RL-TR-94-139
Final Technical Report
August 1994

CASE-BASED REASONING FOR DEPLOYMENT TRANSPORT PLANNING

The University of Chicago

Sponsored by
Advanced Research Projects Agency
ARPA Order No. 7704

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Materiel Command
Griffiss Air Force Base, New York

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

RL-TR-94-139 has been reviewed and is approved for publication.

APPROVED: Northrup Fowler III

NORTHRUP FOWLER III, Ph.D.
Project Engineer

FOR THE COMMANDER: John A. Graniero

JOHN A. GRANIERO
Chief Scientist
Command, Control, and Communications Directorate

If your address has changed or if you wish to be removed from the Rome Laboratory mailing list, or if the addressee is no longer employed by your organization, please notify RL (C3C) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

CASE-BASED REASONING FOR DEPLOYMENT TRANSPORTATION PLANNING

Kristian J. Hammond

Contractor: The University of Chicago
Contract Number: F30602-91-C-0028
Effective Date of Contract: 4 February 1991
Contract Expiration Date: 3 February 1994
Short Title of Work: Case-Based Reasoning for Deployment Transportation Planning
Program Code Number: 1E20
Period of Work Covered: Feb 91 - Feb 94

Principal Investigator: Dr. Kristian J. Hammond
Phone: (312) 702-1571

RL Project Engineer: Northrup Fowler III
Phone: (315) 330-3011

This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by Northrup Fowler III, RL (C3C), 525 Brooks Road, Griffiss AFB NY 13441-4505 under Contract F30602-91-C-0028.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 1994	Final Feb 91 - Feb 94

4. TITLE AND SUBTITLE

CASE-BASED REASONING FOR DEPLOYMENT TRANSPORT PLANNING

5. FUNDING NUMBERS

C - F30602-91-C-0028
PE - 62301E & 62702F
PR - G704
TA - 00
WU - 03

6. AUTHOR(S)

Kristian J. Hammond

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

The University of Chicago
Dept. of Computer Science, Artificial Intelligence Lab
1100 East 58th Street
Chicago IL 60673

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington VA 22203-1714

Rome Laboratory (C3C)
525 Brooks Road
Griffiss AFB NY 13441-4505

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

RL-TR-94-139

11. SUPPLEMENTARY NOTES

Rome Laboratory Project Engineer: Northrup Fowler (315) 330-3011

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

In the report we describe work done in the application of case-based reasoning (CBR) techniques to large scale planning problems. We report on systems that demonstrate the role of CBR in four areas of planning:

CBR as Gatekeeper. The core idea of case-based reasoning is the re-use of plans by relating past experience to present problems. This makes CBR ideal as the "gate-keeper" between users and specialized problem solvers in two ways: the retrieval of past solutions, and the retrieval of past solution strategies.

CBR as Guide. With the complex constellation of problem solvers and information sources available in large-scale planning, guiding users among available resources becomes of crucial importance. By retrieving solution strategies, case-based reasoning can provide structure even in the absence of past solutions.

CBR as editor. Specialized problem solvers make use of specialized problem representations, incomprehensible to those unfamiliar with the technology. We demonstrate that case-based reasoning can serve as an "editor" of this specialized "jargon," presenting results and options in terms that are clear and useful.

CBR as Opportunist. It is an axiom of large-scale planning that solutions must always be partial and approximate, to be further developed over time. (See Reverse)

14. SUBJECT TERMS

Case-based planning, Artificial intelligence, Integrated systems

15. NUMBER OF PAGES

96

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	UNLIMITED

Contents

List of Figures

Abstract

In the report we describe work done in the application of case-based reasoning (CBR) techniques to large scale planning problems. Our work has resulted in a set of systems that demonstrate the role of CBR in four areas of planning:

- *CBR as Gatekeeper.* The core idea of case-based reasoning is the re-use of plans by relating past experience to present problems. This makes CBR ideal as the "gatekeeper" between users and specialized problem solvers in two ways: the retrieval of past *solutions*, and the retrieval of past *solution strategies*.

  This is demonstrated in the ROENTGEN system.

- *CBR as Guide.* With the complex constellation of problem solvers and information sources available in large-scale planning, guiding users among available resources becomes of crucial importance. By retrieving *solution strategies*, case-based reasoning can provide structure even in the absence of acceptable past solutions.

  This is demonstrated in the IOPS and ROENTGEN systems.

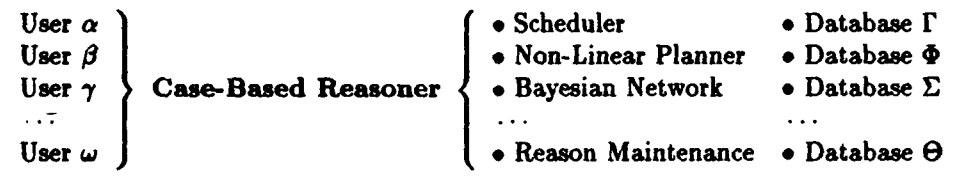
- *CBR as Editor.* Specialized problem solvers make use of specialized problem representations, incomprehensible to those unfamiliar with the technology. With its traditional focus on higher-level abstractions, case-based reasoning can serve as an "editor" of this specialized "jargon," presenting results and options in terms that are clear and useful to users.

  This is demonstrated in the DMAP system.

- *CBR as Opportunist.* It is an axiom of large-scale planning that solutions must always be partial and approximate, to be further developed over time. Case-based reasoning's focus on anticipation, expectation, and repair make it well-suited to organizing the incomplete work of specialized problem solvers and recognizing when new data is relevant to a particular computation.

  This is demonstrated in the RUNNER and SHOPPER systems.

The fundamental conclusion we have arrived at as a result of our research on deployment transport planning is that Case-Based Reasoning (CBR) offers a viable *lingua franca* for large-scale planning.

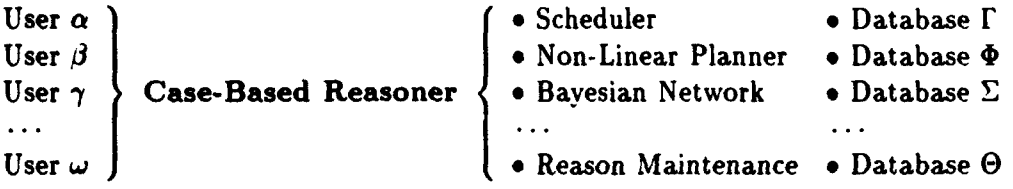


Occupying a place between groups of users and the many specialized planning and decision-making systems and databases, the case-based reasoner provides the best interface for coordinating large-scale planning efforts. Our final report is organized around this central idea.

1 Introduction

Large-scale planning must employ many disparate planning and decision-making systems, operating over disparate databases to solve many disparate problems. The problem of coordinating so many systems and so many users over so many different sites is immense.

The fundamental conclusion we have arrived at as a result of our research on deployment transport planning is that Case-Based Reasoning (CBR) offers a viable *lingua franca* for large-scale planning.



Occupying a place between groups of users and the many specialized planning and decision-making systems and databases, the case-based reasoner provides the best interface for coordinating large-scale planning efforts. Our final report is organized around this central idea.

There are a number of broad aspects unique to case-based reasoning that make it a desirable *lingua franca* for large-scale planning:

- *CBR as Gatekeeper.* The core idea of case-based reasoning is the re-use of plans by relating past experience to present problems. This makes CBR ideal as the "gatekeeper" between users and specialized problem solvers in two ways: the retrieval of past *solutions*, and the retrieval of past *solution strategies*.
- *CBR as Guide.* With the complex constellation of problem solvers and information sources available in large-scale planning, guiding users among available resources becomes of crucial importance. By retrieving *solution strategies*, case-based reasoning can provide structure even in the absence of acceptable past solutions.
- *CBR as Editor.* Specialized problem solvers make use of specialized problem representations, incomprehensible to those unfamiliar with the technology. With its traditional focus on higher-level abstractions, case-based reasoning can serve as an "editor" of this specialized "jargon," presenting results and options in terms that are clear and useful to users.
- *CBR as Opportunist.* It is an axiom of large-scale planning that solutions must always be partial and approximate, to be further developed over time. Case-based reasoning's focus on anticipation, expectation, and repair make it well-suited to organizing the incomplete work of specialized problem solvers and recognizing when new data is relevant to a particular computation.

These aspects are the organizing points of this report. As we discuss individual research projects, we will relate them to these aspects and point out how the research contributes to our overall understanding of CBR as *lingua franca* for large-scale planning.

1.1 Background: Large-Scale Planning

Large-scale planning, including deployment transport planning, has certain characteristics which pose challenges and opportunities for traditional AI planning and decision-making approaches.

- *Scale.* The size of large-scale plans are many orders of magnitude larger than those dealt with historically by traditional AI techniques.
- *Similarity.* The broad outlines of large-scale planning problems are highly similar to previous and prototypical situations.
- *Difference.* The specific details of large-scale planning problems are radically different from previous and prototypical situations.
- *Timeliness.* Solutions to large-scale planning problems must often be presented within a specific window of response time.
- *Uncertainty and Failure.* In large-scale planning there is always uncertainty about the world and always failures in execution.
- *Dependencies.* Operations of large-scale plans typically contain mutual dependencies and cannot be decomposed into independent subgoals.
- *Historicity.* Solutions to large-scale planning problems must be consistent with the standard operating procedure of the planning organization and individuals within it.
- *Opacity.* Users of large-scale planning systems cannot be aware of all the details that go into particular scheduling decisions.
- *Clarity.* Despite the opacity of large-scale planning systems, they must be able to clearly explain their decisions to their users.
- *Interactivity.* Large-scale planning systems must be responsive to user input, modifying plans and schedules to meet user-defined criteria.

These characteristics of large-scale planning define the requirements of large-scale planning systems. In the main part of this report, we will point out how these projects address these considerations.

1.2 Background: Case-Based Reasoning

The fundamental insight of case-based reasoning is that reasoning about new problems can make use of past solutions. In case-based planning, this insight is used to address traditional problems of plan generation, projection, and optimization. Case-based planners have the following basic features:

- New plans are built from old plans.

- Old plans are selected on the basis of the goals that they satisfy, the problems that they avoid, and the features in the world that have been associated with them in the past.
- Planning operators are used, not to build plans, but to explain plans that fail so that failures can be anticipated and avoided in the future.
- Old plans are modified to deal with conjunctive goals.
- Problems that arise due to modification are repaired, and the repairs are themselves used in later retrieval of old plans.

These features motivate a seven-module planning architecture that is the foundation of case-based reasoning.

1. *Anticipator.* Past failures due to goal interactions are used to predict planning problems in advance. Often the "anticipation" is used in retrieval by associating features predictive of a problem with a plan that avoids it.
2. *Retriever.* Searches plan memory for a plan that satisfies as many of the current goals as possible while avoiding predicted problems.
3. *Modifier.* Alters the plan to achieve additional goals from the input that the old plan did not satisfy.
4. *Projector.* Uses cases indexed by solutions rather than problems to predict outcomes.
5. *Storer.* Indexes new plans in memory by the goals that they satisfy and the problems that they avoid.
6. *Repairer.* Explains execution failures and alters plans to avoid the failure in the future.
7. *Assigner.*. Uses the explanation of a failure to index the problem for later anticipation and avoidance.

The flow of control through these modules is depicted in Figure 1.

In the most basic case, goals enter the case-based planner through the *Anticipator*, which tries to predict any problems that might occur as a result of planning for them. If a problem is predicted, a goal to avoid it is added to the set of goals.

The goals now pass to the *Retriever*, which searches for a plan that satisfies as many of the planner's goals as possible, including any goals to avoid the problems predicted by the *Anticipator*. In order to do this, the *Retriever* makes use of a memory of plans indexed by the goals they satisfy and the problems they solve.

Once an old plan is found, the *Modifier* alters it to satisfy any unsatisfied goals. Modification uses modification rules indexed by the goal to be added and the type of plan being altered. It also uses a set of critics for domain-specific modifications.

Case-based reasoning is the foundation for the work described in this report, and for our vision of large-scale planning systems.

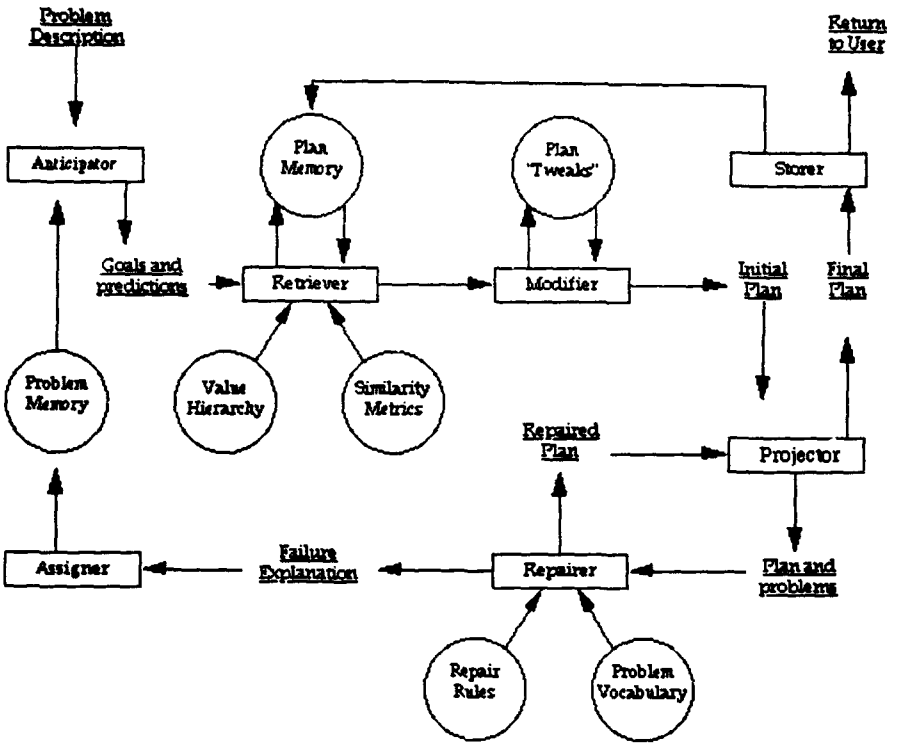


Figure 1: Case-Based Planning

1.3 A Vision of Large-Scale Planning

It is self-evident that there can be no single all-purpose panacea for large-scale planning. Instead, a host of planning and decision-making techniques from artifical intelligence and operations research must be brought to bear where they are most applicable. Some problems are the province of scheduling algorithms, while others may be most amenable to linear programming, Bayesian network analysis or non-linear planning.

Each such technique poses its own requirements on the structure of inputs and outputs; each solves a different kind of problem. Even within one paradigm, there may be many different applications of a technology, each requiring different databases, each with its own format and structure. In effect, they all speak different languages.

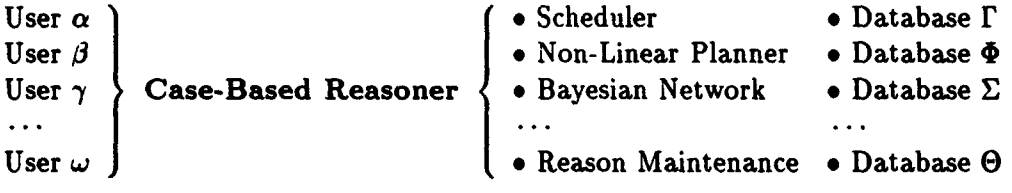
These techniques and applications can be brought together to build large-scale planning systems, but they require a common language, a *lingua franca*, to avoid a "Tower of Babel" problem. We believe that case-based reasoning can provide such a common language.

1.3.1 A *lingua franca* for large-scale planning

Proposals for common languages and environments have a poor history of acceptance. For example, the Knowledge Interchange Format proposal of a few years ago failed miserably, and the Common Prototyping Environment has had only lukewarm support from the research teams of the ARPI community.

This should not be unexpected. Forcing technologies to adopt a specific representation paradigm can only hamstring the advantages that technology has to offer. In some cases, the representation paradigm *is* the technology in question.

Our vision of case-based reasoning as a *lingua franca* is quite different. We propose that CBR be considered as a technology that stands between the users of large-scale planning systems and the host of technologies used in specific applications.



Here we outline the broad architectural aspects of this view:

- Specific technologies are "black boxes," and do not change.

  No requirements are put on the representations, algorithms, and implementations used for specific technologies (e.g., scheduling, non-linear planning). These are simply "black boxes" from the point of view of the case-based reasoning system.

- Each technology defines its interface with the main system.

  For each specific technology, there must be a specific input and output format. Again, no specific requirements are imposed from the point of view of the case-based reasoning system.

- Interfaces are mapped into case-based representations.

  For each technology, the interface imposed by that technology must be mapped into the knowledge structures used by the case-based reasoning system. This will usually require new CBR representations for each specific technology.

- The case-based interfaces are integrated with memory.

  This is the primary representational task. Given the representation of a specific technology's interface, the knowledge engineer must evolve specific case-based representations and abstract cases that allow the technology to be integrated into the larger case-based system.

1.3.2 An example: CBR and specific technologies

The ROENTGEN project described in the main body of this report provides a succinct example of the application of CBR as a *lingua franca* for specific technologies. ROENTGEN is a case-based system for planning radiation therapy in the treatment of cancer.

The dose calculator. The ROENTGEN system relies on the *dose calculator* as a separate technology. The dose calculator computes the degree to which tissue has been irradiated, given a specific radiation treatment plan. It is implemented in a different programming language, its internal workings are a mystery, it has idiosyncratic input and output formats, and it must be called as a "black box" from ROENTGEN.

To make use of this technology, the ROENTGEN system has case representations that are specific to the dose calculator. These represent the input, the output, and *the use* of the dose calculator. With these representations, the case-based ROENTGEN system is capable of making use of the dose calculator.

Case-based reasoning as user interface. By doing so, ROENTGEN effectively supplies a *smart user interface* to the dose calculator in the form of the entire case-based system. It is possible for radiation dosimetrists to make use of the dose calculator directly—but highly unlikely, given the difficulty of creating, modifying, and interpreting the input and output of the dose calculator.

Instead, the user interacts with the case-based ROENTGEN system, which makes use of the dose calculator whenever required. This is much more effective for the user. One of case-based reasoning's great strengths is the intuitive nature of its operations; users find the basic *Retriever/Modifier* cycle to be clear and intelligible.

1.3.3 Prospects for large-scale planning

Our hope is that case-based reasoning can serve the same role in the larger task of large-scale planning. When able to reason about all the specific technologies that make up the large-scale planning system, the central case-based reasoning system will present a unified, intuitive, powerful system for organizing the large-scale planning task.

1.4 Guide to the report

The report is organized around three research paths towards this goal:

- *Airline Irregular Operations.* To develop an understanding of the requirements of large-scale planning, we examined how schedulers for *United Airlines* identify and repair scheduling problems in real time.
- *Case-Based Interfaces.* To develop an understanding of how case-based reasoning can provide a unified, powerful, and intuitive interface, we examined the problem of using CBR to integrate natural language with specialized systems for planning and vision.

- *Planning and Execution.* To develop case-based reasoning along the lines required by large-scale planning, we examined the problems of case-based reasoning in a number of dynamic environments with real-time plan execution, failure and repair, information gathering, and opportunism.

In the sections that follow we will describe the results from each of these research paths.

2 Airline Irregular Operations

2.1 Summary

The Chicago research group identified *airline irregular operations* (or *IOPS*) as a domain that shares many significant features with deployment transport planning. In order to exploit this similarity, we formed a cooperative research relationship with United Airlines. Owens, assisted by John Borse, a graduate student on leave at the time from the Operations Research group at United, spent time at United's operations center studying the techniques used by experienced schedulers to identify, diagnose and repair schedule problems triggered by bad weather, equipment breakdown, traffic congestion and similar inevitable but unpredictable occurrences(Borse and Owens, 1992). They also built software to allow operating data from the airline to be incorporated into a planning testbed.

The primary scientific focus of this work is on representation(Owens, 1991; Owens, 1990)

Specifically, we are determining how to represent schedules, schedule failures, and repair strategies so as to enable the IOPS advisor to:

- Identify and characterize schedule problems so as to determine the applicability of prior solutions or specific quantitative techniques.
- Acquire new descriptive features as they become necessary to discriminate among otherwise indistinguishable situations.
- Compare the applicability of multiple, competing solutions to the same problem.

The results of this work were threefold:

- An analysis of failure and dynamic repair in a complex transportation planning environment. The analysis is based on real-world failures and repairs as diagnosed and implemented by skilled experts. The analysis contributes to a machine-manipulable vocabulary for representing failures and repair strategies.
- Software that makes it possible to capture real operating data from the airline, to represent and reason about the downstream consequences of schedule modifications, and, in limited measure, to graphically display and interactively manipulate scheduling information[1].

[1]Because the University of Chicago's software has fallen out of synchronization with ongoing work at the airline, additional engineering effort will be needed to restore the software to proper operation for any future use.

- A research relationship with an organization that has demonstrated expertise in solving complex dynamic rescheduling problems every day. Such a source of real-world (as opposed to realistic but simulated) data is critically important to the Chicago group's theoretical approach to learning from experience.

2.2 The problem domain

An operating schedule for an airline the size of United involves roughly [illegible] aircraft making approximately 2000 flight operations per day. The individual components of [illegible]ule (i.e. flight segments) are densely interconnected by a network of dependencies. An outbo[illegible]flight may depend upon the arrival of a prior inbound flight because the outbound flight may use the same aircraft, because it may use the same cockpit or cabin crew, because there may be connecting passengers or cargo, or because the inbound flight may be carrying necessary repair parts for the aircraft to be used on the outbound flight.

Because of this densely interconnected nature, airline operating schedules are highly sensitive to perturbation. Even a single point failure, such as an aircraft delayed for 30 minutes due to a flat tire, can cause a cascade of downstream problems.

Although the *static schedule* (the one the airline would fly if everything worked perfectly according to plan) is carefuly optimized in advance using well-understood quantitative techniques, controllers must take reactive steps (i.e. by cancelling, rerouting, or rescheduling flights, swapping aircraft and crews, etc) to counteract the effects of inevitable, but individually unpredictable, disruptions. The goal is to contain (or to repair, if possible) the damage to the schedule, minimizing passenger delay and costs.

It is this dynamic repair that is of relevance to the task domain of the Planning Initiative. In a large-scale deployment transport environment, uncertainty is high, and dynamic recovery from failure is essential. Assets may become temporarily or permanently unavailable due to weather or military action, requirements may change, and scheduling errors born of inexperience with the particular routes being flown are likely. Scientific knowledge gained by studying the irregular operations problem in civilian air transport is very likely to be useful in building systems to support dynamic rescheduling and replanning in the large-scale military deployment transport domain.

2.3 Irregular Operations Scheduling (IOPS)

Airline schedules are highly complex, structured objects, with large numbers of internal interdependencies. Airlines must confront the consequences of uncertainty in the execution of their daily schedules — uncertainty stemming from inclement weather, sick calls from crew members, mechanical problems with aircraft, constraints on airport resources, and other problems. A snowstorm at a key airport, for example, can have devastating consequences on the operations of an airline, effects from which it may take days to recover. The interdependencies among factors like crew scheduling, maintenance routing, and congestion at airports add further complication to the daily planning problem. Because of these interdependencies, even a single disruption and the consequent attempts at recovery typically involve widespread and long-lasting downstream effects. The search space of possible recoveries to a schedule disruption is enormous.

Airlines employ schedule planners who attempt to mitigate the effects of schedule disruptions. Their main goals are to minimize both passenger inconvenience and the cost of implementing the repair, while accounting for crew work rules, aircraft maintenance schedules, and other factors. An additional goal is to minimize the overall complexity of a repair.

Controllers attempt to balance the trade-offs and uncertainties of irregular events, typically using information provided by various decision support systems such as real-time scheduling displays and passenger booking data. However, very few, if any, of these systems provide the planner with decision-making advice in the form of strategies or specific recommendations to counteract the adversity of a particular event. The goal of our research is to develop the scientific foundations for a new class of decision support tool to address this problem(Owens, 1992a).

From the viewpoint of Artificial Intelligence planning and decision support, the key features of the irregular operations planning task are:

- Airline schedules are large, complex, and highly interdependent.
- Solving schedule problems by exhaustive search is generally infeasible.
- Current situations typically share more with past situations than they differ from them.
- While they may be similar, no two situations are ever entirely identical. This means that simply storing and reusing a "library" of solutions will not suffice.

The size of the search space, together with the recurring nature of typical problems, suggests a solution based on the re-use of plans. But re-using plans means more than just retrieving and replaying old solutions. Because the details of situations change over time, the system will need to be able to notice that a retrieved plan does not exactly fit the current situation, therefore it will need to modify its retrieved plans to fit new situations.

Our approach to plan repair is to provide qualitative expertise in the form of a case library linking descriptions of stereotypical problems with appropriate recovery strategies, and quantitative expertise in the form of optimization techniques drawn from the Operations Research (OR) community. The goal of our research is to develop the scientific foundations for a new class of decision support tool. The IOPS Advisor, currently under development, couples the experiential knowledge of schedulers, which is essential in generating strategies for solving a schedule problem, with the quantitative power of operations research techniques, which are effective in comparing the costs and effectiveness of the potential solutions generated by those strategies. Furthermore, the quantitative models may be responsible for optimizing the details missing from a sketchy solution suggested by a qualitative strategy. For example, if a strategy is "stop to refuel", a quantitative analysis may indicate where to stop and how much fuel to take on.

The IOPS Advisor is intended to represent schedules, failures, and repairs so that both sets of techniques can cooperate using the same representational constructs.

2.4 Field study

Real data (as opposed to realistic but simulated data) are essential to building systems that reason from experience. A key theoretical aspect of the Chicago approach is that a system should reduce

the otherwise intractable complexity of plan repair by representing and reasoning about some core set of stereotypical plan failures and corresponding repairs, with the core set chosen to cover the plan failures that typically occur in real situations, as opposed to attempting to cover the entire distribution of theoretically possible failures.

Accordingly, an important part of the work was to study experienced operations controllers as they solved real schedule problems as encountered in the day-to-day operation of the airline.

The space of planning operators available to schedule controllers is limited. They can cancel, reschedule, or reroute flights. They can skip or add intermediate stops. They can substitute one aircraft or crew for another. It is these operators that would form the basis of a simple generative planner operating in the domain of airline irregular operations.

From our study, however, it appears that controllers consciously and deliberately use abstract problem solving strategies built up out of the primitives to, for example:

- **Distribute** the effects of a problem over a broad geographic area so that no one point in the route system suffers severely, for example by introducing minor delays in a large number of incoming flights to prevent major congestion at a hub.
- **Localize** a problem to prevent it from propagating to other portions of the route system. For example, if an airport is expected to be closed due to bad weather, route aircraft around it to prevent them from being stranded there and delaying their future assignments.
- **Defer** the effect of a problem in the expectation that an opportunistic or cheap solution will ultimately present itself. For example, if an aircraft is late arriving at a hub, cover the outbound flight to which that aircraft had been assigned by borrowing an aircraft from a later flight, which can in turn be covered by borrowing from a still later flight, by which time the original aircraft will have arrived and can be re-introduced into the schedule. This strategy is interesting because, while often suboptimal, it is perceived as valuable because of the degree to which it reduces the cognitive complexity of plan repairs.

It is also clear that the applicability of these strategies depends upon the presence of abstract features that characterize situations. Certain strategies, for example, might be applicable at hub airports during peak travel times. Other strategies might be applicable to infrequently-flown routes to remote cities at the end of the day. These features, while not explicitly present in the schedule data, can in general be inferred from it.

These higher level strategies, and the mechanisms for detecting their applicability and utility, would form the core of a planner built upon the results of this study.

A further observation is that experienced controllers use specific experience to rapidly prune the search space of possible solutions to a problem, for example:

> "I won't even look at Westbound flights out of this city right now as possible places to borrow an airplane, because I know from experience that those flights will be fully booked."

or

"I know, almost without checking, that I'll be able to cover these 5 stranded passengers quickly because there is hourly service between those two cities, and we're still in the middle of the day."

The abstract characterizations of situations and problems that we learn by observing controllers solving real problems, becomes the core of a machine-manipulable representation vocabulary useable for building plan repair systems.

2.4.1 Knowledge Representation Issues:

The main knowledge representation issue, and the primary focus of our current activity, is to categorize and represent the heuristic knowledge used by controllers and Operations Research analysts, specifically:

- How problems are detected and described.
- What problem-solving strategies exist.
- What aspects of a problem indicate the applicability of one strategy over another.

In order to gather a realistic set of failures and repairs, we have been observing controllers as they detect, diagnose, and repair schedule problems. Our initial study has suggested to us that controllers build and use sophisticated, high-level repairs from a small number of primitive operators. The primitives form the basic representation vocabulary used to describe actions, and it is anticipated that the list will be stable over time. The higher-level strategies, on the other hand, are more dynamic, and one of our tasks is to model the acquisition of new high-level strategies.

Typical primitive operators represent concrete actions like:

- Cancel a segment
- Delay a segment
- Divert a flight to a different airport
- Substitute one aircraft for another
- Substitute one crew for another
- Ferry an empty aircraft from one airport to another

Higher-level strategies, on the other hand, may involve both primary actions and secondary actions designed to mitigate the side-effects of the primary actions. Or, they might involve a series of steps taken to defer the impact of a problem, in the expectation that an opportunistic solution may present itself in the intervening time. Other high-level strategies include geographically localizing the impact of a problem or, conversely, diluting the impact of a problem by spreading a minor delay across several geographic points.

As we gather more high-level strategies from our observation of controllers and from our encoding of quantitative techniques, our plan is to encapsulate the strategies in knowledge structures that also include descriptions of appropriate situations for the strategies. The IOPS advisor will extract from the user a description of the current situation, propose repair strategies based upon the match between the current situation and the stored descriptions, and quantitatively evaluate the utility of situations generated by competing strategies. As it performs this selection and comparison, it can acquire, from the user, information about features of the world that determine the applicability of one strategy over another. These newly-acquired features can then become part of the selection criteria encoded with the strategies in memory.

2.5 Software development

We have developed a set of software tools that allow us to:

- Copy electronic operating and schedule data from the airline to our own systems.
- Translate the data into a representation language enabling automated reasoning about dependencies, temporal constraints, and the downstream consequences of schedule perturbations and attempts at repair.
- Graphically display various aspects of the data, including calculated dependencies and constraints.

This suite of software tools exists as a potential building block of future systems. Unfortunately, due in part to personnel changes, it is no longer synchronized with changes in the way United represents its internal data, so additional engineering effort is required in this area.

We have also conducted a preliminary study of the applicability of the Honeywell implementation of Dean's Time Map Manager (Boddy and Dean, 1989) as a platform for temporal inference.

2.5.1 Knowledge acquisition

While the list of primitives is expected to remain relatively static, an important aspect of the IOPS Advisor is that it will be able to acquire new descriptive features as it is used. If the system erroneously suggests a prior case as being a good match to the current situation, the user can correct this by supplying a descriptive feature that would differentiate the current situation from the case stored in memory. The error might have occurred either because the discriminating feature was not mentioned in the description of the current situation, or because it was not mentioned in the stored case. In the latter scenario, it can be added.

In general, a longer-range goal for the IOPS advisor is that, in having a human user interact with a planning tool, we have an opportunity to record information about plan accessing strategies, modification techniques and typical failures that can, in turn, become the heuristics used by a more autonomous system. A system that observed human schedulers in action and recorded their responses to specific planning problems, and which indexed those responses in memory using the functional criteria discussed above, would become a powerful expert assistant — an assistant with a good memory for what worked and what didn't in the past.

2.5.2 Case-Based Planning Issues

While case-based planning addresses many of the qualitative problems in the irregular scheduling domain, much work must be done before a practical system could be put in the hands of a human scheduler. Fortunately, the core idea in case-based planning, that of incremental modification, is one aspect of the technology that could be usefully applied in the near term as a way to deal with the type of changes that have to be made to schedules during execution.

One of the recurring problems of automated planning is the issue of the repairs that have to be made during execution as a result of unforeseen circumstances. There are always unexpected problems that arise. Weather, crew sickness, and equipment failures cannot be predicted. Bottlenecks show up where none was suspected. Each of these classes of problems can be recognized using a specific set of symptoms, and each requires a specific type of repair.

Run-time repair and optimization, while useful, has to be traded-off against the overall stability of an existing plan. If a single aircraft is unexpectedly grounded, one form of optimization might be to rebuild the entire system schedule, minus that aircraft. But even if such a repair were computationally feasible, implementing it would be preposterous. A planner that deals with unexpected changes in the state of the world by completely replanning will be constantly creating new plans that will do little more than confuse the people that are using them. What is needed instead is incremental, local plan repair, coupled with local optimization. One wants to perturb the schedule as little as possible in the achievement of an acceptable response to an unexpected occurrence.

Much of the emphasis of CBR research to date has been on issues of plan indexing, retrieval and modification. While these issues are clearly present in this domain, our emphasis is primarily on plan evaluation through objective analytical (e.g., OR) tools which are also under development. Specifically, we are focusing on how to direct the search for relevant cases based on the OR model's assessment of the feasibility or "utility" of previously proposed solutions. Because the two sets of techniques tend to characterize the problems differently, integrating them is a challenge.

2.5.3 Operations Research Issues

Operations research analysts tend to think in terms of opportunities for optimization. One of our preliminary findings is that schedule planners do not readily identify these opportunities. Accordingly, an important aspect of the integrative research is to identify classes of situations in which particular optimization techniques are appropriate, and to select descriptive features that allow the system or planners to differentiate among these classes. We intend to codify this knowledge in the form of cases which couple the relevant optimization techniques with characteristic features of the appropriate class of situation.

2.6 Case Study

The following hypothetical case study is based on observations of airline planners. The case illustrates the interplay between qualitative and quantitative reasoning tha was the focus of our work. Airports are designated by the following three letter codes: SFO = San Francisco, EUG = Eugene, and MED = Medford.

A runway construction project at EUG has imposed a weight restriction on departing flights. A departing flight EUG-SFO is over the weight limitation by approximately 20 passengers. The flight is scheduled to depart on time, however, inbound flow control is in effect at SFO (due to fog) and is imposing a 53 minute pre-takeoff delay on the EUG-SFO flight.

The planner generates some alternative solutions:

1. Move the excess passengers to a later EUG-SFO flight.
2. Have a flight enroute to SFO passing nearby EUG stop to pick up the excess passengers.
3. Remove enough fuel to carry the excess passengers, and stop at an intermediate point to refuel.

At this stage, the alternatives are qualitative: they simply match a problem with a strategy. Although in many cases this step of the solution process is trivial (e.g., weather-related IOP forces cancellations), we believe that in general this step is non-trivial and it is one aspect of the planner's job which distinguishes an experienced planner from an inexperienced one.

The next step of the planning process involves evaluating the relative merits of each proposed strategy with respect to the planner's goals. In this case the planner chose not to solve the problem using strategy (1) because pushing the problem to a later flight would most likely cause weight restriction problems downline and would disservice the excess passengers. Strategy (2) was not chosen since it would involve delaying a large number of passengers on a different flight to accommodate a relatively small number of connecting passengers on the EUG-SFO flight. On further analysis of strategy (3), the controller determined that, since SFO air traffic control had already imposed a 53-minute delay on the inbound flight for reasons of airspace crowding, the flight could in fact refuel at MED and carry all passengers to SFO as planned without incurring additional delays. The cost of landing and departing at MED was considered negligible in comparison to the alternative costs of delaying passengers and causing misconnections of aircraft and people (although this calculation was not performed explicitly).

Notice that the planner's analysis in choosing among alternatives remains highly qualitative. The planner uses various sources of information to determine the viability of each approach, however, he rarely explicitly calculates the cost impact of various strategies. We believe that at this stage the planner could be greatly aided by OR models which:

- provide an objective analysis of the relative merits of each strategy based on utility measures.
- determine optimal implementations of high-level strategies, for example, given strategy (2), choosing an appropriate flight, or, given strategy (3), choosing an appropriate airport.

2.6.1 Evaluation

The bases against which we can evaluate the IOPS advisor project are:

- Does the system enable a controller to produce good schedule repairs? In particular, can a controller use the system's prepackaged strategies and OR evaluation methods to improve upon solutions produced using the controller's own judgment?
- How good are the high-level strategies that the experienced planners employ? How often do controllers choose the best strategy? While the strategies obviously work, are they applied inappropriately? Does post-facto analysis repeatedly indicate that some other strategy might have been preferable?
- Are individuals able to make use of the canned strategies? Can one individual recognize and re-use canned strategies? Is there any transfer across individuals, such that one individual can use strategies developed by another? If so, how should the strategies be presented to the user?
- Can novices use the strategies and optimizations from the IOPS advisor to generate expert-like repairs? In general, how do solutions built by novices differ from solutions built by experts? Does the availability of a library of expert solutions improve a novice's performance?
- Does the integrative AI/OR approach provide a better method than either technique applied alone? Is it even possible to model the IOPS problem using either technique alone? What form would these models take (e.g. large scale linear programming, expert-system)? How would each of these approaches compare to the integrative approach?

2.7 Future directions

This work has established several building blocks usable in future work on planning systems:

- A taxonomy of stereotypical, recurring failure types, (e.g. *single-aircraft delay during peak hour at a hub airport* or *weather delay at non-hub airport at the end of the day*) and associated repair strategies (e.g. *borrow aircraft on a rolling basis from later scheduled flights*)
- Major building blocks of a software environment for representing, displaying, and manipulating complex schedules and for inferring the consequences of perturbations.

These failure types and repair strategies will constitute the high-level operators of an automated plan repair system. Initially, the system would need to ask the user for assistance with the task of diagnosing the failure (i.e. by selecting the appropriate failure categorization from a displayed list). Future research will identify a set of diagnostic features that make it possible for the system to autonomously identify the applicable failure type in many cases.

The software enviroment brings in house a rich repository of real-world data, which can be used as a testbed for a variety of planning tasks.

2.8 Large Scale Transportation Planning

A key aspect of the Chicago approach is the use of **real** data, as opposed to randomly-generated test cases or data that is merely realistic. This is because our approach gains leverage on the intractability of planning by addressing the *expected case* rather than the *general case*.

Planning and scheduling, as has been argued in (CHAPMAN, 1985) and elsewhere, are, in the general case, computationally intractable. Given a representation language rich enough to capture the PI's task domain, the space of syntactically valid plans that could be generated is enormous.

Fortunately, the space of plans likely to occur in the real world, while still large, represents a tiny fraction of this space of possible plans. Our approach depends upon capturing these likely and commonly-recurring situations as cases or stereotypes, and thereby avoiding the need to resort to the kind of bottom-up plan construction or repair that the general case requires.

Accordingly, we set out to find a set of task domains that shared enough with the Planning Initiative's task domain to be a reasonable match, but that represented opportunities for us to capture data about real-world plan failure repair. One such opportunity came in the form of the Irregular Operations problem at United Airlines.

Airline scheduling and schedule repair is, obviously, a task with many parallels to deployment transport. The scale of operations at United (approximately 400 aircraft making 2000 flight operations per day) yields a level of complexity and plan element interdependency sufficient to make the problem comparable to the scale of a deployment plan. And we had the advantage of being to study human experts diagnosing plan failures and making repairs every day, in an unclassified setting.

Of course there are important differences between an airline schedule and a deployment transport plan. Most importantly, an airline schedule is a continuous operation, cycling approximately daily. A deployment transport plan is much more of a one-shot nature. Whereas the entire deployment plan is dynamic, a routine schedule is dynamic only to the extent that weather, equipment failure or unexpected demand for irregular service causes it to deviate from the norm. Nevertheless, we believe that the lessons learned about plan failure and repair in the latter domain are applicable to the former.

3 Case-Based Interfaces

The research on case-based interfaces has been directly aimed at examining how case-based reasoning systems can provide the interface between users and specialized technologies for large-scale planning. There are two types of interfaces required:

- The *user interface*, for which we have concentrated on natural language as the most general (and difficult) user interface modality.
- The *system interface* to specialized technologies, for which we have used reactive planning and active vision as test cases.

In both cases, the research has addressed three characteristics of large-scale planning:

- *Opacity.* Users of large-scale planning systems cannot be aware of all the details that go into particular scheduling decisions.

- *Clarity.* Despite the opacity of large-scale planning systems, they must be able to clearly explain their decisions to their users.
- *Interactivity.* Large-scale planning systems must be responsive to user input, modifying plans and schedules to meet user-defined criteria.

Since natural language has been the focus of the user interface, the research in case-based interfaces is based on a case-based parsing technology called Direct Memory Access Parsing (DMAP) and a specialized technology for reactive execution called Reactive Action Packages (RAP).

The DMAP system is designed to function within a large body of already-represented knowledge; rather than determine the meaning of a text, it uses the text to recognize relevant existing knowledge structures and modify them to represent what is unique about a particular communicative act. It is fundamentally case-based in its theoretical design, though its implementation differs dramatically from traditional case-based systems(Martin, 1990).

The RAP system provides a coherent means of representing planning and execution knowledge; its semantics of interpretation are simple and do not require maintaining complex dependency structures. For these reasons, it provides an ideal framework upon which to build a DMAP-style natural language system(Firby, 1989).

We have developed a hybrid system in which the structures and algorithms of DMAP are used to represent the model of planning and execution expressed in the RAP system. In addition, the system represents the knowledge of how to *talk about* its own model of planning and execution, allowing the system to learn by taking advice(Martin and Firby, 1991a).

The resulting system is capable of effective real-time response due to the underlying reactive system, but is capable of reasoning about and modifying the reactive system in response to natural language input(Martin and Firby, 1991b; Martin and Firby, 1991c; Martin and Firby, 1992).

3.1 Challenges

In large-scale planning, specialized technologies play a critical role in integrating the disparate elements of large plans. These technologies, such as scheduling, non-linear planning, and others, are akin to the "primitive actions" of the overall planner. They are separate, each with its own purpose and ability, and together they enable large-scale planning.

Unfortunately, the interaction of users with large-scale planning systems will bear little resemblance to the the requirements of such specialized technologies. User interaction is, in general, defined by more global concerns that focus on critical objectives rather than minute details.

Once these objectives pass to the large-scale planner, they must be represented in exactly that minute detail so that specialized technologies can be brought to bear on the objectives. The challenge is to determine how the high-level objectives that characterize users' interactions with the case-based reasoner can be mapped into the detailed specifications that characterize specialized technologies. The remainder of this section of the report details our work on exactly this problem.

In the remainder of this section, we first present background on the specialized RAP technology, background on the DMAP case-based parsing technology, and then an detailed explication of how the case-based reasoner can interface to the specialized system through uniform representations.

3.2 Background: Reactive Action Packages

The RAP situation-driven plan execution system (Firby, 1989) uses Reactive Action Packages, or RAPs, as the basic mechanism for packaging methods. A RAP groups together and describes all of the known ways to carry out a task in different situations. When a task is generated, its goal is used as an index to select a RAP, and when the task is selected for execution, the situation surrounding the robot is used to index into that RAP and select the most appropriate method it packages. A RAP includes the methods for a task and the contexts in which they are appropriate in a single package, so there needs to be only one RAP for every task type. However, there may be many independent instantiated tasks using the same RAP definition.

A RAP also includes a description of the situations in which its task is satisfied. A task has two parts: a goal, or index, and a test that defines when the task should be active. The activation test itself consists of two parts: a satisfaction test. The satisfaction test tells whether or not the task's goal is satisfied in a given situation. If a task's goal is not satisfied, the task should be an active candidate for execution. A satisfaction test is the property of a particular task but since there is only one RAP for each task, the satisfaction test is described in the RAP. Thus, a RAP is a complete description for the execution of a task. It contains the task's goal satisfaction test and all of the methods to achieve the task's goal in different situations.

```
(DEFINE-RAP
   (INDEX  (arm-pickup ?arm ?object))
   (SUCCESS (arm-holding ?arm ?object))
   (METHOD
      (CONTEXT (location ?object tool-caddy))
      (TASK-NET
         (t1 (arm-move-to-caddy ?arm)
             ((at ?arm tool-caddy) for t2))
         (t2 (arm-grasp ?arm ?object))))
   (METHOD
      (CONTEXT (not (location ?object tool-caddy)))
      (TASK-NET
         (t1 (locate ?object)
             ((location ?object ?place) for t2))
         (t2 (arm-move-to ?arm ?place)
             ((at ?arm ?place) for t3))
         (t3 (arm-grasp ?arm ?object)))))
```

A simple RAP is shown above. Its definition is split into three major sections: the INDEX which corresponds to the task-goal that the RAP is to satisfy, the SUCCESS clause which describes a test on the memory to determine if the goal is satisfied in the current situation, and any number of METHOD clauses that describe possible ways of carrying out the behavior under different circumstances.

Each method within a RAP is further broken down into two sections: the CONTEXT which describes a test on the memory that can be used to determine whether this is an appropriate way to achieve the desired behavior, and a TASK-NET which contains the detailed steps involved in the behavior. Task net steps may be primitive actions to be executed directly, calls to other RAPs to perform some behavior, or constructs that allow loops and conditionals.

In this example, the RAP defines a task to pickup a given object. This RAP is used for tasks with index `(arm-pickup ?arm ?object)` and will be satisfied when `(arm-holding ?arm ?object)` is true in the memory. There are two different ways of actually picking up the object and the one chosen depends on whether the object is in the robot's tool caddy. Special operations are required to move the arm into the tool caddy and additional sensing operations are required to locate the object if it is not known to be in the caddy.

3.3 Background: Direct Memory Access Parsing

The implementation of the DMAP case-based parsing technology is critical to the success of case-based reasoning as a *lingua franca* for specialized technologies. Since the DMAP algorithm does not have provisions for traditional AI concepts such as "plans," "goals," or "syntactic rules," all such knowledge must be explicitly represented in the uniform memory format of the DMAP system.

The fundamental processing of the DMAP technology is consistent with traditional case-based reasoning, although the approach is somewhat different. Inputs to the system are used to *recognize* existing knowledge structures and *modify* them to represent what is unique about the current situation. At a high level, the algorithm may be expressed as two coroutines:

coroutine PROMOTE *concept*	
if *concept* is a primitive operation	P1
then apply it and INTERPRET input	P2
else for each *index* of *concept*	P3
do PROMOTE the first *item* of the *index*	P4
coroutine INTERPRET *item*	
for each *concept index item* that matches *item*	I1
do begin	
refine the *concept* based on the *item*	I2
if the *concept index* is complete	I3
then INTERPRET *concept*	I4
else PROMOTE the next *concept index item*	I5

The basic algorithm may be compared with the basic cycle of case-based reasoning (Section 1.2).

In DMAP, concepts are *retrieved* at statements I3–I4. Prior to this point, they have been anticipated through the cumulative promotion and intepretation of their indices (statements P3–P4 and I5). *Modification* of concepts due to differences between past experience and the present situation occurs at statement I2.

Three aspects of the DMAP architecture important to understanding how case-based reasoning interfaces with users and specialized technologies are further explained with respect to examples drawn from the DMAP/RAP agent architecture project.

3.3.1 Primitive operations and input

For the high-level agent, primitive operations are interactions with the lower-level RAP interpreter; many of these ultimately result in changes to the underlying control routines, while others communicate information back to the high-level agent.

This is critical for the success tests. The indices for **rap** both have a reference to **the-success** specified; since the concept referenced by this relation will always be a **state** structure, some means must exist for entering the INTERPRET coroutine with a **state** as input.

The communication is handled by primitives whose effect is to establish a monitor condition in the lower-level RAP interpreter. For example, if the success test of a hypothetical **rap** were that there was a blue object in front of the robot, the primitive would establish a monitor condition in the RAP interpreter that, in turn, sent a particular color historgram to the vision processor. When the vision processor signalled successful recognition of the histogram, the RAP interpreter would respond by resuming the INTERPRET coroutine with the appropriate **the-success state** structure as as input.

As another example, here are some of the definitions for **fold-arm**:

```
(DEFINE-NODE fold-arm (ISA rap)
  (SLOTS (the-arm arm)
         (the-success arm-folded
           (WHERE (the-arm = the-arm)))))

(DEFINE-NODE arm-folded (ISA state)
  (SLOTS (the-arm arm))
  (INDEX (MONITOR (folded-p the-arm))))
```

Where "**folded-p**" is a predicate in the world-model of the lower-level RAP interpreter. Note that no **the-method** relation appears in **fold-arm**; as with **move-arm**, they would appear as specializations.

3.3.2 Gathering indices

This step updates indices that can refer to a node. By doing this dynamically, the state of promoted nodes is constantly changing to reflect the current state of the DMAP system. This is crucial for resolving ambiguity of reference.

As an example, consider the previous reference to part of the **m/rap/prog2** structure: "moving it inside the bay." "It" in this case refers to the arm, but how is that reference established? Recall the **move-arm** definition:

```
(DEFINE-NODE m/move-arm (ISA mtrans)
  (SLOTS (the-info move-arm))
  (MTRANS move
          (the-info the-arm)
          (the-info the-destination)))
```

After the recognition of the lexical item "move," the memory reference to the arm causes the promotion of the node representing that arm. The lexical item "it" is associated with the general communication of nouns; although there are many nouns, the recent promotion of the arm node disambiguates the lexical item.

3.3.3 Refining concepts

Refining concepts implements a core part of the case-based reasoning theory, in which specific prior instances are used to guide the current interpretation. This allows the system to make use of more specific knowledge as soon as possible in the interpretation process.

For example, an index used to recognize `move-arm` may be adjusted after the relation `the-destination` is found to refer to an internal location. Since the more specific `arm-move-internal` represents this more specific information, the system will adjust the node from `arm-move` to `arm-move-internal`. This means that *immediately*, the additional information represented at `arm-move-internal` is available. In this case, the knowledge that `arm-move-internal` requires that the arm be folded may prove to be relevant.

The more specific node also results in updates to the node promotions described above. This usually proves useful in disambiguation. Ease of disambiguation may result in further specialization of the associated node, and the two processes may feed back and forth in this manner many times during PROMOTE-INTERPRET interactions.

3.4 The dmap/rap agent architecture

The lowest-level of the agent architecture is a reactive planner based on the RAP model of Firby (1989). The RAP model assumes that there is a prescribed method for carrying out every task in every situation. Each method is a set of actions that will accomplish the task in a given situation. Execution consists of choosing a task to work on, assessing the current situation, choosing an appropriate method, and carrying it out. The assumption of known methods makes task execution very much like hierarchical plan-expansion except that it takes place at execution time rather than being done in advance, and is done with reference to states of the world model that can be determined without inference. This assures real- time behavior.

The low-level reactive system is somewhat simpler than the original RAP system, since many of the memory issues are the responsibility of the high-level agent. In particular, the ability to connect sensor readings with previously-identified objects is now the responsibility of the high-level system.

Although the RAP system is capable of real-time response to its environment, it cannot cope with fundamental changes in that environment. This is the role of the high-level agent, and constitutes the primary focus of this research.

The fundamental idea is to create a high-level representation of the reactive system. This representation will include both the data structures *and algorithms* of the reactive system, and will be directly manipulable by the algorithms of the high-level agent. In particular, the high-level agent can build new reactive planning data structures to be subsequently incorporated into the low-level reactive system. Agency thus arises out of the interaction between the reactive system and the high-level agent:

- The reactive planning system provides continuous real-time response to the environment.
- The high-level agent responds to environmental changes by creating new reactive planning structures.

The DMAP/RAP agent architecture project is being implemented in two domains: the simulated *TruckWorld* environment (Firby and Hanks, 1987), and the University of Chicago robotic platform.

3.4.1 The *TruckWorld* domain

The *TruckWorld* domain is a simulated delivery domain, in which a series of transport goals are generated during runtime. The agent must coordinate the achievement of these transport goals while dealing with contingencies of the world; for example, a bridge may be out, it may run low on fuel, or other malevolent actors may appear to damage the delivery vehicle or steal objects while in transport.

The level of knowledge and decision making is that of control routines, low-level behaviors, and the underlying RAP architecture. These procedures are inaccessible to the user, comprising as they do the "basic competence" of the system. The following low-level transcript demonstrates the overwhelming detail that the user would have to confront without the case-based intermediary. In this example, the delivery vehicle (truck) receives a new delivery order while trying to move down the road:

```
New top level goal: DELIVER-ROCKS :: #{Goal 19}
 Adding goal to agenda: DELIVER-ROCKS :: #{Goal 19}
 Data+: (ROAD-SEEN W ROAD-136)
 Data+: (ROAD-SEEN E ROAD-140)
 Data+: (ROAD-SEEN S ROAD-139)
 Data+: (OBJECT-SEEN EXTERNAL USER-3 ROCK-CONSUMER)
 Event+: (:EVENT-567 OKAY) - SUCCEED :: #{Goal 26}
  Disabling event :EVENT-568
  Disabling event :EVENT-567
  Disabling skill: EYE-SCAN
 Goal accomplished: (EYE-SCAN-P EXTERNAL) :: #{Goal 26}
  with result: SUCCEED OKAY
 Removing goal: #{Goal 26}
 Goal accomplished: (TRUCK-MOVE-DOWN-ONE-ROAD ROAD-136 NODE-96) :: #{Goal 15}
  with result: SUCCEED OKAY
 Removing goal: #{Goal 15}
Considering goal (TRUCK-MOVE-DOWN-ONE-ROAD ROAD-139 NODE-104) :: #{Goal 20} :NEW
 Instantiating METHOD - METHOD-139
 Adding goal to agenda: PULL-IN-ARM :: #{Goal 15}
 Adding goal to agenda: PULL-IN-ARM :: #{Goal 26}
 Adding goal to agenda: TRUCK-SET-SPEED :: #{Goal 27}
 Adding goal to agenda: TRUCK-SET-HEADING :: #{Goal 14}
 Adding goal to agenda: TRUCK-MOVE-P :: #{Goal 8}
 Adding goal to agenda: EYE-SCAN-P :: #{Goal 29}
 Suspending goal: TRUCK-MOVE-DOWN-ONE-ROAD :: #{Goal 20}
```

```
Considering goal (PULL-IN-ARM ARM1) :: #{Goal 15} :NEW
 Goal accomplished: (PULL-IN-ARM ARM1) :: #{Goal 15}
  with result: SUCCEED ()
 Removing goal: #{Goal 15}
Considering goal (TRUCK-SET-HEADING S) :: #{Goal 14} :NEW
 Instantiating METHOD - METHOD-136
 Adding goal to agenda: TRUCK-TURN-P :: #{Goal 15}
 Suspending goal: TRUCK-SET-HEADING :: #{Goal 14}
Considering goal (TRUCK-TURN-P S) :: #{Goal 15} :NEW
 Instantiating METHOD - PRIMITIVE-10
  Enabling skill: (TRUCK-TURN S) - #{Goal 15}
  Enabling event :EVENT-569 : (SUCCEEDED TRUCK-TURN)
  Enabling event :EVENT-570 : (FAILED TRUCK-TURN)
 Blocking on events: TRUCK-TURN-P :: #{Goal 15}
Processing event queue ...
 Data+: (CURRENT-TIME 186)
 Data+: (CURRENT-STATUS HAPPY)
 Data+: (CURRENT-SPEED FAST)
 Data+: (CURRENT-HEADING S)
 Data+: (CURRENT-FUEL 11.405333333333335)
 Event+: (:EVENT-569 OKAY) - SUCCEED :: #{Goal 15}
  Disabling event :EVENT-570
  Disabling event :EVENT-569
  Disabling skill: TRUCK-TURN
 Goal accomplished: (TRUCK-TURN-P S) :: #{Goal 15}
  with result: SUCCEED OKAY
 Removing goal: #{Goal 15}
 Goal accomplished: (TRUCK-SET-HEADING S) :: #{Goal 14}
  with result: SUCCEED OKAY
 Removing goal: #{Goal 14}
Considering goal (PULL-IN-ARM ARM2) :: #{Goal 26} :NEW
 Goal accomplished: (PULL-IN-ARM ARM2) :: #{Goal 26}
  with result: SUCCEED ()
 Removing goal: #{Goal 26}
```

3.4.2 The University of Chicago robotic platform

The agent architecture is currently being tested on the University of Chicago Animate Agent robotic platform. This testbed provides for many of the same considerations as the overall logistics project: the need for flexible, real-time response, and the need to adjust to longer-term changes in the operating environment.

There are four hardware and software levels to this platform:

1. High-level agency software.
2. Low-level reactive planning software.
3. Control firmware.

4. Sensor and effector hardware.

Robot sensors include a color camera, microphone, and multiple infrared and sonar sensors. Effectors include a mobile base, pan and tilt camera controls, swivel motors for the sonars, and a speech synthesizer. We hope to add a robotic arm in the near future. Direct control of the hardware is handled by real-time control procedures. Vision processing requirements are met by additional hardware and software; visual results are sent to the low-level reactive planning software.

The University of Chicago robotic platform has taken part in AAAI Robot Competitions, and appeared on the cover of *AI Magazine*.

3.4.3 Uniform representation

The central issue in using case-based reasoning as a *lingua franca* is the uniform representation of case-based knowledge and the information necessary for specialized technologies.

In the case of the DMAP/RAP agent architecture, that specialized technology is the RAP system. This section presents a catalog of some of our "cases" of providing such an interface.

Here, for example, is a DMAP representation of the **move-arm** RAP.[2]

```
(DEFINE-NODE move-arm (ISA rap)
  (SLOTS (the-arm arm)
         (the-destination location)
         (the-success at-location
           (WHERE (the-object = the-arm)
                  (the-location =
                   the-destination)))))
```

Note that this definition of **move-arm** does not include context checks or method descriptions. Instead, the DMAP representation makes use of the hierarchical structure of memory to establish more specific instances of **move-arm** with the appropriate information.[3]

```
(DEFINE-NODE move-arm-internal (ISA move-arm)
  (SLOTS (the-destination location/truck-internal)
         (the-method prog2
           (SLOTS (the-1st fold-arm
                    (WHERE (the-arm = the-arm)))
                  (the-2nd primitive-move-arm
                    (WHERE (the-arm = the-arm)
                           (the-destination =
                            the-destination)))))))
```

[2]The system uses a hierarchical representation of memory. For clarity, all keywords are in uppercase, and all relations are of the form "the-". All other symbols are memory node names. The appearance of "=" indicates a variable binding constraint.

[3]The following definition omits some aspects of **move-arm**, *i.e.*, **the-arm** and **the-success**. Constraints and relationships are inherited, so there is no need to respecify identical information.

This more specific version of **move-arm** represents the fact that the arm must first be folded before it can be moved inside the truck. The memory nodes **fold-arm** and **primitive-move-arm** are other **raps**, siblings of **move-arm**.

```
(DEFINE-NODE fold-arm (ISA rap) ...)

(DEFINE-NODE primitive-move-arm
  (ISA primitive-rap) ...)

(DEFINE-NODE primitive-rap (ISA rap) ...)
```

(Primitive **raps** invoke the true low-level RAP interpreter, usually resulting in subsequent changes to the robot control routines and possible sensor or effector actions.)

Method specifications. The method specification for **move-arm-internal** was done by specifying constraints on the **prog2** memory structure. This general structure represents the concept of taking two successive actions.

```
(DEFINE-NODE prog2 (ISA method)
  (SLOTS (the-1st rap)
         (the-2nd rap))
  (INDEX (the-1st) (the-2nd)))
```

The structure of **prog2** is very simple, having as it does two substructures, both of which are **raps**. The fact that the first **rap** should come before the second **rap** is indicated by the presence of an index annotation to the DMAP node description.

The index annotation tells the DMAP system how it can recognize that the method has been achieved. The elements of the index indicate that the DMAP system must first recognize that "**the-1st**" has been achieved, and then recognize that "**the-2nd**" has been achieved. At that point, the DMAP system will be able to conclude, by virtue of the index annotation, that the **prog2** has been achieved.

Context checks. Note that **move-arm-internal** still does not specify a context check. This is because context checks as implemented in the RAP system are not necessary. The DMAP system relies on the structure of memory to perform context checks automatically. In this case, it is the specification of **location/truck-internal** as **the-destination** for **move-arm-internal** that results in the correct choice of method. The general transformation of RAP methods to DMAP memory structures is illustrated below.

RAP representation

```
(define-rap name
  (success success)

  (method                ; method-1
    (context context-1)
    (task-net task-net-1))
```

```
  (method                ; method-2
    (context context-2)
    (task-net task-net-2)))
```

DMAP representation

```
(DEFINE-NODE name (ISA rap)
  (SLOTS (the-success success)))

(DEFINE-NODE method-1 (ISA name)
  (SLOTS context-1
         (the-method task-net-1)))

(DEFINE-NODE method-2 (ISA name)
  (SLOTS context-2
         (the-method task-net-2)))
```

The role played by the context checks in the RAP system is now performed by specialization in the hierarchical memory of the DMAP system. The context checks of the RAP system are translated into slot specification in the DMAP system; for example, in **move-arm-internal**, the RAP context check **(internal ?loc)** becomes a more specific filler for the relation **the-destination**. Multiple methods become multiple sibling descendants of the more general **rap** structure. The DMAP system relies upon its algorithm for concept refinement within the memory hierarchy to perform method disambiguation.

Success tests. The definition of **move-arm** specified a **the-success** relation, which was subsequently inherited by **move-arm-internal**. These relations correspond directly to the success checks of the original RAP system. At the highest level, every **rap** memory structure has a success test and a method specification.

```
(DEFINE-NODE rap (ISA mobject)
  (SLOTS (the-success state)
         (the-method method))
  (INDEX (the-success))
  (INDEX (the-method) (the-success)))
```

(The **mobject** node is the highest level of the memory hierarchy, and exists primarily as a reference point for the general specification of communicative acts.)

The general **rap** node serves as a reference point for two important indices. These represent how the successful execution of a **rap** can be recognized, as in the previously described annotation of **method** achievement. The RAP system specifies that execution is complete when the success test is satisfied; even when the methods are executed, the success test must still be checked upon method completion to determine successful execution.

Similarly, in the DMAP system these two indices represent **rap** success. There are two possibilities:

1. the success condition of the **rap** (the **state** that exists in the **the-success** relation) is recognized, or

2. achievement of the method associated with the rap (the method that exists in the the-method relation) is recognized, followed by recognition of the success condition as in (1).

In either case, the success condition must be successfully recognized in order to qualify as successful execution of the rap.

These two indices are inherited by all descendants in the hierarchy: for example, move-arm-internal can be recognized by its success condition (inherited from move-arm), or by its method followed by its success condition.

Taking action. Although the previous discussion has been in terms of recognition, it is in attempting to recognize the successful achievement of methods that the DMAP system ends up taking action. One way to think about this is to take the prescriptive statement, "DMAP should now do a move-arm-internal." and recast it as a description of a state of affairs that DMAP should recognize, that is, "try to recognize the successful achievement of move-arm-internal."

How can that recognition be achieved? One way is by doing (recognizing) the achievement of the-method. Ultimately, the decomposition of tasks will be grounded in the attempt to recognize a primitive action such as primitive-move-arm. These recognition attempts result in calls to the lower-level RAP interpreter.

The second of the two indices associated with the general rap structure, then, is exactly that used to prompt task decomposition and the ultimate execution of primitive tasks in lower levels of the overall robotic platform.

3.4.4 Uniform representation for natural language

Once planning knowledge has been encoded in DMAP memory structures, it is straightforward to extend the natural language capacities in order to enable human users to interact with the system. For details about natural language understanding in DMAP, see (Martin, 1990). Here we will only demonstrate how references to planning structures can be recognized and disambiguated.

The same memory hierarchy used to represent planning knowledge is also used to represent the knowledge necessary for language understanding. This is necessary since it is exactly this planning knowledge which turns out to be essential to understanding many natural language references. For example, the communication "don't do that" can only be understood in reference to the current planning tasks of the robot.

3.4.5 Communicative actions

The highest level of the language hierarchy in the DMAP system consists of the mtrans marker for communicative actions (Schank and Abelson, 1977).

```
(DEFINE-NODE mtrans (ISA action)
  (SLOTS (the-info mobject)))
```

This structure has many descendents, since it is the attachment point for the linguistic knowledge associated with memory structures. For example, the **move-arm** rap might be referred to using the following structure.

```
(DEFINE-NODE m/move-arm (ISA mtrans)
  (SLOTS (the-info move-arm))
  (INDEX move (the-info the-arm)
         to (the-info the-destination)))
```

In this example, the **the-info** relation has been specialized the **move-arm** structure, and the structure as an index that gives a linguistic pattern that may refer to **move-arm**. The linguistic information is specified in a form much like that of a phrasal lexicon; a kind of template in which the variable elements are composed of memory relations.

In this case, the index begins with the word "move," followed by a memory reference to the **the-arm** of the **the-info** of the current structure. Chasing the relationships in the memory hierarchy, this reference results in the identification of the **arm** structure associated with the particular **move-arm**.

The situation is considerably more complex than we have indicated here, since the reference to the **arm** structure must also be by way of a communicative act. The definition of **m/move-arm** given above is therefore incorrect; a more correct version is the following:

```
(DEFINE-NODE m/move-arm (ISA mtrans)
  (SLOTS (the-info move-arm)
         (m-arm mtrans
           (WHERE (the-info = the-info the-arm)))
         (m-loc mtrans
           (WHERE (the-info =
                   the-info the-destination))))
  (INDEX move (m-arm) to (m-loc)))
```

Careful examination of this definition will reveal that it specified two sub-communicative acts as part of **m/move-arm**, each concerned with communicating a sub-structure of the **move-arm** structure. This allows the DMAP system to correctly resolve these sub-communciative acts with other **mtrans** structures, such as the following which specifies that the word "arm" may be used to refer to a robot arm.

```
(DEFINE-NODE m/arm (ISA mtrans)
  (SLOTS (the-info arm))
  (INDEX arm))
```

If there is more than one arm, of course, the DMAP algorithm will have to disambiguate between them.

The second specification of **m/move-arm** is correct, but is for many reasons not the best choice of representation. Efficiently handling issues such as syntax require more complex solutions, none of which are directly relevant to the subject of this paper. In fact, for simplicity we will specify **mtrans** structures as in the first (incorrect) definition of **m/move-arm**, but use the keyword "**MTRANS**" instead of "**INDEX**" as syntactic sugaring. In this paper we will not attempt to describe how syntactic concerns (subject-verb agreement, number agreement, etc.) are taken into account.

3.4.6 Referring to methods

The kind of interactions we would like to have with our robotic assistants include instructions such as "fold your arm before moving it inside the bay." This might be used to *teach* the robot the **move-arm-internal** structure. The previous **rap** definitions impose the following abstraction structure:

```
(DEFINE-NODE rap (ISA mobject)
  (SLOTS (the-method method) ...) ...)

(DEFINE-NODE move-arm (ISA rap) ...)

(DEFINE-NODE move-arm-internal (ISA move-arm)
  (SLOTS (the-method prog2 ...) ...) ...)
```

In order to be instructed in **move-arm-internal**, the system must already know about the successive execution of two tasks. That is, it must know about **rap** structures whose methods are **prog2** structures:

```
(DEFINE-NODE rap/prog2 (ISA rap)
  (SLOTS (the-method prog2)))
```

This structure is more general than **move-arm-internal**, unrelated (but compatible) with **move-arm**, and more specific that **rap**. The addition of this structure turns the above abstraction structure into a lattice, since **move-arm-internal** will inherit from both **move-arm** and **rap/prog2**. Multiple inheritance is a critical aspect of the DMAP implementation.

The **rap/prog2** structure is the correct referent for the above human-robot interaction. We can define a communicative act to make that reference explicit in the DMAP memory.

```
(DEFINE-NODE m/rap/prog2 (ISA mtrans)
  (SLOTS (the-info rap/prog2))
  (MTRANS (the-info the-method the-1st)
          before
          (the-info the-method the-2nd)))
```

The instruction "fold your arm before moving it inside the bay" has thus been reduced to the phrasal pattern "*action-1* before *action-2*." Recognition of "fold your arm" and "moving it inside the bay" must be the responsibility of the **mtrans** structures associated with **fold-arm** and **primitive-move-arm**.

3.5 Case-based interfaces for large-scale planning

This part of the report has presented many details of the interaction between a case-based reasoner and a specialized technology. We think this was important in order to communicate our vision of case-based reasoning as the *lingua franca* of large-scale planning.

What we have shown is that it is possible to take a technology—reactive execution, in this case—which is unrelated to case-based reasoning in many fundamental ways, and map it into the knowledge structures of a case-based reasoner.

We believe the same can be done for the many technologies that make up the large-scale planning initiative. When brought together under the aegis of case-based reasoning, these technologies can present a common, intuitive, *case-based* face to the user while maintaining their own efficient, special-purpose technological advantages.

4 Planning and Execution

Our work in planning and execution is based primarily in CBR (1), but also incorporates ideas from relatively new areas of AI such as active vision and situated activity as well as more established technologies associated with non-linear planning and reason maintenance. In all of the systems that follow, the CBR core is used as the *lingua franca* that allows the collections of somewhat disparate planning and execution components to to be used together within complete systems.

- In ROENTGEN, the CBR system acts as a intelligent buffer between the user and the details of dose distributions, data retrieval, and plan modification. This project examines the development of new indexing and repair techniques for traditional planners.
- In RUNNER, the CBR architecture enables the integration of high-level declarative plans with low-level reactive opportunism. This project examines the development of execution, modification, and repair techniques for real-time planning and execution systems.
- In SHOPPER, the same architecture coordinates the abstract search plans of the agent with a variety of knowledge sources, both internal and external to the system. This project examines the development of an integrated model of planning and sensing for real-time execution.

All of these systems are hybrids in which the technologies best suited for specific tasks are integrated by the CBR architectures at their cores. The following three sections detail each research project in turn.

4.1 Roentgen

ROENGTEN is a case-based system for planning radiation therapy in the treatment of cancer. The system supports treatment planning by retrieving and suggesting relevant plans from memory, adapting them to fit the details of new cases, and evaluating potential solutions and then repairing any problems that are discovered.

The basic architecture of our earlier work on CHEF(Hammond, 1989a) is now being applied in the ROENTGEN project to the more demanding domain of planning radiation therapy for cancer. As with CHEF, ROENTGEN plans from a memory of actual cases—past successes and failures in treatment planning—to develop plans for new cases. As new problems are presented to it (Figure 2), ROENTGEN retrieves similar, successful cases from memory and then uses them as suggestions

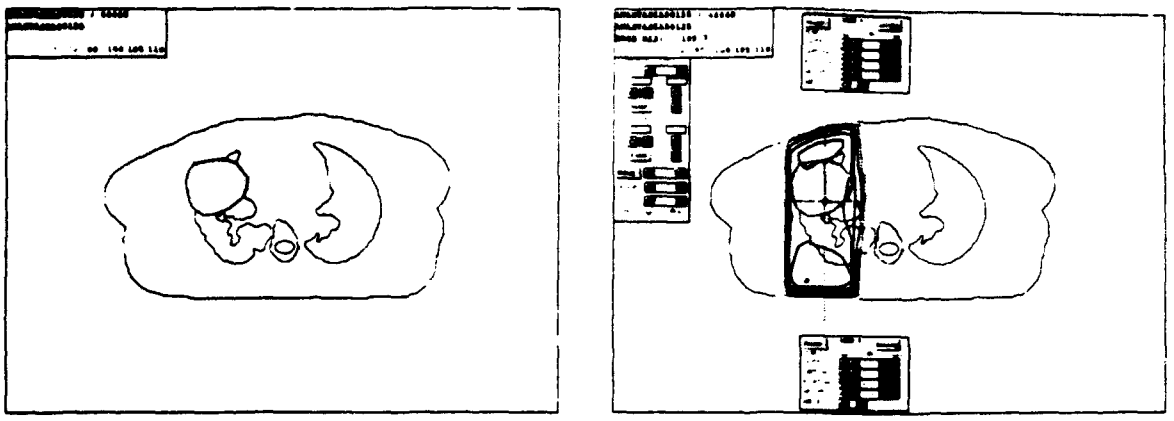


Figure 2: A Roentgen Problem Description and Resulting Plan

to help formulate its first attempt at a plan. It performs standard modifications and then uses a radiation dose calculator to compute the dose distribution over an entire body cross section (Figure 2). One of the innovations in ROENTGEN is our development of an abstracted visual vocabulary for use in indexing and retrieval of existing cases (Figure 3).

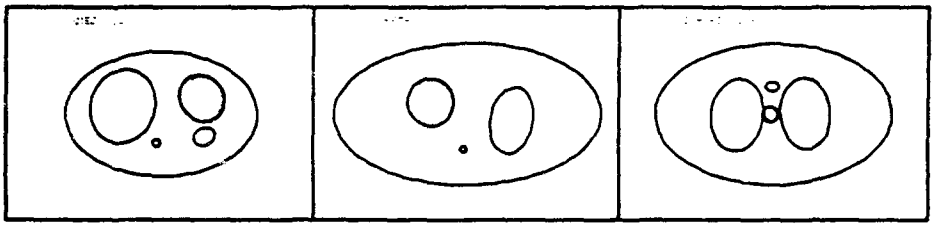


Figure 3: Visual representations of retrieved cases

For the past three years, we have been working on ROENTGEN, a demonstration of the feasibility of case-based radiation therapy planning(Berger *et al.*, 1990; Berger and Hammond, 1991; Berger, 1992; Berger, 1993; Berger, 1994). ROENTGEN can support a human therapy planner (a dosimetrist) or autonomously design therapy plans for cancer patients. The system can suggest first-approximation therapy plans for new patients, detect and suggest repairs, and produce finished plans subject to human evaluation, all for patients with cancer of the thorax.

4.1.1 Planning Radiation Therapy for Cancer Patients

Dosimetrists in oncology clinics are responsible for designing satisfactory therapy plans for cancer patients. In external photon beam therapy, their job is to find an arrangement of high energy photon beams which delivers a prescribed dose to a "target" region in the patient's body while ensuring that the doses to radiation-sensitive tissues in the body are below those tissues' tolerances and while sparing unnecessary dose to other healthy tissue generally. They must take into account any previous doses to normal tissue that have been delivered by earlier stages in radiotherapy when constructing the therapy plan. The important elements in the definition of the problem they are to

solve consist of a two-dimensional geometric cross section of the patient passing through the center of the target (see Figure 4), the dose prescribed by the physician, and any previous doses that have been administered to the target and normal tissues by previous stages.

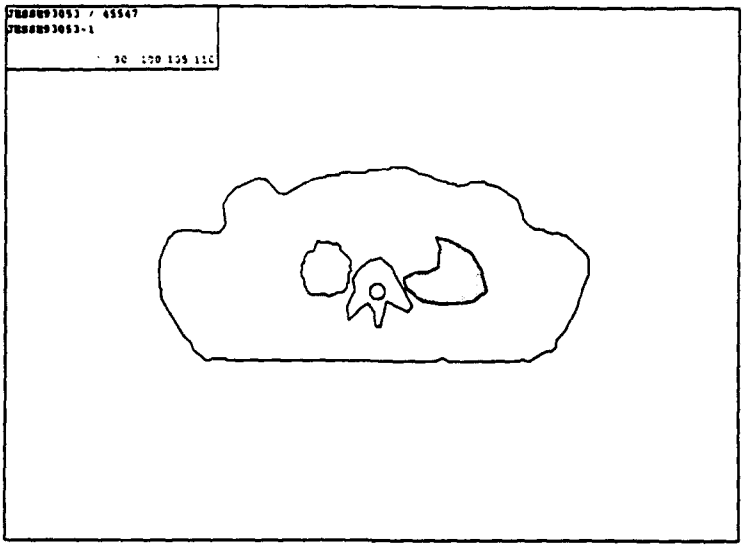


Figure 4: A treatment cross section in the upper thorax with target shaded.

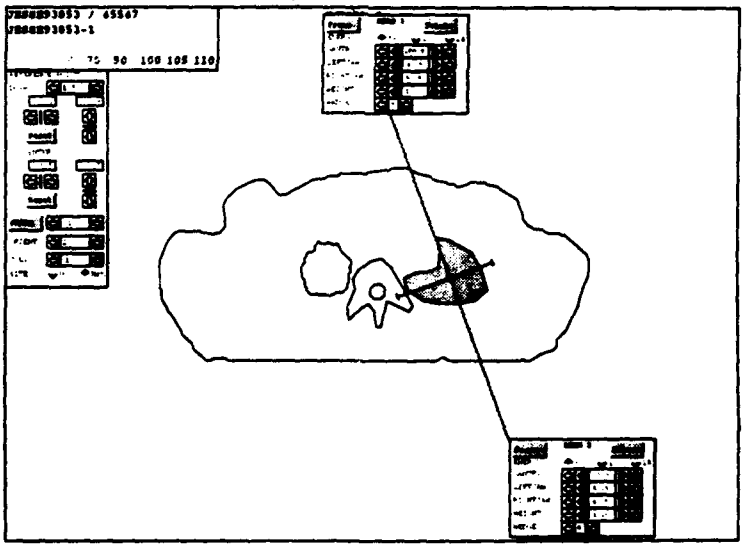


Figure 5: The plan for the patient JESSE93053-1 from the RT plan case-base.

Based on this information, dosimetrists develop therapy plans using a two step process. The first step is to arrive at a first-approximation therapy plan for the patient. This plan is a rough attempt to satisfy the treatment goals. When the dosimetrist looks at the simulated results of the first-approximation plan, she is almost certain to find flaws: underdosed target tissue; overdosed

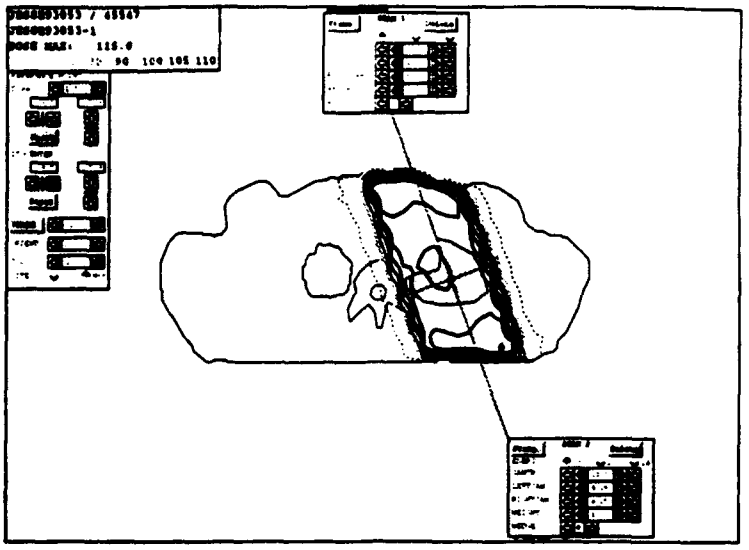


Figure 6: The results of the plan for the patient JESSE93053-1 from the RT plan case-base. Nested isodose contours show the level of dose delivered by the plan.

sensitive tissues; or other problems. This fact of therapy planning life naturally leads to the second stage of plan development, the iterative repair of the first-approximation plan. (Figures 5 and 6 illustrate a good plan and its resulting dose distribution.)

To help them carry out the first step in therapy planning, clinic dosimetrists frequently maintain notebooks that contain past therapy cases they have planned. When designing a plan for a new patient, the dosimetrist will often look in the notebook in order to find suggestions for a first-approximation plan for a new patient. She will seek a past case that is similar to the new patient in terms of the geometry of the patient cross section and the dose requirements (the prescribed dose to the target and dose limits to the sensitive tissues). In general, the closer the similarity, the fewer flaws the first-approximation plan is likely to exhibit; since, for a given therapy plan, the patient geometry absolutely determines the resulting dose distribution. The dosimetrist then uses the dose requirements to see how well the plan performs. Hence, both geometry and dose requirements are important in finding a good match to the current case.

4.1.2 Case-Based Therapy Planning

The naturally ocurring case-based reasoning of the dosimetrists provided part of the impetus for attempting to build ROENTGEN, a case-based reasoning *program* for therapy planning. ROENTGEN also employs the two-stage process used by the dosimetrists. It has a case-base of past therapy plans from which it develops its first-approximation therapy plan for a new patient. And, it has a second case-base of repair episodes that record the corrections made to past therapy plans to eliminate or reduce unwanted conditions in plan results. ROENTGEN uses the second case-base to support the repair of plans currently being developed. Figure 7 gives a system diagram for ROENTGEN. The *Retriever* and *Adapter* together are responsible for producing the first-approximation plan. The *Evaluator* and *Repairer* for repairing the suggested plan to produce an acceptable result.

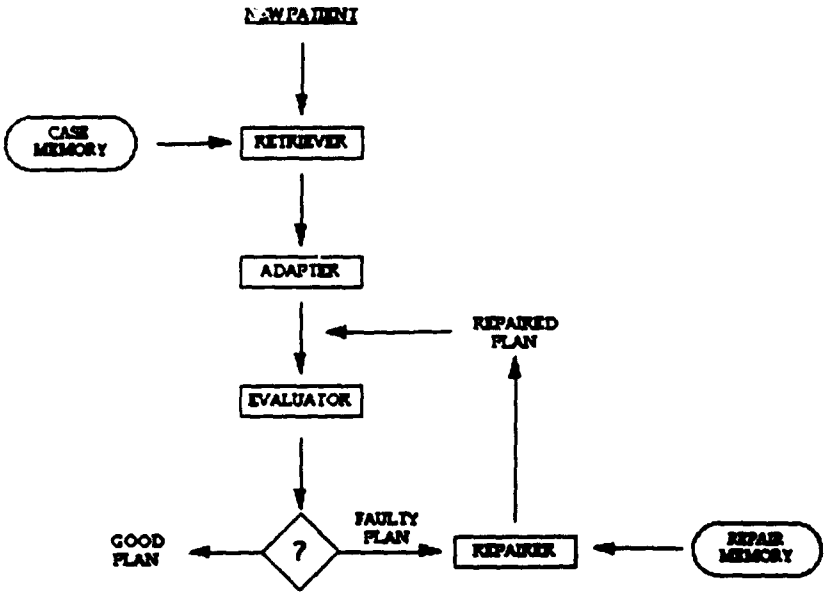


Figure 7: The ROENTGEN System

4.1.3 The *Retriever*

The *Retriever* is the ROENTGEN module that is responsible for finding past cases in case-memory that can be used as the basis for first-approximation therapy plans. Like the dosimetrist, the *Retriever* looks for cases that are geometrically similar to the current patient while keeping in mind the current dose requirements. The *Retriever* compares a description of the important geometric features of the new patient with descriptions of the important geometric features of the cases in case-memory and returns the cases that match the best and which do the best job of achieving the dose requirements of the current patient.

To carry out geometric matching, we have developed a vocabulary of features based on elliptical approximations to the tissue shapes in the cross section of the patient. For each of the tissues of interest—target, spinal cord, lungs—the system approximates the tissue and then computes the following associated parameters:

- **area.** The area of the polygon/ellipse. For the body outline, this is the absolute area in square centimeters; for the other tissues it is the area as a proportion of the total body area.
- **eccentricity.** This feature parallels the subjective perception of how "elongated" the corresponding tissue is. Ellipses with an eccentricity of 0 are circles; those with an eccentricity of 1 are line segments.
- **orientation.** This feature is associated with the subjective perception of the direction in which a tissue points. It is the angle formed by the major axis of the approximating ellipse with the X-axis in radians.

- **rho.** The distance from the target centroid (center of gravity) to the centroid of the polygon/ellipse of the tissue. This is also the first polar coordinate of the tissue centroid. This distance is normalized by the posterior to anterior extent of the body outline.

- **theta.** The angle formed by the vector from the target centroid to the tissue centroid with the positive X-axis (radians). This is the second polar coordinate of the tissue centroid.

The *Retriever* uses these features to find geometric matches to the current patient among its past cases. Among those cases which match geometrically, the *Retriever* seeks cases whose plans are likely to satisfy the dose reqirements—the prescribed dose to the target, and the limiting doses permitted to sensitive vital tissues—of the current patient.

The *Retriever* estimates how well plans in its memory are likely to satisfy these limits by looking at information stored with each case that summarizes the dose distribution in the case. This summary information is in the form of dose-area histograms for each important tissue in the cross section. For example, if the current requirement is that the lung be kept below 50 percent of the current dose prescribed for the target, the *Retriever*, when considering a case from memory checks to see what fraction of the lung was at or below that level when the plan from the case was applied. The fractions for each tissue are added to give an overall "satisfaction" score for the case. This satisfaction score provides an estimate of how well the plan from the case under consideration satisfies the current requirements. The higher the score, the better the requirements are (theoretically) met. The *Retriever* favors cases which are geometrically similar to the current patient and which have a high satisfaction score.

4.1.4 The *Adapter*

The *Adapter* takes the case returned by the *Retriever* and tailors the therapy plan it contains to the basic geometric facts of the current patient. No two patients are exactly alike. The location and size of the target in one may differ by only a couple of centimeters from the location and size in another, but the plan from the first will have to be adjusted to account for those centimeters if it is to be used on the second.

The *Adapter* also takes account of symmetry when tailoring plans. If the targets in the retrieved case and the current patient are on different sides, the *Adapter* will correct the retrieved plan accordingly. It will also adjust the retrieved plan for a prone patient to one that can be used on a supine patient and *vice versa*.

The *Adapter* adjusts five types of plan parameters when tailoring a retrieved plan for a new patient: 1) the coordinates of the isocenter; 2) the coordinates of the reference point; 3) the gantry angles for the beams in the plan; 4) the collimator jaw settings for each beam; and 5) the orientation of any wedges used in the plan.

Thus the *Adapter* tailors the output of the *Retriever* to produce a first-approximation plan for a new patient.

4.1.5 The *Evaluator*

The *Evaluator* has the job of detecting flaws in the results of the proposed RT plan. The module produces a list of flaws (and possibly relevant facts about the plan) that can be used by the *Repairer* or dosimetrist as problems to overcome through RT plan repair. The *Evaluator* can detect the following sorts of flaws:

- Underdosed target tissue. Some area of the target may be receiving less than the prescribed dose. This is a serious flaw especially if the degree of underdose or the area of underdose are significant.
- Overdosed vital tissue. A plan should not dose a tissue that is vital to the life or well-being of the patient beyond its dose tolerance. In the thorax, the lungs and the spinal cord are radiation-sensitive. Overdosing the lungs can lead to a loss of pulmonary capacity. Serious loss can lead to death. Overdosing the spinal cord will cause the nerve fibers to stop functioning resulting in motor and sensory deficits below the point of injury.
- High or unnecessary dose to normal tissue. All tissues in the body suffer damage and some loss of function if subject to high enough levels of radiation. Tissue should not be dosed unless necessary.
- High dose close to a sensitive structure. If an area of high dose lies within a centimeter or two of the spinal cord, a small error in positioning the patient on the treatment table can result in accidentally delivering a damaging dose to it.
- Uneven dose to the target. A rule of thumb in the clinic is that the dose in the target region should not vary by more than 10%.
- High dose maximum. It is desirable that the dose not exceed 110% of the prescription anywhere in the cross section. The absolute limit is often set at 115%.
- Significant dose to normal tissue. Since radiation is potentially harmful to normal tissue, significant dose to any structure—even though it is below the tissue's tolerance—should be noticed so it can be avoided when possible.

In addition to describing the nature of the flaw, the *Evaluator* also specifies its location. For example, "Overdosed2 Cord PA-beam cord-side-outershoulder-entry" indicates that a part of the spinal cord has been dosed at between 110 and 140 percent of its tolerance. The overdosed region lies in the entry portion of the outer-shoulder of the Posterior to Anterior beam.

4.1.6 The *Repairer*

The *Repairer* uses the list of flaws produced by the *Evaluator* (and possibly modified by the dosimetrist) to transform the first-approximation therapy plan into an acceptable one for treatment. It is a case-based system in its own right and finds the closest match between the current situation and past situations for which it has repair plans in its memory. Current and past situations are characterized by: 1) the flaws of concern; 2) the type of RT plan in use; and 3) the

patient geometry and dose requirements of the patient. During retrieval, the *Repairer* measures the suitability of each past repair episode in its memory by considering:

- The match between the set of flaws of current concern and the set of flaws which the past repair episode aimed at correcting.
- The match between the current RT plan and the RT plan of the repair episode it is considering.
- The similarity between the patient geometries and dose requirements of the current and past situations.

The *Repairer* applies the best match from the past episodes to the current flawed RT plan. It does this by executing each step in the retrieved repair plan. Each step is a "tweak" that performs an adjustment to the therapy plan like raising or lowering beam energy, rotating a beam source to a slightly differnt position, opening or closing the collimator jaws which control the size of the beam cross section. More complex tweaks can be constucted by the user by combining predefined predicates which test for the existence of important conditions in the dose distribution and previously defined tweaks using four standard programming constructs: IF, UNLESS, BLOCK, and UNTIL. These more complex tweaks can perform higher level repair actions like rotating a beam until it no longer impacts the spinal cord.

Tweaks are the building blocks for the repair plans which make up the *Repairer*'s memory. And this memory is the key to the *Repairer*'s ability to create acceptable RT plans from the first-approximation plans produced by the *Retriever* and *Adapter*.

4.1.7 Status of ROENTGEN

ROENTGEN consists of code and data. The code for the system includes that for the four modules described in the previous section. The data represents the system's knowledge in the form of case memories.

ROENTGEN is built around an existing archive of 165 therapy plans created by dosimetrists at a hospital oncology clinic for treating patients with thoracic lesions. These plans and the patients for whom they were designed comprise the case-base used by the *Retriever* as described above. The existing archive also provided a development problem set. We chose 15 cases from the archive by an arbitrary process (the first 15 in alphabetical order of patient name for whom we could find the hard-copy clinic records) that should give us a representative sample. This set of problems was used to tune the *Retriever* so it produced reasonable first-approximation plans for the patients in it. In addition, as therapy plans were produced for the problems in the development problem set, the repair episodes necessary to correct the flaws in the first-approximation plans were stored in the repair plan case-base for future use.

Both the code and data portions of ROENTGEN are complete. The current focus of work on the project is producing an account of the research in the form of a Phd dissertation by Jeff Berger. This is to be completed by August, 1994.

4.1.8 Evaluation

ROENTGEN clearly demonstrates the feasibility of case-based radiation therapy planning. While we have not performed a formal clinical evaluation of its capabilities, it gives every indication of being up to the task of supporting the efforts of human dosimetrists and of designing reasonable therapy plans on its own. We are planning to do a formal evaluation in the future. The evaluation will involve submitting a representative set of new (to ROENTGEN) therapy problems to: 1) clinic dosimetrists, both expert and novice; 2) to ROENTGEN; and 3) clinic dosimetrists who use ROENTGEN as an aide to the planning process. A physician will evaluate the plans produced, without knowledge of the source of the plans. We will compare the performance of ROENTGEN to aided and unaided human dosimetrists of different levels of experience.

4.2 Large Scale Transportation Planning

While ROENTGEN is not a transportation planning system, the overall design of the system and the leasons learned are directly applicable to crucial aspects of the transportation problem. In particular, the issues of integration dealt with by ROENTGEN are the same issues that are faced in transportation planning when using tools such as DART, where the feedback to a user is primarily numeric. As in IOPSthe ROENTGEN solution of using qualitative descriptions of known configuations of problems as indicies to known solutions, is exactly what is needed in the transportation planning domain. Likewise, the integrated use of specific cases along side more general repair rules is also applicable to the transportation planning domain.

In general, the ROENTGEN research stands as an example of how CBR can be used as a *lingua franca* for large-scale planning.

4.3 Runner

RUNNER is a system that uses plans in a commonsense domain (a simulated kitchen). As a research project it lies between traditional planning research and more recent work on situated and reactive systems, and draws from both areas.

Traditional planning systems construct action sequences (plans) that, when executed, will satisfy the goals given to the system. These systems need complete symbolic descriptions of the state of the world in order to construct the plans, and generally assume an exact model of the effects of actions. This sort of plan construction can be very computationally expensive, and is probably of limited utility in uncertain worlds.

More recent work (by Brooks, Chapman and Agre, and Rosenschein and Kaelbling) responds to the uncertainty of execution and the cost of plan construction by tying current action as directly as possible to current perception, maintaining little state over time(Agre, 1988; Agre and Chapman, 1987). While they avoid the intractabilities of planning, these systems rely on delicate characterizations of agent-environment interactions by the designer, as well as on the assumption that everything needed to determine action is perceptually immediate. While robust to environmental uncertainty, these systems are brittle to patterns of interaction with the environment that have not been explicitly anticipated by the designer (*e.g.* problems such as looping, or becoming trapped

in local maxima). Finally, since there is little declarative representation of knowledge in these architectures, it is difficult to see how to integrate anything but the most knowledge-poor kinds of learning within them. Reinforcement learning is the main technique that has been looked at within this framework (e.g. Brooks and Maes, Chapman and Kaelbling, Sutton), but there is good reason to believe that pure reinforcement learning suffers from combinatoric problems analogous to those that make classical planning intractable.

We draw from both traditions in the RUNNER project: We agree with the "situated" camp that it is not reasonable to assume that an action system has a complete symbolic model of the world before any action takes place, or that the behavior of agents can be well characterized without some characterization of the environments they are to inhabit. We also agree that traditional planning is too expensive to be invoked every time an agent wants to achieve a goal. But we argue for the utility of explicit symbolic representations of plans, largely because this sort of representation provides the information necessary for run-time recovery and repair, as well as for learning by incremental modification of those plans. These plans should be re-used as much as possible (thereby amortizing the cost of planning over time), so that routine action is computationally cheap.(Converse and Hammond, 1992) Nevertheless, the structure and causal dependencies of plans should be represented explicitly, so that problems can be diagnosed and repaired.

4.3.1 Research issues

The central research issues in the RUNNER project are:

- Flexible plan execution, with learning from execution-time failures and unexpected opportunities.
- "Extrinsic" planning—recognizing in the midst of activity when it is appropriate to use a stored plan (as opposed to reasoning about the internals of the plan).
- The representation of plans to permit effective reuse in a changing environment.
- Stabilization of the environment by the enforcement of policies.

Of these, probably only the last requires explanation: the idea here is that plans to be reused need certain preconditions to be re-established, and also that the problem of plan reuse is simplified by ensuring that certain of these preconditions are "always true". This is an abstract phrasing of a simple idea: for example, the fact that most people make sure that they always have clean drinking glasses in a particular kitchen cupboard probably makes it much easier for them to decide what to to do when they decide they want to have a drink of water. This sort of "stabilization" of the immediate environment trades off with the need for flexible plan representation—the more standardized an environment is with respect to the preconditions of a plan, the less flexibility is needed in the plan's execution(see (Hammond and Converse, 1991; Hammond *et al.*, 1994).

4.3.2 Project domain

The RUNNER program interacts with a simulator, which provides for it the illusion of inhabiting a conventional kitchen. Although a simulation, it is a fine-grained one—the RUNNER program controls a "head" with a visual focus and a limited field of view, and "hands" which must be directed by visual attention to manipulate objects in the field of view. We hope to make the simulator available to other research labs soon.

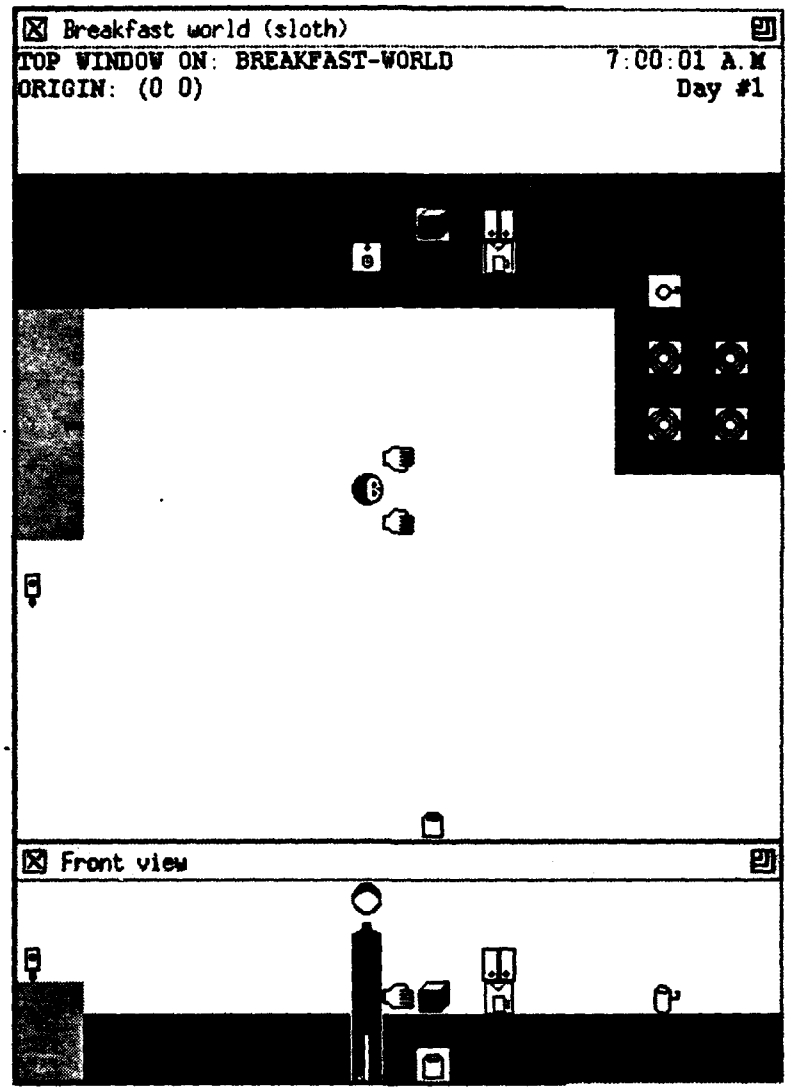


Figure 8: Runner's World

In this simulated world, RUNNER must perform simple tasks like making coffee and breakfast cereal. RUNNER starts out with a small set of hand-coded plans. Its job is to use these plans (and interleave their actions when necessary) to actually achieve the goals in its world. These plans represent the causal dependencies of their different parts, but do not completely determine the action of the agent: ordering of steps can depend on discovered opportunities, steps from different plans can

be interleaved, and the appropriate times for particular plans must be recognized on the basis of perceptual cues.

4.3.3 Memory and Agency

Our model of planning and understanding rises out of three pieces of work: Schank's structural model of memory organization(Schank, 1982), our own work in case-based planning and dependency directed repair (Hammond, 1989a; Hammond, 1990; Hammond and Seifert, 1993; Hammond and Seifert, 1992; Seifert *et al.*, 1994), and the work of Martin and Riesbeck in Direct Memory Access Parsing (Martin, 1989). Our model has been articulated in two programs, TRUCKER and RUNNER(Hammond *et al.*, 1988; Hammond *et al.*, 1990)

The model was first developed to deal with the problem of recognizing execution-time opportunities in the context of a resource-bound agent that is forced to suspend planning in order to attend to execution (Hammond, 1989b; Hammond and Seifert, 1994; Hammond *et al.*, 1993b). The goal of this model was to capture the ability of an agent to suspend goals, yet still recognize execution-time opportunities to satisfy them.

To accomplish this goal, we use a single set of memory structures both to store suspended goals and to understand the agent's circumstances in the world. In response to a blocked goal, an agent's first step is to do a planning-time analysis of the conditions that would favor the satisfaction of the goal. The agent then *suspends* the goal in memory, indexed by a description of those conditions. For example, a goal to buy eggs that was blocked during planning would be placed in memory associated with the condition of the agent being at a grocery store.

During execution, the agent performs an ongoing "parse" of the world in order to recognize conditions for action execution. Following DMAP (Martin, 1989), this parse takes the form of passing markers through an existing episodic memory. Because suspended goals are indexed in the memory used for understanding the world, the goals are activated when the conditions favoring their execution are recognized. Once active, the goals are then reevaluated in terms of the new conditions. Either they fit into the current flow of execution or they are again suspended.

We called the initial model *opportunistic memory* because the agent's recognition of opportunities depends on the nature of its episodic memory structures. Having turned to the broader issues of integrating planning and action, we now refer to our work as the study of *agency.*

We use the term *agency* to comprise the spawning of goals, selection of plans, and execution of actions. Our process model of agency is based on Martin's DMAP understander as well as its antecedent, Schank's *Dynamic Memory.* DMAP uses a memory organization defined by part/whole and abstraction relationships. Activations from environmentally supplied features are passed up through abstraction links and predictions are passed down through the parts of partially active concepts. Subject to some constraints, when a concept has only some of its parts active, it sends predictions down its other parts. When activations meet existing predictions, the node on which they meet becomes active. Finally, when all of the parts of a concept are activated, the concept itself is activated.

To accommodate action, we have adued the notion of PERMISSIONS. PERMISSIONS are handed down the parts of plans to their actions. The only actions that can be executed are those that

are PERMITTED by the activation of existing plans. Following McDermott (McDermott, 1978), we have also added POLICIES. POLICIES are statements of ongoing goals of the agent. Sometimes these take the form of maintenance goals, such as "Glasses should be in the cupboard." or "Always have money on hand." The only goals that are actively pursued are those generated out of the interaction between POLICIES and environmental features. We would argue that this is, in fact, the only way in which goals can be generated.

Most of the processing takes the form of recognizing circumstances in the external world as well as the policies, goals and plans of the agent. The recognition is then translated into action through the mediation of PERMISSIONS that are passed to physical as well as mental actions.

Goals, plans, and actions interact as follows:

- Features in the environment interact with POLICIES to spawn goals.

  For example, in RUNNER, the specific goal to HAVE COFFEE is generated when the system recognizes that it is morning. The goal itself rises out of the recognition of this state of affairs in combination with the fact that there is a policy in place to have coffee at certain times of the day.

- Goals and environmental features combine to activate plans already in memory.

  Any new MAKE-COFFEE plan is simply the activation of the sequence of actions associated with the existing MAKE-COFFEE plan in memory. It is recalled by RUNNER when the HAVE-COFFEE goal is active and the system recognizes that it is at home.

- Actions are **permitted** by *plans and are associated with* the descriptions of the world states appropriate to their performance. Once a set of features has an action associated with it, that set of features (in conjunct rather than as individual elements) is now predicted and can be recognized.

  Filling the coffee pot is **permitted** when the MAKE-COFFEE plan is active; it is associated with the features of the pot being in view and empty. This means not only that the features are now predicted but also that their recognition will trigger the action.

- Actions are specialized by features in the environment and by internal states of the system. As with Firby's RAPs(Firby, 1989), particular states of the world determine particular methods for each general action.

  For example, the specifics of a GRASP would be determined by information taken from the world about the size, shape and location of the object being grasped.

- Action level conflicts are recognized and mediated using the same mechanism that recognizes information about the current state of the world.

  For example, when two actions are active (such as filling the pot and filling the filter), a mediation action selects one of them. During the initial phases of learning a plan, this can in turn be translated into a specialized recognition rule which, in the face of a conflict, will always determine the ordering of the specific actions.

- Finally, suspended goals are associated with the descriptions of the states of the world that are amenable to their satisfaction.

  For example, the goal HAVE-ORANGE-JUICE, if blocked, can be placed in memory, associated with the conjunct of features that will allow its satisfaction, such as being at a store, having money and so forth. Once put into memory, this conjunct of features becomes one of the set that can now be recognized by the agent.

RUNNER's Representation: The knowledge and memory of the agent is captured in the conjunction of three types of semantic nets, representing knowledge of goals, plans and states. Nodes in these networks are linked by abstraction and packaging links, as in DMAP. In addition, we propose an additional SUSPEND link, which connects suspended goals to state descriptions that may indicate opportunities for their satisfaction. For example, the goal to have eggs would be suspended in association with the description of the agent being at a grocery store. In addition to being passed to abstractions of activated concepts, activation markers are always passed along SUSPEND links.

In general, the only other way in which these nets are interconnected is via *concept sequences*. A node may be activated if all of the nodes in one of its concept sequences is activated – a concept sequence for a given node can contain nodes from any of the parts of memory. The following is a partial taxonomy of the types of concept sequences we currently allow:

- Activation of a goal node can activate a node representing a default plan.
- Activation of a plan node and some set of state nodes can activate a further specialization of the plan.
- Activation of a goal node and some set of state nodes can activate a further specialization of the goal.
- Activation of any state node that has a SUSPEND link will activate the associated goal.

An Example: Making Coffee The above discussion of representation may make more sense in the context of an example, currently implemented in RUNNER, of how a particular action is suggested due to conjunction of plan activation and environmental input.

One of the objects in RUNNER's simulated kitchen is a coffeemaker, and one of the plans it has available is that of making coffee with this machine. This plan involves a number of subsidiary steps, some of which need not be ordered with respect to each other. Among the steps that are explicitly represented in the plan are: fetching unground beans from the refrigerator, putting the beans in the grinder, grinding the beans, moving a filter from a box of filters to the coffeemaker, filling the coffeemaker with water from the faucet, moving the ground beans from the grinder to the coffeemaker, and turning the coffeemaker on.

The subplans of the coffee plan are associated with that plan via packaging links. In this implemented example, the agent starts out with a node activated which represents knowledge that it is morning. This in turn is sufficient to activate the goal to have coffee (this is as close as the program

comes to a theory of addiction). This goal in turn activates a generic plan to have coffee. This turns out to be nothing but an abstraction of several plans to acquire coffee, only one of which is the plan relevant to our kitchen:

```
Sending initial activations to memory
sending activation marker to [morning]
Activating concept: [morning]
concept sequence ([morning])
for node [GOAL: drink-coffee] is completed.
sending activation marker to
   [GOAL: drink-coffee]
Activating concept: [GOAL: drink-coffee]
Asserting new goal: [GOAL: drink-coffee]
sending activation marker to
   [PLAN: coffee-plan]
Node [PLAN: coffee-plan] has both permission
and activation:
   ((MARKER [GOAL: drink-coffee]))
    (TOP-LEVEL-PLAN)
Activating concept: [PLAN: coffee-plan]
Asserting new plan: [PLAN: coffee-plan]
Plan has no steps -- insufficiently specific
```

"Visual" input, in terms of atomic descriptions of recognizable objects and their proximities, is passed to memory. For example, the agent "sees" the following visual types:

```
countertop, white wall, box of filters
```

Among sets of possible visually recognized objects are concept sequences sufficient for recognition that the agent is in the kitchen. The recognition of the white wall and the countertop completes one of these sequences. The "kitchen" node in turn passes activation markers to its abstractions, activating the node corresponding to the agent being at home:

```
-------------------------------------------------
Straight ahead I see:
 a countertop, up close;
 a countertop, fairly close;
 a green square filter-box, up close;
 a countertop, fairly close;
 a countertop, far away;
 a white wall, far away;
 a countertop, fairly close;
 a countertop, far away;
 a white wall, far away
To the left is a countertop, up close
To the right, there's a countertop, up close
Straight ahead, there's a countertop, up close
-------------------------------------------------
MEMORY:
Active plans: coffee-plan
```

```
sending activation marker to [wall]
Activating concept: [wall]
sending activation marker to [filter-box]
Activating concept: [filter-box]
sending activation marker to [countertop]
Activating concept: [countertop]
concept sequence ([wall] [countertop])
for node [in-kitchen] is completed.
sending activation marker to [in-kitchen]
Activating concept: [in-kitchen]
sending activation marker to [at-home]
Activating concept: [at-home]
```

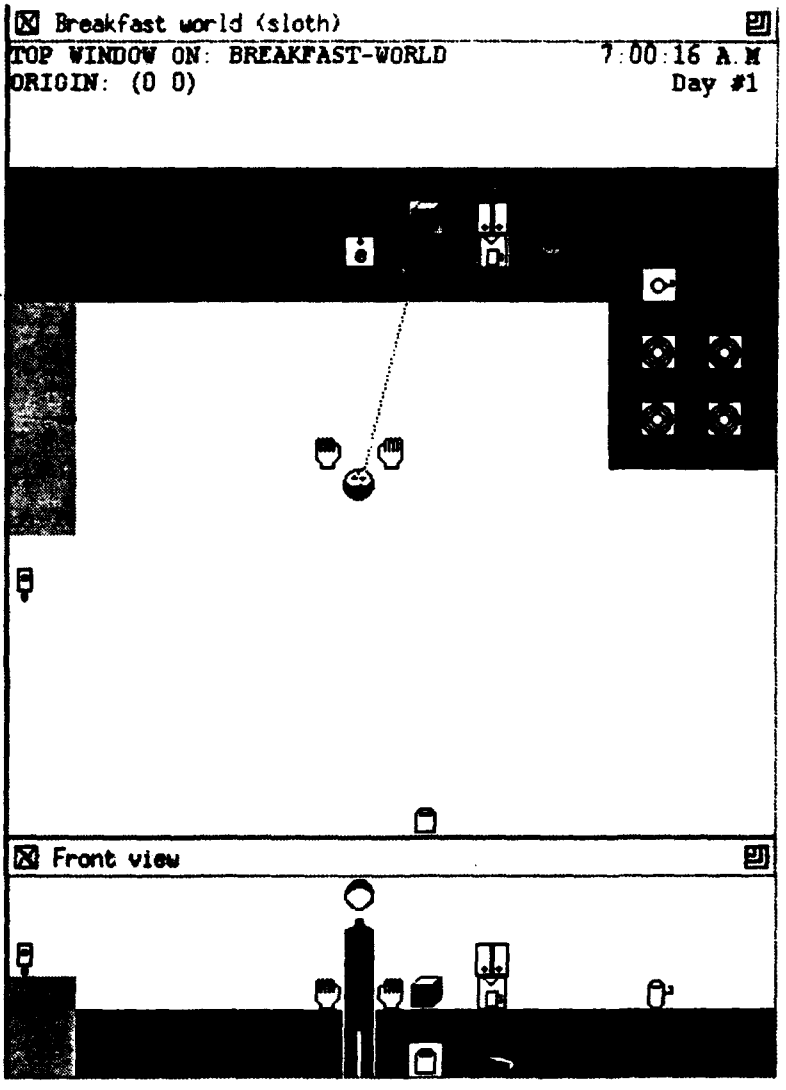


Figure 9: The agent's "visual" focus of attention

The activation of this node in conjunction with the activation of the generic coffee goal completes

the concept sequence necessary for the goal for making coffee at home, which in turn activates the default plan for that goal. In this way a specialized plan is chosen in response to a conjunction of a recognized state and a more generic goal:

```
MEMORY:
concept sequence
    ([GOAL: drink-coffee] [at-home])
for node
     [GOAL: drink-coffee-at-home] is completed.
sending activation marker to
     [GOAL: drink-coffee-at-home]
Activating concept:
     [GOAL: drink-coffee-at-home]
Asserting new goal:
     [GOAL: drink-coffee-at-home]
sending activation marker to
     [PLAN: make-coffee-at-home]
Node [PLAN: make-coffee-at-home]
has both permission and activation:
     ((MARKER [GOAL: drink-coffee-at-home]))
     (TOP-LEVEL-PLAN)
Activating concept:
     [PLAN: make-coffee-at-home]
```

The activation of the coffee-plan causes permission markers to be sent down packaging links to the nodes representing the parts of the plan. The activation of the other object concepts from the "visual" input in turn have sent activation markers to the actions that contain them in their concept sequences. Among these is the plan step for taking a filter from the box and installing it in the coffeemaker, which is activated by seeing box of filters itself. In this way a sub-plan is suggested by the intersection of permission from its parent plan and cues from the environment that indicate that it is easily satisfiable:

```
Asserting new plan:
     [PLAN: make-coffee-at-home]
Sending permissions to steps of plan
Sending permission markers from
     [PLAN: make-coffee-at-home]
to steps
     FILL-CARAFE
     PUT-BEANS-IN-GRINDER
     MOVE-GROUNDS-TO-COFFEE-MAKER
     TURN-ON-COFFEE-MAKER
     GRIND-BEANS
     PUT-IN-FILTER
     GET-COFFEE-BEANS
concept sequence
     ([filter-box]
      [PLAN: make-coffee-at-home])
for node [PLAN: put-in-filter] is completed.
```

```
sending activation marker to
     [PLAN: put-in-filter]
Node [PLAN: put-in-filter]
has both permission and activation:
     ((MARKER ([filter-box]
               [PLAN: make-coffee-at-home])))
     ((MARKER [PLAN: make-coffee-at-home]))
Activating concept:
     [PLAN: put-in-filter]
Asserting new plan: [PLAN: put-in-filter]
Sending permissions to steps of plan
Sending permission markers from
     [PLAN: put-in-filter]
to steps
     PUT-FILTER-IN-COFFEEMAKER
     GET-FILTER
concept sequence
     ([filter-box]
      [PLAN: put-in-filter])
for node [PLAN: get-filter] is completed.
sending activation marker to
     [PLAN: get-filter]
Node [PLAN: get-filter]
has both permission and activation:
     ((MARKER ([filter-box]
               [PLAN: put-in-filter])))
     ((MARKER [PLAN: put-in-filter]))
Activating concept: [PLAN: get-filter]
```

After another level of passing permission markers to sub-plans, the process "bottoms out" in the suggestion of the primitive action of picking up the box of filters. With no suggestions to the contrary, the action is taken:

```
Asserting new plan:
     [PLAN: get-filter]
Sending permissions to steps of plan
Sending permission markers from
     [PLAN: get-filter]
to steps
      TAKE-OUT-FILTER
      PICK-UP-BOX
      LOOK-FOR-FILTER-BOX
concept sequence
     ([filter-box] [PLAN: get-filter])
for node [PLAN: pick-up-box] is completed.
sending activation marker to
     [PLAN: pick-up-box]
Node [PLAN: pick-up-box]
has both permission and activation:
     ((MARKER ([filter-box] [PLAN: get-filter])))
```

```
     ((MARKER [PLAN: get-filter]))
Activating concept: [PLAN: pick-up-box]
Suggesting action: (GRASP 'FILTER-BOX)
------------------------------------------------
ACTION:
Performing action: (GRASP 'FILTER-BOX)
------------------------------------------------
To the left is a countertop, up close
To the right, there's a countertop, up close
Straight ahead, there's a countertop, up close
Result of action: I'm holding on to a filter-box
------------------------------------------------
```

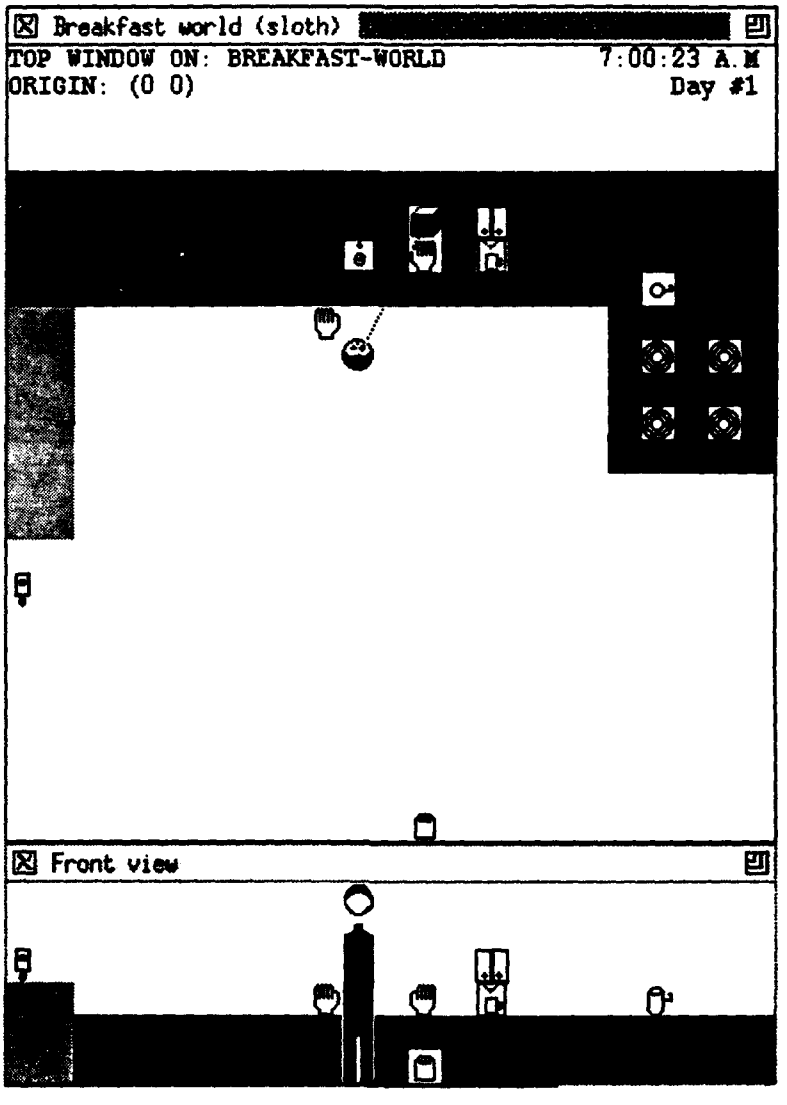


Figure 10: The agent reaches for a box of filters

The final action is chosen both on the basis of active plans and goals, and in response to the

immediate circumstances in which the agent finds itself. Given a change in either the top-down guidance or the bottom-up recognition, the selection of plan and action will change in response.

4.3.4 Results

Achievements include:

- Development and extension of RUNNER's simulated world. Some features of this simulator are:

  1. A fairly sophisticated "physics", in which unsupported objects fall, struck objects acquire momentum and move independently, and objects approach the temperature of the surrounding environment.
  2. A simulation of visual search, which enables RUNNER's agent to look initially for objects on the basis of color or texture, and then focus on candidate objects to see if they are of the right type.
  3. Complete decoupling of the simulator and the AI program; the two programs can now run on different machines, and communicate over UNIX sockets. Multiple AI programs (not just RUNNER) in the Chicago AI lab can "attach" to the simulator and inhabit the same simulated world.

- Successful use of plans that result in action sequences of four to five hundred steps.
- Use of more than one plan simultaneously, without the intermediate construction of an interleaved plan.
- Successful "multiple day" examples, where the program uses the same plan representation to perform a task even though its own activity has changed the circumstances of the relevant objects (*e.g.* an object which was taken from a cabinet is now found left on a counter on the next run of the plan).

4.3.5 Current work

We are currently performing experiments on learning the appropriateness conditions for using particular plans—the conditions that should be checked before the plan is committed to(Converse and Hammond, 1991b) This set of conditions is not the same as the logical preconditions, since some preconditions are effectively "always true", and others can be addressed and fixed easily if encountered in the midst of execution. Also, the agent may not know the truth value of all preconditions in advance, and the effort of physically verifying them may in some cases not be worth the expected cost of plan failure. Our approach involves two kinds of complementary learning, one of which learns new preconditions from failure, and the other of which drops preconditions that turn out persistently to be true.

4.4 Shopper

Planning and perception have generally been treated as separate topics in the AI literature. However, recent work has cast many doubts on the practicality of either research track. Planning researchers have begun to realize that they cannot expect a complete, perfect model of the world when they begin to plan and, even if they did have such a model, the computational complexity involved in using all of its detail would be prohibitive. Without a complete world model, future states of the world cannot be predicted accurately and detailed plans cannot be constructed in advance.

Similarly, perception researchers are also coming to suspect that it may not be possible to generate a complete model of the visual world in real time. The amount of detail in a visual scene is enormous and modeling it all in a rapidly changing environment would appear hopeless. Furthermore, the complete interpretation of a scene is seldom necessary because only a fraction of the total information available is usually relevant for any given task. Faced with this flood of often irrelevant data, we are turning to techniques that focus sensing and processing resources on just those aspects of the world that are important at any given time.

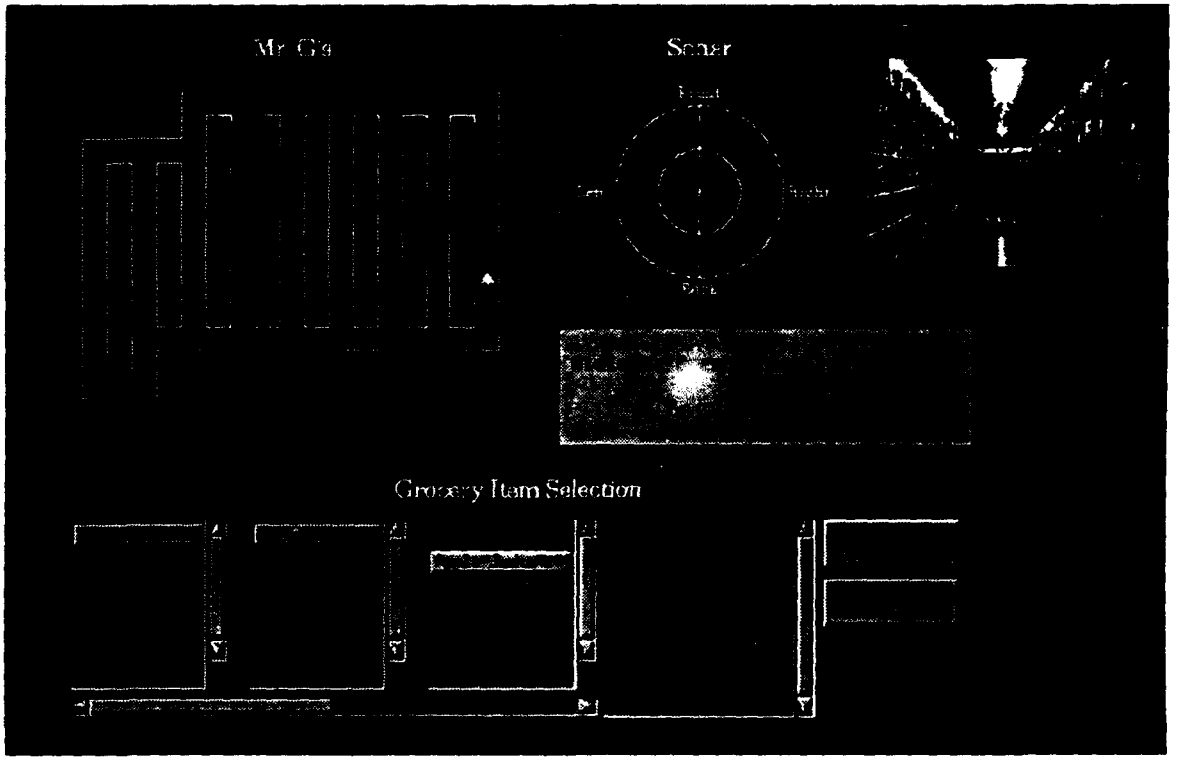


Figure 11: The Shopper selection and search screen

The SHOPPER project considers the planning problems associated with navigation and search in a virtual world into which an agent is thrust. The world itself is a grocery store through which an agent must navigate, move, and shop for groceries. Our grocery store simulator, based on video images, provides a realistic domain in which to explore ideas about navigation, planning, and opportunism while avoiding the headaches associated with robots (see Figure 11). As in RUNNER

we are exploring ideas of opportunism in SHOPPER without having to depend on a robot. By using video images, we raise interesting questions of how one "notices" items in realtime. This project thus brings together ideas from several areas of AI, including work in case-based reasoning, animate agency, and active vision(Fu *et al.*, 1994).

Initially with the SHOPPER project, we are examining the types of functional knowledge needed for an agent to work in a man-made domain as well as the sensing and control mechanisms needed to use this knowledge. In the remaining sections, we describe the SHOPPER system: an integrated system incorporating planning and vision techniques for the task of grocery store shopping.

Grocery store shopping is a common task everyone does at least occasionally. Since everybody is able to accomplish their shopping needs fairly quickly, we are interested in what functional knowledge people use in order to shop efficiently as possible. Since all grocery store managers presumably want customers to find items without much trouble, they place and index items in some consistent manner. They do so according to the features they deem the most functional in terms of satisfying their customer's needs, and their own needs for selling as much food as possible.

A customer intending to leave in a reasonable time has to know how his food will be used. For example, suppose a customer who wants to bake a cake needs cake mix and cake frosting. He'll find the cake frosting nearby the cake mixes. He also might find the cake mixes near the flour, sugar, and baking tins. This arrangement is anything but accidental: it's *intentional.* The cake mixes, as well as the rest of the items in the store, are indexed according to the features most useful in serving the needs of their customers and their stores.

To a greater and lesser extent, other man-made environments will exhibit regularities of organization. Examples are ubiquitous: kitchens, offices, bedrooms, stores, streets, cars, etc. In each of these instances, people can and will use their knowledge of regularities in order to facilitate their task. Consider a robot whose job is to tidy up several desks in offices. He needs to know where pens, pencils, papers, and books should go. Placing pens and pencils are simple. Filing papers and books are much harder because it involves knowledge of a person's method of organizing their literature. Books can be arranged according to several criteria such as: author, title, subject, shape/size, frequency of use, etc. We are interested in using knowledge such as this to aid in accomplishing tasks.

4.4.1 GroceryWorld

The SHOPPER agent works in a simulated grocery store called GROCERYWORLD. We wanted to build a world which offers the same challenges and opportunities as a real grocery store. However, we wish to avoid all the problems associated with real robots – problems like fixing broken hardware, writing motor driver code, having to transport the robot to an available grocery store, etc.[4]

The GROCERYWORLD simulator satisfies these design criteria. By using video footage from an actual store, we are able to base our simulator on real images of a grocery store. The simulator is complete in that the entire store (excluding checkout counter areas) is modeled by the simulator. Any object in the image database is accessible by moving through the world.

[4]We also, for now, are able to ignore problems such as noise in sonar readings and wheel slippage. However, we will eventually incorporate similar problems into GROCERYWORLD.

In addition, the simulator provides range information on the relative proximity of walls and aisles with respect to the agent's current location. Sign information is also given. When an agent is at the end of an aisle and looking down that aisle, he automatically receives the text of those signs. The signs in GROCERYWORLD are a faithful reproduction of the signs of the specific grocery store filmed.

The remainder of this section is organized as follows. We first illustrate the kinds of regularities present in grocery stores. Next, we describe the visual routines implemented in SHOPPER and how regularities aid in processing visual information. Then we discuss the control mechanism for execution of plans and visual routines. We later demonstrate a more complicated example of search which uses a combination of regularities and visual mechanisms. In the final section we relate our project to similar work, and discuss its implications.

4.4.2 Regularities in grocery stores

Because a moderately-sized grocery store can stock at least 10,000 items, grocery stores need to organize their food items in consistent ways so customers can easily find them. In this section we illustrate the different types of knowledge used for finding goods. In the Raisin Bran example, we were relying on organization by *type*. Below are the regularities we have identified so far:

- **Type:** The most important strategy for the Raisin Bran example. Typically, items that either serve nearly the same function, or are very similar are nearby each other. This is a most basic organization principle under which many items fall under; e.g. McIntosh apples are near Rome apples; a jar of Gerber baby food will be found with other baby foods; a tomato clustered with other vegetables; an apple placed with other fruits; coffee is near tea.
- **Brand:** Within a section of a specific type, the maker of the food will also be clustered together. For example, in a typical grocery store aisle, soups of the same brand (e.g. Campbell's, Progresso) will be clustered with each other no matter how similar they. So, Campbell's vegetable soup is not placed adjacent to Progresso vegetable soup.
- **Counterparts:** Items that complement each other. For example, salad and salad dressing, pancakes and maple syrup, pasta and tomato sauce, etc.
- **Physical Constraints:** Perishable or bulky items that require special storage considerations like orange juice, eggs, frozen entrees, etc.
- **Ethnic foods:** For items commonly associated with other countries or cultures: e.g. soy sauce, curry, matzah, water cress. These foods tend to be placed nearby each other in an "ethnic" section.[5]
- **Packaging:** Bulk items such as bags of oranges, apples, and potatoes will be placed separate from their individual versions.

[5] A point of clarification is needed here. While SHOPPER makes use of the idea of "ethinc foods" we are more interested in the generalization of the undelying concept. That is, the world is organized around seemly arbitrary conections that can be exploited by an agent trying to make its way around an environment. In an art museum, "ethnic foods" would be replaced with "schools of art." No matter what the specifics are, the generalization is to use the organization that the world gives you.

These regularities are general rules of thumb - not hard and fast rules. But they provide direction for finding items. The point is to avoid exhaustive search by using regularities as fixed points from which we can base the search for an item.

At one time or another, each of these regularities can prove useful. But they can also be wrong. Since SHOPPER works within the structure of a store organized by someone else, this can lead to mistaken beliefs about the locations of objects. Eventually, though, an agent can incrementally learn and optimize its plans of action over several visits. And when new grocery stores are encountered, the agent can be better prepared since its knowledge of particular grocery stores serves as a field from which it can reap the benefits of past experience.

4.4.3 Control of action and perception

The planning and acting mechanism is a version of that used in RUNNER (Hammond *et al.*, 1990). SHOPPER's control structures are composed of plans. Figure 12 shows the basic algorithm. Initially, a plan is given a *permission* to activate. An active plan first checks to see if its objectives (a *success* clause) are met. If so, it finishes. If not, it selects a method based on current sensor and state information. Each method will have a sequence of plans or actions. These plans and actions will then be permitted (activated) in sequence, as successive plans succeed.

```
procedure permit (plan)
if succeeded(plan) then
   return true;
else
   pick applicable method m;
   for i in m's plan sequence
      do if i is an action then
         execute action i;
      else
         permit(i);
```

Figure 12: A simplified control mechanism for Shopper.

Execution of this control mechanism behaves in a very "depth-first search" manner by permitting abstract plans which become more and more concrete depending on sensor/state conditions. The resulting "leaves" are either physical, visual, or mental actions. For example "(align-body-to-head)" is a physical action which orients the direction of travel to the direction the head is facing.

4.4.4 Example

In this section we illustrate a more involved example of finding an item: looking for pancake mix. According to the "type" regularity discussed earlier, we should expect that Mrs. Butterworth's pancake mix be placed near other pancake mixes. However, there's no sign saying "pancake mix". In that case, we know if other regularities will (or won't) apply:

- counterparts - Maple syrup is often used with pancakes.

- physical constraints - Not applicable since neither needs to be refrigerated.
- packaging - Both are small and can reside near each other.

Because of these regularities a *good place to look* is nearby the maple syrups. As we will demonstrate, this belief is correct for this particular example in GROCERYWORLD.

The following is an edited trace of SHOPPER finding a box of pancake mix. Of the 159 primitive actions done, only illustrative ones are reported here.

```
Permitting (find-item mrsbutterworths)
 Permitting (find-sign)
  [Action: (turning body left)]
```

At this point, SHOPPER is looking down an aisle. *Sign information* is passed from the simulator:

```
(see sign aisle-1) (see sign bread)
(see sign cracker) (see sign cookie)
(see sign meat) (see sign frozen-entree)
(see sign baked-good)
```

From here, SHOPPER needs to turn his body toward an open area so he can move across the aisles. He'll then turn back toward the aisle in order to see signs.

```
  [Action: (turning body left)]
  [Action: (turning head to look right)]
Permitting (move-across-aisles-looking-for
            mrsbutterworths)
  [Action: (moving forward)]
  [Action: (moving forward)]
```

At this point, SHOPPER keeps moving forward until a relevant sign is seen: syrup.

```
(see sign aisle-5) (see sign syrup)
(see sign oil) (see sign shortening)
(see sign bakery-need) (see sign tea)
(see sign coffee) (see sign sugar)
(see sign flour)
```

SHOPPER will now use a visual routine we term a *type recognizer*: a routine which indicates the type of items in an image without recognizing any single item. Because of the regularity that items of the same type are grouped together, we sample color histograms from the regions above shelves and compare them across the color histogram database of items. Active items in the database keep track of their best match value and then vote for their associated type. If enough responses are registered for a particular type, we consider the type to be recognized. Because of the sign information, SHOPPER only considers those histograms directly related to the signs by type. This constraint improves type recognition by reducing the number of false positives.

```
Permitting (move-down-aisle-looking-for
            mrsbutterworths)
 [Action: (aligning body to head)]
 [Action: (disabling items of type all)]
 [Action: (enabling items of type syrup
           oil shortening bakery-need
           coffee sugar flour tea)]
Permitting (look-for-type pancake-mix
            syrup mrsbutterworths)
 [Action: (turning head to look left)]
 [Action: (checking for shelves)]
 [Action: (checking for type at shelf
           position 175)]
 [Action: (checking for type at shelf
           position 305)]
 [Action: (moving forward)]
 [Action: (moving forward)]
 [Action: (turning head to look right)]
```

Next, SHOPPER processes images on the left and right as it moves down the aisle. Eventually, shopper will reach the syrup section and then begin a local search routine within the local region of the aisle.

```
Permitting (search-vicinity mrsbutterworths)
 [Action: (turning body full around)]
 [Action: (moving forward)]
 [Action: (moving forward)]
 [Action: (checking for shelves)]
```

Now, SHOPPER is looking for Mrs. Butterworth's. It does so using an *item recognizer*. By taking color histograms across and above a shelf location, it can quickly tell if the box is not present if all resulting intersections are low in value. In contrast, if the intersection values are high, we bound the regions of high response and use Hausdorff distance comparison by first using a precomputed edge image of Mrs. Butterworth's and computing an edge image of the region of high response. If the edge images match well, we have verified the location of the item. If not, we consider the item to be absent from the image and continue on.

```
 [Action: (checking for mrsbutterworths
           at height 93)]
 [Action: (checking for mrsbutterworths
           at height 239)]
 [Action: (verifying if mrsbutterworths
   in boundaries 72 to 113 at height 168)]
 [Action: (verifying if mrsbutterworths
   in boundaries 236 to 339 at height 168)]
 [Action: (found mrsbutterworths item)]
```

In this example, we used regularities of type and counterpart in order to design more complicated routines. This merging of simpler visual routines into more sophisticated routines results in more

robust performance at a smaller cost. The color histogram intersection routine could be scanned across the entire image and produce many possible locations for an object. However, by itself, it is not enough to reliably verify the existence of the object. The Hausdorff distance between a model edge image and an entire image yields more accurate results, but at a prohibitive time cost.

The combined routines of shelf detection, color histogramming, and Hausdorff distance not only lessen computation time, but they also provide more reliable performance as a whole. The regularities of the domain allow these visual routines to be combined into more complex routines. Thus, an examination of the task makes SHOPPER not only more reliable, but also permits us to use simpler machinery (Agre, 1988).

4.4.5 Status of Shopper

SHOPPER currently uses four out of the six regularities outlined earlier: type, counterpart, physical constraint, and ethnic foods. Of the 825 food items in the database, we initially tested for thirty items. Out of the thirty, SHOPPER correctly found eighteen items (60 percent found). For all but one of the trials, a wrong item was picked. Since many of these items were relatively small (about 40x50 pixels), we then tried twenty-five items of larger sizes (cereals, laundry detergents, etc.). Of the twenty-five items: twenty were found (80 percent found), one was missed by color histogramming, one wrong item was picked and the other three didn't match correctly using our set thresholds for Hausdorff matching. Since we used a wide-angle lens for filming, items appearing close to the borders of any image will be warped. Larger items' edge models will suffer from this problem. We believe we can alleviate this matching problem by de-warping the image.

We are also investigating the uses of texture for noticing cans and bottles. Since cans and bottles can be rotated and stacked in several ways, comparing edge images using Hausdorff distance is unreliable for both detection and verification. Although the use of color histograms is still robust, we are still missing a good verifier. By characterizing the textures of items, we believe we can use texture in another routine together with the other existing routines to find those kinds of items.

4.4.6 Related work

Everyday tasks have been studied before in AI – most recently in the realm of cooking tasks. Agre and Horswill (Agre and Horswill, 1992) have built a system, TOAST, which specializes in making breakfast food in a simulated kitchen. They demonstrate how activity in the midst of cultural artifacts can be improvised to produce nontrivial behavior. They do this by noticing regularities, or constraints, on cooking tools and materials. Hammond and Converse (Hammond and Converse, 1991) have also noted that our environments are designed to aid activity, rather than hinder. Regularities, if maintained, can greatly simplify a person's interactions with the world. They demonstrate the efficacy of this approach for the task of making coffee in a simulated kitchen.

4.4.7 Discussion

SHOPPER, as well as TOAST and RUNNER , are untraditional programs in that they actively participate in their domains. These domains have been engineered for human use, and are replete

with tools for facilitating tasks. As a consequence, an agent's activity cannot be characterized independent of its relationship to its surroundings.

SHOPPER is differentiated from traditional planning domains since GROCERYWORLD provides real visual information while still being a controllable simulation. GROCERYWORLD is very unique in that respect: the richness of visual information provides a testbed to try ideas of planning, vision, and activity in the context of an everyday task and domain, but without having to maintain a physical robot and its environment. For the time being, we are not addressing all the problems of robot sensor/actuator uncertainties. By considering some real sensor problems, we have explored some ways in which an account of the regularities can help us design reliable visual routines.

SHOPPER also differs from past vision research in that the vision routines are highly task-based. Every single image is considered in the context of the system's understanding of how the world is organized. Thus SHOPPER can expect to see shelves, classes of items, an unobstructed aisle, etc. Using this knowledge results in visual routines which will always compute relevant information. Moreover, these routines are simple and fast. While not powerful by themselves, a combination of routines can result in robust performance. Since we are currently working on more routines, we expect to analyze the relative utility of routines in order to assemble routines, both in design and at run-time.

Because SHOPPER works in a grocery store, it initially can't know many of the item locations. Moreover, these items can come and go.[6] Since TOAST and RUNNER work in a kitchen, practically anything can be found immediately since the physical search space is much smaller (and we usually make breakfast/coffee in the comfort of our own dwellings). So, SHOPPER copes with a world which is engineered for people, but not specifically for the agent. SHOPPER doesn't control the stocking or the layout of the store, so it must learn/know the organization as opposed to attempting to restructure the store to its own liking.

Eventually, we would like SHOPPER to expand its set of regularities by learning the organization of specific grocery stores. Earlier, we illustrated SHOPPER finding a box of pancake syrup. However, we did not say why a regularity of counterparts should be preferred over regularity of type. Indeed, there could have been a "pancake" sign in the next aisle. The detection and relevance of potential opportunities is the subject of future work.

5 Other Activities

Along with the work done directly on these projects, the Chicago group has also been trying to transfer the Case-based technology developed under this grant into other, non-academic arenas. Three such efforts in particular are worth mentioning here.

Dr. Hammond is working with Owen Research in Boulder, CO in the development of a case-based system for the selection and evaluation of NASA flight crews. The work involves the integration of our CBR indexing technology with existing databases and a fuzzy logic reasoning system. The goal of the project is a system that can use existing libraries of "stories" to predict the results of particular crew pairings. This work is being done by Owen Research under a NASA SBIR.

[6] We are also preparing GROCERYWORLD2: the same grocery store filmed one year later.

Dr. Hammond is also involved with Applied Research Associates in the development of a case-based tool for Munitions Effectiveness Assessment. The work includes the development of a system designed to aid a weaponeer or air campaign planner in the weaponing process. The system is designed to use case-libraries to aid in the selection and evaluation of both targets and weapons. This work is being done under a contract with the Defense Nuclear Agency.

Dr. Martin is working with Intell/Agent Systems in the development of autonomous robotic systems for astronaut assistance. The work requires the integration of the case-based parsing technology with external planning and scheduling systems. The systems are designed to function both autonomously and in cooperation with astronauts in space environments. This work is being done by Intell/Agent Systems under a NASA SBIR.

6 Publications

The research reported in this document has resulted in the following publications:

1. J. Berger. "Roentgen: Case-based reasoning and radiation therapy planning." In *The Proceedings of the 16th Annual Symposium on Computer Applications in Medical Care*, New York, 1992. McGraw-Hill.

2. J. Berger. "Knowledge acquisition and design support in a medical domain." In *Papers from the 1993 Workshop on Case-Based Reasoning.* AAAI Press, Menlo Park, 1993.

3. J. Berger. "Roentgen: Radiation therapy and case-based reasoning." In *The Proceedings of the 10th Conference on Artificial Intelligence for Applications*, Washington, DC, 1994. IEEE, IEEE Computer Society Press.

4. J. Berger, K. Hammond, and G. T. Y. Chen. "Roentgen: A case-based approach to radiation therapy planning." In *The Proceedings of the 32nd Annual Scientific Meeting of the American Society for Therapeutic Radiology and Oncology*, page 220, New York and Oxford, 1990. Pergamon Press.

5. Jeffrey Berger and Kristian J. Hammond. "Roentgen: A memory-based approach to radiation therapy treatment design." In Ray Bareiss, editor, *Proceedings: Case-Based Reasoning Workshop*, San Mateo, California, May 1991. Morgan Kaufmann Publishers, Inc.

6. John Borse and Christopher Owens. "IOPS Advisor: Knowledge-Intensive Methods for Irregular Operations Airline Scheduling." *AAAI Spring Symposium on Planning and Scheduling.* Palo Alto, CA, March, 1992.

7. Timothy Converse and Kristian Hammond, "Preconditions and appropriateness conditions", *Proceedings of the 1992 Meeting of the Cognitive Science Society.* Bloomington, IN: Erlbaum Associates.

8. Timothy Converse and Kristian Hammond, "Learning to Satisfy Conjunctive Goals', *The Proceedings of the Ninth International Conference on Machine Learning.* Aberdeen, Scotland: Morgan Kaufmann, July 1992.

9. Timothy Converse and Kristian Hammond. "Opportunistic Memory and Visual Search", *Proceedings of the 1991 Meeting of the Cognitive Science Society.* Chicago, IL: Erlbaum Associates.

10. Timothy Converse and Kristian Hammond. "Stabilizing environments to facilitate planning and activity: An engineering argument". *Proceedings of the 1991 National Conference of Artificial Intelligence.* AAAI Press and MIT Press.

11. Daniel Fu, Kristian Hammond, and Michael Swain. (1994). "Vision and navigation in man-made environments: Looking for syrup in all the right places." In *Proceedings, Workshop on Visual Behaviors* IEEE Computer Society Press, Washington, DC. (to appear)

12. Patricia Goldweic and Kristian J. Hammond. "Multi-Agent Interaction: A Vocabulary of Engagement." In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society.* Hillsdale, NJ, 1992. Lawrence Earlbaum.

13. Kristian Hammond, Timothy Converse, and Joshua Grass, "The Stabilization of Environments", *Artificial Intelligence Journal.* Elsevier Science Publishers, North-Holland, Amsterdam. In press.

14. Kristian Hammond and Colleen Seifert. "Opportunistic memory." In *Beliefs, Reasoning, and Decision, Making: Psycho-logic in Honor of Robert Abelson.* R. C. Schank and E. Langer (Eds.) Hillsdale, NJ: Erlbaum Asso. 1994.

15. Kristian Hammond, Colleen Seifert, Mitchell Marks, and Timothy Converse, "Opportunism and Learning", *The Journal of Machine Learning.* Vol.10, Number 3, March 1993. (Hammond *et al.*, 1993b)

16. Kristian Hammond, Timothy Converse and Mitchell Marks. "Towards a Theory of Agency." Chapter 11 in *Machine Learning Methods for Planning.* Edited by Steven Minton. Morgan Kaufmann Publishers, Inc., San Mateo, CA. 1993. (Hammond *et al.*, 1993a)

17. Kristian Hammond and Colleen Seifert. "A Cognitive Science Approach to Case-Based Planning." In *Foundations of Knowledge Acquisition: Cognitive Models of Complex Learning.* S. Chipman and A. L. Meyrowitz (Eds.), (pp. 245-267). Norwell, MA: Kluwer Academic Publishers. 1993. (Hammond and Seifert, 1993)

18. Kristian Hammond and Colleen Seifert. "A Vocabulary for Indexing Plan Interactions and Repairs." In *Proceedings of the 1992 Meeting of the Cognitive Science Society.* Erlbaum Associates. Bloomington, IN. July, 1992. (Hammond and Seifert, 1992)

19. Kristian Hammond and Timothy Converse. "Stabilizing Environments to Facilitate Planning and Activity: An engineering argument" In *The Proceedings of the 1991 National Conference of Artificial Intelligence.* July 1991.

20. Kristian Hammond, Timothy Converse and Charles Martin. "Integrating Planning and Acting in a Case-Based Framework." In *The Proceedings of the 1990 National Conference of Artificial Intelligence.* August 1990. (Hammond *et al.*, 1990)

21. Charles E. Martin. "Language and Intermediate Vision." In *Working Notes of the 1994 AAAI Spring Symposium on Active Natural Language Processing*, Stanford, CA. March 1994.

22. Charles E. Martin. "The Representation of Experience for Case-Based Reasoning in Subjective Interpretation." In *Proceedings of the AAAI-93 Workshop on Case-Based Reasoning*, Washington, D.C. August 1993.

23. Charles E. Martin and R. James Firby. "An Overview of the Dynamic Predictive Memory Architecture for Robotic Assistants." In *Proceedings of the Third Conference on Cooperative Intelligent Robotics in Space*, Boston, MA. 1992.

24. Charles E. Martin and R. James Firby. "Generating Natural Language Expectations from a Reactive Execution System." In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*, Chicago, IL. July 1991.

25. Charles E. Martin and R. James Firby. "An Integrated Architecture for Planning and Learning." In *Working Notes of the 1991 AAAI Spring Symposium on Integrated Cognitive Architectures*, Palo Alto, CA. Reprinted in *ACM SIGART Bulletin* **2**(4), 125–129. March 1991.

26. Charles E. Martin and R. James Firby. "Advising a Reactive Planner." In *First Annual Workshop on Planning and Learning*, Stanford, CA. 1991.

27. Thomas F. McDougal. "Using case-based reasoning and situated activity to write geometry proofs." In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, pages 60–65. MIT Press, 1993.

28. Thomas F. McDougal and Kristian J. Hammond. "Representing and using procedural knowledge to build geometry proofs." In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 711–716, Hillsdale, NJ, 1993. Lawrence Earlbaum.

29. Tom McDougal and Kristian Hammond. "A recognition model of geometry theorem-proving." In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, pages 106–111, Hillsdale, NJ, 1992. Lawrence Earlbaum.

30. Christopher Owens. Integrating Feature Extraction and Memory Search. *Machine Learning* **10**, 311–339 (1993).

31. Christopher Owens. Qualitative Relevance Feedback and Incremental Query Formulation. *AAAI Spring Symposium on Adaptation and Reuse*. Palo Alto, CA, March, 1992.

32. Christopher Owens. Problem Solving Stereotypes for an Intelligent Assistant. *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*. 1026–1031. Cognitive Science Society. Pittsburgh, PA. (1992)

33. Christopher Owens. A functional taxonomy of abstract plan failures. *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*. 167–172. Lawrence Erlbaum Associates, Hillsdale NJ. (1991)

34. Christopher Owens. Representing abstract plan failures. *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*. 277–284. Lawrence Erlbaum Associates, Hillsdale NJ. (1990)

35. Colleen Seifert, Kristian Hammond, Hollyn Johnson, Timothy Converse, Thomas McDougal, and Scott W. Vanderstoep. Case-Based Learning: Predictive features in Indexing. In *The Journal of Machine Learning*. In press.

References

Agre, P. & Chapman, D. (1987). Pengi: An implementation of a theory of activity. In *The Proceedings of the al Conference on Artificial Intelligence*, pages 268–72. AAAI.

Agre, P. E. & Horswill, I. (1992). Cultural support for improvisation. In *The Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 363–368.

Agre, P. E. (1988). The dynamic structure of everyday life. Technical Report 1085, MIT.

Berger, J. & Hammond, K. J. (1991). Roentgen: A memory-based approach to radiation therapy treatment design. In Bareiss, R., editor, *Proceedings: Case-Based Reasoning Workshop*, San Mateo, California. Morgan Kaufmann Publishers, Inc.

Berger, J., Hammond, K., & Chen, G. T. Y. (1990). Roentgen: A case-based approach to radiation therapy planning. In *The Proceedings of the 32nd Annual Scientific Meeting of the American Society for Therapeutic Radiology and Oncology*, page 220, New York and Oxford. Pergamon Press.

Berger, J. (1992). Roentgen: Case-based reasoning and radiation therapy planning. In *The Proceedings of the 16th Annual Symposium on Computer Applications in Medical Care*, New York. McGraw-Hill.

Berger, J. (1993). Knowledge acquisition and design support in a medical domain. In *Papers from the 1993 Workshop on Case-Based Reasoning*. AAAI Press, Menlo Park.

Berger, J. (1994). Roentgen: Radiation therapy and case-based reasoning. In *The Proceedings of the 10th Conference on Artificial Intelligence for Applications*, Washington, DC. IEEE, IEEE Computer Society Press.

Boddy, M. & Dean, T. (1989). Solving time-dependent planning problems. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*. International Joint Conference on Artificial Intelligence.

Borse, J. & Owens, C. (1992). Iops advisor: Knowledge-intensive methods for irregular operations airline scheduling. In *AAAI Spring Symposium on Planning and Scheduling*, Stanford, California. AAAI Press. Menlo Park, California.

CHAPMAN, D. (1985). Planning for conjunctive goals. Memo AI-802, AI Lab, MIT.

Converse, T. M. & Hammond, K. J. (1991a). Opportunistic memory and visual search. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*, pages 97–101.

Converse, T. M. & Hammond, K. J. (1991b). Preconditions and appropriateness conditions. In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, pages 97–101.

Converse, T. M. & Hammond, K. J. (1992). Learning to satisfy conjunctive goals. In *Proceedings of the Sixth International Workshop on Machine Learning*.

Firby, R. J. & Hanks, S. (1987). A simulator for mobile robot planning. In *Knowledge-Based Planning Workshop*, Austin, TX. DARPA.

Firby, R. J. (1989). Adaptive execution in complex dynamic worlds. Research Report 672, Yale University Computer Science Department.

Fu, D., Hammond, K., & Swain, M. (1994). Vision and navigation in man-made environments: Looking for syrup in all the right places. In *Proceedings, Workshop on Visual Behaviors*, pages 363–368. IEEE Computer Society Press.

Goldweic, P. & Hammond, K. (1992). Multi-agent interaction: A vocabulary of engagement. In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, Hillsdale, NJ. Lawrence Erlbaum.

Hammond, K. J. & Converse, T. M. (1991). Stabilizing environments to facilitate planning and activity: An engineering argument. In *The Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 787–793.

Hammond, K. & Seifert, C. (1992). A vocabulary for indexing plan interactions and repairs. In *Proceedings of the 1992 Meeting of the Cognitive Science Society.* Erlbaum Associates. Bloomington, IN.

Hammond, K. & Seifert, C. (1993). A cognitive science approach to case-based planning. In Chipman, S. & Meyrowitz, A. L., editors, *Foundations of Knowledge Acquisition: Cognitive Models of Complex Learning*, Norwell, MA. Kluwer Academic Publishers.

Hammond, K. & Seifert, C. (1994). Opportunistic memory. In Schank, R. C. & Langer, E., editors, *Beliefs, Reasoning, and Decision Making: Psycho-logic in Honor of Robert Abelson*, pages 19–23, Hillsdale, NJ. Erlbaum Associates. Menlo Park, California.

Hammond, K. J., Converse, T., & Marks, M. (1988). Learning from opportunities: Storing and reusing execution-time optimizations. In *The Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 536–40. AAAI.

Hammond, K., Converse, T., & Martin, C. (1990). Integrating planning and acting in a case-based framework. In *The Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 292–297.

Hammond, K., Converse, T., & Marks, M. (1993a). Towards a theory of agency. In Minton, S., editor, *Machine Learning Methods for Planning*, San Mateo, California. Morgan Kaufman Publishers.

Hammond, K., Seifert, C., Marks, M., & Converse, T. (1993b). Opportunism and learning. In *The Journal of Machine Learning.*

Hammond, K., Converse, T., & Grass, J. (1994). The stabilization of environments. *Artificial Intelligence.* In press.

Hammond, K. (1989a). *Case-Based Planning: Viewing Planning as a Memory Task*, volume 1 of *Perspectives in Artificial Intelligence.* Academic Press, San Diego, CA.

Hammond, K. J. (1989b). Opportunistic memory. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*. IJCAI.

Hammond, K. J. (1990). Explaining and repairing plans that fail. *Artificial Intelligence Journal*. in press.

Martin, C. & Firby, R. (1991a). Advising a reactive planner. In *First Annual Workshop on Planning and Learning*, Stanford, California.

Martin, C. & Firby, R. (1991b). Generating natural language expectations from a reactive execution system. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*. Chicago, IL.

Martin, C. & Firby, R. (1991c). An integrated architecture for planning and learning. In *Working notes of the 1991 AAAI Spring Symposium on Integrated Cognitive Architectures*, Stanford, California.

Martin, C. & Firby, R. (1992). An overview of the dynamic predictive memory architecture for robotic assistants. In *Proceedings of the Third Conference on Cooperative Intelligent Robotics in Space*. Boston, MA.

Martin, C. E. (1989). *Direct Memory Access Parsing*. PhD thesis, Yale University Department of Computer Science.

Martin, C. E. (1990). *Direct Memory Access Parsing*. PhD thesis, Yale University.

Martin, C. (1993). The representation of experience for case-based reasoning in subjective interpretation. In *Proceedings of the AAAI-93 Workshop on Case-based Reasoning*, Stanford, California.

Martin, C. (1994). Language and intermediate vision. In *Working Notes of the 1994 AAAI Spring Symposium on Active Natural Language Processing*, Stanford, California.

McDermott, D. (1978). Planning and acting. *Cognitive Science*, 2:71–109.

McDougal, T. & Hammond, K. (1992a). A recognition model of geometry theorem-proving. In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, pages 711–716. Lawrence Erlbaum.

McDougal, T. & Hammond, K. (1992b). Representing and using procedural knowledge to build geometry proofs. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 106–111. Lawrence Erlbaum.

McDougal, T. (1993). Using case-based reasoning and situated activity to write geometry proofs. In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, pages 60–65. MIT Press.

Owens, C. (1990). Representing abstract plan failures. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*, pages 277–284. Lawrence Erlbaum.

Owens, C. (1991). A functional taxonomy of abstract plan failures. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*, pages 167–172. Lawrence Erlbaum.

Owens, C. (1992a). Problem-solving stereotypes for an intelligent assistant. In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, pages 1026–1031. Lawrence Erlbaum.

Owens, C. (1992b). Qualitative relevance feedback and incremental query formulation. In *AAAI Spring Symposium on Adaptation and Reuse*, Palo Alto, CA.

Owens, C. (1993). Integrating feature extraction and memory search. *Machine Learning*, 10:311–339.

Schank, R. C. & Abelson, R. (1977). *Scripts, Plans, Goals and Understanding.* Lawrence Erlbaum Associates, Hillsdale, New Jersey.

Schank, R. (1982). *Dynamic Memory: A Theory of Reminding and Learning in Computers and People.* Cambridge University Press.

Seifert, C., Hammond, K., Johnson, H., Converse, T., McDougal, T., & Vanderstoep, S. (1994). Case-based learning: Predictive features in indexing. In *The Journal of Machine Learning.* In Press.

DISTRIBUTION LIST

addresses	number of copies
DR NORTHRUP FOWLER III ROME LABORATORY/C3C 525 BROOKS RD GRIFFISS AFB NY 13441-4505	10
THE UNIVERSITY OF CHICAGO DEPARTMENT OF COMPUTER SCIENCE ATTN: DR KRISTIAN HAMMOND 1100 EAST 58TH STREET CHICAGO IL 60637	5
RL/SUL TECHNICAL LIBRARY 26 ELECTRONIC PKY GRIFFISS AFB NY 13441-4514	1
ADMINISTRATOR DEFENSE TECHNICAL INFO CENTER DTIC-FDAC CAMERON STATION BUILDING 5 ALEXANDRIA VA 22304-6145	2
ADVANCED RESEARCH PROJECTS AGENCY 3701 NORTH FAIRFAX DRIVE ARLINGTON VA 22203-1714	1
NAVAL WARFARE ASSESSMENT CENTER GIDEP OPERATIONS CENTER/CODE QA-50 ATTN: E RICHARDS CORONA CA 91718-5000	1
HQ ACC/DRIY ATTN: MAJ. DIVINE LANGLEY AFB VA 23665-5575	1
WRIGHT LABORATORY/AAAI-4 WRIGHT-PATTERSON AFB OH 45433-6543	1

WRIGHT LABORATORY/AAAI-2 1
ATTN: MR FRANKLIN HUTSON
WRIGHT-PATTERSON AFB OH 45433-6543

AFIT/LDEE 1
2950 P STREET
WRIGHT-PATTERSON AFB OH 45433-6577

WRIGHT LABORATORY/MTEL 1
WRIGHT-PATTERSON AFB OH 45433

AAMRL/HE 1
WRIGHT-PATTERSON AFB OH 45433-6573

AUL/LSE 1
BLDG 1405
MAXWELL AFB AL 36112-5564

US ARMY STRATEGIC DEF 1
CSSD-IM-PA
PO BOX 1500
HUNTSVILLE AL 35807-3801

COMMANDING OFFICER 1
NAVAL AVIONICS CENTER
LIBRARY D/765
INDIANAPOLIS IN 46219-2189

COMMANDING OFFICER 1
NCCOSC RDTE DIVISION
CODE 02748, TECH LIBRARY
53560 HULL STREET
SAN DIEGO CA 92152-5001

CMDR 1
NAVAL WEAPONS CENTER
TECHNICAL LIBRARY/C3431
CHINA LAKE CA 93555-6001

SPACE & NAVAL WARFARE SYSTEMS COMM 1
WASHINGTON DC 20363-5100

CDR, U.S. ARMY MISSILE COMMAND 2
REDSTONE SCIENTIFIC INFO CENTER
AMSMI-RD-CS-R/ILL DOCUMENTS
REDSTONE ARSENAL AL 35898-5241

ADVISORY GROUP ON ELECTRON DEVICES 2
ATTN: DOCUMENTS
2011 CRYSTAL DRIVE,SUITE 307
ARLINGTON VA 22202

LOS ALAMOS NATIONAL LABORATORY 1
REPORT LIBRARY
MS 5000
LOS ALAMOS NM 87544

AEDC LIBRARY 1
TECH FILES/MS-100
ARNOLD AFB TN 37389

COMMANDER/USAISC 1
ATTN: ASOP-DO-TL
BLDG 61801
FT HUACHUCA AZ 85613-5000

AIR WEATHER SERVICE TECHNICAL LIB 1
FL 4414
SCOTT AFB IL 62225-5458

AFIWC/MSO 1
102 HALL BLVD STE 315
SAN ANTONIO TX 78243-7016

SOFTWARE ENGINEERING INST (SEI) 1
TECHNICAL LIBRARY
5000 FORBES AVE
PITTSBURGH PA 15213

DIRECTOR NSA/CSS 1
W157
9800 SAVAGE ROAD
FORT MEADE MD 21055-6000

NSA 1
E323/HC
SAB2 DOOR 22
FORT MEADE MD 21055-6000

NSA 1
ATTN: D. ALLEY
DIV X911
9800 SAVAGE ROAD
FT MEADE MD 20755-6000

DOD 1
R31
9800 SAVAGE ROAD
FT. MEADE MD 20755-6000

DIRNSA 1
R509
9800 SAVAGE ROAD
FT MEADE MD 20775

ESC/IC 1
50 GRIFFISS STREET
HANSCOM AFB MA 01731-1619

ESC/AV 1
20 SCHILLING CIRCLE
HANSCOM AFB MA 01731-2816

FL 2807/RESEARCH LIBRARY 1
OL AA/SULL
HANSCOM AFB MA 01731-5000

TECHNICAL REPORTS CENTER 1
MAIL DROP D130
BURLINGTON ROAD
BEDFORD MA 01731

DEFENSE TECHNOLOGY SEC ADMIN (DTSA) 1
ATTN: STTD/PATRICK SULLIVAN
400 ARMY NAVY DRIVE
SUITE 300
ARLINGTON VA 22202

MS. KAREN ALGUIRE 1
RL/C3CA
525 BROOKS RD
GRIFFISS AFB NY 13441-4505

JAMES ALLEN 1
COMPUTER SCIENCE DEPT/BLDG RM 732
UNIV OF ROCHESTER
WILSON BLVD
ROCHESTER NY 14627

MR. ROGER ALEX 1
DIGITAL SYSTEMS RSCH INC
4301 NORTH FAIRFAX DRIVE
SUITE 725
ARLINGTON VA 22203

YIGAL ARENS 1
USC-ISI
4676 ADMIRALTY WAY
MARINA DEL RAY CA 90292

MR. RAY BAREISS 1
THE INST. FOR LEARNING SCIENCES
NORTHWESTERN UNIV
1890 MAPLE AVE
EVANSTON IL 60201

MR. JEFF BERLINER 1
BBN SYSTEMS & TECHNOLOGIES
10 MOULTON STREET
CAMBRIDGE MA 02138

MARIE A. BIENKOWSKI 1
SRI INTERNATIONAL
333 RAVENSWOOD AVE/EK 337
MENLO PRK CA 94025

DR MARK S. BODDY 1
HONEYWELL SYSTEMS & RSCH CENTER
3660 TECHNOLOGY DRIVE
MINNEAPOLIS MN 55418

PIERO P. BONISSONE
GE CORPORATE RESEARCH & DEVELOPMENT
BLDG K1-RM 5C-32A
P. O. BOX 8
SCHENECTADY NY 12301 1

MR. DAVID BROWN
MITRE
EAGLE CENTER 3, SUITE 8
O'FALLON IL 62269 1

MR. MARK BURSTEIN
BBN SYSTEMS & TECHNOLOGIES
10 MOULTON STREET
CAMBRIDGE MA 02138 1

MR. GREGG COLLINS
INST FOR LEARNING SCIENCES
1890 MAPLE AVE
EVANSTON IL 60201 1

MR. RANDALL J. CALISTRI-YEH
ORA CORPORATION
301 DATES DRIVE
ITHACA NY 14850-1313 1

DR STEPHEN E. CROSS
SCHOOL OF COMPUTER SCIENCE
CARNEGIE MELLON UNIVERSITY
PITTSBURGH PA 15213 1

MS. JUDITH DALY
ARPA/ASTO
3701 N. FAIRFAX DR., 7TH FLOOR
ARLINGTON VA 22209-1714 1

THOMAS CHEATHAM
HARVARD UNIVERSITY
DIV OF APPLIED SCIENCE
AIKEN, RM 104
CAMBRIDGE MA 02138 1

MS. LAURA DAVIS
CODE 5510
NAVY CTR FOR APPLIED RES IN AI
NAVAL RESEARCH LABORATORY
WASH DC 20375-5337 1

MS. GLADYS CHOW 1
COMPUTER SCIENCE DEPT.
UNIV OF CALIFORNIA
LOS ANGELES CA 90024

THOMAS L. DEAN 1
BROWN UNIVERSITY
DEPT OF COMPUTER SCIENCE
P.O. BOX 1910
PROVIDENCE RI 02912

WESLEY CHU 1
COMPUTER SCIENCE DEPT
UNIV OF CALIFORNIA
LOS ANGELES CA 90024

MR. ROBERTO DESIMONE 1
SRI INTERNATIONAL (EK335)
333 RAVENSWOOD AVE
MENLO PRK CA 94025

PAUL R. COHEN 1
UNIV OF MASSACHUSETTS
COINS DEPT
LEDERLE GRC
AMHERST MA 01003

MS. MARIE DEJARDINS 1
SRI INTERNATIONAL
333 RAVENSWOOD AVENUE
MENLO PRK CA 94025

JON DOYLE 1
LABORATORY FOR COMPUTER SCIENCE
MASS INSTITUTE OF TECHNOLOGY
545 TECHNOLOGY SQUARE
CAMBRIDGE MA 02139

DR. BRIAN DRABBLE 1
AI APPLICATIONS INSTITUTE
UNIV OF EDINBURGH/80 S. BRIDGE
EDINBURGH EH1 LHN
UNITED KINGDOM

MR. SCOTT FOUSE 1
ISX CORPORATION
4353 PARK TERRACE DRIVE
WESTLAKE VILLAGE CA 91361

MR. STU DRAPER
MITRE
EAGLE CENTER 3, SUITE 8
O'FALLON IL 62269
1

MARK FOX
DEPT O INDUSTRIAL ENGRG
UNIV OF TORONTO
4 TADDLE CREAK ROAD
TORONTO, ONTARIA, CANADA
1

MR. GARY EDWARDS
4353 PARK TERRACE DRIVE
WESTLAKE VILLAGA 91361
1

MS. MARTHA FARINACCI
MITRE
7525 COLSHIRE DRIVE
MCLEAN VA 22101
1

MR. RUSS FREW
GENERAL ELECTRIC
MOORESTOWN CORPORATE CENTER
BLDG ATK 145-2
MOORESTOWN NJ 08057
1

MICHAEL FEHLING
STANFORD UNIVERSITY
ENGINEERING ECO SYSTEMS
STANFORD CA 94305
1

MR. RICH FRITZSON
CENTER OR ADVANCED INFO TECHNOLOGY
UNISYS
P.O. BOX 517
PAOLI PA 19301
1

MR KRISTIAN J. HAMMOND
UNIV OF CHICAGO
COMPUTER SCIENCE DEPT/RY155
1100 E. 58TH STREET
CHICAGO IL 60637
1

MR. ROBERT FROST
MITRE CORP
WASHINGTON C3 CENTER, MS 644
7525 COLSHIER ROAD
MCLEAN VA 22101-3481
1

RICK HAYES-ROTH 1
CINFLEX-TEKNOWLEDGE
1810 EMBARCADERO RD
PALO ALTO CA 94303

RANDY GARRETT 1
INST FOR DEFENSE ANALYSES (IDA)
1801 N. BEAUREGARD STREET
ALEXANDRA VA 22311-1772

MR. JIM HENDLER 1
UNIV OF MARYLAND
DEPT OF COMPUTER SCIENCE
OLLEGE PARK MD 20742

MS. YOLANDA GIL 1
USC/ISI
4676 ADMIRALTY WAY
MARINA DEL RAY CA 90292

MR.MAX HERION 1
ROCKWELL INTERNATIONAL SCIENCE CTR
444 HIGH STREET
PALO ALTO CA 94301

MR. STEVE GOYA 1
DISA/JIEO/GS11
CODE T3D
11440 ISAAC NEWTON SQ
RESTON VA 22090

MR. MORTON A. HIRSCHBERG, DIRECTOR 1
US ARMY RESEARCH LABORATORY
ATTN; AMSRL-CI-CB
ABERDEEN PROVING GROUND MD
21005-5066

MR. MARK A. HOFFMAN 1
ISX CORPORATION
1165 NORTHCHASE PARKWAY
MARIETTA GA 30067

MR. RON LARSEN 1
NAVAL CMD, CONTROL & OCEAN SUR CTR
RESEARCH, DEVELOP, TEST & EVAL DIV
CODE 444
SAN DIEGO CA 92152-5000

DR. JAMES JUST
MITRE
DEPT. W032-M/S Z360
7525 COLSHIER RD
MCLEAN VA 22101 — 1

MR. CRAIG KNOBLOCK
USC-ISI
4676 ADMIRALTY WAY
MARINA DEL RAY CA 90292 — 1

MR. RICHARD LOWE (AP-10)
SRA CORPORATION
2000 15TH STREET NORTH
ARLINGTON VA 22201 — 1

MR. TED C. KRAL
BBN SYSTEMS & TECHNOLOGIES
4015 HANCOCK STREET, SUITEE 101
SAN DIEGO CA 92110 — 1

MR. JOHN LOWRENCE
SRI INTERNATIONAL
ARTIFICIAL INTELLIGENCE CENTER
333 RAVENSWOOD AVE
MENLO PARK CA 94025 — 1

DR. ALAN MEYROWITZ
NAVAL RESEARCH LABORATORY/CODE 5510
4555 OVERLOOK AVE
WASH DC 20375 — 1

ALICE MULVEHILL
MITRE CORPORATION
BURLINGTON RD
M/S K-302
BEDFORD MA 01730 — 1

ROBERT MACGREGOR
USC/ISI
4676 ADMIRALTY WAY
MARINA DEL REY CA 90292 — 1

WILLIAM S. MARK, MGR AI CENTER
LOCKHEED MISSILES & SPACE CENTER
1801 PAGE MILL RD
PALO ALTO CA 94304-1211 — 1

RICHARD MARTIN 1
SOTWARE ENGINEERING INSTITUTE
CARNEGIE MELLON UNIV
PITTSBURGH PA 16213

DREW MCDERMOTT 1
YALE COMPUTER SCIENCE DEPT
P.O. BOX 2158, YALE STATION
51 PROPSPECT STREET
MEW HAVEN CT 06520

MS. CECILE PARIS 1
USC/ISI
4676 ADMIRALTY WAY
MARINA DEL RAY CA 90292

DOUGLAS SMITH 1
KESTREL INSTITUTE
3260 HILLVIEW AVE
PALO ALTO CA 94304

DR. AUSTIN TATE 1
AI APPLICATIONS INSTITUTE
UNIV OF EDINBURGH
80 SOUTH BRIDGE
EDINBURGH EH1 1HN - SCOTLAND

MS. REGINA SMITH 1
MERIDIAN CORPORATION
4001 NORTH FAIRFAX DRIVE
SUITE 200
ARLINGTON VA 2203-1714

EDWARD THOMPSON 1
ARPA/SISTO
3701 N. FAIRFAX DR., 7TH FL
ARLINGTON VA 22209-1714

MR. STEPHEN F. SMITH 1
ROBOTICS INSTITUTE/CMU
SCHENLEY PRK
PITTSBURGH PA 15213

LTCOL RAYMOND STACHA 1
DEPUTY SCIENTIFIC & TECHNICAL
ADVISOR
HQ USCINCPAC/STA
CAMP H. M. SMITH HI 96861

DR. ABRAHAM WAKSMAN 1
AFOSR/NM
110 DUNCAN AVE., SUITE B115
BOLLING AFB DC 20331-0001

JONATHAN P.STILLMAN 1
GENERAL ELECTRIC CRD
1 RIVER RD, RM K1-5C31A
P. O. BOX 8
SCHENECTADY NY 12345

MR. EDWARD C. T. WALKER 1
BBN SYSTEMS & TECHNOLOGIES
10 MOULTON STREET
CAMBRIDGE MA 02138

MR. BILL SWARTOUT 1
USC/ISI
4676 ADMIRALTY WAY
MRINA DEL RAY CA 90292

GIO WIEDERHOLD 1
PROGRAM MGR FOR KB SYSTEMS
ARPA/SISTO
3701 NORTH AIRFX DRIVE, RM 739
ARLINGTON VA 22203-1714

KATIA SYCARA/THE ROBOTICS INST 1
SCHOOL OF COMPUTER SCIENCE
CARNEGIE MELLON UNIV
DOHERTY HALL RM 3325
PITTSBURGH PA 15213

MR. DAVID E. WILKINS 1
SRI INTERNATIONAL
ARTIFICIAL INTELLIGENCE CENTER
333 RAVENSWOOD AVE
MENLO PRK CA 94025

DR. PATRICK WINSTON 1
MASS INSTITUTE OF TECHNOLOGY
RM NE43-817
545 TECHNOLOGY SQUARE
CAMBRIDGE MA 02139

HUA YANG 1
COMPUTER SCIENCE DEPT
UNIV OF CALIORNIA
LOS ANGELES CA 90024

LTCOL DAVE NEYLAND 1
ARPA/ISTO
3701 N. FAIRFAX DRIVE, 7TH FLOOR
ARLINGTON VA 22209-1714

MR. RICK SCHANTZ 1
BBN SYSTEMS & TECHNOLOGIES
10 MOULTON STREET
CAMBRIDGE MA 02138

LTC FRED M. RAWCLIFFE 1
USTRANSCOM/TCJ5-SC
BLDG 1900
SCOTT AFB IL 62225-7001

JOHN P. SCHILL 1
NAVAL COMMAND, CONTROL & OCEAN
SURVEILLANCE CENTER/CODE 423
EVALUATION DIVISION
SAN DIEGO CA 92152-5000

MR. DONALD F. ROBERTS 1
RL/C3CA
BLDG 3
525 BROOKS RD
GRIFFISS AFB NY 13441-4505

MR. DAVE SCHNEEGAS 1
USPACOM/J3
CAMP SMITH, HI 9686L-5025

ALLEN SEARS 1
MITRE
7525 COLESHIRE DRIVE, STOP Z289
MCLEAN VA 22101

STEVE ROTH 1
CENTER FOR INTEGRATED MANUFACTURING
THE ROBOTICS INSTITUTE
CARNEGIE MELLON UNIV
PITTSBURGHJ PA 15213-3890

JEFF ROTHENBERG 1
SENIOR COMPUTER SCIENTIST
THE RAND CORPORATION
1700 MAIN STREET
SANTA MONICA CA 90407-2138

YOAV SHOHAM 1
STANFORD UNIVERSITY
COMPUTER SCIENCE DEPT
STANFORD CA 94305

MR. DAVID B. SKALAK 1
UNIV OF MASSACHUSETTS
DEPT OF COMPUTER SCIENCE
RM 243, LGRC
AMHERST MA 01003

MR. BILL ROUSE 1
SEARCH TECHNOLOGY
4725 PEACH TREE CORNER CIRCLE
SUITE 200
NORCROSS GA 30092

MR. MIKE ROUSE 1
AFSC
7800 HAMPTON RD
NORFOLD VA 23511-6097

MR. DAVID E. SMITH 1
ROCKWELL INTERNATIONAL
444 HIGH STREET
PALO ALTO CA 94301

JEFF ROTHENBERG 1
SENIOR COMPUTER SCIENTIST
THE RAND CORPORATION
1700 MIN STREET
SANTA MONIA CA 90407-2138

DR LARRY BIRNBAUM 1
NORTHWESTERN UNIVERSITY
ILS
1890 MAPLE AVE
EVANSTON IL 60201

MR RANDALL J. CALISTRI-YEH 1
ORA
301 DATES DR
ITHACA NY 14850-1313

MR WESLEY CHU 1
COMPUTER SCIENCE DEPT
UNIVERSITY OF CALIFORNIA
LOS ANGELES CA 9002

MR PAUL R COHEN 1
UNIVERSITY OF MASSACHUSETTS
COINS DEPT, LEDERLE GRC
AMHERTS MA 01003

MR DON EDDINGTON 1
NAVAL COMMAND, CONTROL & OCEAN
SURV CENTER
RDT&E DIVISION, CODE 404
SAN DIEGO CA 92152-5000

MR. LEE ERMAN 1
CIMFLEX TECKNOWLEDGE
1810 EMBARCARDERO RD
PALO ALTO CA 94303

MR DICK ESTRADA 1
BBN SYSTEMS & TECHNOLOGIES
10 MOULTON ST
CAMBRIDGE MA 02138

MR HARRY FORSDICK 1
BBN SYSTEMS AND TECHNOLOGIES
10 MOULTON ST
CAMBRIDGE MA 02138

MR MATTHEW L. GINSBERG 1
CIRL, 1269
UNIVERSITY OF OREGON
EUGENE OR 97403

MR IRA GOLDSTEIN 1
OPEN SW FOUNDATION RESEARCH INST
ONE CAMBRIDGE CENTER
CAMBRIDGE MA 02142

MR MOISES GOLDSZMIDT 1
INFORMATION AND DECISION SCIENCES
ROCKWELL INTL SCIENCE CENTER
444 HIGH ST, SUITE 400
PALO ALTO CA 94301

MR JEFF GROSSMAN, CO 1
NCCOSC RDTE DIV 44
5370 SILVERGATE AVE, ROOM 1405
SAN DIEGO CA 92152-5146

JAN GUNTHER
ASCENT TECHNOLOGY, INC.
64 SIDNEY ST, SUITE 380
CAMBRIDGE MA 02139 1

DR LYNETTE HIRSCHMAN
MITRE CORPORATION
202 BURLINGTON RD
BEDFORD MA 01730 1

MS ADELE E. HOWE
COMPUTER SCIENCE DEPT
COLORADO STATE UNIVERSITY
FORT COLLINS CO 80523 1

DR LESLIE PACK KAELBLING
COMPUTER SCIENCE DEPT
BROWN UNIVERSITY
PROVIDENCE RI 02912 1

SUBBARAO KAMBHAMPATI
DEPT OF COMPUTER SCIENCE
ARIZONA STATE UNIVERSITY
TEMPE AZ 85287-5406 1

MR THOMAS E. KAZMIERCZAK
SRA CORPORATION
331 SALEM PLACE, SUITE 200
FAIRVIEW HEIGHTS IL 62208 1

PRADEEP K. KHOSLA
ARPA/SSTO
3701 N. FAIRFAX DR
ARLINGTON VA 22203 1

MR CRAIG KNOBLOCK
USC-ISI
4676 ADMIRALTY WAY
MARINA DEL RAY CA 90292 1

MRS CARLA LUDLOW
ROME LABORATORY/C3CA
525 BROOKS RD
GRIFFISS AFB NY 13441-4505 1

DR MARK T. MAYBURY
ASSOCIATE DIRECTOR OF AI CENTER
ADVANCED INFO SYSTEMS TECH G041
MITRE CORP, BURLINGTON RD, MS K-329
BEDFORD MA 01730 1

MR DONALD P. MCKAY
PARAMAX/UNISYS
P O BOX 517
PAOLI PA 19301 1

DR KAREN MYERS
AI CENTER
SRI INTERNTIONAL
333 RAVENSWOOD
MENLO PARK CA 94025 1

DR MARTHA E POLLACK
DEPT OF COMPUTER SCIENCE
UNIVERSITY OF PITTSBURGH
PITTSBURGH PA 15260 1

RAJ REDDY
SCHOOL OF COMPUTER SCIENCE
CARNEGIE MELLON UNIVERSITY
PITTSBURGH PA 15213 1

EDWINA RISSLAND
DEPT OF COMPUTER & INFO SCIENCE
UNIVERSITY OF MASSACHUSETTS
AMHERST MA 01003 1

MR NORMAN SADEH
CIMDS
THE ROBOTICS INSTITUTE
CARNEGIE MELLON UNIVERSITY
PITTSBURGH PA 15213 1

MR ERIC TIFFANY
ASCENT TECHNOLOGY INC.
237 LONGVIEW TERRACE
WILLIAMSTOWN MA 01267 1

MANUELA VELOSO
CARNEGIE MELLON UNIVERSITY
SCHOOL OF COMPUTER SCIENCE
PITTSBURGH PA 15213-3891 1

MR DAN WELD 1
DEPT OF COMPUTER SCIENCE & ENG
MAIL STOP FR-35
UNIVERSITY OF WASHINGTON
SEATTLE WA 98195

MR CRAIG WIER 1
ARPA/SISTO
3701 N. FAIRFAX DR
ARLINGTON VA 22203

MR JOE ROBERTS 1
ISX CORPORATION
2231 CRYSTAL DRIVE, SUITE 500
ARLINGTON VA 22202

COL JOHN A. WARDEN III 1
ASC/CC
225 CHENNAULT CIRCLE
MAXWELL AFB AL 36112-6426

DR TOM GARVEY 1
ARPA/SISTO
3701 NORTH FAIRFAX DRIVE
ARLINGTON VA 22203-1714

MR JOHN N. ENTZMINGER, JR. 1
ARPA/DIRO
3701 NORTH FAIRFAX DRIVE
ARLINGTON VA 22203-1714

LT COL ANTHONY WAISANEN, PHD 1
COMMAND ANALYSIS GROUP
HQ AIR MOBILITY COMMAND
402 SCOTT DRIVE, UNIT 3L3
SCOTT AFB IL 62225-5307

DIRECTOR 1
ARPA/SISTO
3701 NORTH FAIRFAX DRIVE
ARLINGTON VA 22203-1714

*U.S. GOVERNMENT PRINTING OFFICE: 1994-510-117-50019

MISSION

OF

ROME LABORATORY

Mission. The mission of Rome Laboratory is to advance the science and technologies of command, control, communications and intelligence and to transition them into systems to meet customer needs. To achieve this, Rome Lab:

a. Conducts vigorous research, development and test programs in all applicable technologies;

b. Transitions technology to current and future systems to improve operational capability, readiness, and supportability;

c. Provides a full range of technical support to Air Force Materiel Command product centers and other Air Force organizations;

d. Promotes transfer of technology to the private sector;

e. Maintains leading edge technological expertise in the areas of surveillance, communications, command and control, intelligence, reliability science, electro-magnetic technology, photonics, signal processing, and computational science.

The thrust areas of technical competence include: Surveillance, Communications, Command and Control, Intelligence, Signal Processing, Computer Science and Technology, Electromagnetic Technology, Photonics and Reliability Sciences.