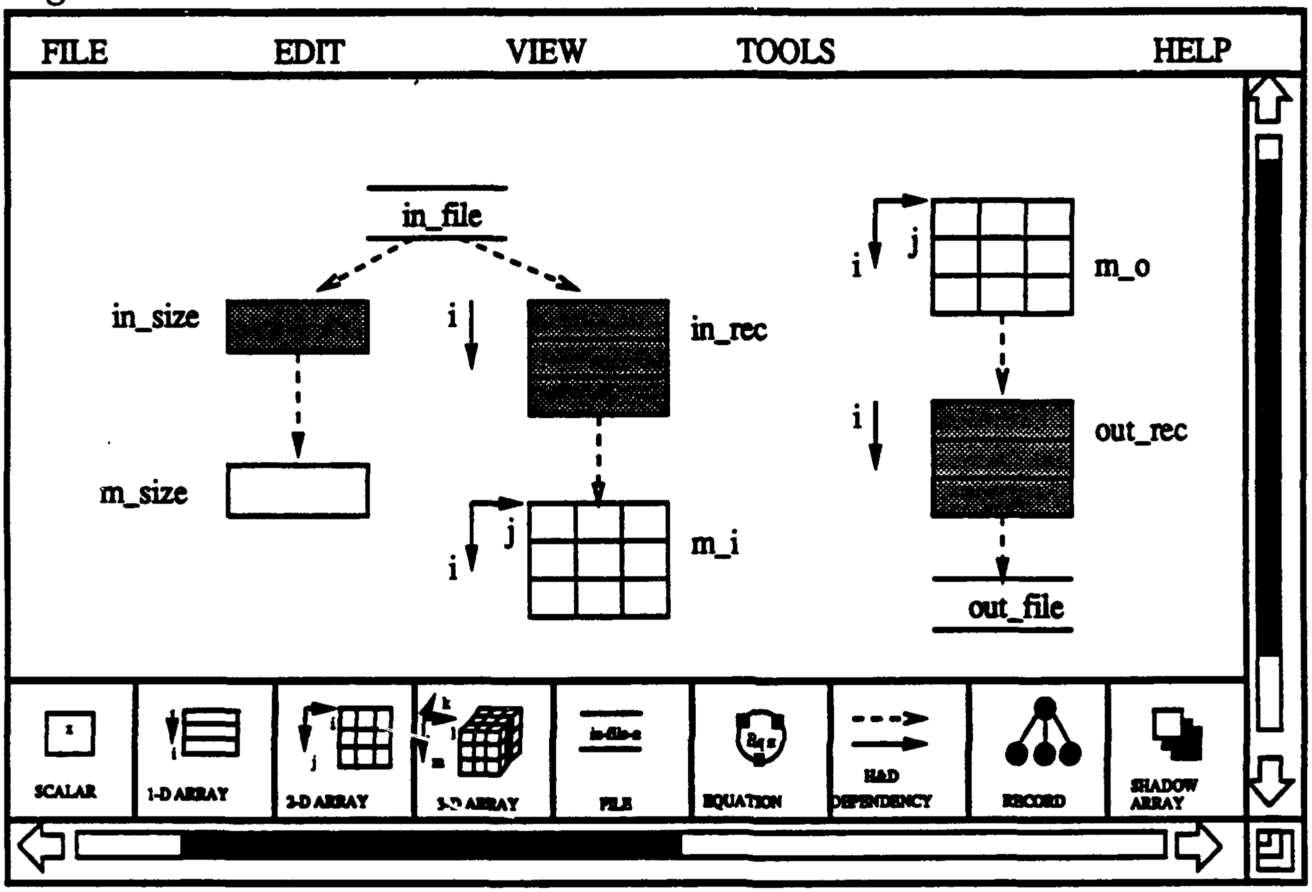
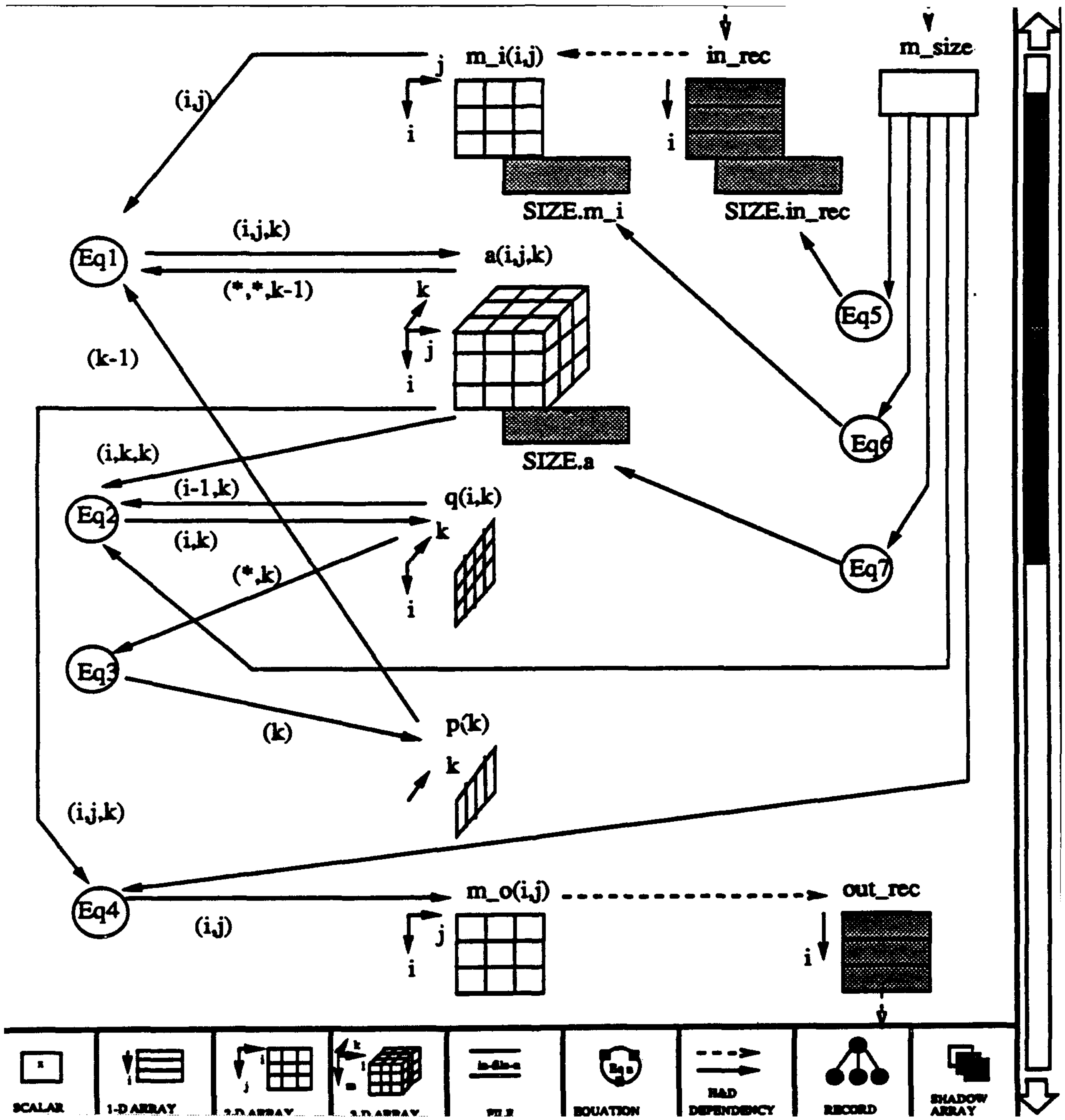


Figure 4: **Data Structures.**



DATA DECLARATIONS:

```
1 in_file IS FILE,            /* input data file */
 2 in_size IS RECORD,
   3 m_size IS FIELD (NUM),      /* size of the input matrix, n */
 2 in_rec(*) IS RECORD,
   3 m_i(*) IS FIELD (FLOAT DEC);  /* input matrix, n x (n+1) */
1 out_file IS FILE,    /* output data file */
 2 out_rec(*) IS RECORD,
   3 m_o(*) IS FIELD (FLOAT DEC));
    /* output matrix is upper-triangular if the input has unique solution,
       otherwise, all elements of the matrix are zero. */
```



m_i(i,j)
in_rec
m_size
(i,j)
j
i
SIZE.m_i
SIZE.in_rec
(i,j,k)
Eq1
a(i,j,k)
(*,*,k-1)
k
Eq5
(k-1)
j
i
Eq6
(i,k,k)
SIZE.a
(i-1,k)
Eq2
q(i,k)
(i,k)
k
(*,k)
Eq7
i
Eq3
(k)
p(k)
k
(i,j,k)
Eq4
m_o(i,j)
out_rec
(i,j)
j
i
SCALAR
1-D ARRAY
EQUATION
R&D DEPENDENCY
RECORD
SHADOW ARRAY

Figure 6: **Equations (Text).**

```
EQUATIONS:

/* Eq1: array elements */
a(i,j,k) = IF k=1 THEN m_i(i,j) /* load input matrix */
        ELSE IF p(k-1)=0 THEN 0 /* no pivoting mean no unique solution */
        ELSE IF i=p(k-1) THEN a(k-1,j,k-1)   /* switch pivoting row p(k-1)with (k-1) row */
        ELSE IF i=(k-1) THEN a(p(k-1),j,k-1)   /*switch (k-1) row with pivoting raw p(k-1) */
        ELSE IF i<(k-1) THEN a(i,j,k-1) /* no need to pivot */
        ELSE a(i,j,k-1)-a(p(k-1),j,k-1)*a(i,k-1,k-1)/a(p(k-1),k-1,k-1);
                                                  /* zeroing the lower left triangle */

/* Eq2: non-zero array elements for pivoting */
q(i,k) = IF i=1 THEN IF a(i,k,k)=0 | k>1 THEN 0 ELSE 1
        ELSE IF i<k THEN 0 ELSE IF a(i,k,k)=0 THEN q(i-1,k) ELSE 1;

/* Eq3: position of pivoting element */
p(k) = IF k=1 & i=1 & q(i,k)=1 THEN i
       ELSE IF i>1 & k<=i THEN IF q(i-1,k)=0 & q(i,k)=1 THEN i
                                ELSE IF q(m_size,k)=0 THEN 0;
                                   /* p(k)=0, if there is no no-zero elements */

/* Eq4: output matrix */
m_o(i,j) = IF k=m_size THEN a(i,j,k);

/* Eq5: size of subscript, i */
SIZE.in_rec = m_size;

/* Eq6: size of subscript, j */
SIZE.m_i = m_size+1;

/* Eq7: size of subscript, k*/
SIZE.a = m_size;
```

igure 7: **Translation of a MODEL Equation into a CLIPS Rule.**

TRANSLATION TO CLIPS RULES:

```
/* Eq1: array elements */
a(i,j,k) = IF k=1 THEN m_i(i,j) /* load input matrix */
        ELSE IF p(k-1)=0 THEN 0 /* no pivoting elements mean no unique solution */
        ELSE IF i=p(k-1) THEN a(k-1,j,k-1)   /* switch pivoting row p(k-1) with (k-1) row */
        ELSE IF i=(k-1) THEN a(p(k-1),j,k-1)   /*switch (k-1) row with pivoting raw p(k-1)*/
        ELSE IF i<(k-1) THEN a(i,j,k-1) /* no need to pivot */
        ELSE a(i,j,k-1)-a(p(k-1),j,k-1)*a(i,k-1,k-1)/a(p(k-1),k-1,k-1);
                                                  /* zeroing the lower left triangle */
===>

/* Rule1: computing a triangular matrix, a(i,j,k) */
(defrule rule1
   (p =(- ?k 1) ?p)          /* does p(k-1) exist? if so, ?p has its value */
   (a =(- ?k 1) ?j =(- ?k 1) ?akj)   /* ?akj = a(k-1,j,k-1) */
   (a ?p ?j =(- ?k 1) ?apj)      /* ?apj = a(p(k-1),j,k-1) */
   (a ?i ?j =(- ?k 1) ?aij)      /* ?aij = a(i,j,k-1) */
   (a ?i =(- ?k 1) =(- ?k 1) ?aik)      /* ?aik = a(i,k-1,k-1) */
   (a ?p =(- ?k 1) =(- ?k 1) ?apj)      /* ?apk = a(p(k-1),k-1,k-1) */
   =>
   (if (= ?p 0) then (assert (a ?i ?j ?k 0))  /* a(i,j,k)=0, i.e., no unique solution */
          else (if (= ?i ?p) then (assert (a ?i ?j ?k ?akj)) /* the pivoting row */
          else (if (= ?i =(- ?k 1)) then (assert (a ?i ?j ?k ?apj)) /* the (k-1)th row */
          else (if (< ?i =(- ?k 1)) then (assert (a ?i ?j ?k ?aij)) /* no pivoting */
          else
             (assert (a ?i ?j ?k =(- ?aij (* ?apj (/ ?aik ?apk))))))))))
                                                /* zeroing the lower left triangle */
```

4 TRANSLATION OF EQUATIONS INTO RULES

Each equation is translated into a respective rule and added to the knowledge base. Figure 7 shows the translation of the equation that defines `a(i,j,k)` into the respective CLIPS rule. As shown, the equational representation is more compact, however it conveys the same information and same computational interpretation.

The translation is based on the following: Computation of the left hand side variable of an equation requires pre-existence of the right hand side variables. This is the basis for the condition part of the rule; namely that the right hand side variables exist. The assertion part of the rule is similar to the equation and it asserts a value for the left hand side variable. As elements of array are asserted as facts in a knowledge base, they are recognized as existing and therefore satisfying the condition part of some rules which can then be interpreted next, and so on. The causality check, previously mentioned, assures completion of the overall computation.

5 VISUAL PROGRAM TESTING USING AN EXPERT SYSTEM

Program testing is viewed as essentially a test of program branches and the synthesis of these tests. A key consideration is the choice of test data [DeMillo,87, Howden,87, Hamlet,88].

Use of an expert system facilitates such a process through the following features:

(1). A user can assert the test data for any of the variables in the program.

(2). Once these variables exist, the expert system may be "fired" to execute "branches" (equation chains) of the program that depend on these variables. This may continue until a user specified breakpoint (stopping point), which limits the computation, is reached.

(3). The expert system together with the graphics system display the values of computed variables, which are shown for the last values of the indices, and "explain" the sequences of the fired rules to obtain those values.

To support this process, the testing environment has a number of operations that can be selected by the user from menus. The breakpoint delimiting the computation is specified by the user pointing with the mouse to selected equations or by key-in of delimiting index values.

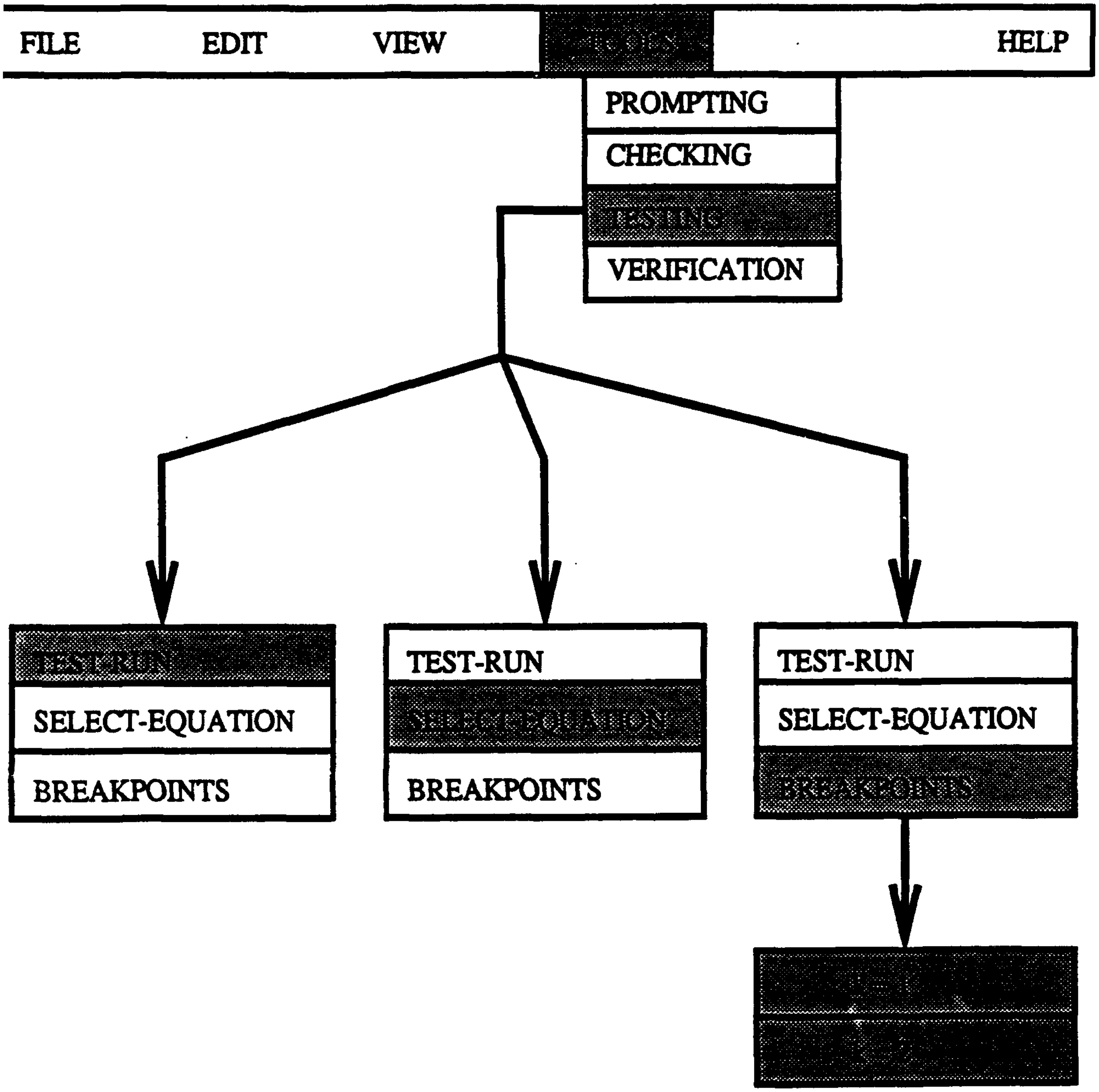
The choice of operations in menus is shown in Figure 8. The exercising of two operations for the Gauss Elimination example is illustrated in Figure 9 and Figure 10, respectively. Figure 9 illustrates not delimiting the testing and allowing it to "fire" from the asserted input to the completed output. It shows asserting the input (`m_i(i,j)`) with the results in eventually evaluating `m_o(i,j)`. Figure 10 illustrates use of breakpoint delimit the computation to `k=2`, namely computing up to the second `(i,j)` matrix of the three dimensional array, `a(i,j,k)`, i.e., `a(i,j,2)`.

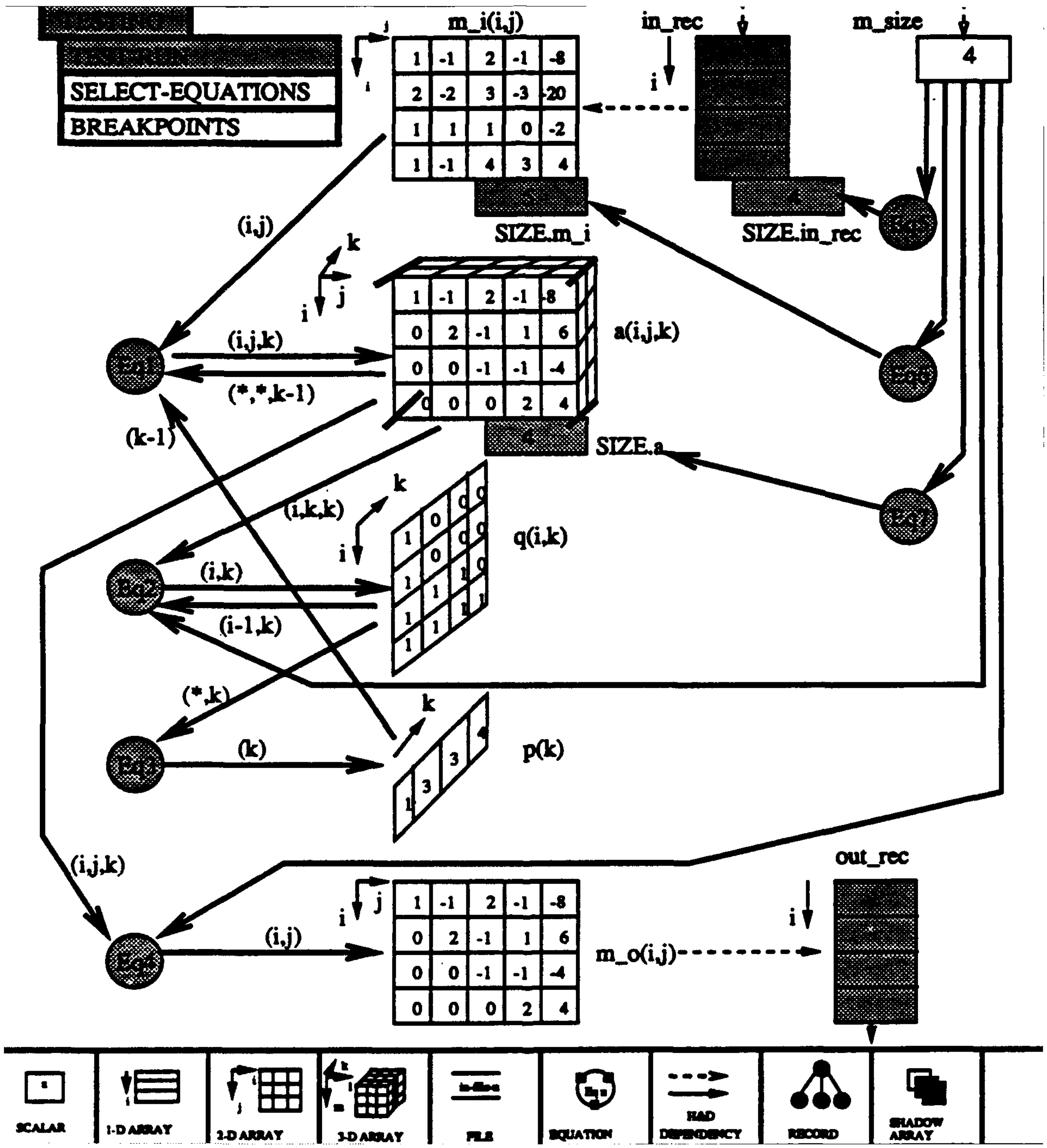
6 IMPLEMENTATION APPROACH

The key features of a prototype system under development are depicted in Figure 11.

As already noted, the choice of languages is:

Figure 8: **Menus and Submenus.**





SELECT-EQUATIONS
BREAKPOINTS
m_i(i,j)
in_rec
m_size
4
SIZE.m_i
SIZE.in_rec
(i,j)
a(i,j,k)
(i,j,k)
(*,*,k-1)
(k-1)
SIZE.a
(i,k,k)
q(i,k)
(i,k)
(i-1,k)
(*,k)
(k)
p(k)
(i,j,k)
out_rec
(i,j)
m_o(i,j)
SCALAR
1-D ARRAY
2-D ARRAY
3-D ARRAY
FILE
EQUATION
HAD DEPENDENCY
RECORD
SHADOW ARRAY

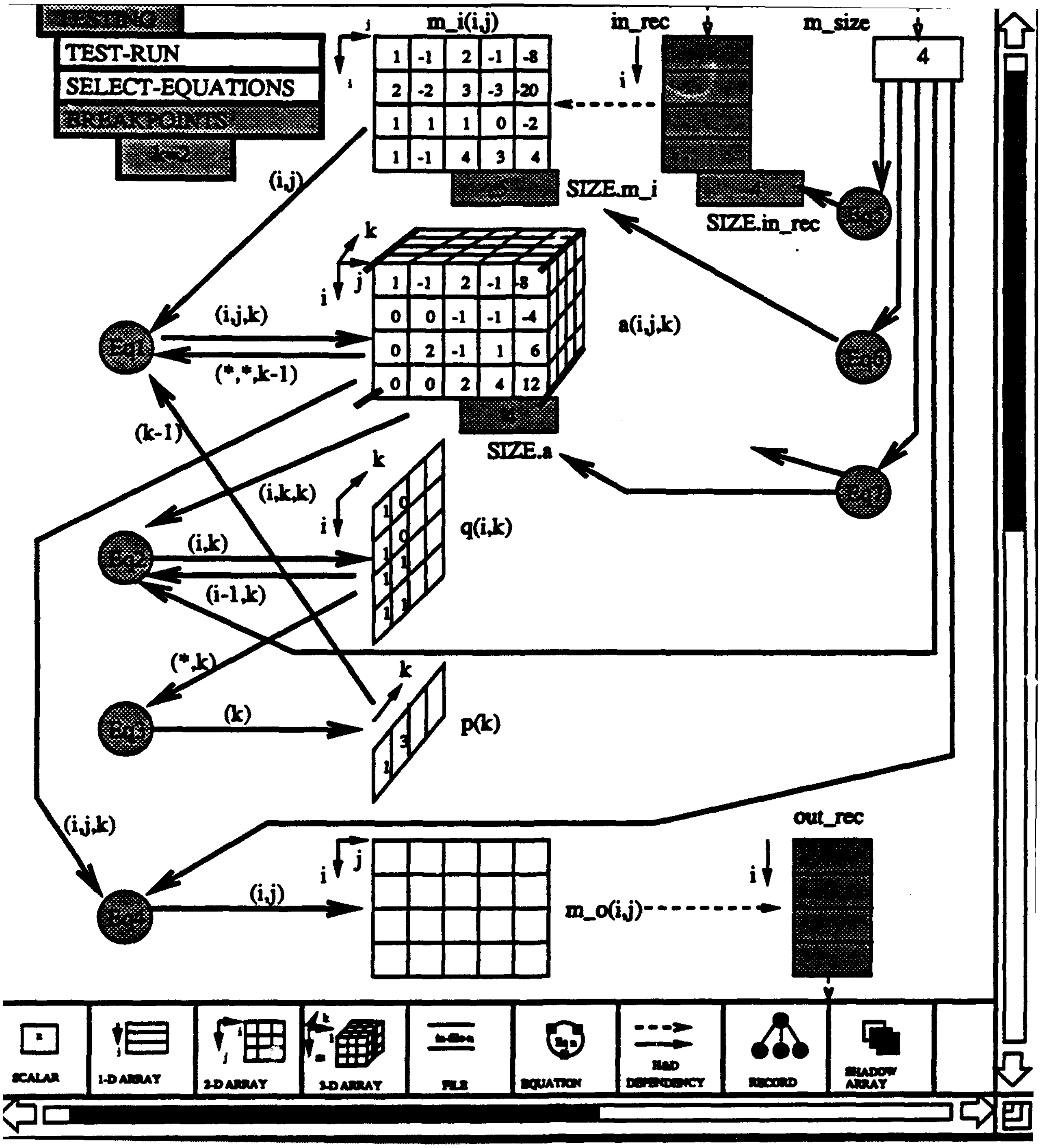

TEST-RUN
SELECT-EQUATIONS
BREAKPOINTS
m_i(i,j)
in_rec
m_size
4
SIZE.m_i
SIZE.in_rec
(i,j)
(i,j,k)
(*,*,k-1)
a(i,j,k)
(k-1)
SIZE.a
(i,k,k)
q(i,k)
(i,k)
(i-1,k)
(*,k)
(k)
p(k)
(i,j,k)
out_rec
(i,j)
m_o(i,j)
SCALAR
1-D ARRAY
2-D ARRAY
3-D ARRAY
FILE
EQUATION
DEPENDENCY
RECORD
SHADOW ARRAY

Figure 11: **Implementation.**

Implemantation Approach

Languages:

Procedural Language: FORTRAN

Equational Language: MODEL

Rule-Based Language: CLIPS

Graphics:

DECdesign - Graphics Environment for programming and design.

Methodology Implementation Facility (MIF) - Program generation for graphics methodologies.

1). FORTRAN for the Procedural language.

2). MODEL for the Equational language.

3). CLIPS for the Rule-Based language.

he translation from FORTRAN to MODEL was described in [Prywes,90]. The environment, and this ıper, focus on the latter two classes of languages. The choice of MODEL is primarily due to existence of compiler/procedural-program-generator for MODEL [Prywes&Pnueli,83, Szymanski&Prywes,88] as well the above mentioned checking algorithms. The choice of CLIPS is primarily due to existence of an ficient language interpreter written in C with a high degree of portability [Giarratano,89].

he DECdesign graphics environment was selected primarily due to existence of Methodology Implementaon Facility (MIF). This is a program generator that accepts as input definition of a graphic methodology Entity-Relation graph structures, icons, menus, and routines). It generates programs incorporated in ECdesign graphics system which implement the defined methodology. Use of this graphics approach was escribed in [Prywes,91].

Ceferences

)EDdesign,90] *Guide to DECdesign.* digital equipment corporation, Maynard, Massachusetts, May 1990.

)eMillo,87] R. A. DeMillo, W. M. McCracken, R. J. Martin, and J. F. Passafiume. *Software Testing and Evaluation.* Benjamin/Cummings Publishing Co., Menlo Park, California, 1987.

'orgy,82] C. L. Forgy. Rete: a fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, 19(1):17–37, 1982.

ielernter,90] D. Gelernter and S. Jagannathan. *Programming Linguistics.* The MIT Press, Cambridge, Massachusetts, 1990.

iiarratano,89] J. C. Giarratano and G. Riley. *Expert Systems: Principles and Programming.* PWS-KENT Publishing Company, 1989.

Iamlet,88] R. Hamlet. Special section on software testing. *Communications of the ACM*, 31(6):662–667, June 1988.

Iowden,87] W. E. Howden. *Functional Program Testing and Analysis.* McGraw-Hill Inc., New York, New York, 1987.

Cim,91] Jee-In Kim. *Equational and Rule-Based Programming: Visualization, Reliability, and Knowledge Base Generation.* Technical Report MS-CIS-91-60, Department of Computer and Information Science, University of Pennsylvania, 1991.

'rywes,90] Noah S. Prywes, Xiang Ge, Insup Lee, and Mitchel Song. *Procedural to Equational Language Translation.* Technical Report Contract AFSOR-88-0116, Department of Computer and Information Science, University of Pennsylvania, 1990.

'rywes,91] Noah S. Prywes, Evan Lock, and Xiang Ge. Automatic abstraction of real-time software and re-implementation in ada. October 1991. Prepared for the Tri-Ada Conference.

[Prywes&Pnueli,83] Noah S. Prywes and A. Pnueli. Compilation of nonprocedural specifications into computer programs. *IEEE Trans. on Software Engineering*, SE-9(3):267–279, May 1983.

[Szymanski&Prywes,88] B. K. Szymanski and Noah S. Prywes. Efficient handling of data structures in definitional languages. *Science of Computer Programming*, 10:221–245, 1988.

[Szymanski,91] B. K. Szymanski. *Parallel Functional Languages and Compilers.* ACM Press, 1991.

APPENDIX F

H-KB

Heterogenous Knowledge Bases

Gio Wiederhold
DARPA
on leave from
Stanford University
October 21, 1991

Between the minds that plan and the hands that build there must be a mediator

Intertitle from Fritz Lang: *Metropolis*, UFA 1926.

This research was supported by DARPA N39-84-C211, National Library of Medicine (LM04836).

Complementary support by IBM and Hewlett-Packard Co's.

Outline

- Definitions
- Problem statement
- An Architectural Solution
- Research Needs

Definitions

We make some definitions
for the purpose of this exposition.

1. knowledge:
   a. definition of concepts and their attributes
      aircraft: vehicle
         speed: 100 mph — 1600 mph provides default
      transport aircraft: aircraft
         speed: 400 mph — 700 mph

. . .

trip: . . .

. . .

begin: . . .

. . .

   b. rules or constraints relating concepts
      trip.begin(aircraft): airport

. . .

2. knowledge base
   a. A collection of definitions and rules
      for some domain, using these concepts
   b. A database of instances for that domain

Problem Statement

Knowledge pertains to specific
application domains
transport, manufacturing, . . .
and subdomains
passenger transport, freight transport, . . .
and sub-subdomains
military . ., corporate . ., commercial . .,
with distinct, possibly overlapping sets of
concepts:
aircraft, load,
and instances: *B747, C141, C17, F16, . . .*

so that rules for one domain cannot
be assumed to hold in another domain

Partial Solutions

1. Define all terms globally
   i.e., establish standards
   Problems: affects all rules, un-natural,
   greatly increases number of terms
2. Delimit all rules by domain predicates
   i.e., limit applicability of rules
   Problems: weakens rules, slows processing
3. Keep knowledge-bases distinct and
   fuse their resuilts
   i.e., permit heterogeneity
   Problems: most decisions require
   information from multiple domains
   creates uncertainty due to doamin mismatch
   (result [not domain] scope,
   abstraction level)

We focus now on the third alternative
(not to be used exclusively)

Heterogenous KBs

Advantages:

1. coherent maintenance by specialist(s)
2. manageable size for processing
3. parallel processing is natural
4. effective linkages to one or few

databases	*the past*
sensor-based sources	*the present*
simulations	*the future*

Disadvantages:

1. results must be merged/fused by higher level knowledge processing
2. uncertainty will be created when fusing

Paradigm

Participating modules provide

Information Services

not direct knowledge or data

Are independent *contractors*,
not integrated subsystems

These notions depend on

1. specialized processors,
   associated with knowledge maintainers
2. effective networks
3. catalog nodes describing the services

versus knowledge integration,
database integration
integration of
databases,
sensor-based processing
simulations

Mediators

are knowledge-based processes

1. which *understand*
   lower level service providers
2. can formulate information requests
   so that
   useful and
   mergable information is returned
3. can schedule good* execution sequences
   (depends on having bound arguments
   and performance estimates)
4. and report combined results
   with reliability assessments

to the end-users' applications.

*optimal requires global knowledge,
partitioning induces suboptimality
but improves computability of local optima
(large systems → heuristics ~~RIGHTARROW~~ → also suboptimal)

Why mediators?

End-user applications must focus on
decision making and planning
not on management of information resources

Follows management principles,
where the decision-maker relies on
specialists, who use
base resources, as
databases
current findings
projections from what-if tasks

recall:
Between the minds that plan and the hands that build
there must be a mediator

Architecture

Knowledge-based modules: **Mediators**

between

User Applications (private programs)

and

Autonomous information resources (public)

All Mediator modules are combinations of

PEOPLE and MACHINES

Arranged top-down:

---

result → decision making

1 Independent APPLICATIONS on workstations

network services to information servers

2 Multiple MEDIATORS

network services to data servers

3 Multiple INFORMATION SOURCES, ETC.

input ← real-world changes, effects

Support modules

Revisting the resource layer

All modules use distinct data
    if shared database,
        synchronized through database locking

All modules use distinct knowledge
    if shared, have local working copy

Many modules use different reasoning
    depends on type of resources, knowledge

Up to now we have implied that
    support modules use AI paradigms

but this architecture does not require that

- knowledge is also encoded in
  - algorithmic programs
    fine if rules are stable
  - OR programs
    great for dealing with large number of variables
  - databases
    use efficient, simple inference techniques
        for large volumes
  - spreadsheets
    for balancing projections

as long as their interfaces are machine-friendly

Mediator modules

A mediation module should be small and simple
(as a any software module)
Maintained by one expert
or coherent group of experts.
assisted by monitors on the data.

Mediators are *inspectable* by the potential users
for selection, evaluation

Results

Abstractions
constraints filters
selection criteria deduction
closure for completeness ↓
monitors for correctness ↑ for expert maintenance

Source Information

Distribution of Mediators

Homebase: at expert

Active nodes: arbitrary

independent copies

bound to workstation

bound to information sources

In general, independent of sources:

1 The mediator contains knowledge beyond the scope of the sources.

2 Often must deal with uncertainty about sources

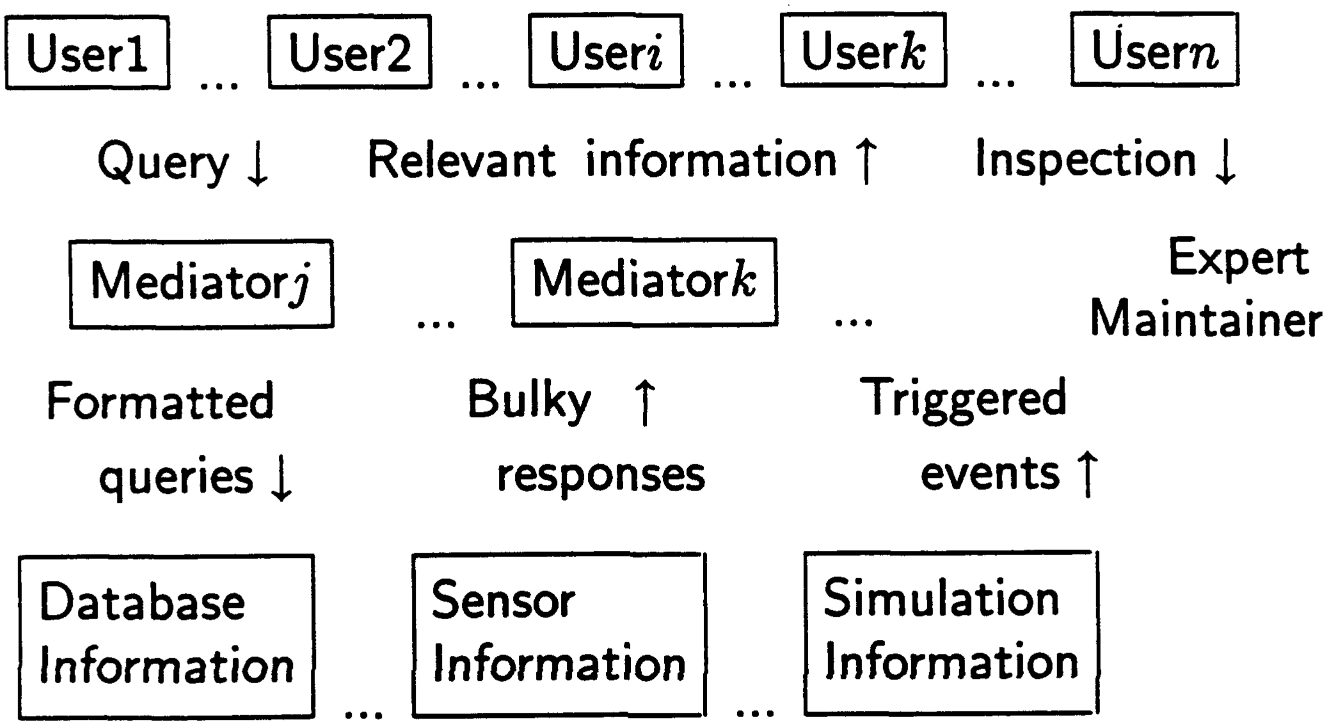
3 Mediators often access multiple sources to perform appropriate fusion

In general, independent of workstation:

1 too large, too many

2 Intrusive maintenance

Architecture with Modules



All *modules* are distributed over nationwide networks.

Alternate AI Paradigms

This architecture separates also
two competing AI paradigms

- Pragmatic systems — for the end-user
  that assist in decision-making
  and must face the *uncertain* future

but
  that are difficult to scale up

- Formal systems — for the mediators
  that behave predictably
  are subject to optimization

but
  that can only deal in bounded scopes

you can never formally predict the future
in an open world

Research Issues

The Interface to Mediators requires

a. A new language level beyond SQL, 4GLs

(Why a language?

Breadth in accessing resorces

Flexibility in composition,

Rearrangement for efficiency

Modularity for reallocation to

computing nodes, ...)

b. Machine-readable descriptions

of needed paarameters

candidate results

c. Uncertainty computations

caused by domain mismatch

d. Dealing with performance issues:

Obtaining estimates (mean,sd) of

costs, result cardinalities

Caching

Binding

Compiling

Reorganization

APPENDIX G

SIMS:
Services and Information Management for decision Support

Yigal Arens and Craig Knoblock
Information Sciences Institute
University of Southern California
Marina del Rey, CA 90292

October 1991

The Problem

Support the retrieval of information in complex domains, which require the integration of multiple databases.

One must:

- Know what databases are available
- Speak the language of each individual database
- Represent the task in terms of the databases
- Find a combination of queries that correspond to the task
- Obtain some degree of "optimality"
- Match the outputs of one query with the requirements of others
- Interpret the results in terms of the original task.

SIMS' Solution: "Soft" Integration

SIMS is a **broker** which coordinates access to the databases. Important features of SIMS are:

- **Database-independent** interaction language: the user does not need to be familiar with the databases or their organization.
- **Logical** integration: the databases remain separate and are not reprogrammed into a custom system.
- **Dynamic** integration via **planning**: the combination of queries that perform a task is determined when the task is proposed to the system.
- **Models** of databases and the application domain are consulted by the SIMS planner.
- **Reformulation** may be performed on task statements.

Current SIMS Status

SIMS now:

- Operates in the Navy briefing domain and in transportation planning domain.
- Models four Oracle database tables, and seven primitive databases.
- Accesses databases automatically over a local area network.
- Uses domain model information to correct user errors: for example, upon encountering an inappropriate term, user is offered a list of semantically acceptable alternatives.

Reformulation Component

Goal Repair strategies:

- Involve replacing elements of the goal specification until a correct one is arrived at.
- Are based on structure of domain and goal knowledge.

$\star$ *Ships*, *Area* = *South China Sea*

$\Rightarrow$ $\star$ *Area* : Employment Area, *or* Location?

Current SIMS Status, cont.

- Uses domain semantic constraints to optimize access: for example, LOOM *number restrictions* affect use of looping in retrieval.

- Uses domain model structure to reformulate goals as equivalent conjunction or disjunction when necessary.

Reformulation Component

Some **Goal Replacement** strategies:

- **Conjunction:** A combination of goals provide for the original one.

  Ship status ⇒

  Location, course, employment schedule

- **Specification:** A set of goals requiring additional information covers the original one.

  Ship location ⇒

  Ship location by country

- **Subsumption:** A super-ordinate goal subsumes the original one.

  Ship displacement ⇒

  Displacement by class

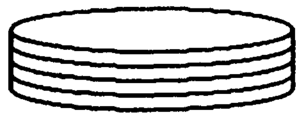
Sample Query

```
(:AND (CURRENT-MAJOR-EMPLOYMENT ?Ship "OPS")
      (CLASS ?Ship ?Class)
      (OVERALL-READINESS ?Ship 5))
```

Sims

UIC	ETERM
N03361	OPS
N03363	OPS
N03651	OPS
N04629	OPS
N04709	OPS
N04806	OPS
N05117	OPS
N05121	OPS
N05138	OPS
N05596	OPS
N09286	OPS
N09618	OPS
N09644	OPS
N09930	OPS
N20043	OPS
N20202	OPS
N20541	OPS
N20883	OPS
N20976	OPS
N20994	OPS
N21016	OPS

ETERM



UIC

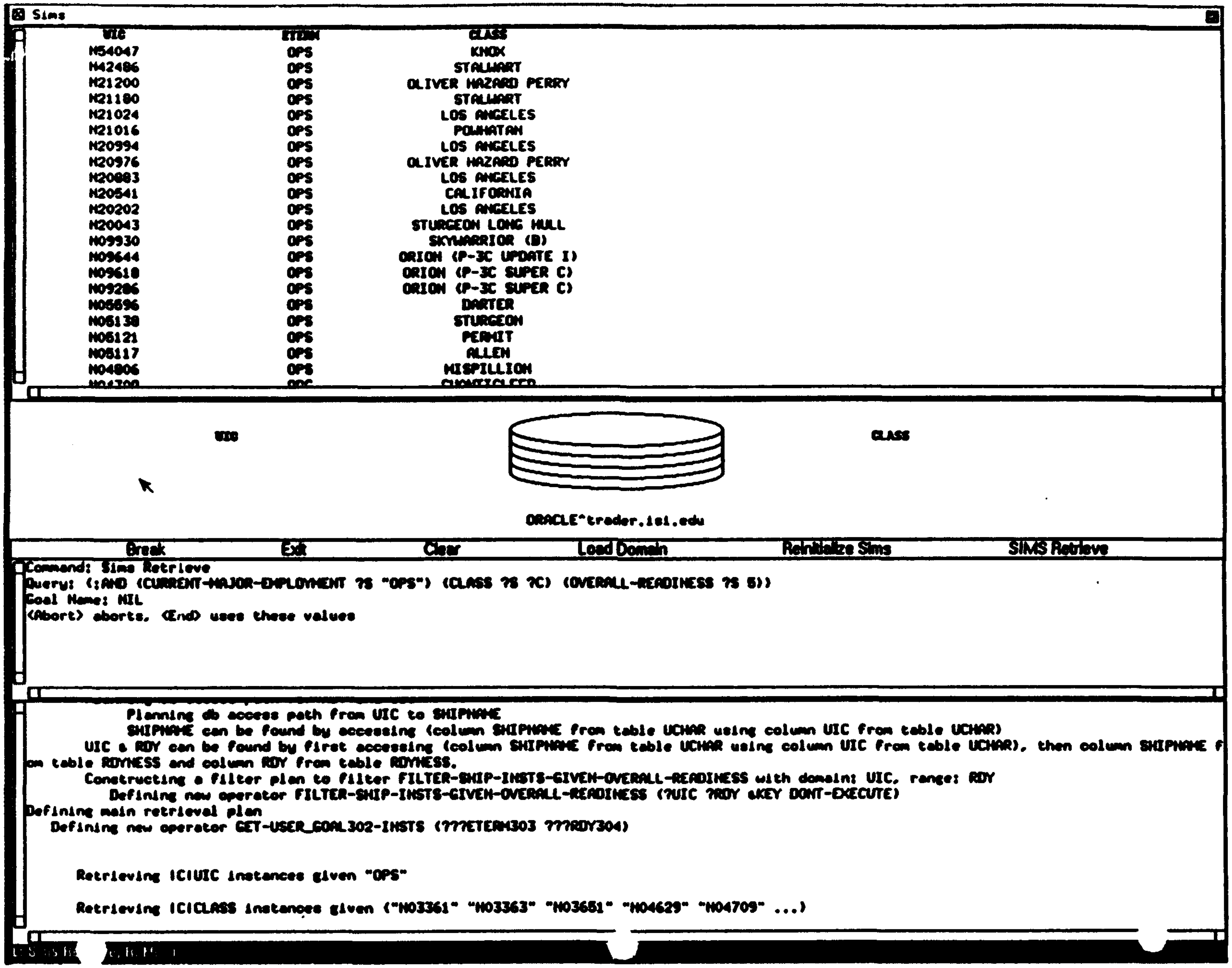
ORACLE^trader

Break Exit Clear Load Domain Reinitialize Sims SIMS Retrieve

```
Command: Sims Retrieve
Query: (:AND (CURRENT-MAJOR-EMPLOYMENT ?S "OPS") (CLASS ?S ?C) (OVERALL-READINESS ?S 5))
Goal Name: NIL
<Abort> aborts, <End> uses these values
```

```
 Invoking meta operator FILTER-SHIP-INSTS-GIVEN-OVERALL-READINESS when given UIC and RDY
      Planning db access path for RDY and UIC
           Planning db access path from UIC to SHIPNAME
           SHIPNAME can be found by accessing (column SHIPNAME from table UCHAR using column UIC from table UCHAR)
      UIC & RDY can be found by first accessing (column SHIPNAME from table UCHAR using column UIC from table UCHAR), then column SHIPNAME f
om table RDYNESS and column RDY from table RDYNESS.
      Constructing a filter plan to filter FILTER-SHIP-INSTS-GIVEN-OVERALL-READINESS with domain: UIC, range: RDY
        Defining new operator FILTER-SHIP-INSTS-GIVEN-OVERALL-READINESS (?UIC ?RDY &KEY DONT-EXECUTE)
Defining main retrieval plan
   Defining new operator GET-USER_GOAL302-INSTS (???ETERM303 ???RDY304)


     Retrieving |C|UIC instances given "OPS"
```

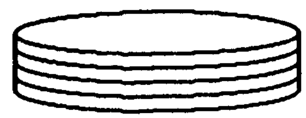

Sims
UIC ETERM CLASS
N54047 OPS KNOX
N42486 OPS STALWART
N21200 OPS OLIVER HAZARD PERRY
N21180 OPS STALWART
N21024 OPS LOS ANGELES
N21016 OPS POWHATAN
N20994 OPS LOS ANGELES
N20976 OPS OLIVER HAZARD PERRY
N20883 OPS LOS ANGELES
N20541 OPS CALIFORNIA
N20202 OPS LOS ANGELES
N20043 OPS STURGEON LONG HULL
N09930 OPS SKYWARRIOR (B)
N09644 OPS ORION (P-3C UPDATE I)
N09618 OPS ORION (P-3C SUPER C)
N09286 OPS ORION (P-3C SUPER C)
N05696 OPS DARTER
N05138 OPS STURGEON
N05121 OPS PERMIT
N05117 OPS ALLEN
N04806 OPS NISPILLION
UIC
CLASS
ORACLE^trader.isi.edu
Break
Exit
Clear
Load Domain
Reinitialize Sims
SIMS Retrieve
Command: Sims Retrieve
Query: (:AND (CURRENT-MAJOR-EMPLOYMENT ?S "OPS") (CLASS ?S ?C) (OVERALL-READINESS ?S 5))
Goal Name: NIL
<Abort> aborts, <End> uses these values
Planning db access path from UIC to SHIPNAME
SHIPNAME can be found by accessing (column SHIPNAME from table UCHAR using column UIC from table UCHAR)
UIC & RDY can be found by first accessing (column SHIPNAME from table UCHAR using column UIC from table UCHAR), then column SHIPNAME f
om table RDYNESS and column RDY from table RDYNESS.
Constructing a filter plan to filter FILTER-SHIP-INSTS-GIVEN-OVERALL-READINESS with domain: UIC, range: RDY
Defining new operator FILTER-SHIP-INSTS-GIVEN-OVERALL-READINESS (?UIC ?RDY &KEY DONT-EXECUTE)
Defining main retrieval plan
Defining new operator GET-USER_GOAL302-INSTS (???ETERM303 ???RDY304)
Retrieving |C|UIC instances given "OPS"
Retrieving |C|CLASS instances given ("N03361" "N03363" "N03651" "N04629" "N04709" ...)

Sims

UIC	ITEM	CLASS	SHIPNAME
N54047	OPS	KNOX	KNOX
N42486	OPS	STALWART	ASSURANCE
N21200	OPS	OLIVER HAZARD PERRY	VANDEGRIFT
N21180	OPS	STALWART	CONTENDER
N21024	OPS	LOS ANGELES	OLYMPIA
N21016	OPS	POWHATAN	NAVAJO
N20994	OPS	LOS ANGELES	HOUSTON
N20976	OPS	OLIVER HAZARD PERRY	LEWIS B PULLER
N20883	OPS	LOS ANGELES	PORTSMOUTH
N20541	OPS	CALIFORNIA	CALIFORNIA
N20202	OPS	LOS ANGELES	LOS ANGELES
N20043	OPS	STURGEON LONG HULL	WILLIAM H BATES
N09930	OPS	SKYWARRIOR (B)	VQ-1
N09644	OPS	ORION (P-3C UPDATE I)	VP-9
N09618	OPS	ORION (P-3C SUPER C)	VP-1
N09286	OPS	ORION (P-3C SUPER C)	VP-50
N05596	OPS	DARTER	DARTER
N05138	OPS	STURGEON	QUEENFISH
N05121	OPS	PERMIT	FLASHER
N05117	OPS	ALLEN	SAM HOUSTON
N04806	OPS	MISPILLION	NAVASOTA

SHIPNAME

RDY



FLAT-FILE^/home/chee/rdyness

Break Exit Clear Load Domain Reinitialize Sims SIMS Retrieve

Command: Sims Retrieve
Query: (:AND (CURRENT-MAJOR-EMPLOYMENT ?S "OPS") (CLASS ?S ?C) (OVERALL-READINESS ?S 5))
Goal Name: NIL
<Abort> aborts, <End> uses these values

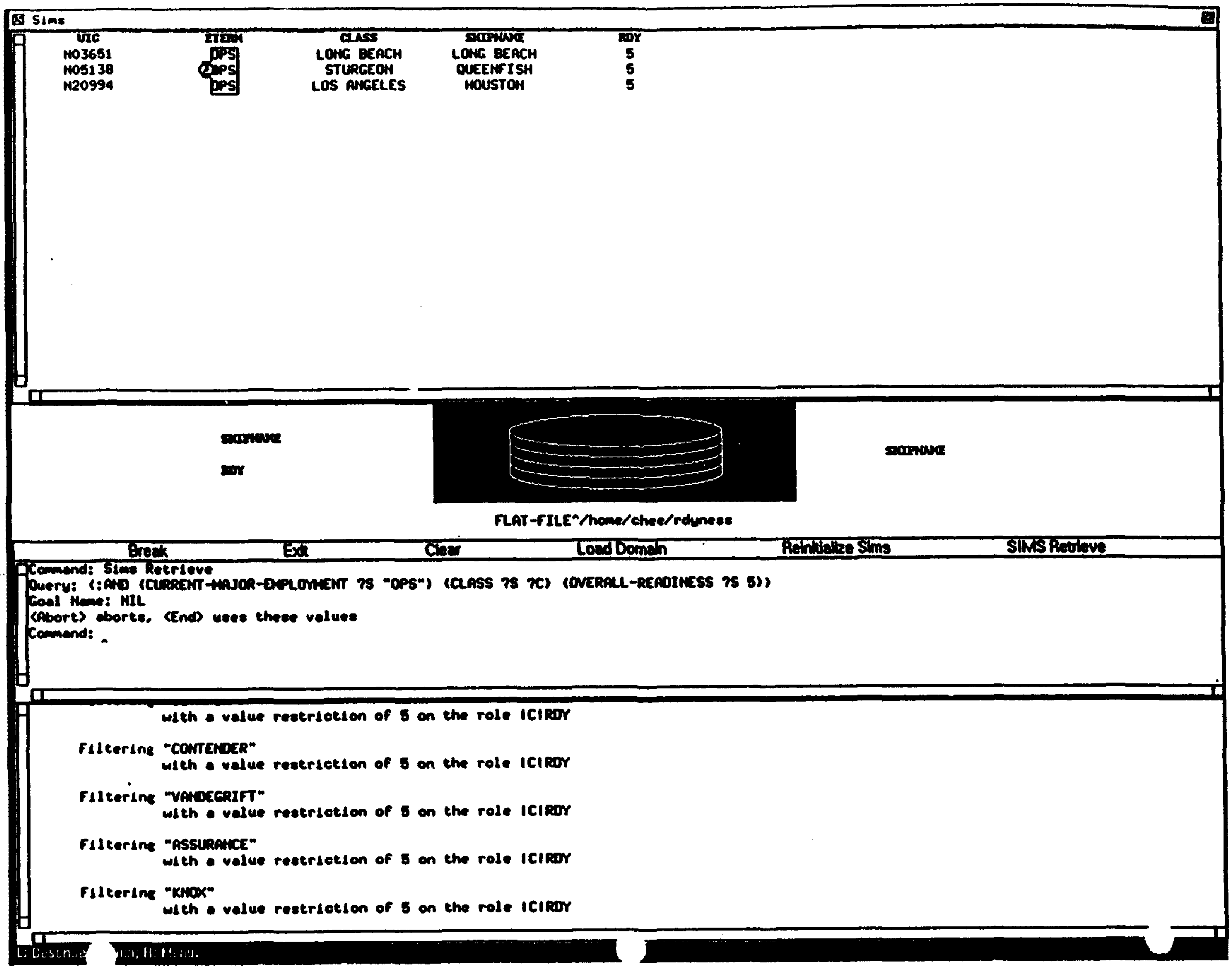
Retrieving |C|UIC instances given "OPS"

Retrieving |C|CLASS instances given ("N03361" "N03363" "N03651" "N04629" "N04709" ...)

Retrieving |C|SHIPNAME instances given ("N54047" "N42486" "N21200" "N21180" "N21024" ...)

Filtering "RANGER"
with a value restriction of 5 on the role |C|RDY

Filtering "KITTY HAWK"
with a value restriction of 5 on the role |C|RDY

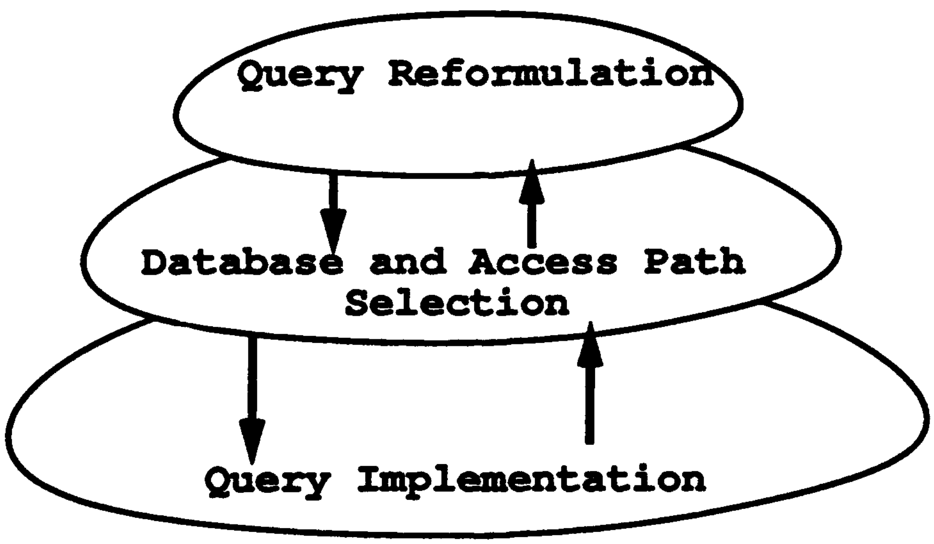

Sims
VIC ETERM CLASS SHIPNAME RDY
N03651 OPS LONG BEACH LONG BEACH 5
N05138 OPS STURGEON QUEENFISH 5
N20994 OPS LOS ANGELES HOUSTON 5
SHIPNAME
RDY
SHIPNAME
FLAT-FILE^/home/chee/rdyness
Break
Exit
Clear
Load Domain
Reinitialize Sims
SIMS Retrieve
Command: Sims Retrieve
Query: (:AND (CURRENT-MAJOR-EMPLOYMENT ?S "OPS") (CLASS ?S ?C) (OVERALL-READINESS ?S 5))
Goal Name: NIL
<Abort> aborts, <End> uses these values
Command:
with a value restriction of 5 on the role ICIRDY
Filtering "CONTENDER"
with a value restriction of 5 on the role ICIRDY
Filtering "VANDEGRIFT"
with a value restriction of 5 on the role ICIRDY
Filtering "ASSURANCE"
with a value restriction of 5 on the role ICIRDY
Filtering "KNOX"
with a value restriction of 5 on the role ICIRDY

Hierarchical Query Reformulation

<u>Given:</u> A query to a database expressed in a high-level KR language.

<u>Find:</u> A formulation of the query that can be implemented efficiently.

<u>Approach:</u> Exploit the hierarchical structure of the search space to guide reformulation.



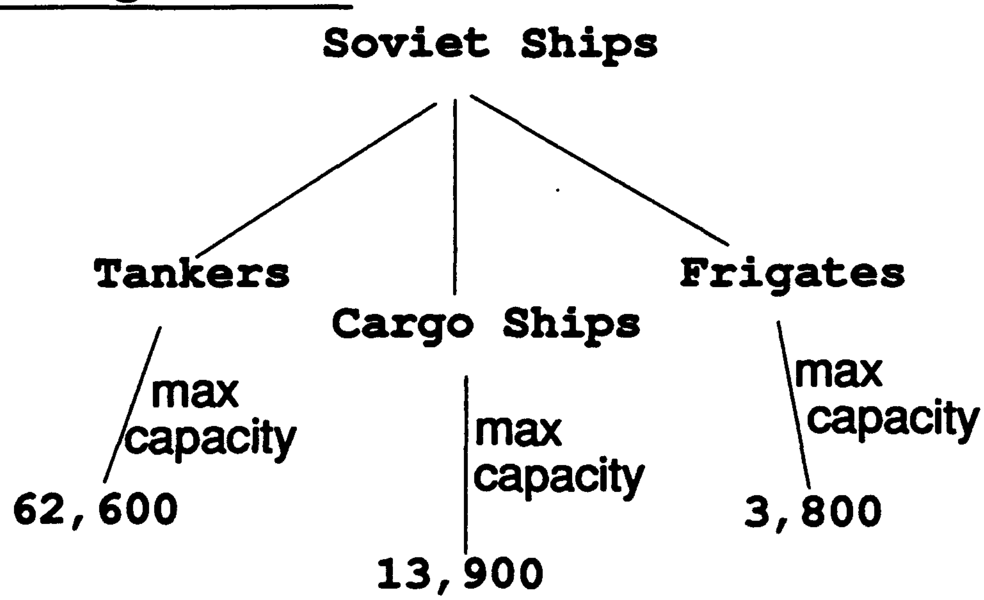
Example

<u>Query:</u>

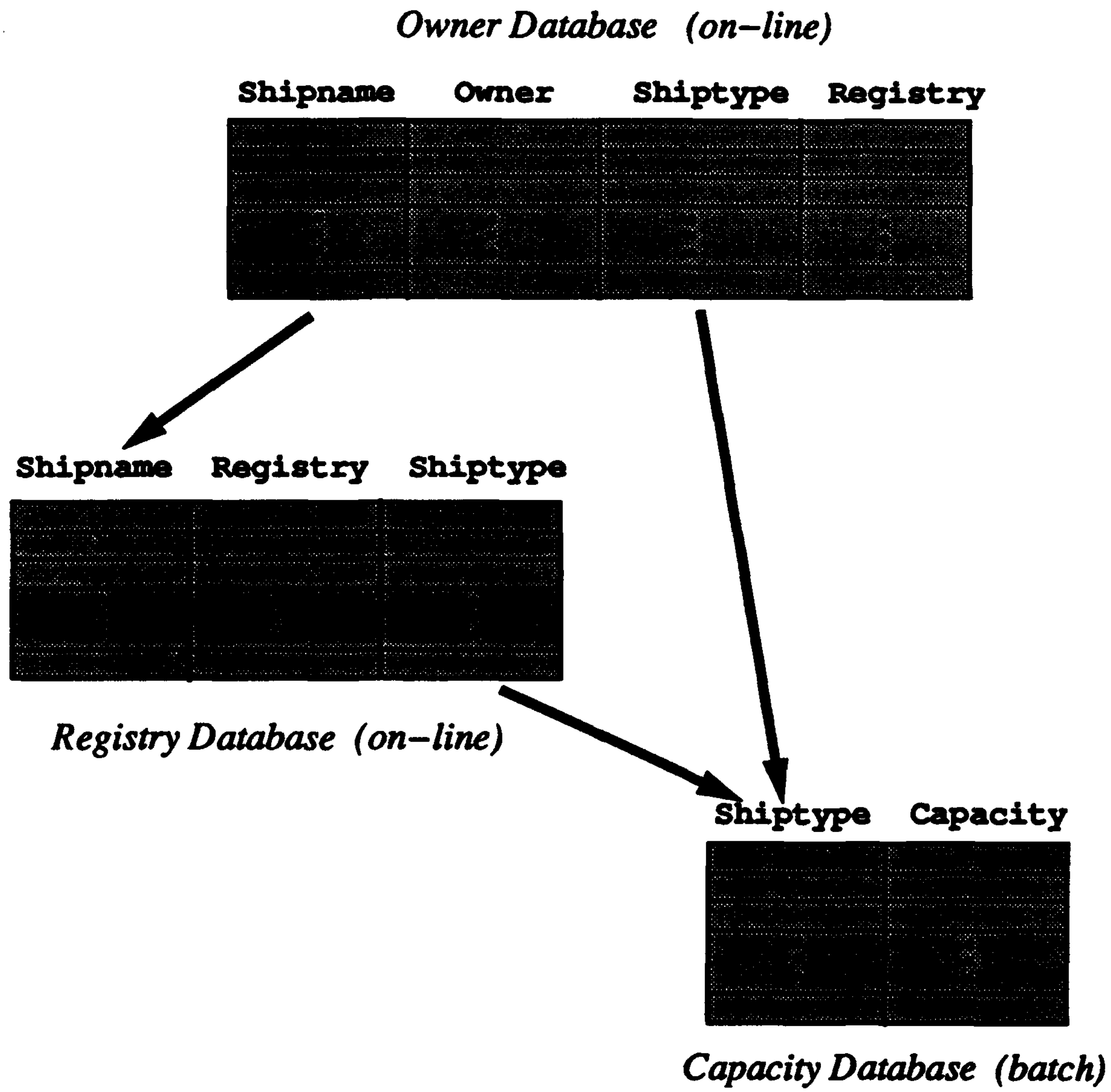
List the Soviet ships with a capacity greater than 40,000 tons.

```
(and (Capacity > 40,000)
     (shiptype = ship)
     (Registry = Soviet)
     (Shipname = ?))
```

<u>Knowledge Base:</u>



Databases:



Owner Database (on-line)
Shipname Owner Shiptype Registry
Shipname Registry Shiptype
Registry Database (on-line)
Shiptype Capacity
Capacity Database (batch)

The Search Process

Query Processing:

1. Select a formulation of the query.
2. Select the databases and access paths.
3. Estimate the cost of executing this query for the selected databases and paths.
4. If the estimated cost is too high, search for:
   (a) Alternative implementations.
   (b) Alternative databases and access paths.
   (c) Reformulations of the query.

Reformulation Example

- Capacity database is batch, so processing the original query will take several hours.
- The system decides to spend a few minutes attempting to reformulate the query to eliminate the capacity constraint.
- KB model yields:
  capacity $> 40{,}000 \rightarrow$ tanker
- Capacity constraint can be dropped from query.

```
(and (shiptype = tanker)
     (registry = Soviet)
     (shipname = ?))
```

Query Reformulation

Problem: Given a query in a high-level language, find equivalent formulations of the query.

Approach:

- Add, delete, and replace constraints in a query.
- Knowledge base makes it possible to perform complex reformulations on a query.
- Search is directed by implementation at lower levels.
- Similar to work by King, 1981, but not limited to semantic integrity constraints.

Database and Access Path Selection

Problem: Given a high-level database query, find a set of databases and corresponding access paths that can be used to answer the query.

Approach:

- Search through the models of the databases to select the databases and access paths.
- Databases may be overlapping and contain multiple access paths.
- Search is directed by the implementation costs.
- Allows flexible access in contrast to other systems, which either assume a single, fixed access path or expect the access path to be given in the query (as in SQL).

Query Implementation

Problem: Given a query and the selected databases and access paths, find the low-level database operations that can be used to implement the query.

Approach:

- This level corresponds to implementing an SQL query.
- Requires determining the Select, Project, Join, Scan, and Sort operations and finding an efficient ordering to implement a query.
- Problem has been studied extensively in the database field, so we will use existing techniques.

Summary

- Use the knowledge base to reformulate the database queries.
- Consider alternative databases and access paths for extracting the desired information.
- Exploit the hierarchical structure of the problem to guide the search for better queries.
- Use existing query implementation techniques to determine when to search for a reformulation and how much time to spend expend.

APPENDIX H

APPENDIX H

Causal Process Modeling

B. Chandrasekaran

LAIR, Ohio State University

Functional Representation of Devices

Hypothesis: For the tasks of design & diagnosis, devices should be represented by explicit indicators of how its **functions** arise as a result of **causal sequences** (**behavior**) made by the functions of the **components** & their relations. What role first principle (physical laws) knowledge plays in explaining causal sequences should be part of this structure.

CPD is A Causal Story

1. Causal Process Descriptions and Functional Representations of Device

* (Partial) State of a physical system:

 Predicates over state variables of interest

* A Causal Process Description (CPD) or a Causal Story is a directed graph of partial states of a system, where the initial state is a predicate over an "exogenous" state variable, and the following states are causal consequences of interest

<u>HOW</u> THINGS WORK

CPD Reflects Agent's Explanatory Goals

* The CPD has to be obey certain rules of composition, that is, there is a story grammar.

* CPD is an **agent-specific** description of a process, not agent-independent, abstract representation of the process.

That is, the choice of the state variables in the process decription **reflects the agent's explanatory goals.**

Annotations of CPD Links: 1. Process Details

CPD's links are ANNOTATED using the following taxonomy of links:

- **1. By <CPD>**

  --- This is a pointer to another causal process description that is already present as part of prior knowledge..

```
  o --------------> o
A   By  CPD-1        B

CPD-1 : o -------->o--------> o..........---> o
        A          a1         a2              B
```

Example usage: Patil's work on Abel

Helps us to reuse CPD's.

Hierarchical ways of organizing processes.

Annotations of CPD Links: 2. Component Function

- **2. By <function> of <component>**

  -- If the physical system can be decomposed into subsystems (components) and the components can be described in terms of input/output relations called Functions (in the sense of "roles", not necesarily "intentions"), then, a causal transition at the device level can be explained by pointing to the function of a component.

  Helps us to understand system behaviors as a hierarchical composition of subsystem behaviors. The component function in turn may be explained as a CPD at the time of component modeling.

Annotations of CPD Links: 3. First Principles

* 3. **As-per** <domain-law>

-- Some causal transitions are simply explained by pointing to a physical law, e.g., Ohm's law or F = ma, etc., as instantiated to the current configuration.

{Voltage = 5 at terminals L1 and L2} -------->

{*Current* = 1 *amp thru* W1}

(Place where Math Models

& *numerical* calculations can come *in*)

Annotations of CPD Links: "By-Abstraction" Link

The next two are not causal transitions, but are often part of causal process descriptions.

* 4. By <abstraction>

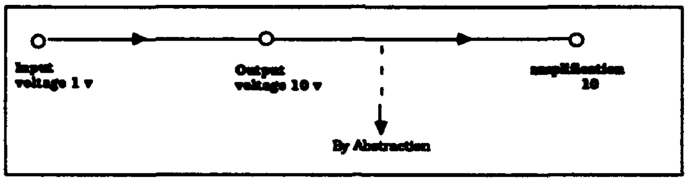
This transition enables the agent to change levels of abstraction in the CPD. For example, to explain the state called "amplification is 10" from the level of analysis in terms of currents and voltages, an appropriate abstraction operator is defined.

We have a number of types of abstraction:

state abstraction, process abstraction

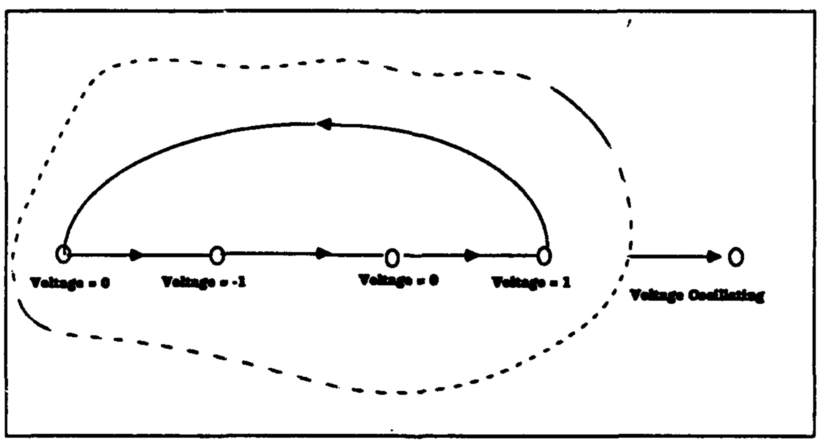
(e.g., a repeated transition between two values may be abstracted as oscillation of certain type.)

Example of "By-Abstraction" Link



Laboratory for AI Research, The Ohio State University

Example of Asbstraction of Process Into State



Laboratory for AI Research, The Ohio State University

Example of "By-Inference" Link

• 5. By -Inference

Many CPD's have transitions which are state descriptions arrived at by inference from previous state descriptions, and general domain knowledge.

Thus a CPD is a comprehensible story about how the causal conclusion is drawn about a relevant behavior of the system as a whole from knowledge of the behaviors of the parts, their connections, and domain laws.

Laboratory for AI Research, The Ohio State University

FUNCTIONAL REPRESENTATIONS (FR)

A Family of Representations Called Functional Representations have Been Designed and Complex Biologicxal and Engineering Systems have been Represented at our Lab. Given Observations to be explained, such representations have been used to do Diagnostic Reasoning.

Laboratory for AI Research, The Ohio State University

Device Representation as a Causal Process

Device Representation (Functional Representation)

We understand device functioning as a causal process. <u>HOW</u> **A DEVICE WORKS**

* **Function :** Pred(output state variables)

  Makes <function>

  If <exogenous state variables = .>

  Provided <certain background conditions on structure>

  * **By** <CPD>
  * **CPD description**
  * **Structure:** Component names and associated function names, and connections between components

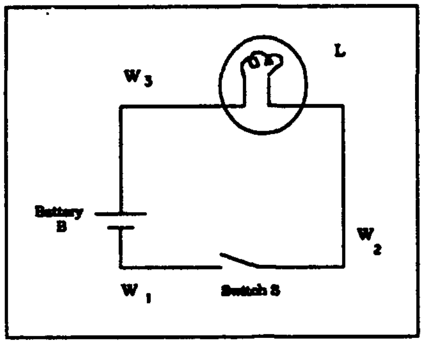
Laboratory for AI Research, The Ohio State University

Device Representation as Hierarchies of CPD'S

Since CPD descriptions use functions of components for parts of their annotations, how a device works is represented as hierarchical composition of CPD's, each of which point to other CPD's and to domain laws for causal explanation and to state and process abstractions.

Laboratory for AI Research, The Ohio State University

Lamp Circuit



<u>Structure</u>:

Serially-Connected {Switch, battery, Lamp, w1, w2, w3}

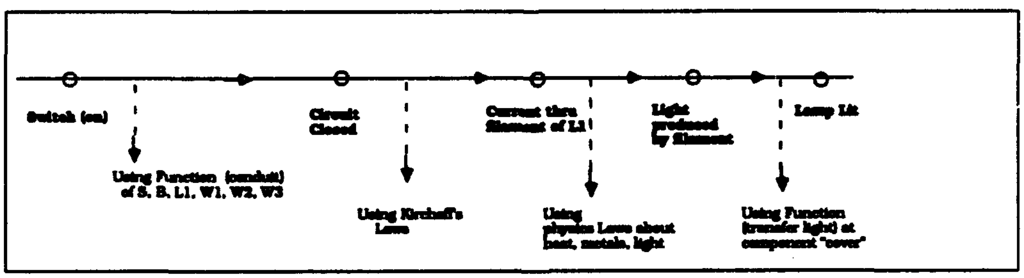
<u>Switch</u> **Functions:**

1. **Make (close connection)   If (on)**
2. **Make (open-connection)   If (off)**

Battery: Function: Make (voltage at terminals)
If (connected to terminals)

Lamp:

Causal Account of How Lamp Circuit Works



Laboratory for AI Research, The Ohio State University

Properties of FR

- Component-Subcomponent structure representation can be repeated several layers
- "No-function-in-structure" obeyed for each level
- Behavioral spec's of components not mentioned, only functions. Parts functionally equivalent but working differently can be substituted.

Functional Representations and Diagnosis

Work at LAIR:

- **Sticklen, 1987.** Compilation of Diagnostic Knowledge. Given a functional model of a device, malfunctions and states indicative of malfunctins can be derived. Because of the hierarchical, system subsytem-nature of CPD representation, we get a hierarchical malfunction representation.

Sticklen has gone on to develop a family of models of parametric simulationalgorithms, and has applied to ecological and materials modeling.

Work at LAIR (cont'd)

- **Keuneke, 1989.** Use of CPD's to generate explanations of diagnoses by demostrating a causal story.
- **Goel 1989.** Use of CPD and functional primitives to index design cases in memory (why a design works), to retrieve relevant cases from memory, and to identify possible substructures to modify.

- **Allemang 1990.** Represent algorithms as generic devices, with alternativecode-level implementations. Used for program debugging.

Work at LAIR (cont'd)

* **Darden, 89, 90.** (Philosopher of science)
  Representation of scientific theories in biology
  and tracing debugging of theories.

Open Issues

* Semantics of the primitives
* Automatic generation of CPD's and functional representations from structure. Involves recognition of functions by using functional templates.
* Application to more complex devices
* Grounding of states in a general theory of substances (Goel) and visual representations (BC and Narayanan).

APPENDIX I

COOPERATIVE ANSWERING: THEORY AND PRACTICE

by

Terry Gaasterland[1]

Jack Minker[1,2]

Department of Computer Science[1]

and

Institute for Advanced Computer Studies[1,2]

University of Maryland

College Park, Maryland 20742

OVERVIEW OF COOPERATIVE ANSWERING METHODS

- PRINCIPLE OF COOPERATION
- EARLY APPROACHES
  - AI APPROACHES
  - ADDRESSING USER GOALS
  - RELATIONAL DATABASE APPROACHES
  - DEDUCTIVE DATABASE APPROACHES
- AN INTEGRATED COOPERATIVE ANSWERING SYSTEM
- SUMMARY

THE PRINCIPLE OF COOPERATION

(Grice 1975):

A *COOPERATIVE ANSWER* IS A

- CORRECT
- USEFUL
- NON-MISLEADING

ANSWER TO A QUERY.

THE PRINCIPLE OF COOPERATION

EXAMPLE:

QUERY:

"In Spring 1990, how many international flights arrived on time in Cedar Rapids, Iowa?"

RESPONSE:

"None."

COOPERATIVE RESPONSE:

"None, there are no international flights into Cedar Rapids, Iowa."

EARLY APPROACHES TO COOPERATIVE ANSWERING

JOSHI/WEBBER '82:

- Consider user beliefs to anticipate user expectations.
- Provide extra info to prevent misconceptions.

  Q: "Is Sam an associate professor?"

  System: *User believes most associate professors have tenure.*

  System: *Sam is not tenured.*

  System: *Sam is an associate professor.*

  A: "Yes, but he doesn't have tenure."

KAPLAN '82:

- Detect and present false presuppositions

  Q: "How many students failed CS 401 last semester?"

  A: "There was no such course."

EARLY APPROACHES TO COOPERATIVE ANSWERING

HIRSHBERG '85: Use scalar implicature to answer yes/no questions

MAYS '81: Use schema to correct false presuppositions

McCOY '85: Use world knowledge to correct object oriented misconceptions

EARLY APPROACHES TO COOPERATIVE ANSWERING

JANAS '81:

- Compute subqueries of a query and see if they fail or not.
- Integrity constraints can eliminate some subqueries.

Q: "Find all red cars in the database owned by employees."

(Q fails)

SUBQUERY 1: "Find all red cars."

SUBQUERY 2: "Find all cars owned by employees."

IC: "All cars are owned by employees."

(SUBQUERY 1 fails)

(SUBQUERY 2 true by IC)

ANSWER: "None. There are no red cars in the database."

EARLY APPROACHES TO COOPERATIVE ANSWERING

LEHNERT '81:

- Classify questions into conceptual categories

Q: "How did John take the exam?"

SYSTEM: Enablement Question

A: "He crammed the night before."

SYSTEM: Instrument/Procedural Question

A: "He took it with a pen."

ADDRESSING USER GOALS: AI APPROACHES

POLLACK '85:

- Detect how question fits into user plan
- Offer alternative answer that facilitates plan

Q: "Under VI editor, how can I delete ctrl-Z?"

SYSTEM: *user typed ctrl-Z and wants to undo it*

A: "ctrl-Z has stopped the current process. Type 'fg' to start it again."

ALLEN '82: Detect user goals and potential obstacles

WAHLSTER '83: Detect user goals and anticipate questions

QUILICI/DYER/FLOWERS '88: Correct plan-oriented misconceptions.

CARBERRY '88: User models help form answers

ADDRESSING USER GOALS: RDB APPROACHES

MOTRO '87:

- "Neighborhood Information"
- Compute 'distance' between values in query and values in possible answers.
- Attributes are weighted by user priority.

Q: "What moderately priced, very good, Chinese restaurant is in Chinatown or Westwood?"

SYSTEM:

restaurant(Jasmine_Gardens,Chinese,Chinatown,Mod,Good).

distance = (Weight * |Very_Good - Good |)

SYSTEM:

restaurant(Nippon,Japanese,Downtown,Expensive,VeryGd).

distance = (W1 * |Chinese - Japanese|) +
(W2 * |Downtown - Chinatown|) +
(W3 * |Moderate - Expensive|)

A: (All restaurants whose tuples are 'near' the query).

ADDRESSING USER GOALS: RDB APPROACHES

CHU/LEE '90:

- Use partitioned generalization hierarchy over DB domain
- For more answers, translate query into new partition
- Formalized thru relational algebra

Q: "Which flights go from DCA to LAX cost less than $300?"

SYSTEM: flight **isa** mode-of-travel

train **isa** mode-of=travel

DCA **is-in-area** DC

LAX **is-in-area** LA

Q: "Which trains go from a train station in DC to a train station in LA?"

A: (list of flights and list of trains).

ADDRESSING USER GOALS: DDB APPROACHES

CUPPENS/DEMOLOMBE '88:

- Entity relationship model with hierarchy
- Add to query new arguments related to current arguments
- Formally defined thru logic

Q: "Which flights from Paris to NY leave between 7 and 11 am.?"

SYSTEM: topic(departure-time,TIME).

topic(arrival-time,TIME).

Q: "... and when do they arrive?"

A: "PanAm flight 101 leaves at 7:30am and arrives at noon".

American flight 733 leaves at 11:00am and arrives at 3:00pm"

TAILORING ANSWERS TO USERS

PARIS '88:

- Address user's level of expertese in content of answer

  Q: "What is a telephone?"

  SYSTEM: *user is an engineer*

  A: (technical description of parts and how they work)

  SYSTEM: *user is an 8-year-old*

  A: (visual description of object and what it does)

INTENSIONAL ANSWERS

IMIELINSKI

- INTENSIONAL ANSWER (IA) IS EQUIVALENT TO QUERY
- IA = SET OF EDB FACTS + SET OF NEW IDB RULES

Q: Q(Z) ← teach(smith,Z).

"List all courses that Smith can teach."

IDB: teach(X,Y) ← teach(X,Z), prerequisite(Y,Z).

"X can teach Y if X can teach Z, and Y is a prerequisite of Z."

EDB: teach(smith,math400).

EDB: prerequisite(math350,math400). prerequisite(math300,math350)

IA: "Math 400 and any prerequisite courses."

Q(Z) ← prerequisite(Z,Y),Q(Y).

Q(math400).

INTENSIONAL ANSWERS: OTHER WORK

- CHOLVY/DEMOLOMBE

- PIROTTE/ROELANTS

INDUCTIVE RESPONSES

CORELLA, SCHUM/MUNTZ:

- After calculating extension, find generalizations.

DB DOMAIN: { ann, sue, mary }

IDB:

technician(ann). technician(sue). researcher(mary).

employee(X) ← researcher(X)

employee(X) ← technician(X)

works_9_to_5(X) ← secretary(X).

works_9_to_5(X) ← technician(X).

QUERY: ← works_9_to_5(X).

ANSWER: {ann,sue}/X

INDUCTIVE ANSWERS

technician(X)

employee(X) + X≠ mary

2/3 employees

APPENDIX J

CoBase: A Cooperative Distributed Database

Wesley W. Chu

Computer Science Department
University of California, Los Angeles

Supported by DARPA contract
N00174-91-C-0107

Cooperative Distributed Database Systems

The next generation of distributed database systems, using Inference techniques to provide:

- Fault Tolerance Capabilities
- Cooperative Query Answering

Validate concepts with prototype cooperative distributed DDBMS

Cooperative Query Answering

Motivation

Conventional query answering

- provides too much data
- may obscure real meaning of data
- time consuming
- needs to know the detailed database schema
- cannot get approximate answer if full data is unavailable
- cannot analyze the intent of the user and derive relevant information if the exact answer is not available
- cannot answer conceptual queries

Cooperative Query Answering

Derive Summary Answers

Derive Intensional Answers

Derive Approximate Answers

Answer Conceptual Queries

Examples of Cooperative Query Answering

Intensional query answering

Q: List all cars with air bags.

Mercedes and Ford Taurus LQs built after 1989.

Approximate query answering

Q: List all flights from LA to New York departing between 8 and 9 AM.

Flight	*Departs*	*From*	*To*
UA #2	9:05	LA	New York
TWA #10	8:00	LA	Newark*

* helicopter flight (15 min, $50) and shuttle bus (50 min, $20) available every 1/2 hour from Newark to New York City.

Examples of Cooperative Query Answering

Conceptual query answering:

Q: What is the best way to travel by air from LA to New York?

Flight	Dep	From	To	Time	Stops	Cost
UA 20	10:00	LA	NY	7	Chi[1]	$325
TWA 5	9:00	LA	NY	7	Wash[2]	$365
TWA 1	8:00	LA	Nwk[3]	5	none	$650
UA 2	9:05	LA	NY	5	none	$725

[1] In winter, stopover time varies with weather conditions.

[2] Recommended for winter season.

[3] Surface transportation available from Newark to New York

Neighborhood Inference

Approach

- Translate a query on missing data into a related query on available data

- Provide answers with different degrees of generality, coverage and approximation

Mechanisms

Based on semantic knowledge, organize data in type abstraction hierarchy

Provide multi-level knowledge representation

Support inference between different knowledge levels

- generalization
- specialization
- association

Type Abstraction Hierarchy

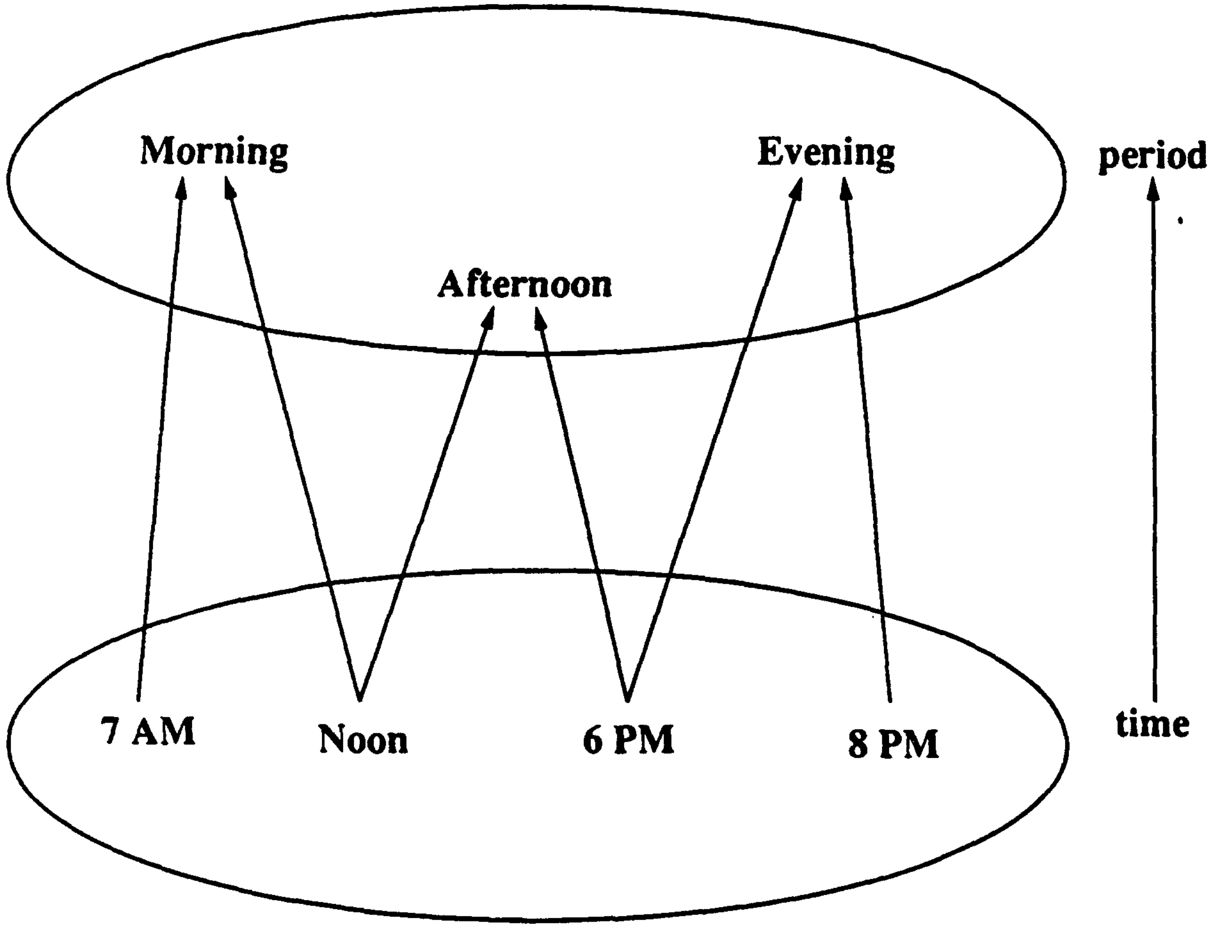
Abstract view of type heirarchy

Integrates:

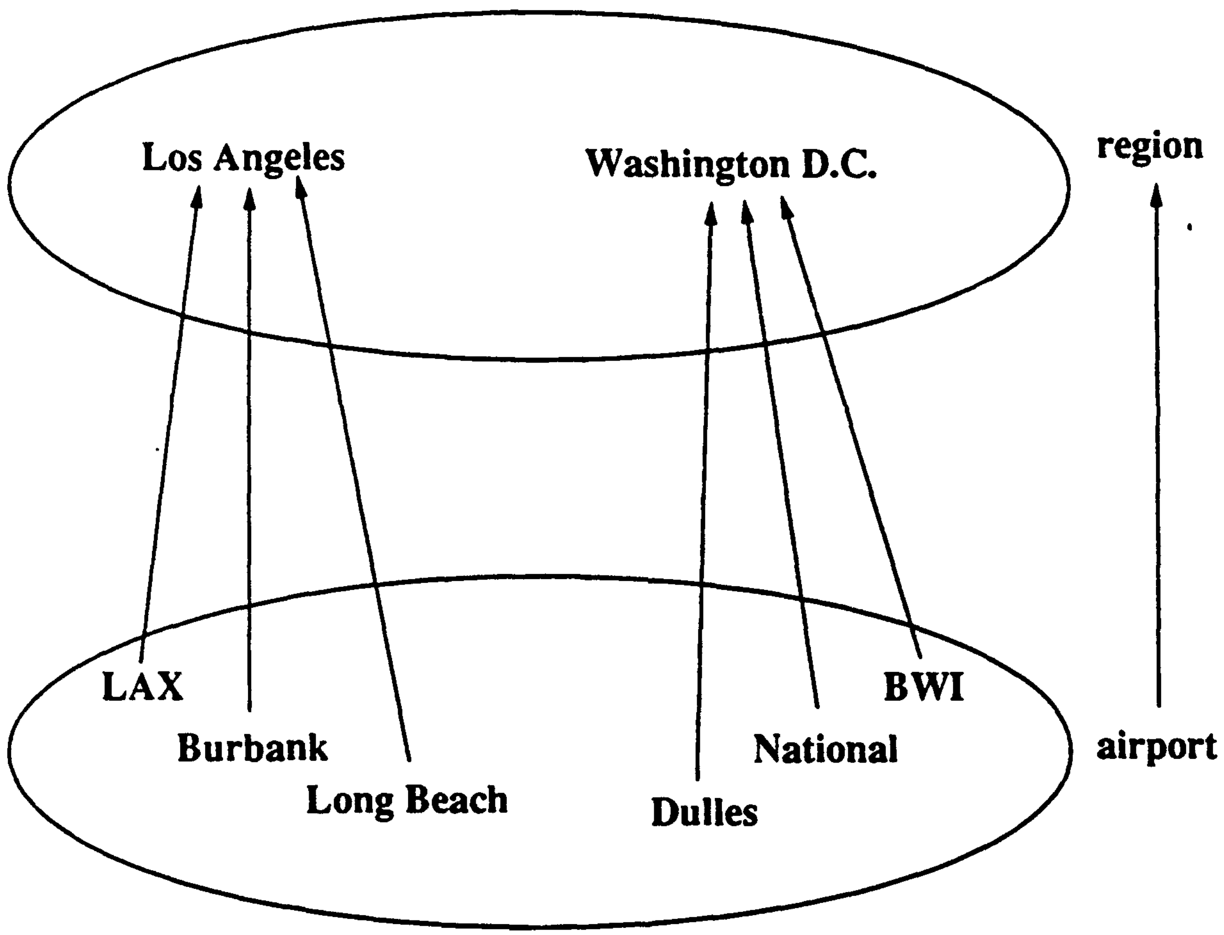
- Subsumption (is_a)
- Composition (part_of)
- abstraction

Provides multi-level knowledge representation

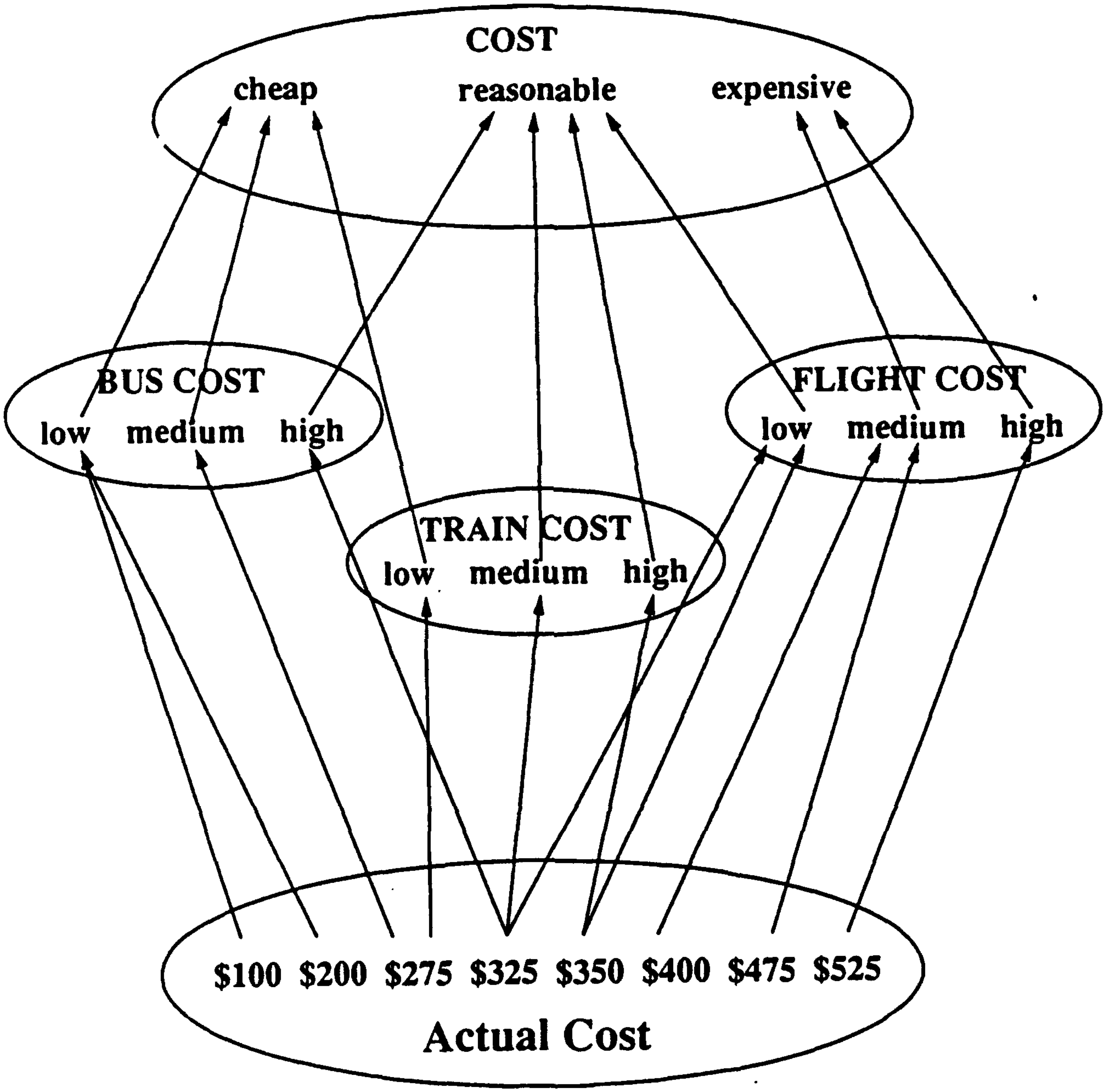
Example Abstraction Hierarchy



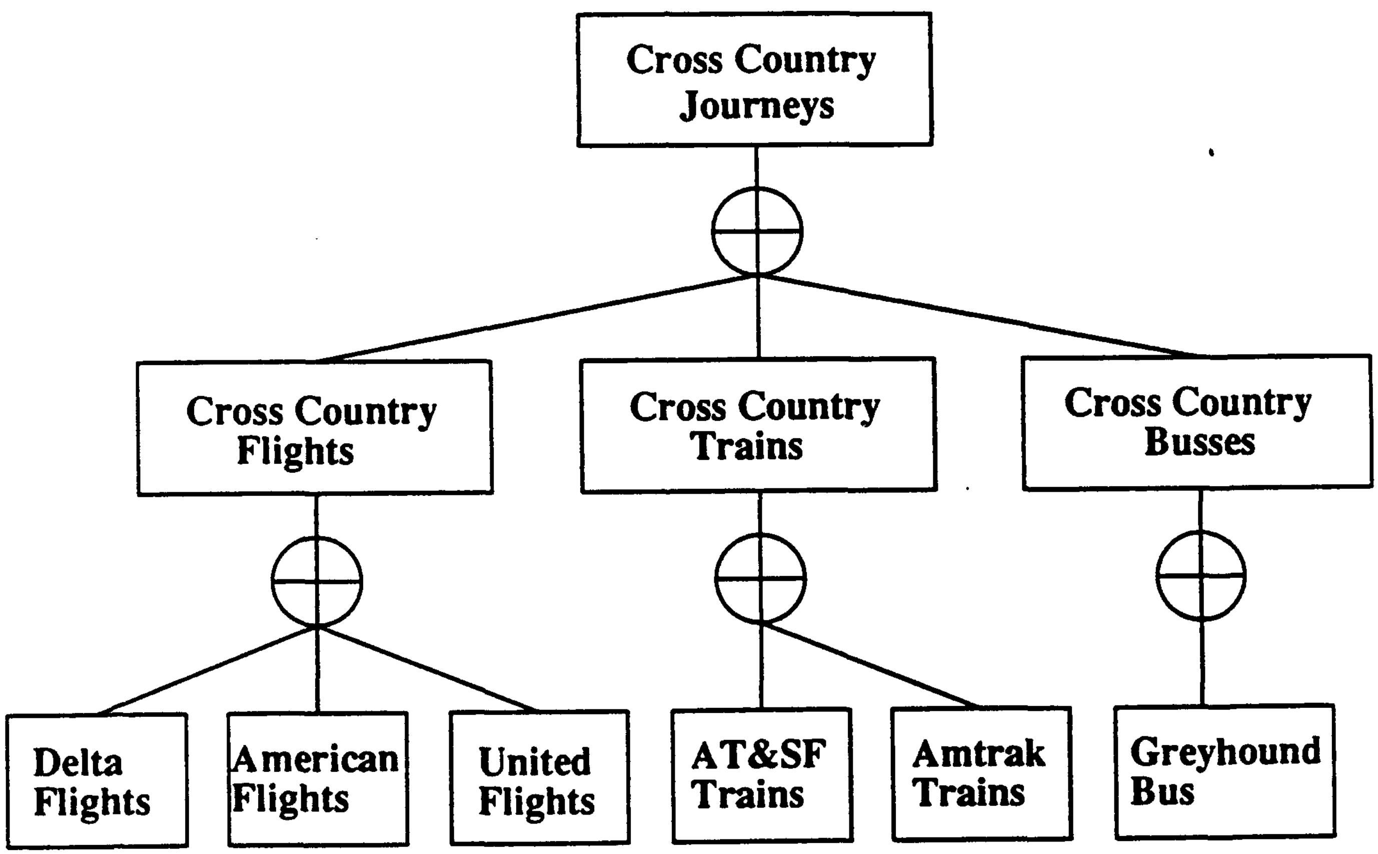
Example Abstraction Hierarchy II



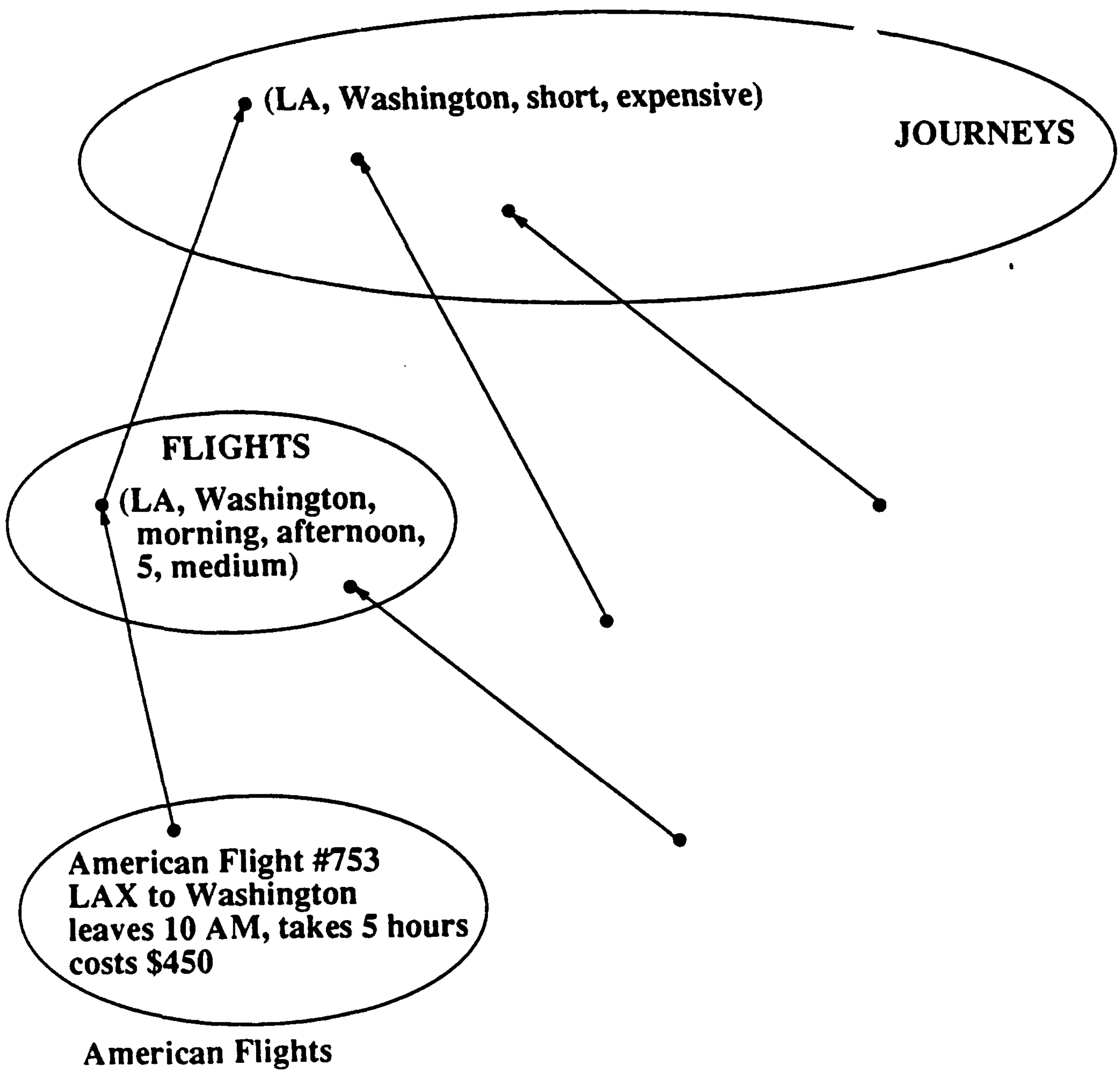
Multilevel Abstraction Hierarchy

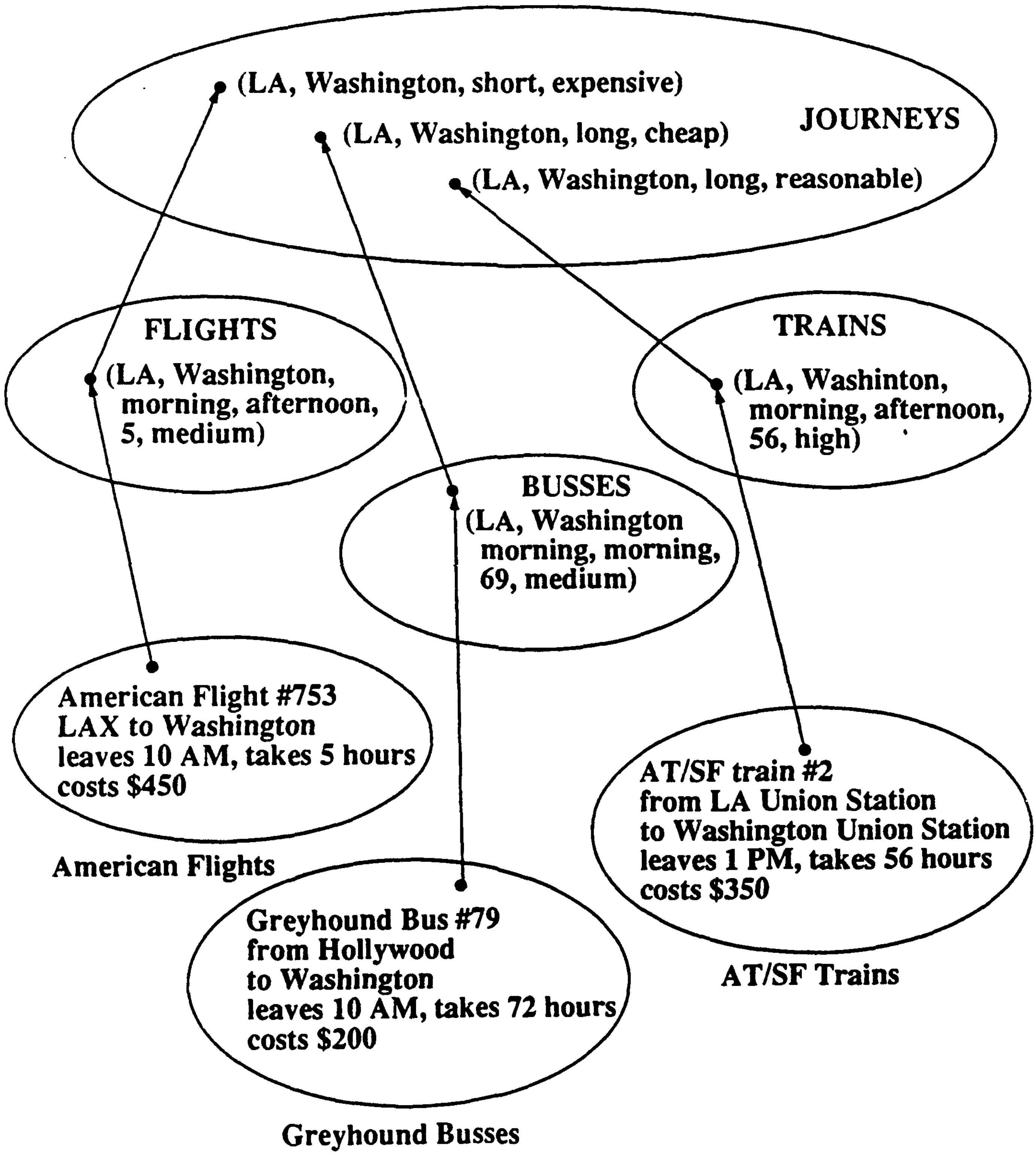


Type Abstraction Hierarchy



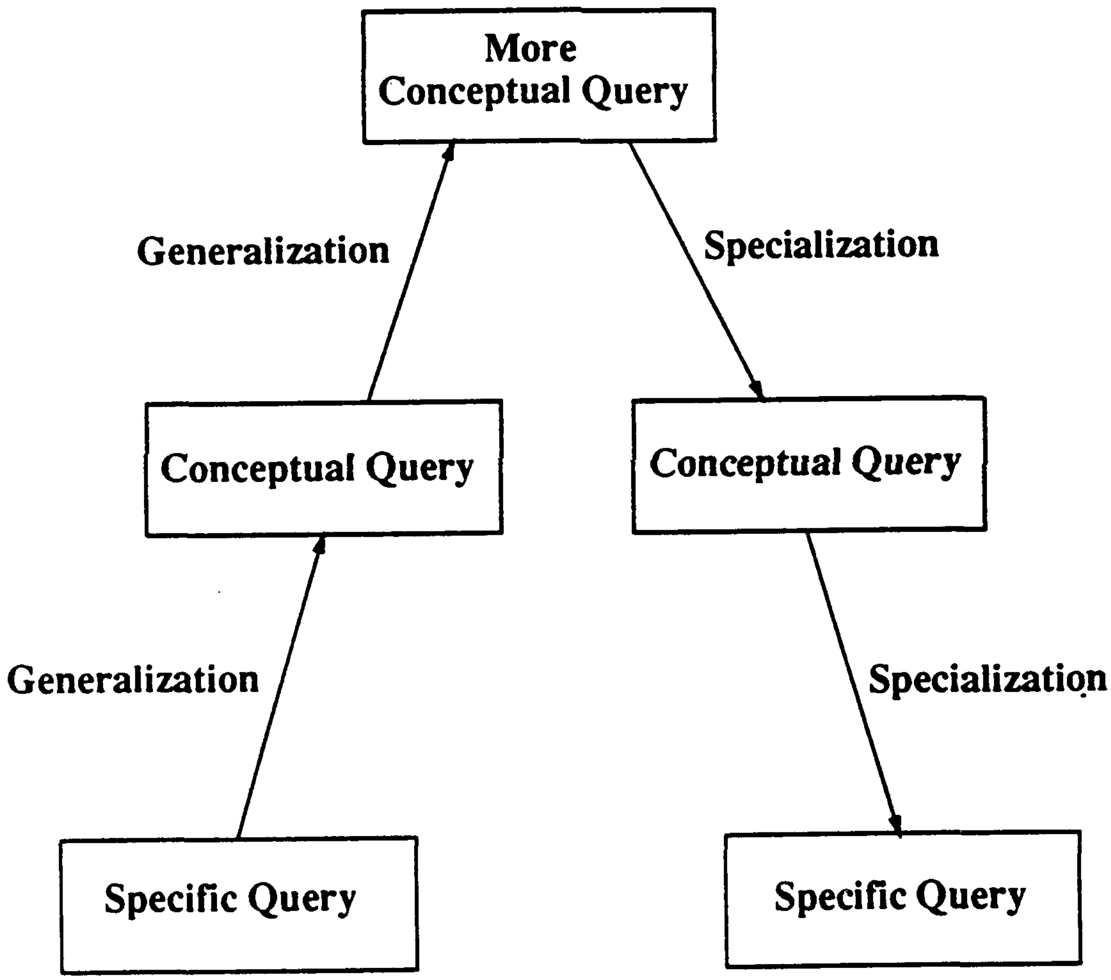
Example of Multilevel Type Abstraction Hierarchy



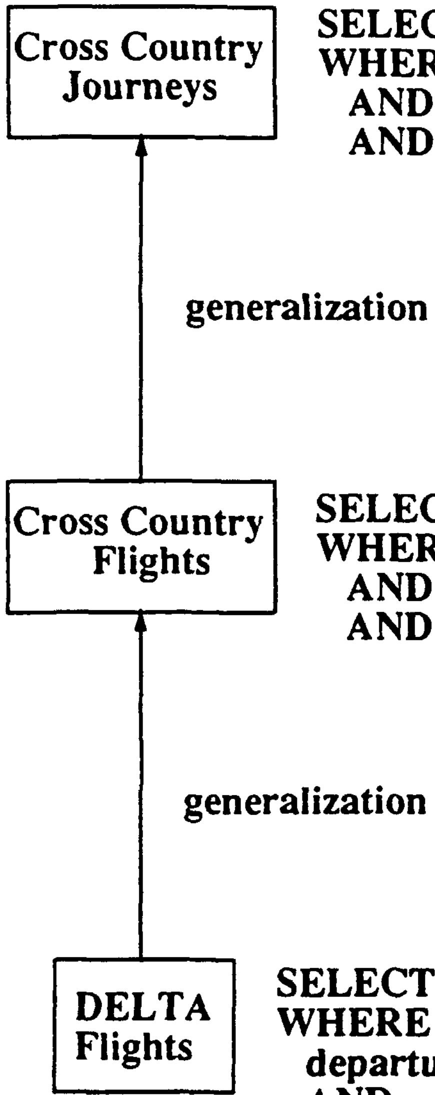


(LA, Washington, short, expensive)
JOURNEYS
(LA, Washington, long, cheap)
(LA, Washington, long, reasonable)
FLIGHTS
(LA, Washington, morning, afternoon, 5, medium)
TRAINS
(LA, Washinton, morning, afternoon, 56, high)
BUSSES
(LA, Washington morning, morning, 69, medium)
American Flight #753 LAX to Washington leaves 10 AM, takes 5 hours costs $450
American Flights
AT/SF train #2 from LA Union Station to Washington Union Station leaves 1 PM, takes 56 hours costs $350
AT/SF Trains
Greyhound Bus #79 from Hollywood to Washington leaves 10 AM, takes 72 hours costs $200
Greyhound Busses

Generalization and Specialization



Generalization




```
SELECT * from cc_journey
WHERE departure_area = "LA"
  AND arrival_area = "Washington"
  AND cost = "reasonable"
```

```
SELECT * from cc_flight
WHERE departure_area = "LA"
  AND arrival_area = "Washington"
  AND flight_cost = "low"
```

```
SELECT * from DELTA_flight
WHERE
  departure_airport = "LAX"
  AND arrival_airport = "National"
  AND fare BETWEEN <$250, $300>
```

Specialization

Cross Country Journeys

```
SELECT * from cc_journey
WHERE departure_area = "LA"
  AND arrival_area = "Washington"
  AND cost = "reasonable"
```

↓ specialization

Cross Country Trains

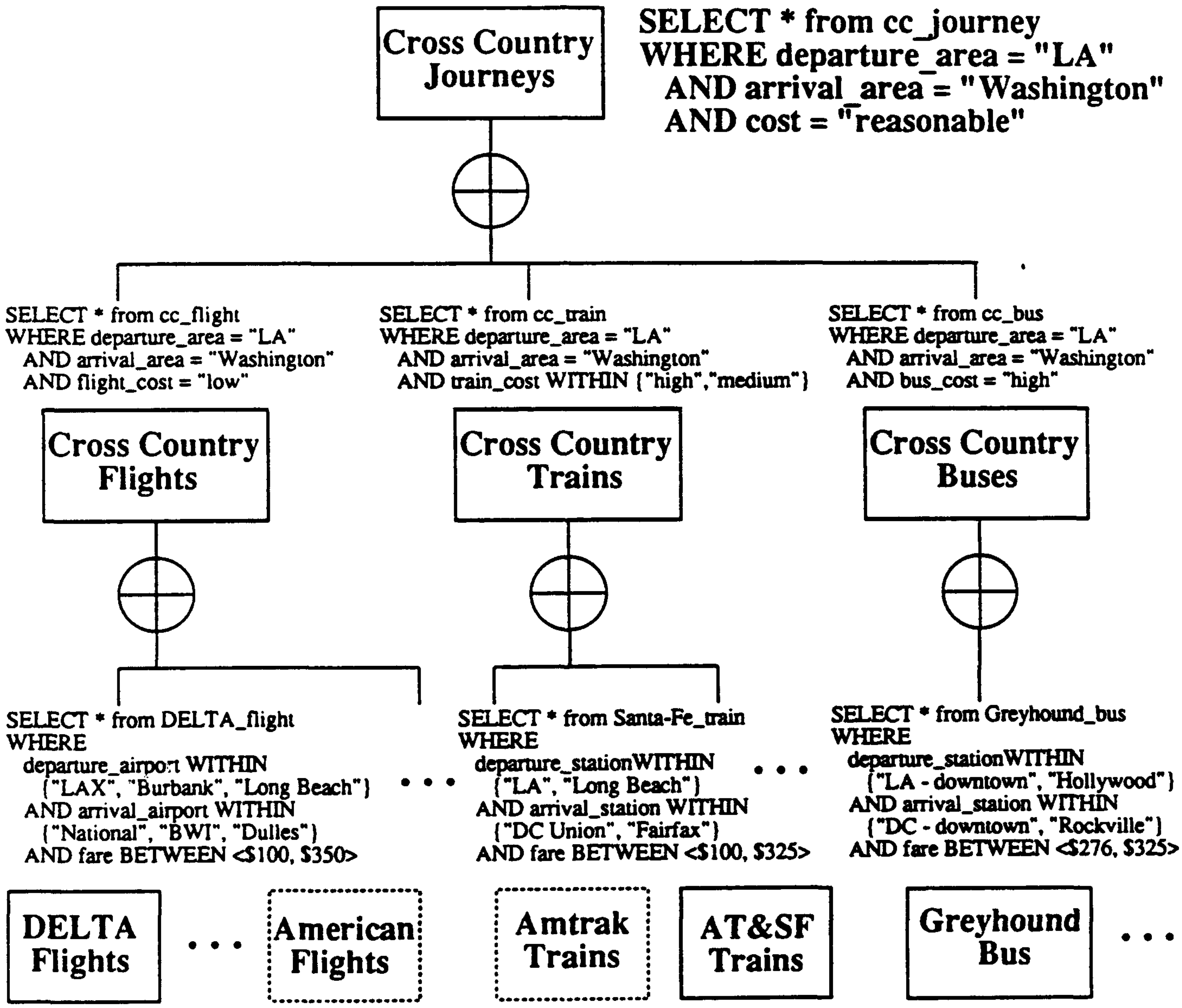
```
SELECT * from cc_train
WHERE departure_area = "LA"
  AND arrival_area = "Washington"
  AND train_cost WITHIN {"high","medium"}
```

↓ specialization

AT&SF Trains

```
SELECT * from Santa-Fe_train
WHERE
  departure_station WITHIN {"LA", "Long Beach"}
  AND arrival_station WITHIN {"DC Union", "Fairfax"}
  AND fare BETWEEN <$275, $400>
```

All possible Specializations



Relaxation Mechanism

Relaxable Variables

- Types
- Attributes
- Attribute Values

Nearness Measure

Context Dependent

- Time
- Space
- Concept

Relaxation Specifics

Explicit (by user)

- relaxable range specified by C-SQL
- primitives (relaxable predicates)

Implicit (by system)

- generalization & specialization
- based on
  - context
  - type abstraction hierarchy
  - abstract domain values

Interactive

- through user/system interaction

C-SQL

An extended Query Language

Primitives (Predicates) for Cooperative Query Answering

Context Free

- **approximate**

  ~9 AM
- **inclusion**

  between (7 AM, 11 AM)
- **set membership**

  within {'LAX','Burbank'}

Context Sensitive (nearness)

- Restaurant **near-to** 'Redondo Beach'
- Airport **near-to** 'LAX'
- Chinese Restaurant **nearest-to** 'UCLA'

Relaxation Order may be specified

- **relaxation-order** (food style, location)

Original Query

select * *from* delta-flight
where dep-airport = "Newark"
and arr-airport **near-to** "Lax"

Relaxed Query

select * *from* delta-flight
where dep-airport = "Newark"
and arr-airport **within**
{"Burbank", "Long-Beach", "Lax"}

delta-flight

Fl#	dep_airport	arr_airport	dep_time	arr_time	fare
151	NEWARK	LAX	845	1319	400
367	NEWARK	BURBANK	1000	1428	350
29	NEWARK	LONG-BEACH	935	1422	300

Nearer

select * *from* delta-flight
where dep-airport = "Newark"
and arr-airport **within**
{"Burbank", "Long-Beach", "Lax"}

delta-flight

Fl#	dep_airport	arr_airport	dep_time	arr_time	fare
151	NEWARK	LAX	845	1319	400
367	NEWARK	BURBANK	1240	1708	350

Nearer

select * *from* delta-flight
where dep-airport = "Newark"
and arr-airport **within**
{"Burbank", "Long-Beach", "Lax"}

delta-flight

Fl#	dep_airport	arr_airport	dep_time	arr_time	fare
151	NEWARK	LAX	845	1319	400

Original Query

select * *from* restaurants
where type = "Chinese"
and location **near-to** "Lax"

Relaxed Query

select * *from* restaurants
where type = "Chinese"
and location **within**
{"Westchester", "Inglewood", "Lax"}

restaurants

name	type	location	address
MANDARIN WOK	Chinese	Inglewood	424, Imperial Hwy.

Original Query

select * *from* american-flight
where dep-airport = "National"
and arr-airport **near-to** "Lax"
and dep-time *between* (1000 ~**1100**)

Relaxed Query

select * *from* american-flight
where dep-airport = "National"
and arr-airport **within**
{"Lax", "Burbank", "Long-Beach"}
and dep-time *between* (1000 **1200**)

american-flight

Fl#	dep_airport	arr_airport	dep_time	arr_time	fare
305	NATIONAL	LAX	1200	1624	385

Patterns

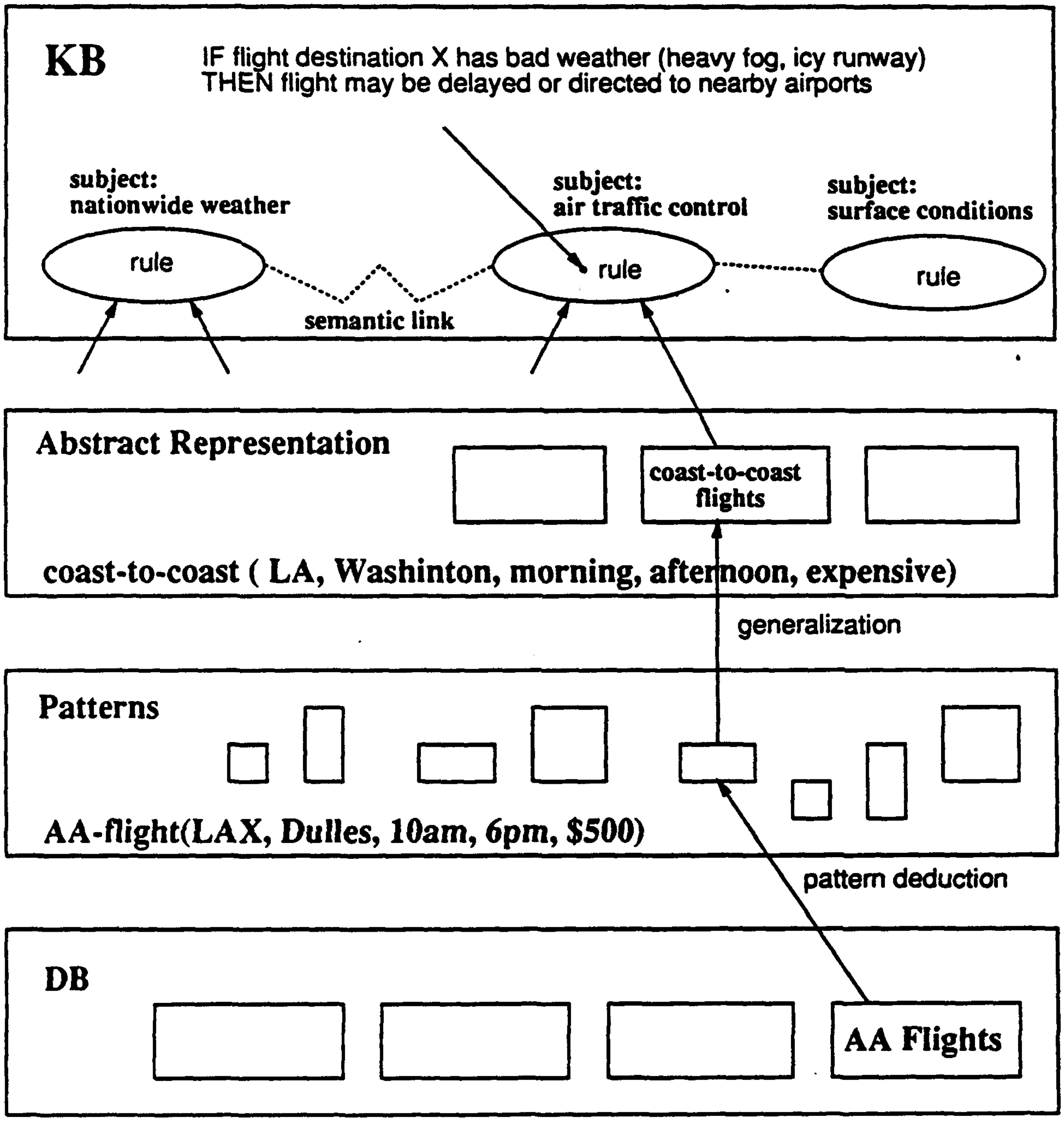
Specified by query conditions one or more attributes

- arrival time = 10 AM
- departure airport = LAX

Advantages

- Finer granularity than types
- More specific knowledge representation
- Complex queries can be derived from logical operations on patterns
- Unified interface between KB and DB

Association



Original Query

select * *from* delta-flight
where dep-airport = "Lax"
and arr-airport = "O'Hare"
and arr-time *between* (1700 1900)

delta-flight

Fl#	dep_airport	arr_airport	dep_time	arr_time	fare
1226	LAX	O'Hare	820	1825	356

Pattern: **arr-time** *between* (1700 1900)
Pattern name: **rush-hour-arrival-time**

Associated Subject: **Surface Traffic Conditions**

Information derived from association:
Extensive traffic delays on major freeways between airport and downtown.

Pattern: **dep-airport = "Lax"**
Pattern name: **los-angeles–dep-airport**

Associated Subject: **Weather**

Information derived from association:
Adverse weather conditions over LA in winter might delay flights.

Pattern: **arr-airport = "O-Hare"**
Pattern name: **Chicago-arr-airport**

Associated Subject: **Weather**

Information derived from association:
Snowstorms over Chicago in winter might divert flights to other airports.

Implementation

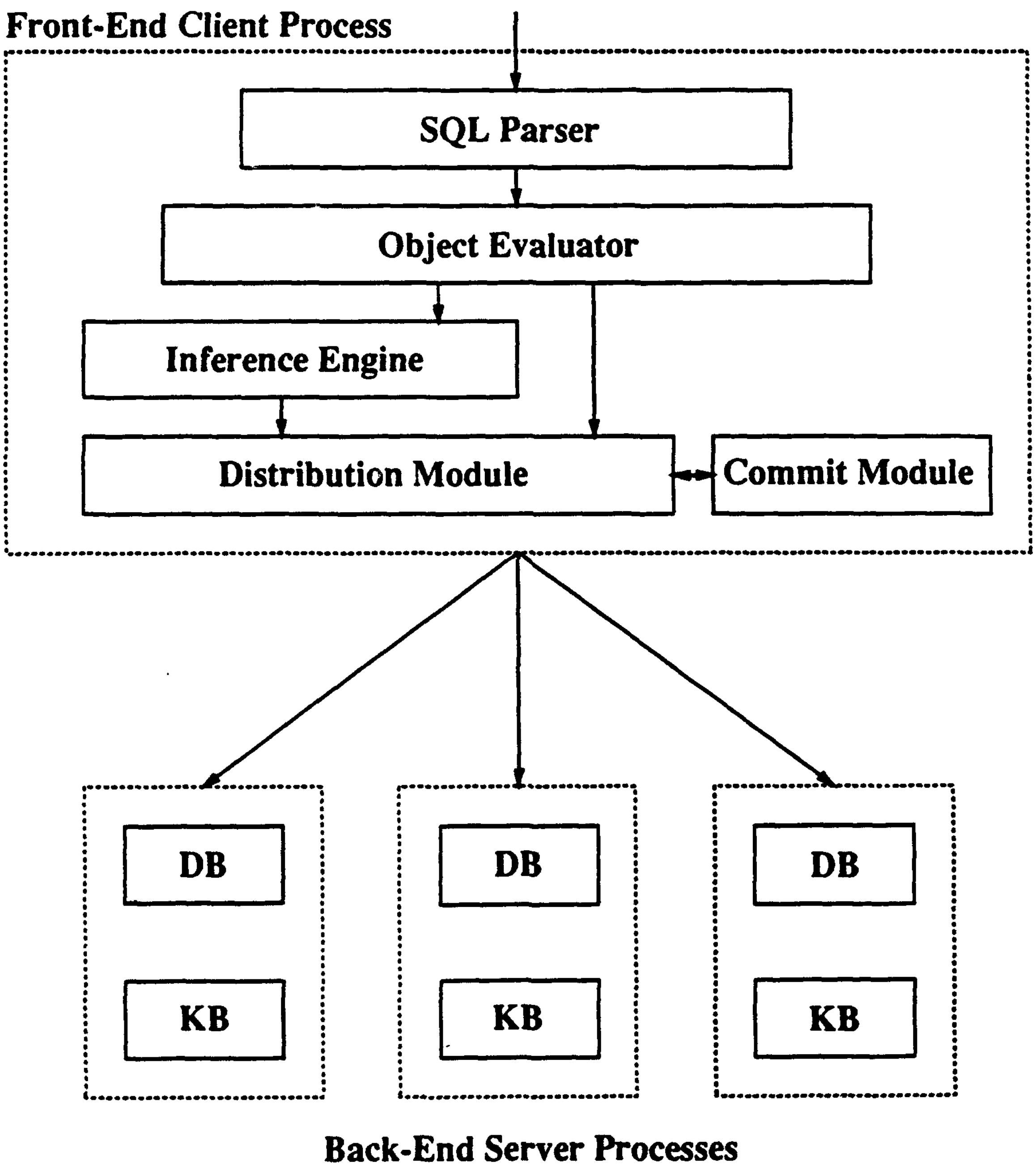
Database Management System

Sybase

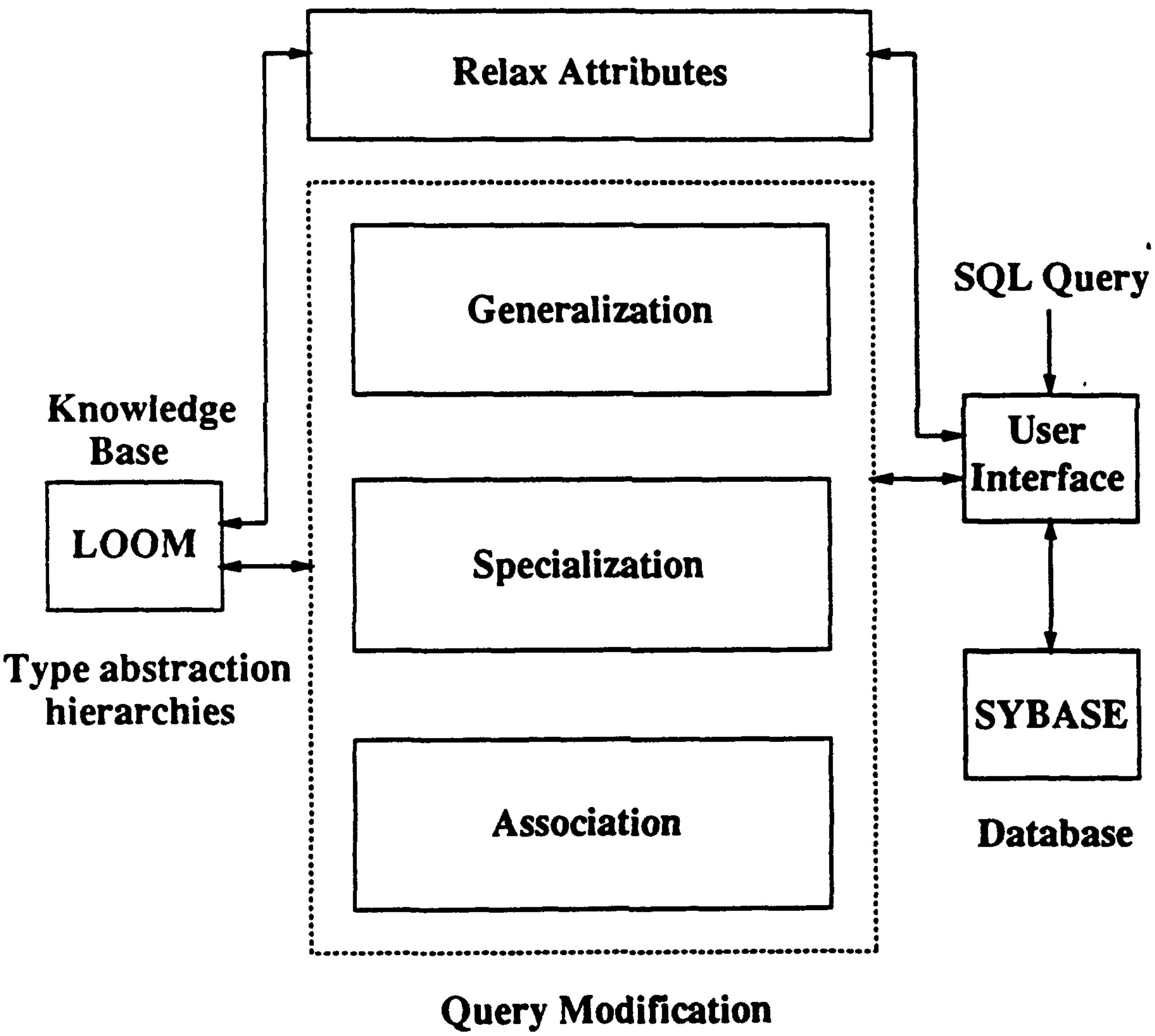
Knowledge Representation Package

LOOM

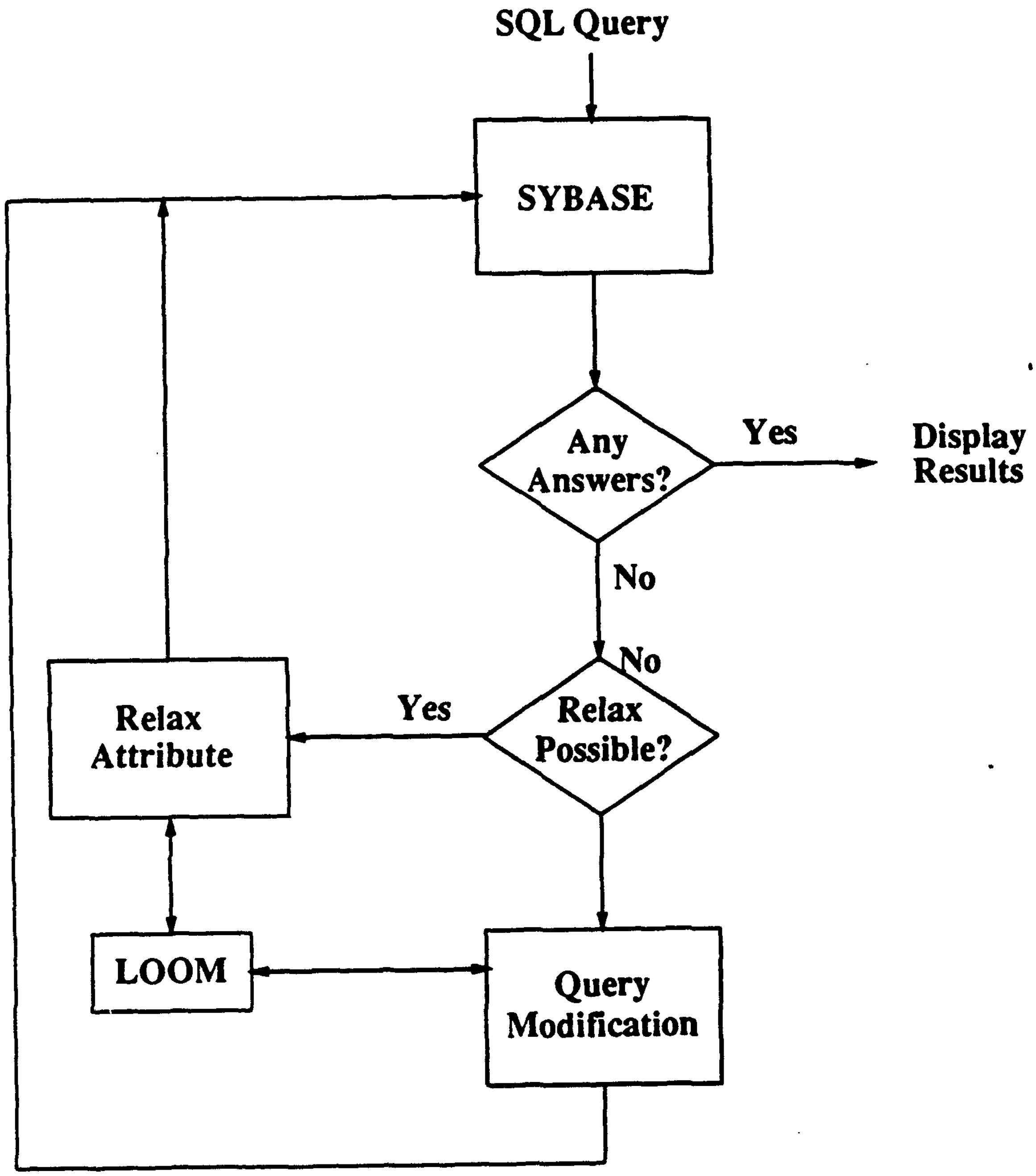
Distributed Architecture



Cooperative Query Answering Architecture

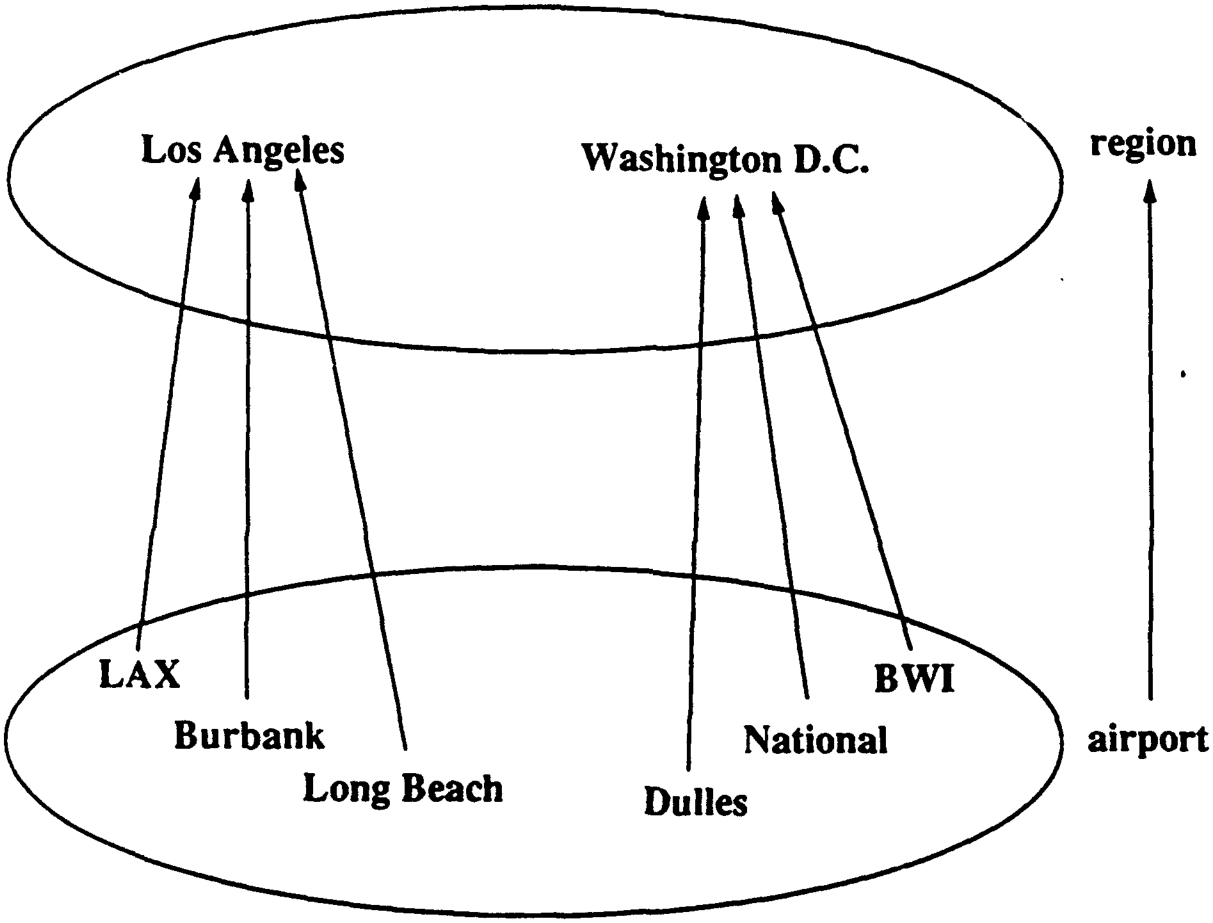


Flow Diagram



Abstraction Representations

Graphical Representation



Logical Representation

Area	*Airport*
Los Angeles Washington	LAX, Burbank, Long Beach Dulles, Balt., National

Abstraction Representations, cntd.

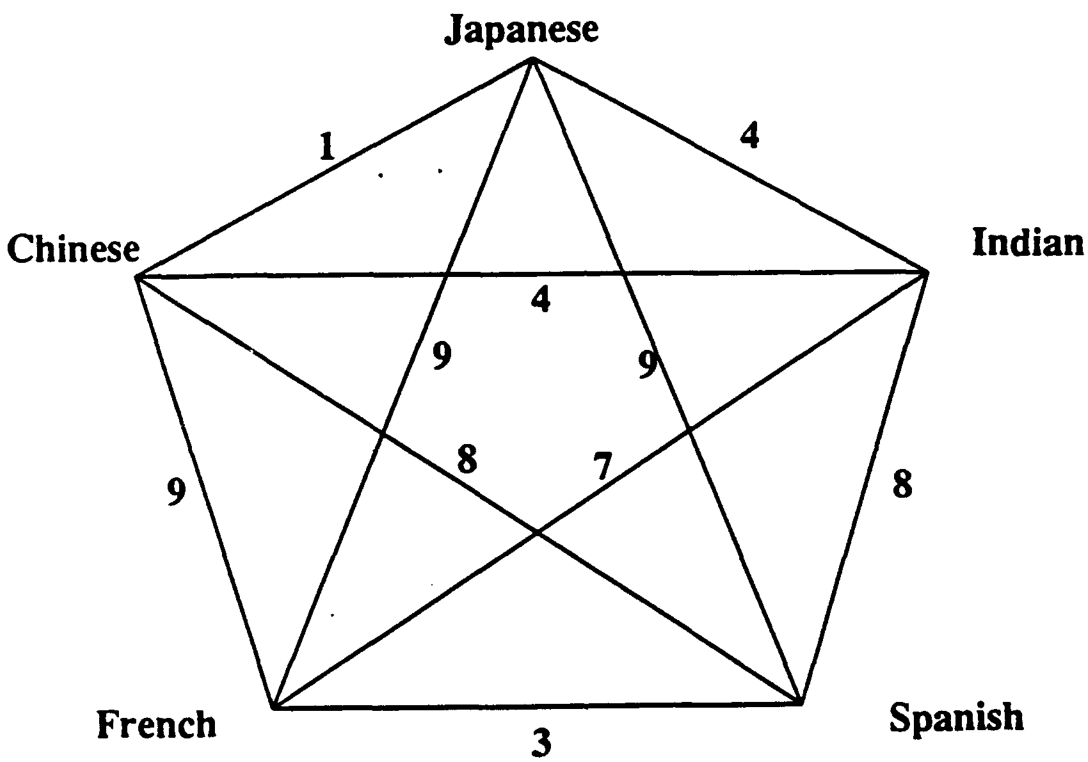
LOOM Representation

- (defconcept area :is :primitive)
- (defconcept airport :is :primitive)
- (defrelation abstraction-of)
- (tellm (area los-angeles))
- (tellm (area washington))
- (tellm (airport lax))
- (tellm (airport national))
- (tellm (abstraction-of los-angeles lax))
- (tellm (abstraction-of washington national))

  •
  •
  •

Nearness Representations

Graphical



Matrix

	Japanese	Chinese	Indian	French	Spanish
Japanese	0				
Chinese	1	0			
Indian	4	4	0		
French	9	9	7	0	
Spanish	9	8	8	3	0

Nearness Measures are Context Sensitive

attribute	*unit of semantic distance*
departure time	one hour
flight time	ten minutes
Airport	100 miles
Restaurant	5 miles

Summary

Cooperative Query Answering provides
- approximate answers
- conceptual answers
- answers to imprecise queries

Type Abstraction Hierarchy
- framework for deriving answers to cooperative queries

CSQL
- provides explicit and implicit relaxation facilities for both types and attributes

Relaxation based on
- query context
- nearness measures
- relaxation order

Continuing Work

Nearness Measures

Association of Subjects

Knowledge maintenance

Scalability

APPENDIX K

MONOTONE DATABASES

(Imprecise Query Processing)

J. W. S. Liu, S. Vrbsky and X. Song

Real-Time Systems Laboratory

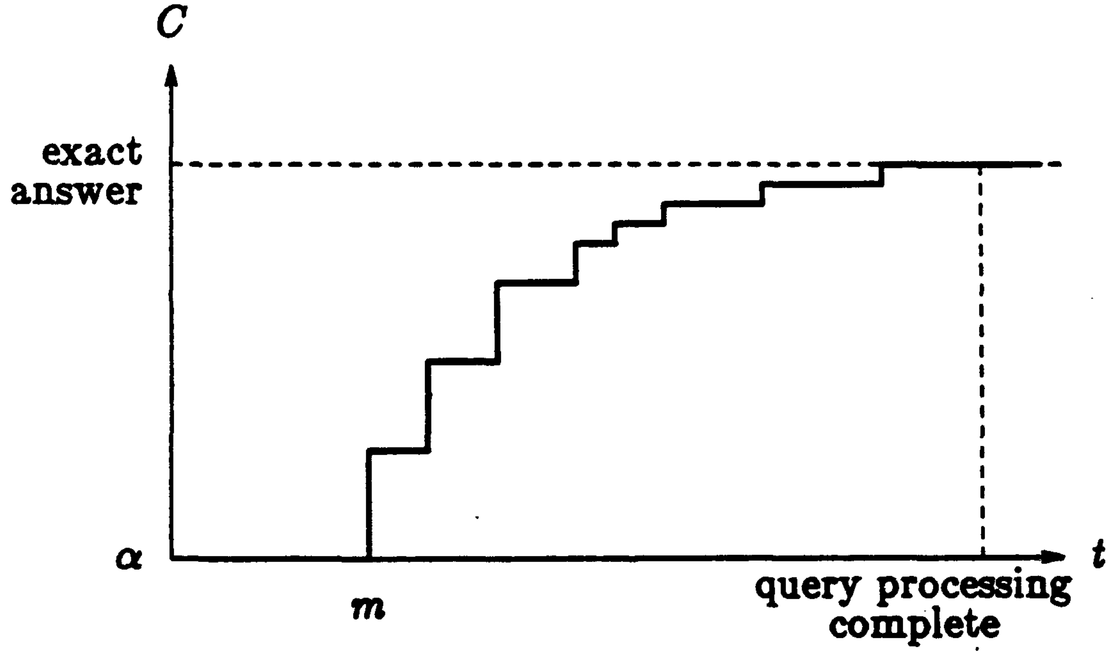
Department of Computer Science

University of Illinois, Urbana, IL

- Imprecise computation technique and monotonicity
- Approximate relational algebra and an approximation semantics for set-valued queries
- Query processing strategy for returning monotonically improving answers
- Future work

Providing Approximate Answers

An approach to providing flexibility in scheduling and concurrency control



Exact Answers Vs *Approximate Answers*

Related Work on Approximations in Databases

- Incomplete databases — *Tzvieli, Lispki, Imielinski, Winslett*
- Vague and generalized queries — *Motro, Chauhuri*
- Disjunctive data — *Liu and Sunderraman*
- Monotone query processing
  - set-valued results, missing data in rule-based systems — *Buneman, Davidson and Watters*
  - statistical databases — *Ozsoyoglu and Ozsoyoglu*

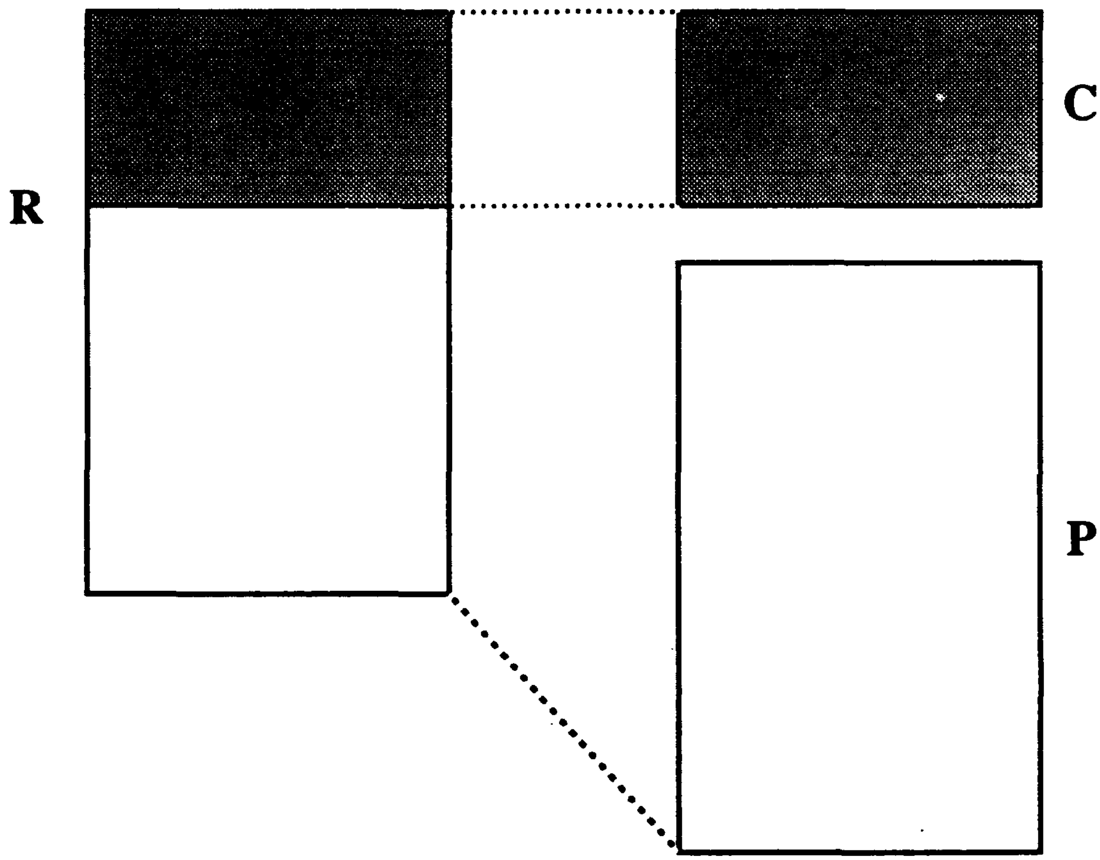
Where can flight United 941 land?

The Exact Relation

Location	Appro. Dist.
Chi-O'Hare, IL	350
Manchester, NH	250
Peoria, IL	400
Syracuse, NY	100

An Approximate Relation

Location	Appro. Dist.	
Chi-O'Hare, IL	350	*Certain tuples*
Peoria, IL	400	
Paris, IL	380	*Possible tuples*
Hell, MI	330	
Manchester, NH	250	
Berlin, MA	300	
Syracuse, NY	100	



An Approximation Relation

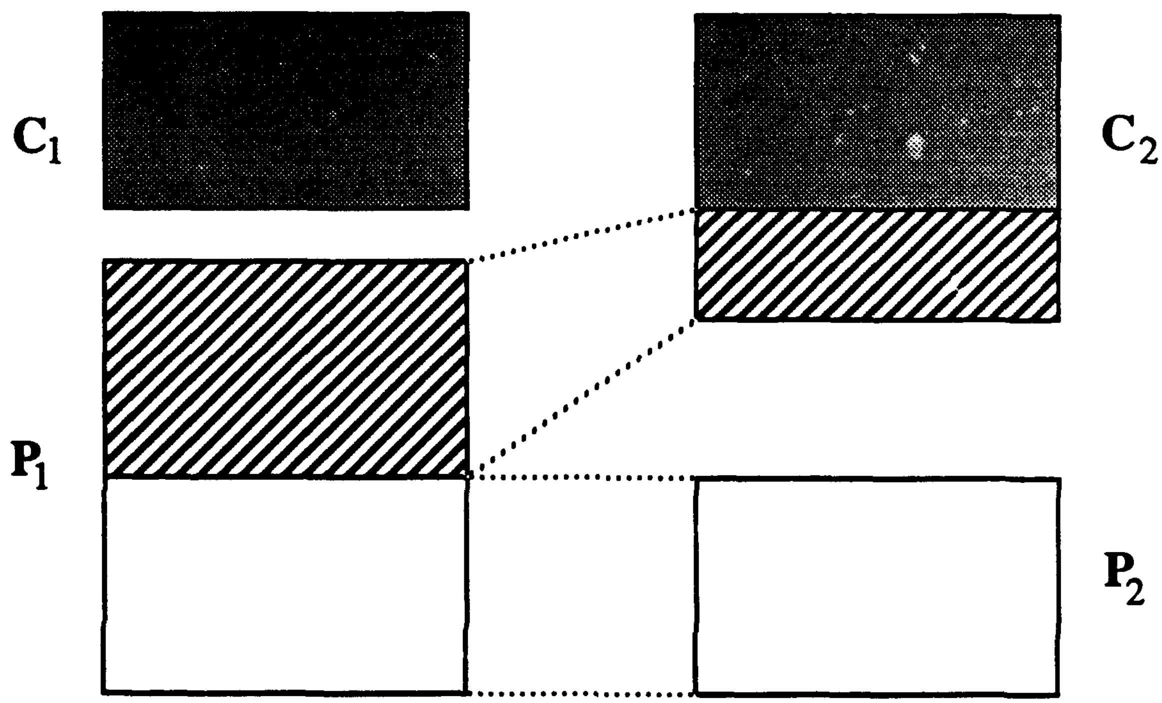
What is an approximate answer?

- Observations:
  - — An answer to a relational query is a relation R, that is, a set of tuples.
  - — A subset of R approximates the set.
  - — A superset of R approximates the set.
- A semantics of approximation for set–valued queries:
  - — An approximation of R has two parts:
    - ◦ The set C of ***certain tuples*** is a subset of R.
    - ◦ The set P contains ***possible tuples*** in R.
    - ◦ The union of C and P is a superset of R.
  - — The worst approximation of R is one which contains no certain tuples.
  - — The best approximation of R is itself.

Location	Approx. Dist.
Chi-O'Hare, IL	350
Peoria, IL	400
Syracuse, NY	100
Paris, IL	380
Hell, MI	330
Manchester, NH	250
Berlin, MA	300

Location	Approx. Dist.
Chi-O'Hare, IL	300
Peoria, IL	400
Paris, IL	380
Manchester, NH	250
Berlin, MA	300
Syracuse, NY	100

Location	Appro. Dist.
Chi-O'Hare, IL	300
Peoria, IL	450
Paris, IL	380
Hell, MI	330
Manchester, NH	250
Berlin, MA	300
Syracuse, NY	100



Improved Approximation

Approximate Relational Algebra Model

- Approximate relation R:
  - — subset and superset of a standard relation S
  - — certain tuples C and possible tuples P
  - — R approximates S if:

    $C \subseteq S$ and $C \cup P \supseteq S$

- Partial order over set of approximate relations:

  $R_i \geq R_j$ if:

  $$C_i \supseteq C_j,\ P_i \subseteq P_j,\ C_i - C_j \subseteq P_j - P_i$$

- Partial order set is a lattice:

  $(\emptyset, \upsilon)$ – worst approximation

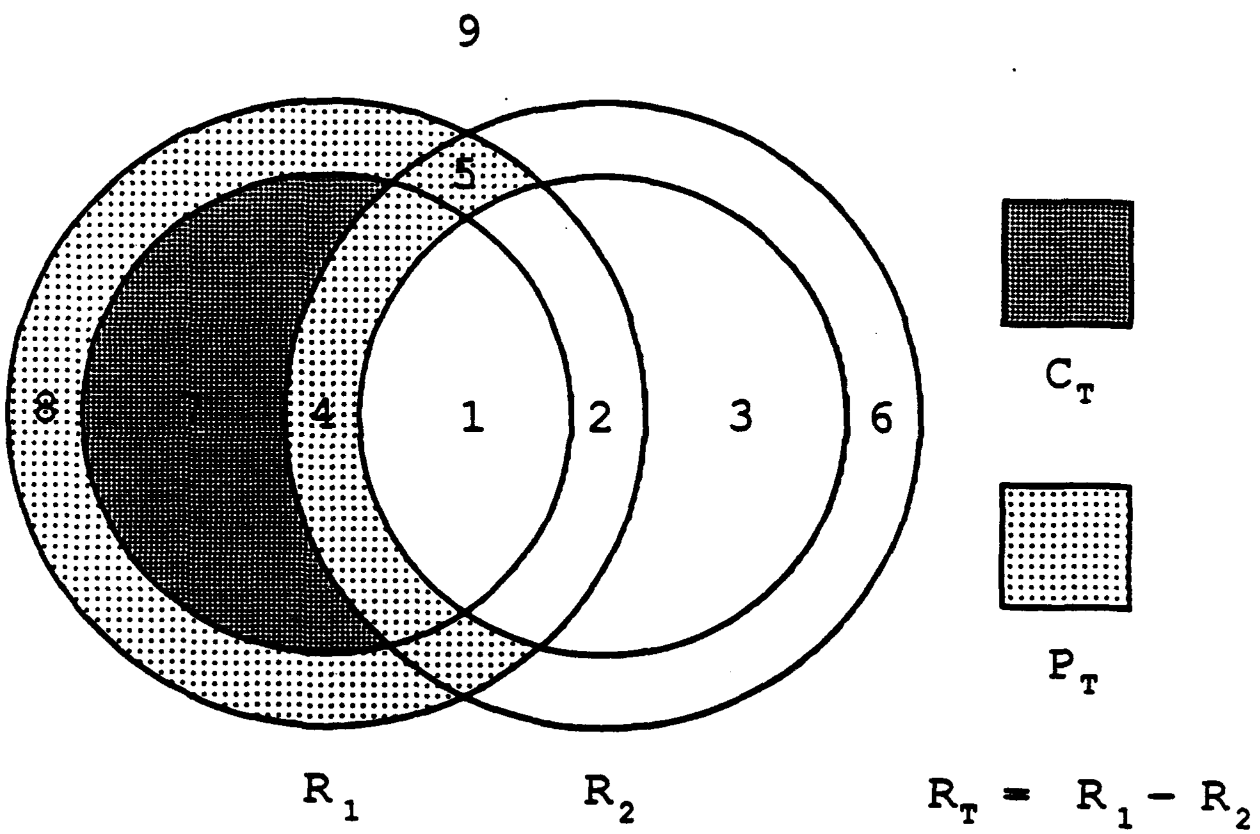
  $(S, \emptyset)$ – best approximation

Approximate Relational Algebra Operations

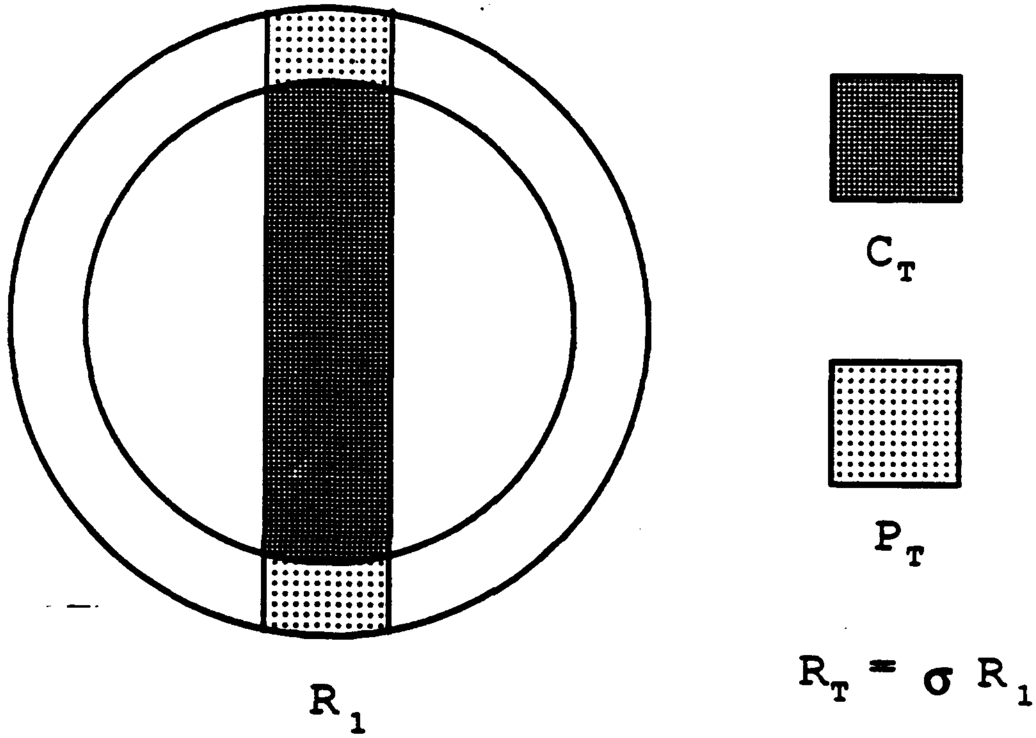
Union, set difference, select, project, cartesian product

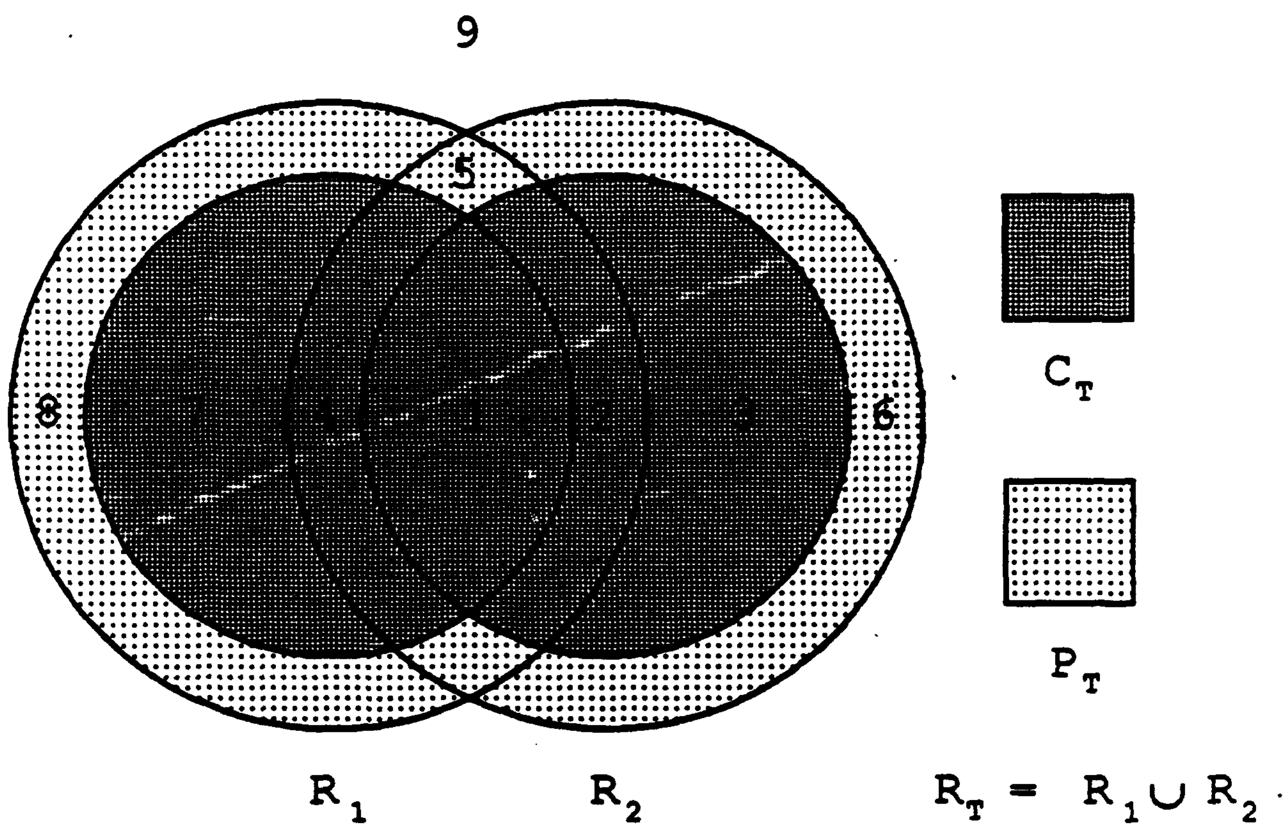
- Every approximate operation is monotone

Approximate Operation	C_T	P_T
Union: $R_T = R_1 \cup R_2$	$C_T = C_1 \cup C_2$	$P_T = (P_1 \cup P_2) - C_T$
Difference: $R_T = R_1 - R_2$	$C_T = C_1 - R_2$	$P_T = (P_1 - R_2) \cup (P_2 \cap R_1)$
Select: $R_T = \sigma_{att=val} R_1$	$C_T = \sigma_{att=val} C_1$	$P_T = \sigma_{att=val} P_1$
Project: $R_T = \pi_{att} R_1$	$C_T = \pi_{att} C_1$	$P_T = \pi_{att} P_1$
Cart. Prod: $R_T = R_1 \times R_2$	$C_T = C_1 \times C_2$	$P_T = (R_1 \times R_2) - C_T$



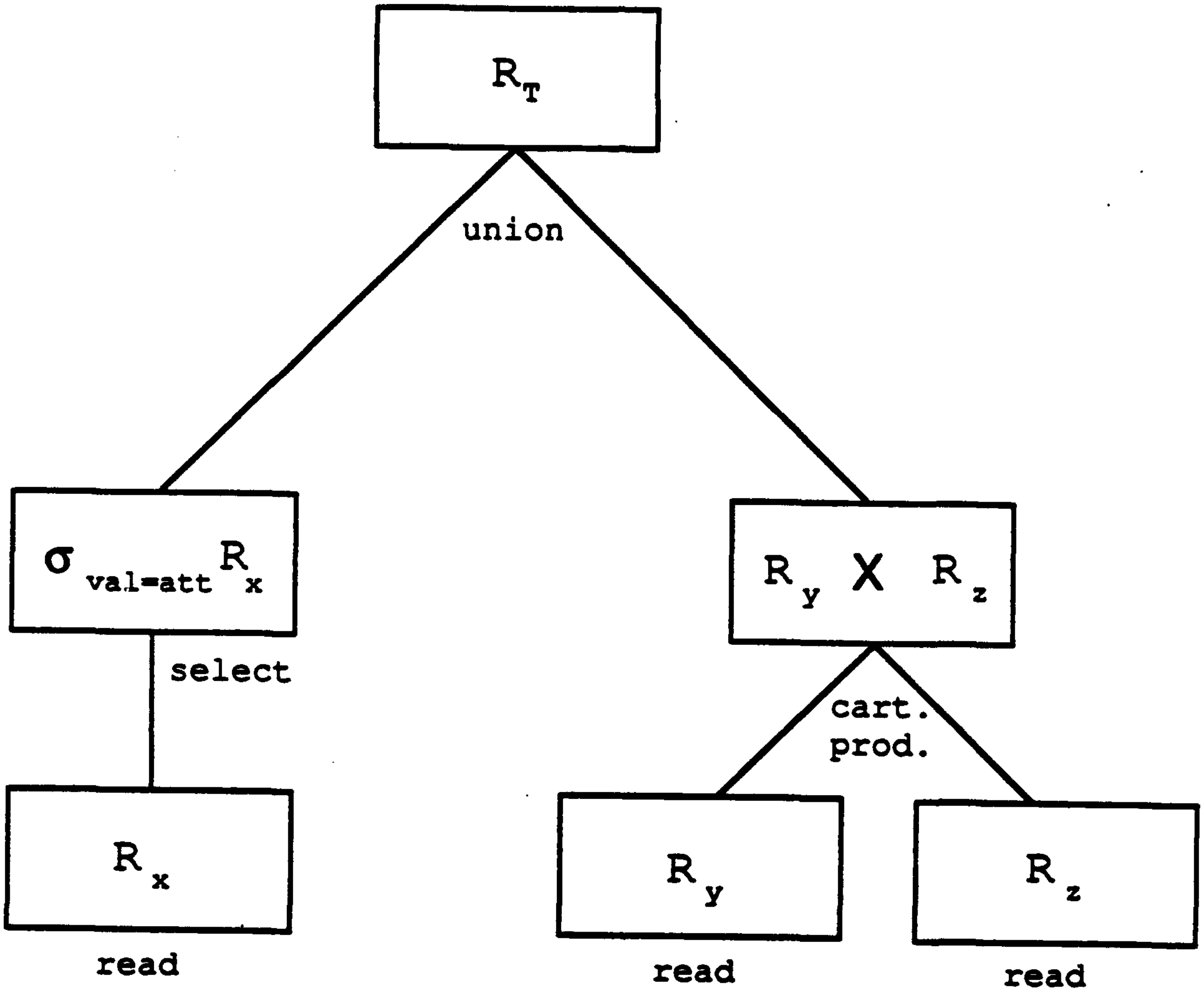
9
5
8
7
4
1
2
3
6
C_T
P_T
R_1
R_2
$R_T = R_1 - R_2$


C_T
P_T
R_1
$R_T = \sigma R_1$


9
5
8
6
C_T
P_T
R_1
R_2
$R_T = R_1 \cup R_2$

Monotone Query Processing At Each Node

- Approximate operations
- Approximate relations initialized to $(\emptyset, \upsilon)$
- Monotonically improving answers
  - tuples migrate from possible to certain part
  - improved values propagate upward with each update



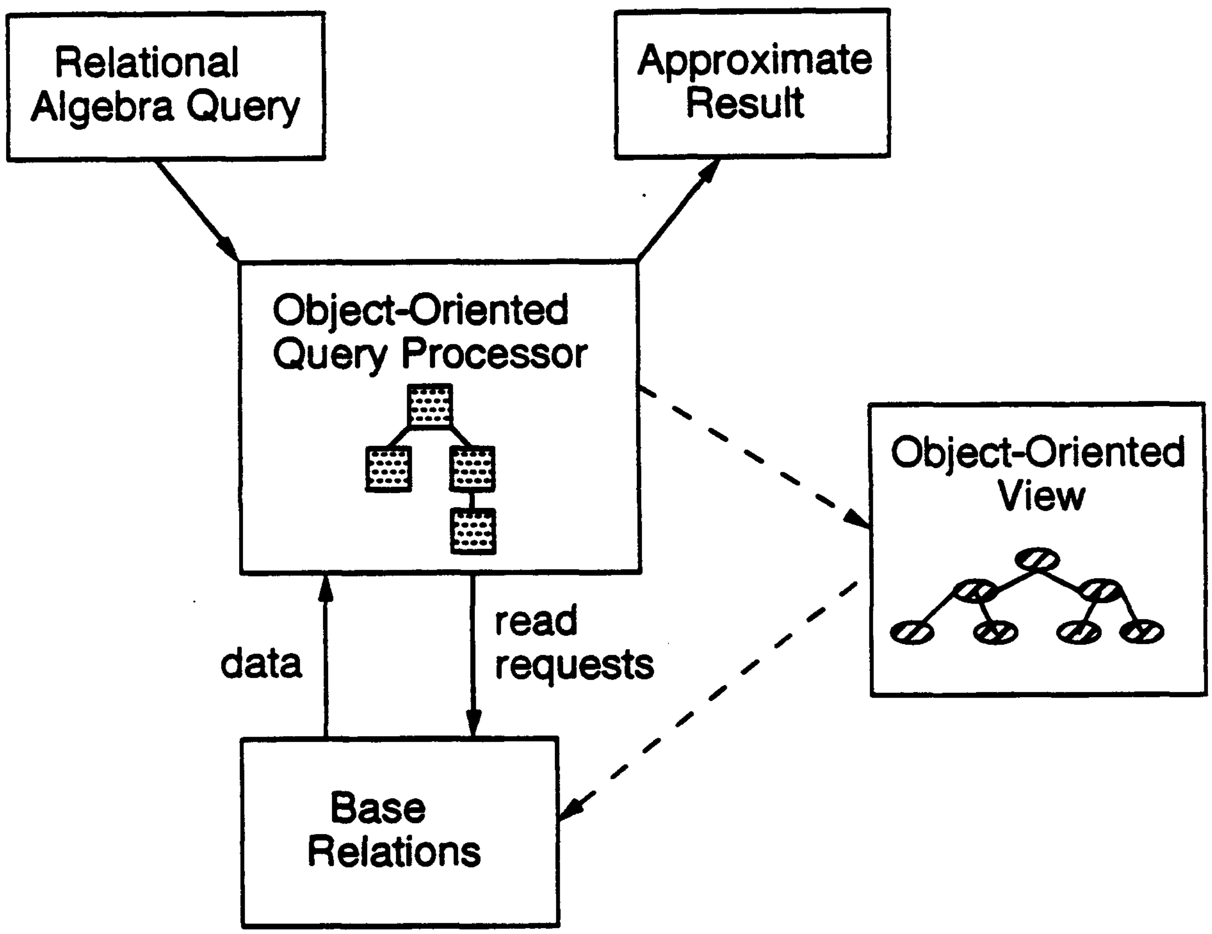
R T
union
σ val=att R x
select
R x
read
R y X R z
cart. prod.
R y
read
R z
read

APPROXIMATE

(*A Prototype Monotone Query Processor*)

- Supports processing of relational algebra queries on relational database systems
- Uses an objected-oriented approach to implementation:
  - — relies on an object-oriented view for semantic support in identification of initial approximations
  - — avoids operations on possible tuples
  - — provides lazy evaluation of possible classes upon user interrupt or faults
- Improves data availability and query processing time predictability

Approximate Query Processing

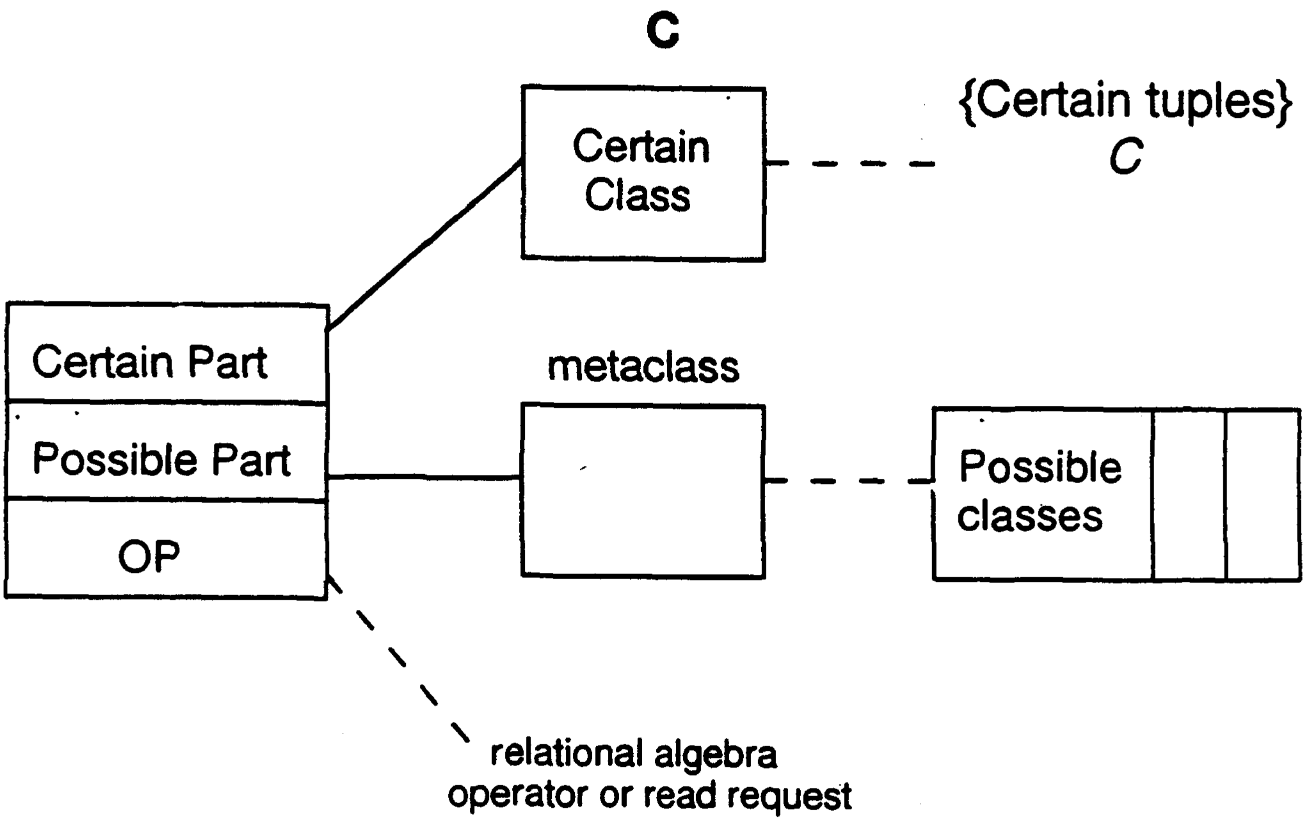


Object–Oriented Monotone Query Processor

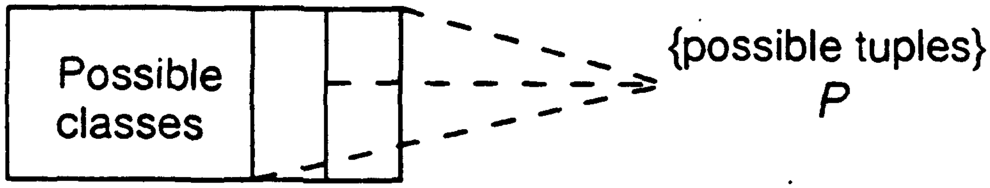
- How to implement the monotone query processor?
  - — initial approximations
    good approximation
  - — approximate relational algebra operations
    operations on possible tuples

- Object–oriented monotone query processor
  - — semantic support for efficient implementation
  - — external view is relational
  - — underlying relational architecture unchanged

Approximate Class



Possible Classes



Approximate Object

- Approximate object $\mathbf{R} = (C, \mathbf{P})$

- Describes a set of approximate relations of S

$\mathbf{R}_i \geq \mathbf{R}_j$ if:

$$C_i \supseteq C_j, \quad \mathbf{P}_i \subseteq \mathbf{P}_j,$$

$$C_i - C_j \subseteq \text{instances of } P_j - P_i$$

If $\mathbf{R}_i \geq \mathbf{R}_j$ then $R_i \geq R_j$

Approximate Class Methods

- Approximate class contains four methods:

  new

  generalization

  specialization

  migration

- Methods are monotone

- Methods invoked during query processing

Approximate Relational Algebra Operations

- Redefined using generalization method (genl)

Approximate Operation	C_T	$\mathbf{P}_T$
Union: $\mathbf{R}_T = \mathbf{R}_1 \cup \mathbf{R}_2$	$C_T = C_1 \cup C_2$	$\mathbf{P}_T = (\mathbf{P}_1 \cup \mathbf{P}_2) - \mathbf{genl}(C_T)$
Difference: $\mathbf{R}_T = \mathbf{R}_1 - \mathbf{R}_2$	$C_T = C_1 - C_2 - \{ t \mid \mathbf{genl}(t) \in \mathbf{P}_2 \}$	$\mathbf{P}_T = (\mathbf{P}_1 - \mathbf{genl}(C_2) - \mathbf{P}_2) \cup (\mathbf{P}_2 \cap \mathbf{genl}(C_1) \cap \mathbf{P}_1)$
Select: $\mathbf{R}_T = \sigma_{att=val}\, \mathbf{R}_1$	$C_T = \sigma_{att=val}\, C_1$	$\mathbf{P}_T = \sigma_{att=val}\, \mathbf{P}_1$
Project: $\mathbf{R}_T = \pi_{att} \mathbf{R}_1$	$C_T = \pi_{att}\, C_1$	$\mathbf{P}_T = \pi_{att}\, \mathbf{P}_1$
Cart. Prod: $\mathbf{R}_T = \mathbf{R}_1 \times \mathbf{R}_2$	$C_T = C_1 \times C_2$	$\mathbf{P}_T = ((\mathbf{genl}(C_1) \times \mathbf{P}_1) \cup (\mathbf{genl}(C_1) \times \mathbf{P}_2) \cup (\mathbf{genl}(C_2) \times \mathbf{P}_1) \cup (\mathbf{genl}(C_2) \times \mathbf{P}_2)) - \mathbf{genl}(C_T)$

Summary

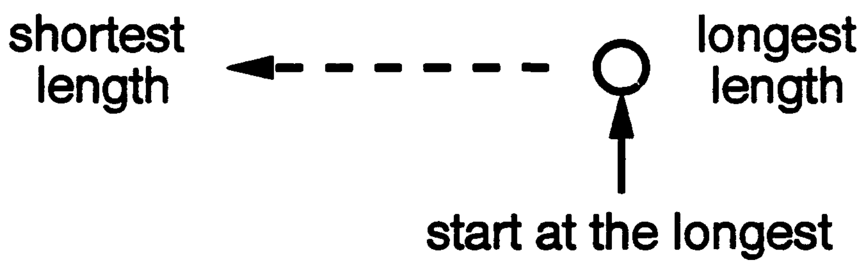
- Object–oriented query processor:
  - identifies initial approximation
  - avoids operations on possible tuples

- Future work:
  - implement object–oriented query processor
    examine overhead due to class hierarchy
  - class lattice with multiple inheritance
  - single–valued queries
  - general approximate model using distance

Single-Valued Query Approximations

Min-Max Queries

"What is the longest runway at the Rome airport?"



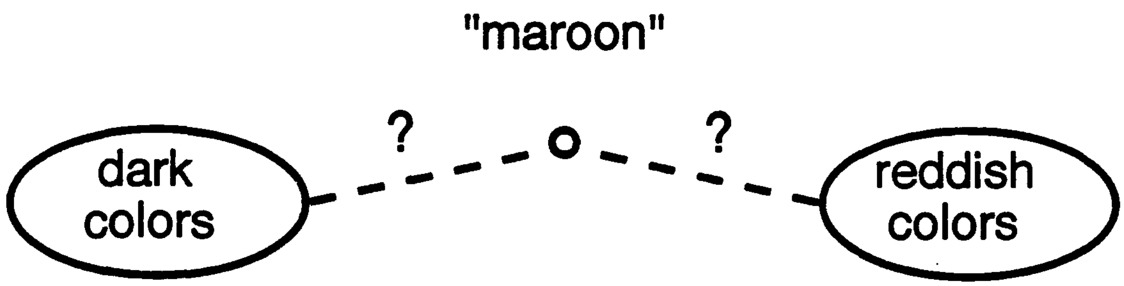
Value-of Queries

"What is the direction of runway #20?"

Semantic Support

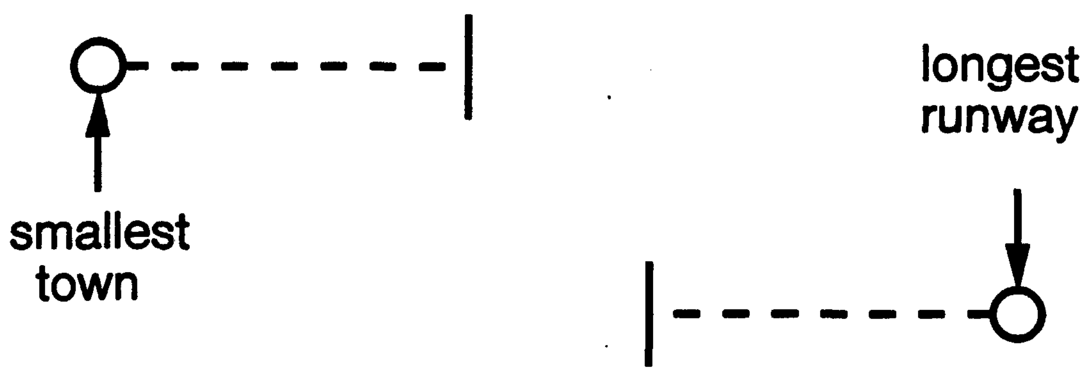
Distance function:

determines which class is closest



Multi-Dimensional (Multiple Attributes):

"What is the airport with the longest runway in the smallest town?"



Multiple Classes - different strategies

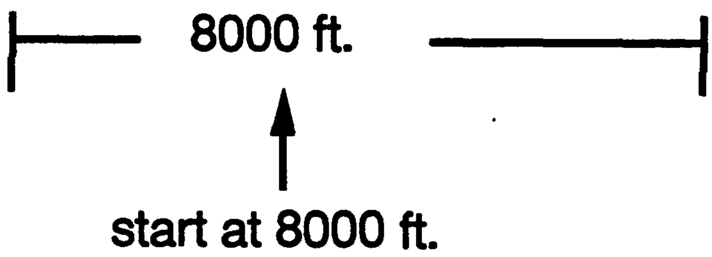
Yes/No Queries

Approximate Answers

No - 30%, No - 80%

Comparison Queries

"Is the runway at Paris at least 8000 feet?"



Process values 8000, 9000, 10000 , ...

or 7000, 6000, 5000, ...

Existential Queries

"Is there an airport in Vienna, IL ?"

- Work in progress
  - — Monotone query processing when the database contains complete information and queries are precise.
- Future work
  - — Monotone query processing in the presence of imprecise or incomplete information
  - — Imprecise updates

APPENDIX L

Temporal Reasoning for Real Applications

Mark Boddy
Honeywell SRC

Basic Temporal Inference

- Retrieving explicitly recorded facts
- Retrieving information about duration
- Conjunctive queries
- Default reasoning (persistence)
- Causal reasoning
- Temporal reason maintenance

Queries and Assertions

Assertions

Propositions: (occurs (status plane41 operational)

Temporal relations: (pt< (begin token1) (end token43))

Queries

Temporal relations: (pt< point1 point2)

"True throughout": (tt p1 p2 (and (inst ?l light) (status ?l on)))

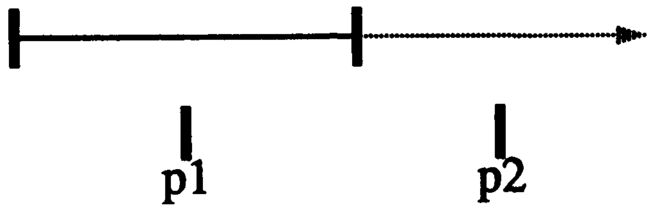
Temporal Reason Maintenance

```
(for-first-answer
  (fetch '(and (occurs (manufacture job14) ?tok)
               (tt (begin ?tok) (end ?tok) (and (installed bit41 machine3)
  (assert '(valid plan1 (task ?tok))))
```

Time Tokens

True throughout:

(on l1)



(tt p1 p2 (on l1)) => T

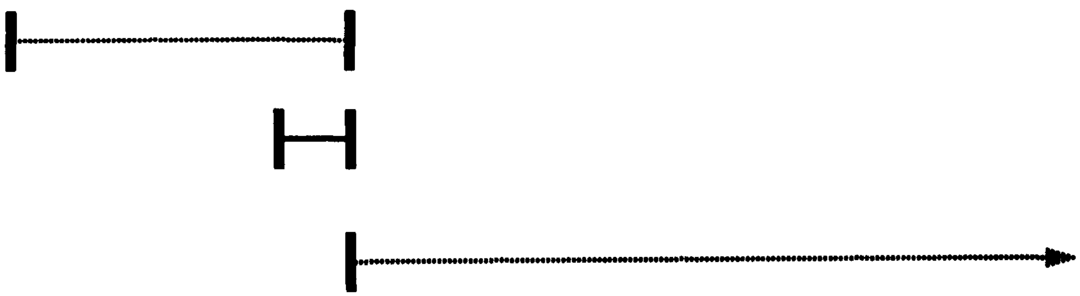
Projection and clipping:

(project (on ?light) (push switch ?light) (off ?light))

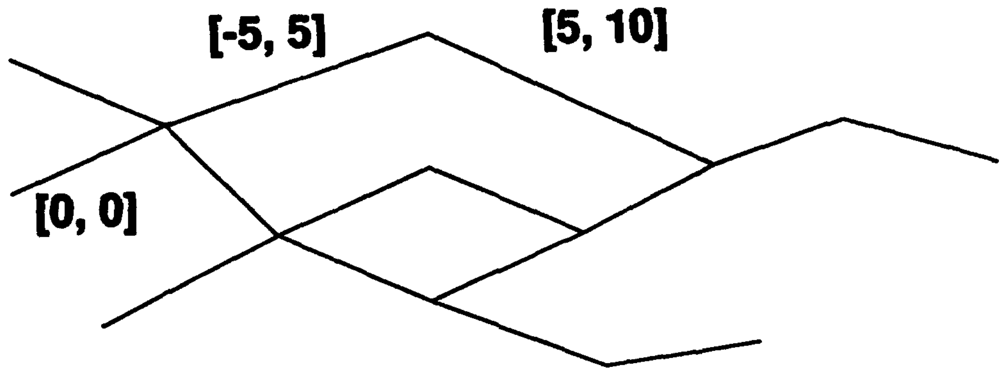
(on l1)

(push switch l1)

(off l1)



Temporal Constraint Graph



Used for specifying, deriving, and monitoring temporal relations.

Basic functions:

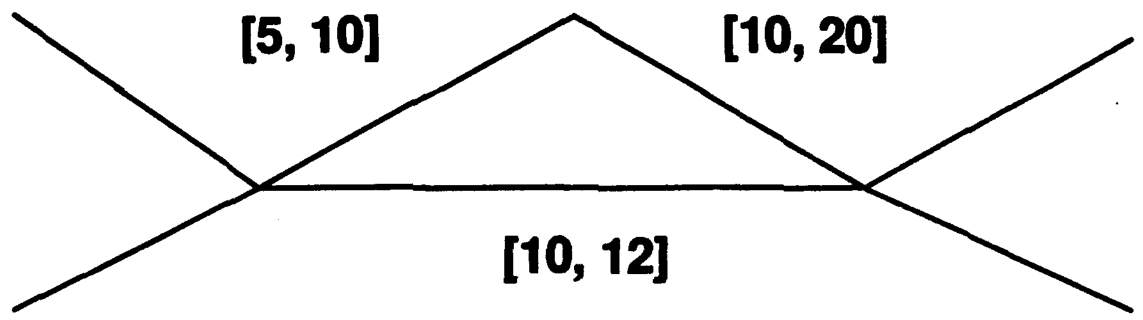
- **Adding points and constraints**
- **Search**
- **Temporal conditions and caching**

Maintaining “Consistency”

We use an ordered set of invariants:

- Path consistency
- Temporal relations and cached distances
- Projection and clipping
- Protections

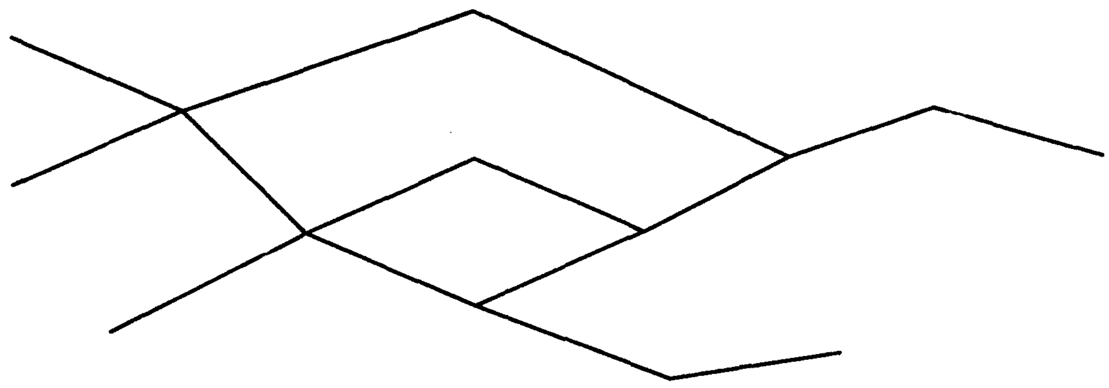
Path Consistency



Detected during constraint propagation and search.

TMM provides a range of choices on system or user's response.

Temporal Conditions and Caching



Monitor bounds on distance between two points.

Initially established by search, updated by propagation.

Re-established using search if a constraint on current path is deleted.

Projection and Clipping

- All the persistences that should be clipped, are clipped.
- All the projection rules that should fire, do.

Protections

A protection is a monitor on the continued truth of a query (either true-throughout or a temporal relation).

Temporal relations are handled by tconditS.

True-throughout is harder; determining whether query is satisfied at any point as the database changes could mean repeatedly invoking the original query.

Currently proposed protection types for true-throughout:

- Try to re-establish temporal relations for the same token.
- Try to find a new token, at the time protection fails.
- Check on whether query can be satisfied, every time database changes.

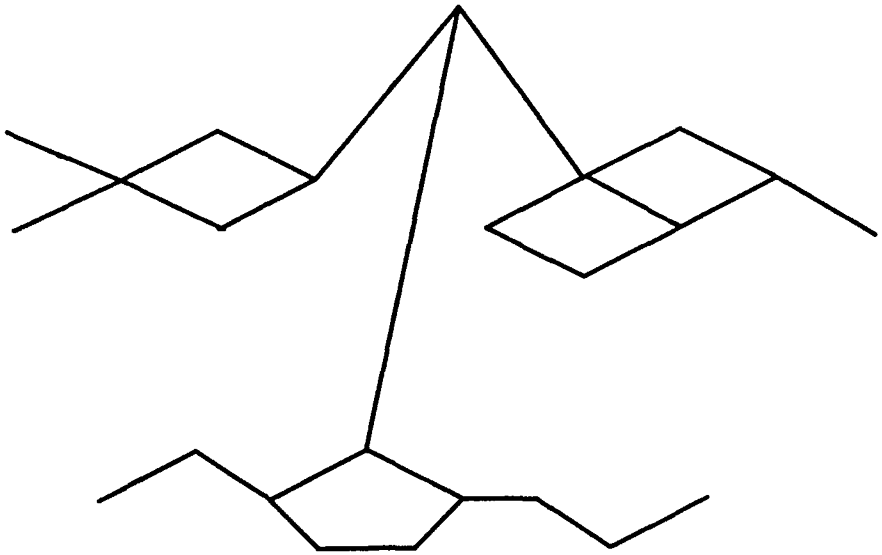
Large databases are a problem

- Tractable temporal reasoning is hard.
  - Projection is NP- complete for partial orders.
- Tractable temporal reasoning is insufficient.
  - Current applications may involve 10K tokens.

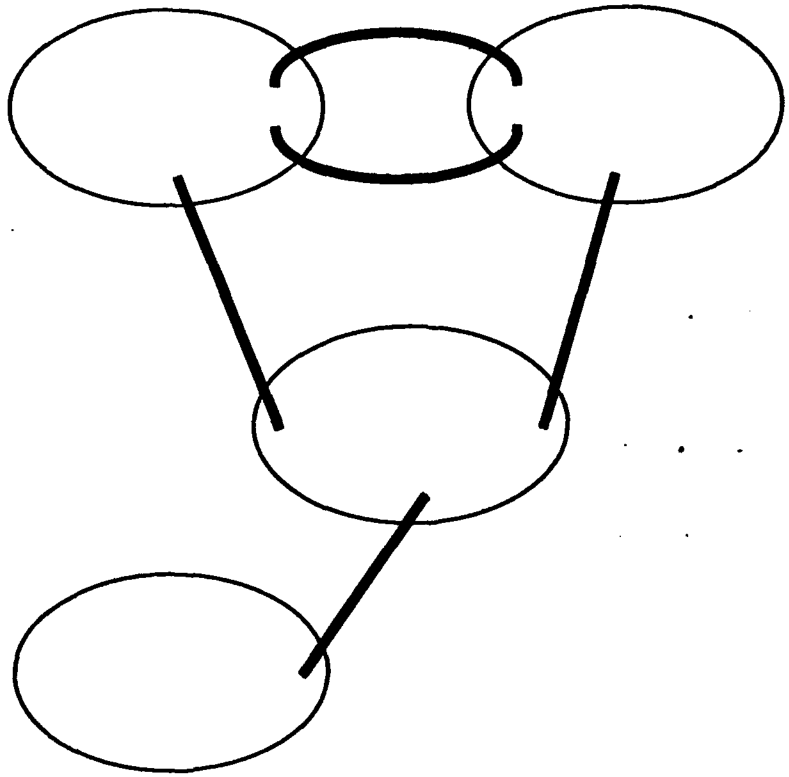
Graph Decomposition

Offrecs

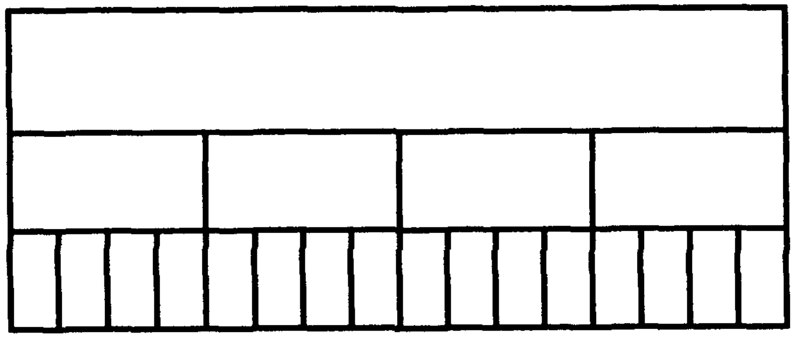
global
reference



Graph partitioning



Temporal Indexing



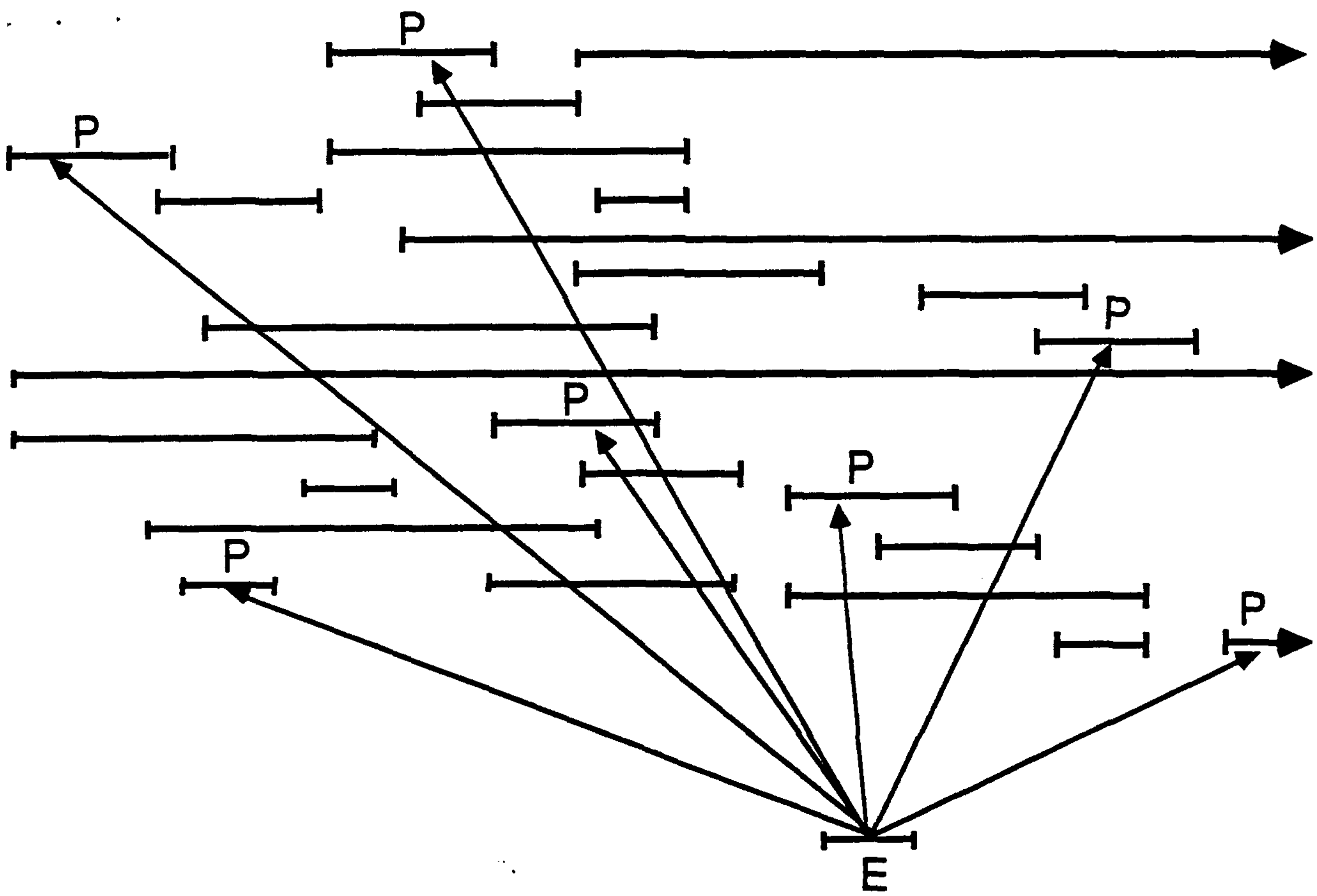
Token endpoints are sorted into buckets.

Cost of projection, clipping, and true-througout is independent of the size of the database.

Summary

- Large-scale temporal reasoning problems require more than "tractable" inference.
- The structure of temporal information provides some hope of being able to decouple the cost of inference from the size of the database.
- The difficulty of maintaining consistency and inferential closure can have a significant impact on the efficiency and expressivity of a temporal reasoner.

Token Retrieval in Large Data Bases



APPENDIX M

Programming Parallel Architectures for Knowledge-Based Applications

Christine A. Montgomery, Ph.D.
Robert E. Stumberger

Language Systems, Inc.

Jean-Luc Gaudiot, Ph.D.

University of Southern California

Intelligent Information System Workshop

Rome Laboratory

October 23, 1991

This material is based upon work supported in part by the U.S. Air Force Rome Laboratory/COES under a Phase I SBIR contract No. F30602-90-C-0053.

Language Systems, Inc.

Highly Programmable Architectures (Phase 1) Objectives

1. Assess status of data-flow research, U.S. and foreign
2. Test data-flow architecture for non-numeric application
3. Define strategy for extrapolation to larger non-numeric applications

Highly Programmable Architectures (Phase 1) Results

1. Micro instruction sets appear sufficient for implementing some knowledge-based applications
2. Macro data-flow actors should provide higher efficiency than micro actors
3. High-level data-flow languages appear to have expressive power required for natural language applications
4. Data-flow principles can deliver runtime parallelism to natural language applications with little compiler effort.
5. Further study should be conducted on larger examples of natural language and knowledge-based applications.

Programming Parallel Architectures (Phase 2) Objectives

1. Specify requirements, design, and implement application-development environment for multiprocessor architectures, using data-flow principles
2. Test environment with selected natural language and knowledge-based applications

Language Systems, Inc.

Simple Grammar for "The parsing algorithm is specified"

NP VP	->	S
Det NG	->	NP
(Adj) N	->	NG
Art	->	Det
Aux VB	->	VP

Grammar Symbols

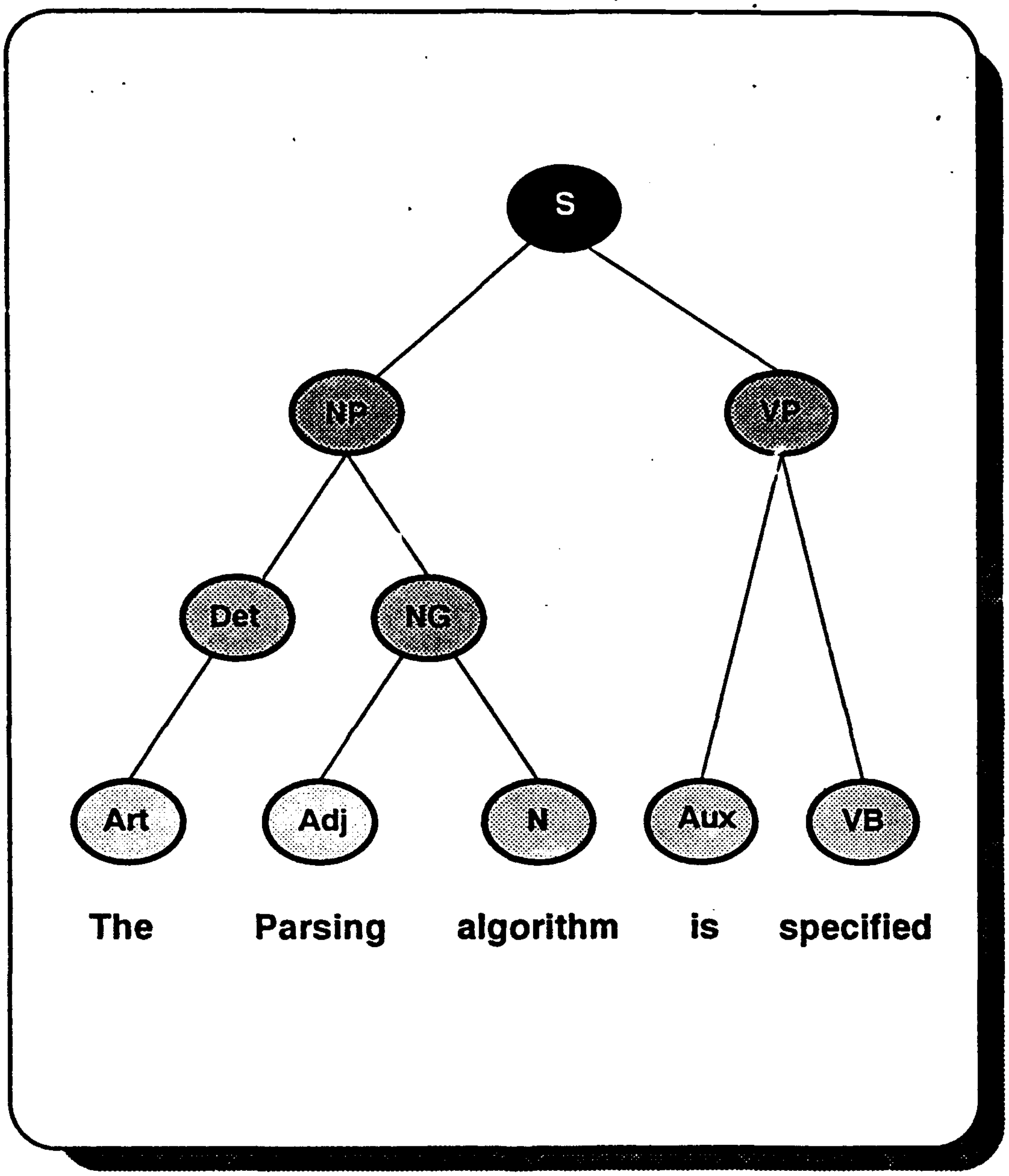
Adj	adjective
Art	article
Aux	auxiliary
Det	determiner
N	noun
NG	noun group
NP	noun phrase
S	sentence
VB	verb
VP	verb phrase

Chart of parses for "The parsing algorithm is specified"

Row Index	(1)	(2)	(3)	(4)	(5)
1.	1	1	Det	1	0
2.	1	2	Adj	2	0
3.	1	2	VB	3	0
4.	1	2	N	4	0
5.	1	3	N	5	0
6.	1	4	Aux	6	0
7.	1	5	Adj	7	0
8.	1	5	VB	8	0
9.	2	1	NG	1	4
10.	2	2	NG	2	5
11.	2	4	VP	6	8
12.	3	1	NP	1	10
13.	5	1	S	12	11

Resulting phrases (for each index)

1. The
2. parsing
3. parsing
4. parsing
5. algorithm
6. is
7. specified
8. specified
9. the parsing
10. parsing algorithm
11. is specified
12. the parsing algorithm
13. the parsing algorithm is specified



Parse tree for "The parsing algorithm is specified"

The Problem

- Reliable interconnection
  (static & dynamic failures)

- Proper synchronization of many
  Processing Units

The Four Challenges

Computer architecture is technology driven.

1. Programmability
2. Communication network design
3. Reliability
4. Performance evaluation & benchmarking

Programmability

- Large number of processing Elements
- Insure deterministic & safe termination of the user program

  "Safe" is defined as the ability to execute a program on a multi-processor & produce the same results as would a single processor architecture

- No centralized control, no shared memory system (inherent bottleneck)
- Communication by "message passing"
- Asynchronous execution

Must find: efficient, architecture-transparent programming paradigms for large-scale multiprocessors

Reliable Operation

- Increasing machine size increases whole system failure rate (for a given technology)
- Identical & autonomous Processing Elements can be used for redundancy as well as for improved performance
- Degraded operation
- Program & data reallocation
- Reliability estimation (execution of a given program)

Performance Evaluation And Benchmarking

- **Early computers: simple benchmarks (gate speed)**
- **Next generations: more complex tests (instruction rate)**

  PROBLEM: Meaningless for very complex architectures (e.g., pipelines)
- **Livermore Loops (CRAY's)**

The history of computing has so far seen an attempt not at doing the same things faster but at doing in similar time larger and larger problems.

Some Possible Solutions

1. New models of computation (data-flow, neural nets)
2. New technologies of inter-connection
3. Fault-tolerant design
4. Larger benchmarks

Alternative Models of Computation

Basic Principles of Data-flow Execution:

- No reliance upon state (central memory system) or central controller (program counter) at the language level
- Asynchronous execution (arrival of the arguments)
- Functional execution (no side-effects)

Research Issues:

- Structure handling
- Program partitioning & allocation
- High-level language translation

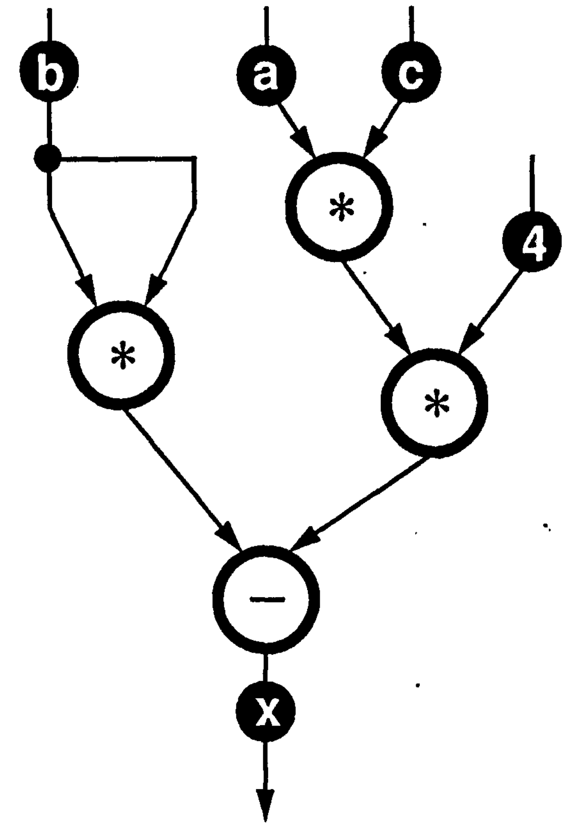
Artificial Neural Networks:

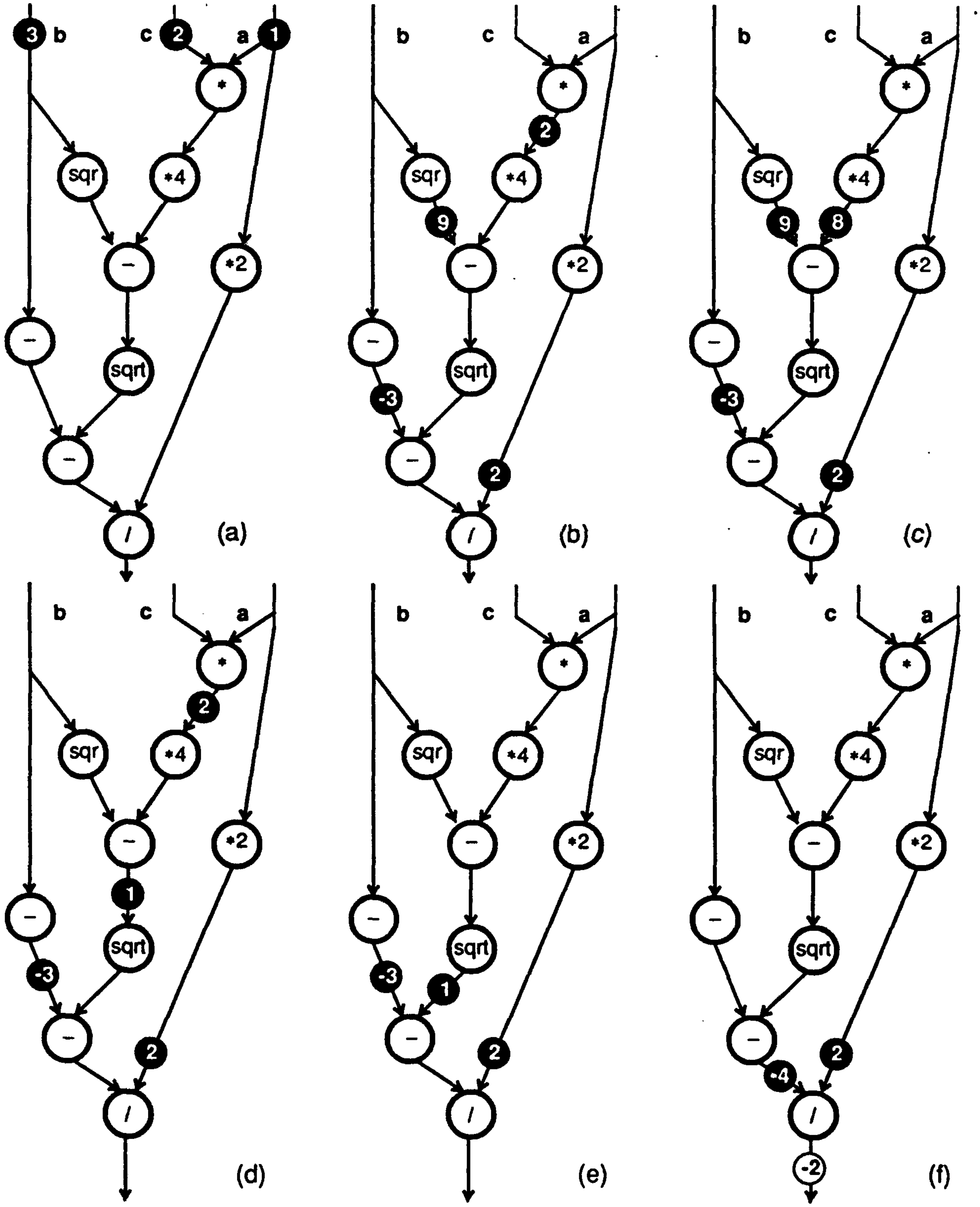
- "Non-algorithmic" computing structures
- Self-organizing features
- Robustness
- Simple Individual Evaluation Elements

Data-flow Principles of Execution

- **High programmability:** Implicit synchronization
- **Operand availability:** Execute whenever data are available.
- **No side effect:** Purely functional

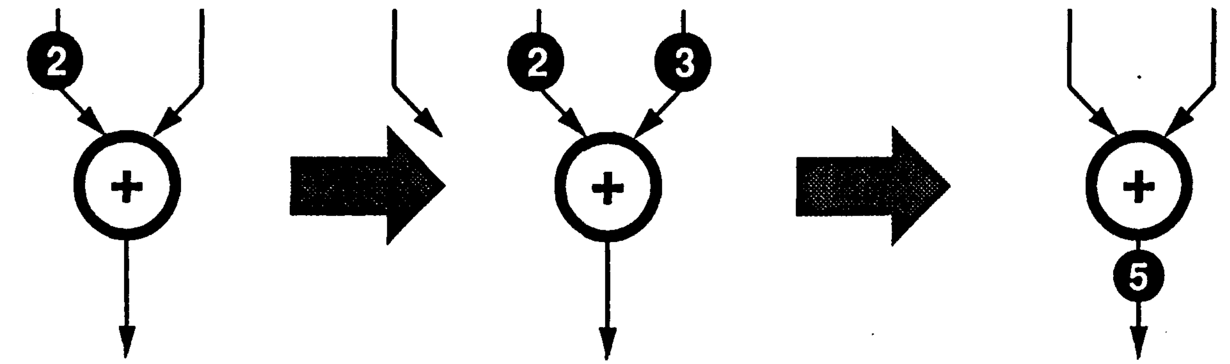
Data-flow graph for
$x = b^2 - 4ac$



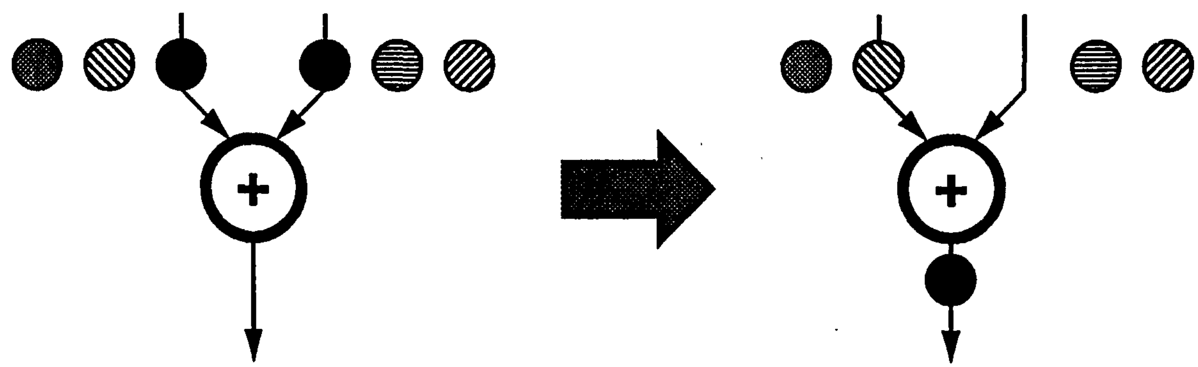

b
c
a
*
sqr
*4
*2
sqrt
/
(a)
(b)
(c)
(d)
(e)
(f)

Interpretation Models

Static data-flow: a single data on an arc allowed



Dynamic data-flow: multiple data on an arc allowed



Processing on Macro Data-flow

What is a Macro?

1. Macro Actor: A collection of micro actors (instructions)
2. Macro Token: A collection of primitive data tokens

Why use Macros for AI processing?

1. Utilize the parallelism in medium grain processing.
2. Utilize the locality in programs, thereby giving a structured way of writing rules.
3. Preserve the fundamental AI list processing.
4. Substantially reduce the communication overhead in multiprocessor environment.

N L Syntactic Processing

Basic Algorithm:

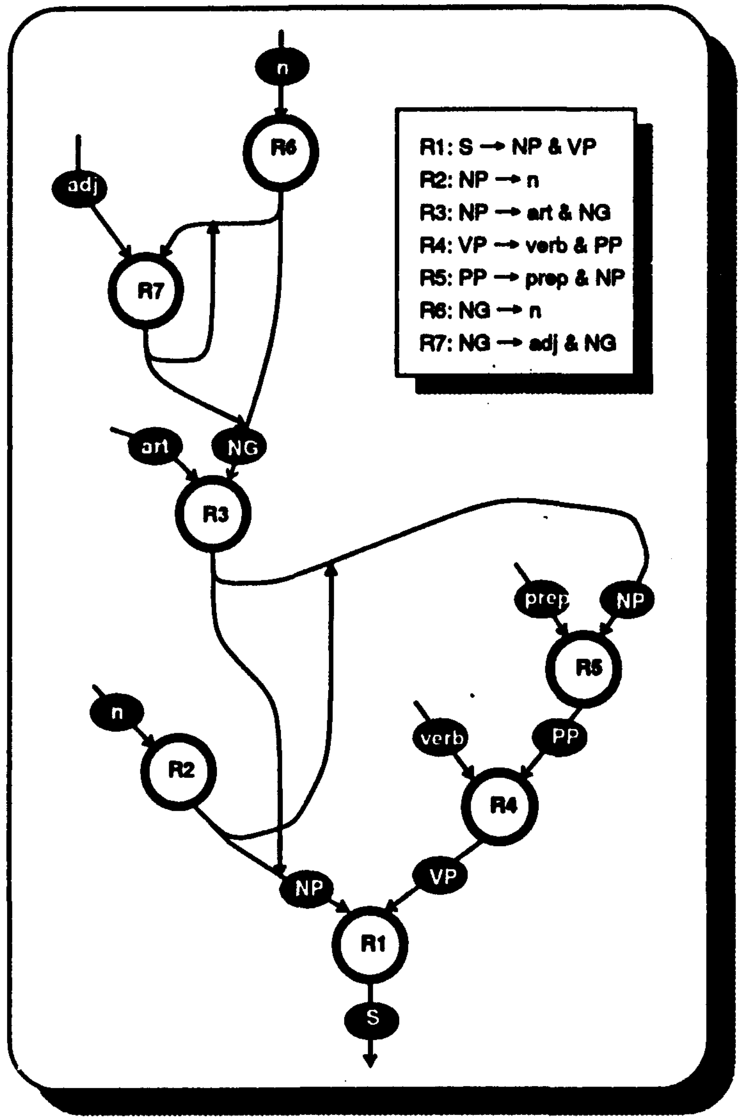
- Maximum Parallelism
- May include high overhead ("useless work")
- Try out all possibilities

Example:

- R1: S → NP & VP
- R2: NP → n
- R3: NP → art & NG
- R4: VP → verb & PP
- R5: PP → prep & NP
- R6: NG → n
- R7: NG → adj & NG

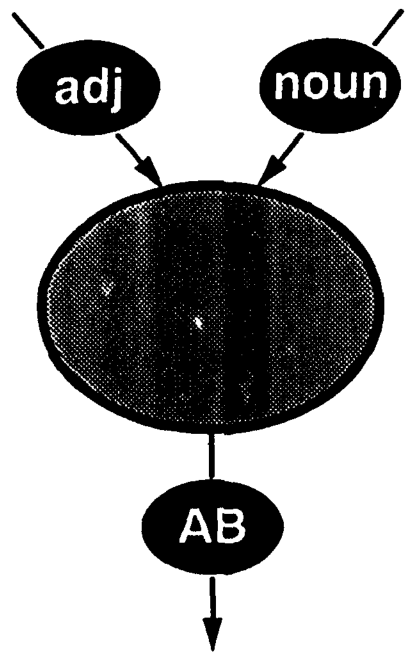
Data-flow Grammar Graph

- All actors are identical and execute the same code.
- Differentiation by the connections between actors.
- Each arc carries a symbol.
- Each actor corresponds to a rule.



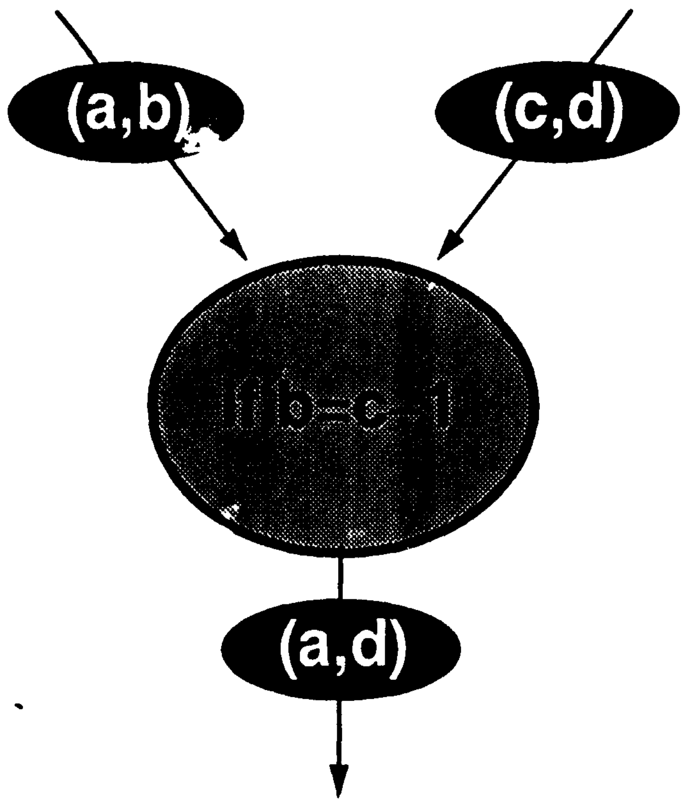
Data-flow Actor

R_{xy}: AB → adj & noun



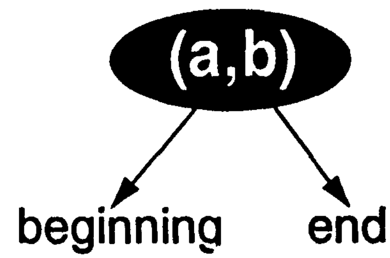
	"The	blue	airplane	flew	over	Kampala"
	art	*adj*	*noun*	*verb*	*prep*	*noun*
Position	1	2	3	4	5	6

Execution of an Actor

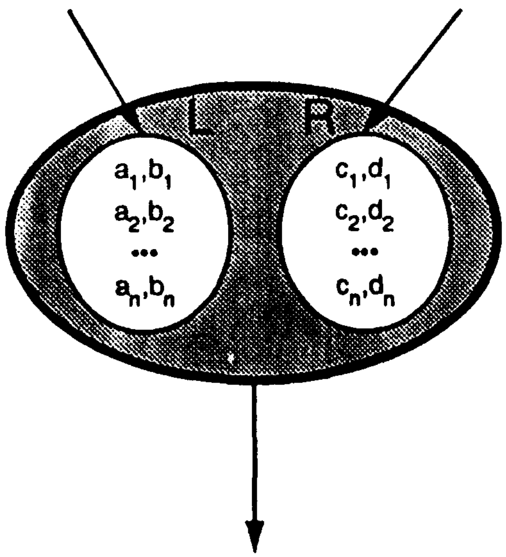


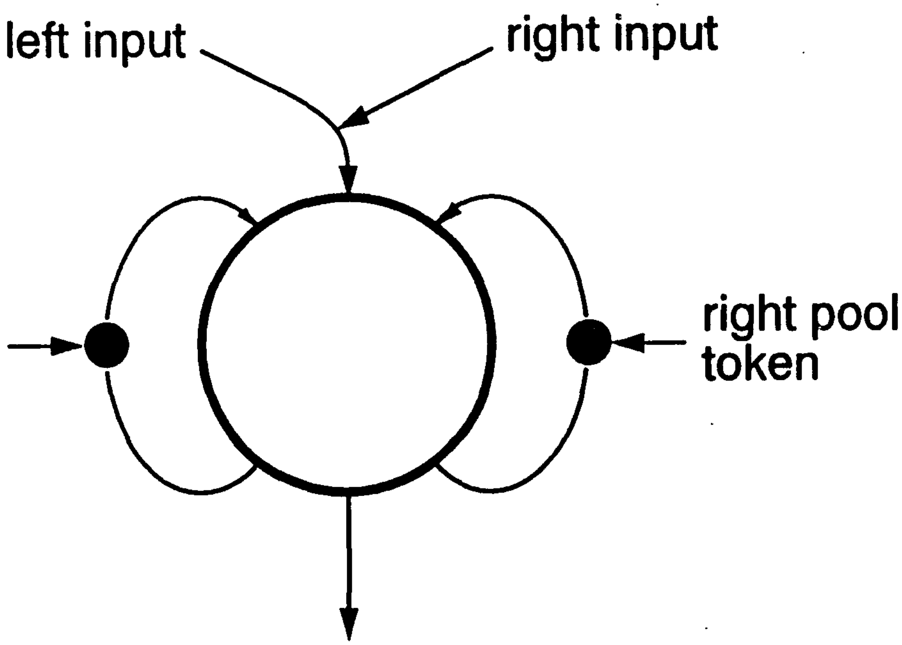
There is a match if b=c-1, then we produce a token.

Token:
- "left" or "right" input
- beginning position
- end position

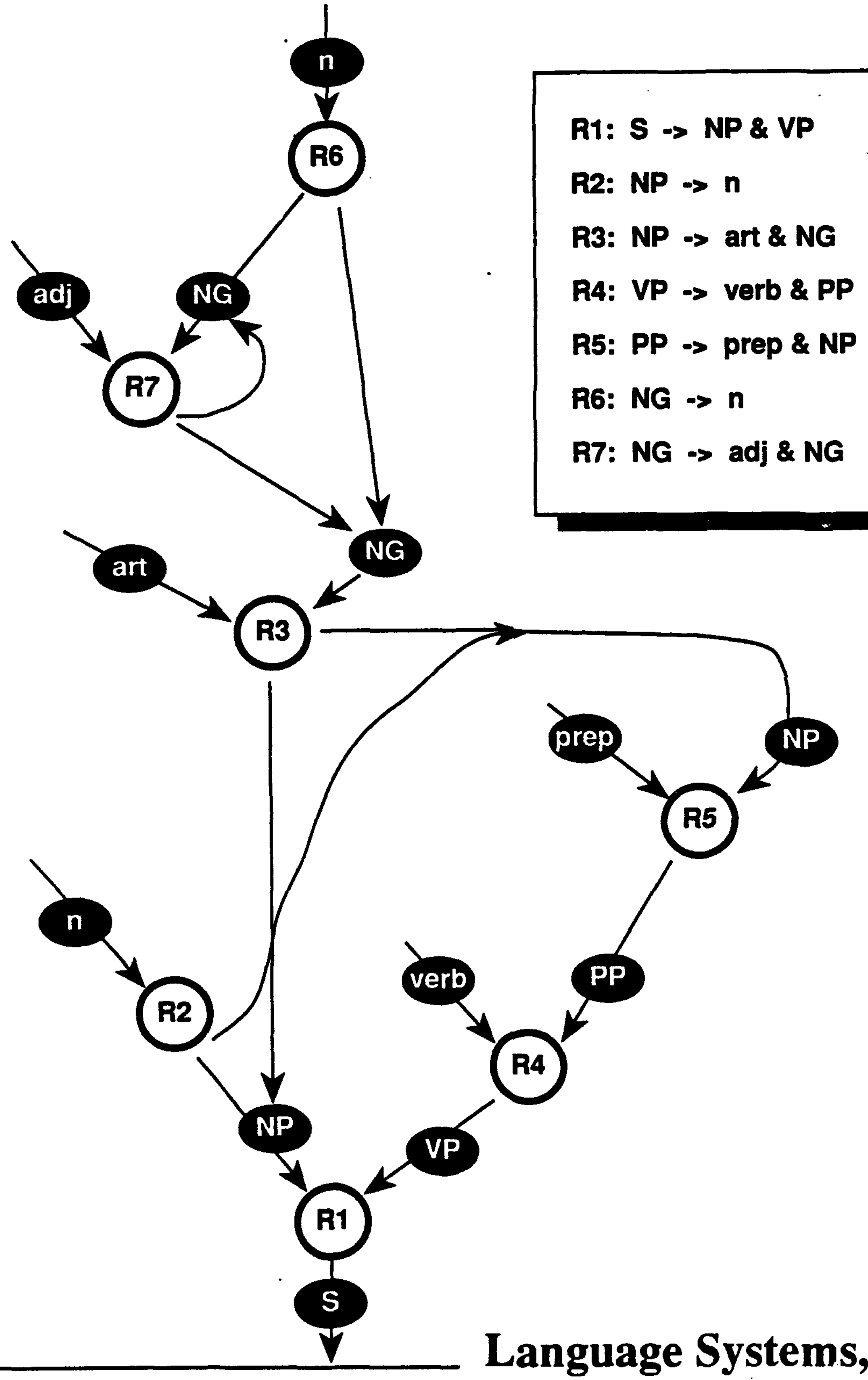


Macro Actors





- **Input token is matched against all the tokens in the other pool.**
- **If match, produce a new token.**
- **If no match, store in the pool.**



R1: S -> NP & VP

R2: NP -> n

R3: NP -> art & NG

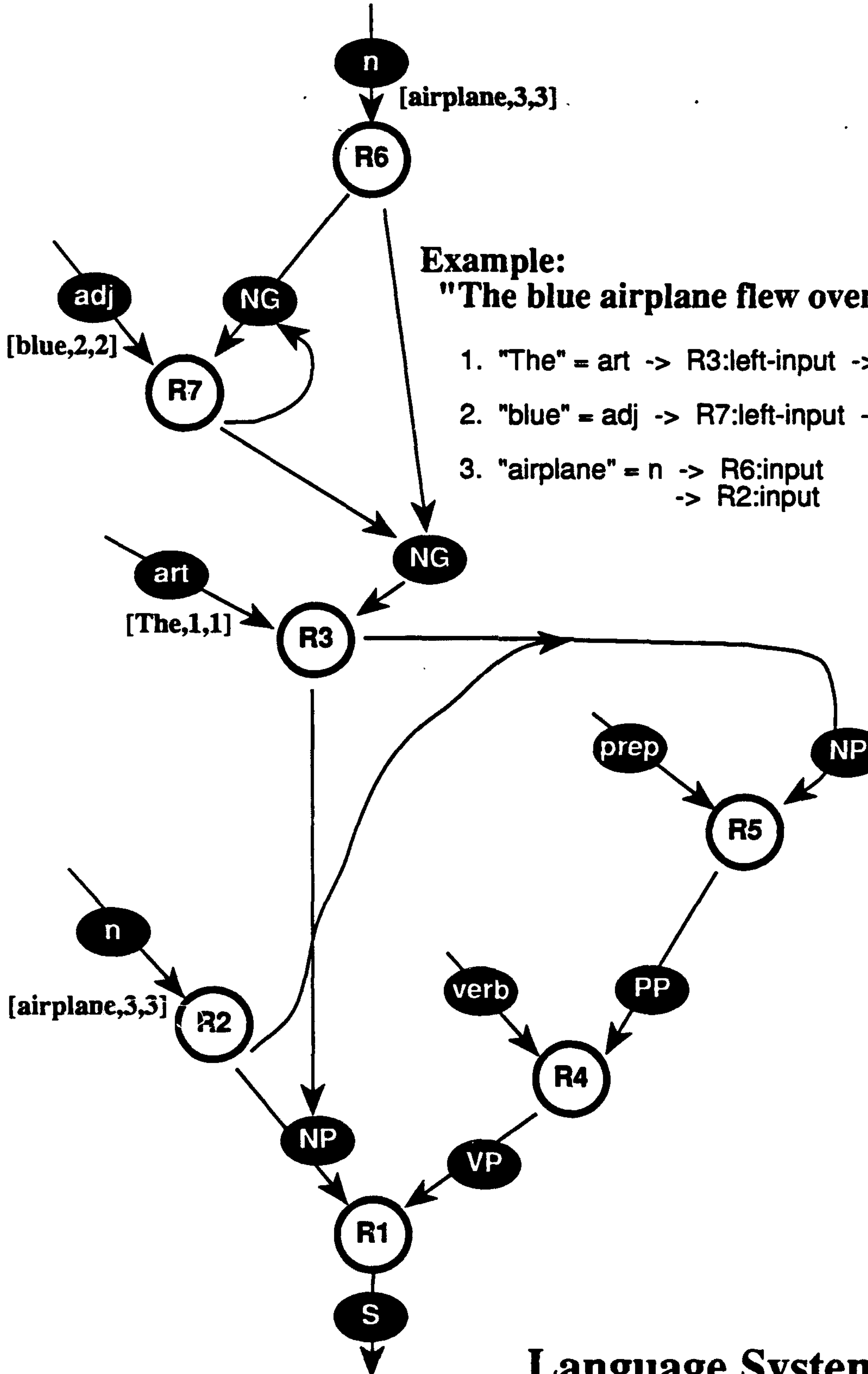
R4: VP -> verb & PP

R5: PP -> prep & NP

R6: NG -> n

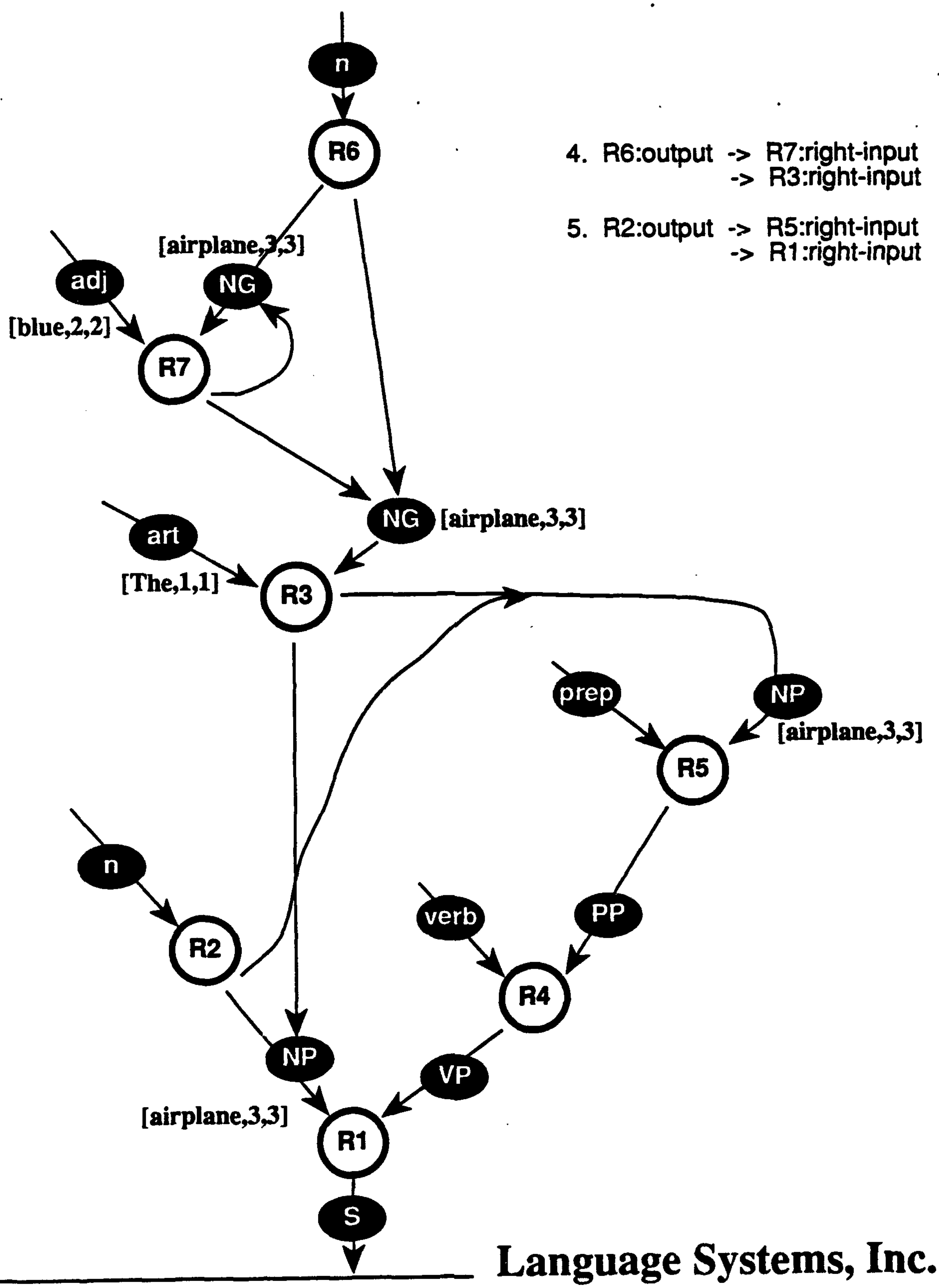
R7: NG -> adj & NG

Language Systems, Inc.



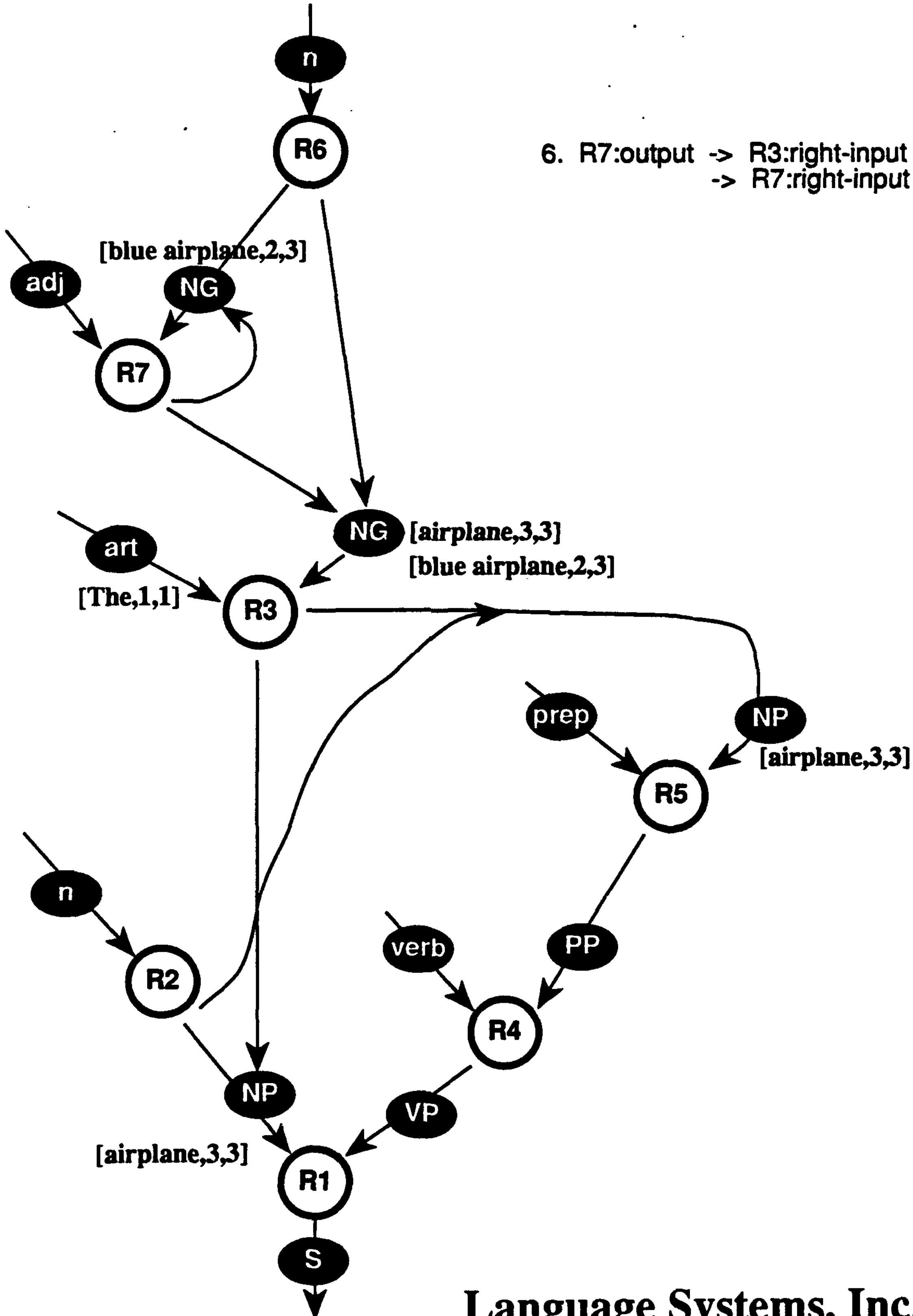
Example:
"The blue airplane flew over Kampala"

1. "The" = art -> R3:left-input -> R3:left-pool
2. "blue" = adj -> R7:left-input -> R7:left-pool
3. "airplane" = n -> R6:input
   -> R2:input

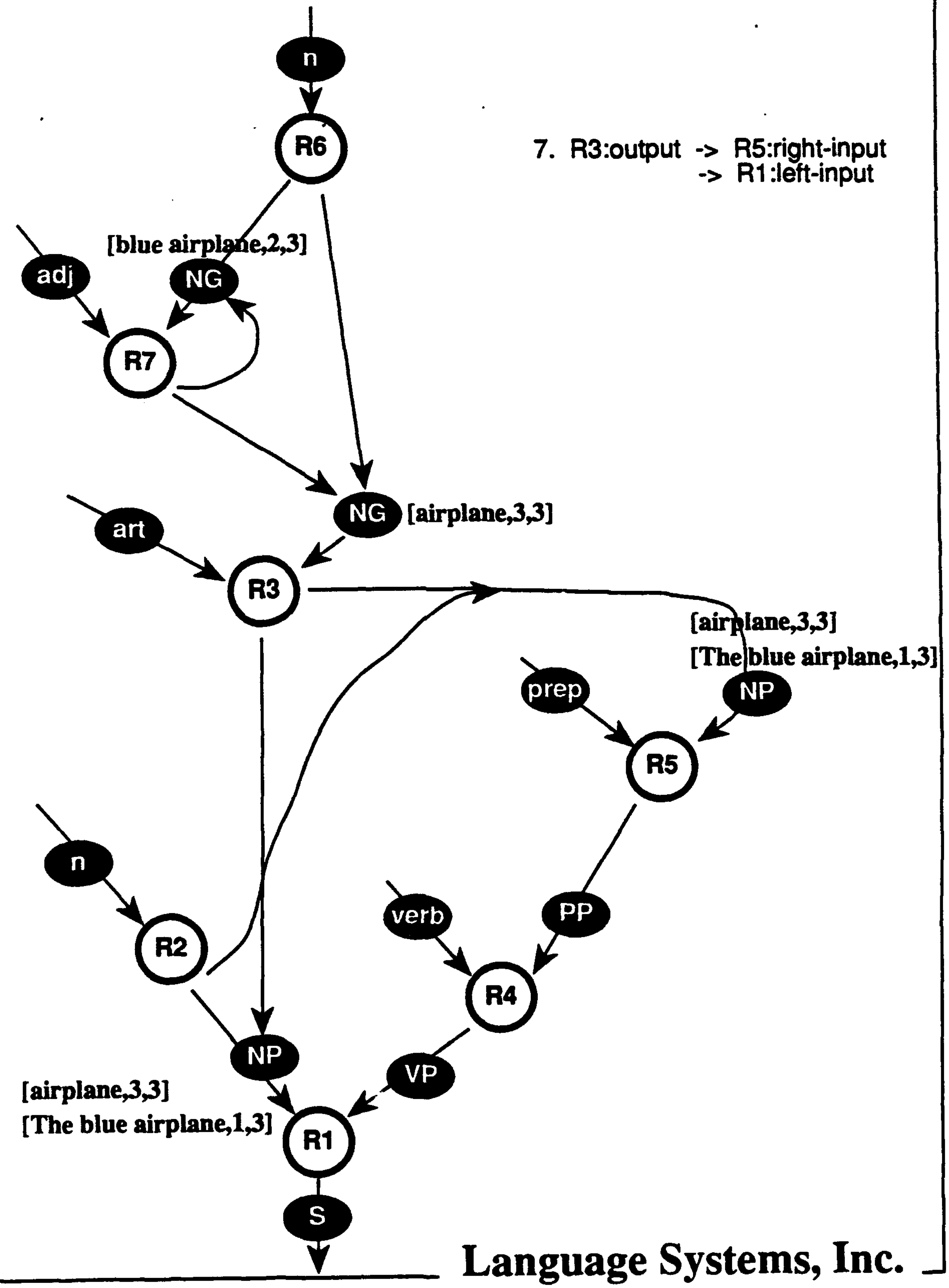


Language Systems, Inc.

6. R7:output -> R3:right-input
-> R7:right-input



Language Systems, Inc.

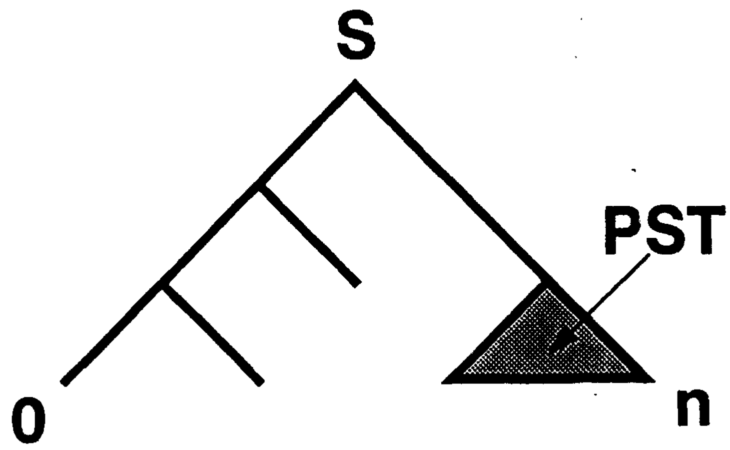


Language Systems, Inc.

Implementation in a HLL

Partial Syntactic Tree (PST) =
Derivation of a subset of the input sentence

- Symbol
- Beginning of PST
- End of PST + 1



Algorithm:

```
initialize M to contain exactly the initial nodes
repeat 'several' times
   begin
   for all nodes x,y,z in parallel do
      if (x,y belong to M and x,y |-- z) then add z to M
   check if (S,0,n) belongs to M
end
```

Implications (1)

1. Conventional (micro) instruction sets appear sufficient for implementation of some analytical algorithms encountered in NL/KB systems.

2. Higher efficiency would result from the introduction of special (macro) actors with greater power than micro data-flow actors.

3. High-level data-flow languages such as SISAL appear to have sufficient expressive power for natural language applications.

Implications (2)

4. Parallelism available in natural language applications can be delivered at runtime by the data-flow principles of execution with little compiler effort.

5. Parallel algorithms for natural language and knowledge base processing need to be studied and developed.

6. Higher-level (i.e., non-numeric) rules may be expressed in the conventional (micro)-instruction sets of the data-flow model. (This has been tested on a small phrase-structure grammar for English).

Directions for Future Research (1)

1. For natural language syntactic processing, parallel algorithms should be investigated and developed. Experimentation with syntactic analysis should be broadened to test the efficiency of data-flow implementations for multiply ambiguous inputs.

2. For knowledge-based systems, a number of possible directions could be explored. Can inheritance hierarchies, such as is-a and part-of, be efficiently embedded in a data-flow model? Can backward-chaining and forward-chaining rules be efficiently encoded as data-flow graphs?

Directions for Future Research (2)

2. (continued)

   Can simple inferencing (or deduction) be performed using these graphs? Can search-pruning strategies, such as spreading activation, be applied to the data-flow graphs (probably at the micro-instruction level) in order to limit the potential combinatorial explosion caused by complex inference tasks?

3. A high-level programming environment would be exceptionally useful in the design, implementation and test of data-flow graphs for natural language processing and knowledge-based systems.

MISSION

OF

ROME LABORATORY

Rome Laboratory plans and executes an interdisciplinary program in research, development, test, and technology transition in support of Air Force Command, Control, Communications and Intelligence (C^3I) activities for all Air Force platforms. It also executes selected acquisition programs in several areas of expertise. Technical and engineering support within areas of competence is provided to ESD Program Offices (POs) and other ESD elements to perform effective acquisition of C^3I systems. In addition, Rome Laboratory's technology supports other AFSC Product Divisions, the Air Force user community, and other DOD and non-DOD agencies. Rome Laboratory maintains technical competence and research programs in areas including, but not limited to, communications, command and control, battle management, intelligence information processing, computational sciences and software producibility, wide area surveillance/sensors, signal processing, solid state sciences, photonics, electromagnetic technology, superconductivity, and electronic reliability/maintainability and testability.