RL-TR-97-34
Final Technical Report
July 1997

MACHINE LEARNING FOR MILITARY OPERATIONS

SRI International

Sponsored by
Advanced Research Projects Agency
ARPA Order No. A005

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Materiel Command
Rome, 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-97-34 has been reviewed and is approved for publication.

APPROVED: 

LOUIS J. HOEBEL
Project Engineer

FOR THE COMMANDER: 

JOHN A. GRANIERO, Chief Scientist
Command, Control & Communications

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/C3CA, 525 Brooks Rd, Rome, NY 13441-4505. 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.

MACHINE LEARNING FOR MILITARY OPERATIONS

Contractor: SRI International
Contract Number: F30602-93-C-0071
Effective Date of Contract: 1 April 1993
Contract Expiration Date: 31 May 1996
Program Code Number: 3E20
Short Title of Work: Machine Learning for Military Operations

Period of Work Covered: Apr 93 - May 96

Principal Investigator: Marie E desJardins
Phone: (415) 859-6323
RL Project Engineer: Louis J. Hoebel
Phone: (315) 330-3655

This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by Louis J. Hoebel, RL/C3CA, 525 Brooks Rd, Rome, NY.

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	July 1997	Final Apr 93 - May 96

4. TITLE AND SUBTITLE

MACHINE LEARNING FOR MILITARY OPERATIONS

5. FUNDING NUMBERS

C - F30602-93-C-0071
PE - 602301E, 603728F
PR - A005
TA - 00
WU - 01

6. AUTHOR(S)

Marie E. desJardins

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

SRI
333 Ravenswood Ave.
Menlo Park CA 94025

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/C3CA
525 Brooks Rd
Rome, NY 13441-4505

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

RL-TR-97-34

11. SUPPLEMENTARY NOTES

Rome Laboratory Project Engineer: Louis J. Hoebel/C3CA/315-330-3655

12a. DISTRIBUTION AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This report covers the application of Machine Learning Techniques to building planning operators. Specifically, developing intelligent tools for acquiring, refining, and validating knowledge bases for operations planning systems.

14. SUBJECT TERMS

Second generation digital optoelectronic computer (DOCII), text search primitives, relational database algorithms, knowledge base

15. NUMBER OF PAGES

48

16. PRICE CODE

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

CONTENTS

FIGURES

INTRODUCTION

SRI International (SRI) is pleased to present this final report under Contract F30602-93-C-0071, Machine Learning for Military Operations Planning, to Rome Laboratory (RL) of the United States Air Force (USAF) and the Defense Advanced Research Projects Agency (DARPA). The work described is part of the DARPA/RL Planning Initiative (ARPI): specifically, the progress made during this contract to develop intelligent tools for acquiring, refining, and validating knowledge bases for a military operations planning system.

As with most large-scale AI projects, one of the most significant obstacles to developing intelligent decision support systems is knowledge engineering. Large quantities of domain knowledge are required for the effective reasoning about large-scale, real-world problems. Therefore, tools for knowledge acquisition must be developed before fully operational crisis response planning systems can be built. We emphasize the use of the term knowledge *acquisition*, rather than knowledge *engineering*, to distinguish our goal, building tools that take an active role in acquiring knowledge, from the more traditional approach of an AI expert, creating a knowledge representation and encoding the knowledge base by hand.

It is our view that knowledge acquisition tools should (1) enable human planners to transfer their expertise to the system, (2) support the acquisition of knowledge from on-line sources, and (3) integrate information from the range of available sources, including human experts, simulators, on-line databases, training exercises, and actual crises. To meet these needs, SRI has developed the Knowledge Acquisition Toolkit (KATY), a package of knowledge acquisition tools for the System for Interactive Planning and Execution (SIPE-2), SRI's generative planning system. KATY includes three knowledge editing tools (the Operator Editor, Object Editor, and Predicate Editor) and two machine learning tools (the Operator Learner and the Probabilistic Autonomous GOal-Directed Agent [PAGODA], an inductive learning system).

The graphical Operator Editor allows users to develop new planning operators and revise existing operators. This tool also supports editing of the object hierarchy via an interface from SIPE-2 to SRI's Generic Knowledge Base (GKB) Editor.* The Predicate Editor allows user to view, add, and modify the predicates that define the world state.

The Operator Learner uses the PAGODA learning model [desJardins 1992] to acquire planning operators via feedback from simulators and from the user's planning processes. The user first enters partially specified operators that reflect an initial rough description of how a subgoal may be solved. The Operator Learner then "fills in the blanks," identifying appropriate preconditions and temporal constraints on the application of the operator. Thus, the user contributes expertise, while the automated learning tool performs much of the tedious work of developing preconditions and identifying precisely when it is appropriate to apply a particular operator.

PAGODA, which was originally developed as part of Dr. Marie desJardins's dissertation research [desJardins 1992], has been enhanced under this contract. Specifically, SRI made a number of extensions and improvements to PAGODA to support the inductive learning required for the Operator Learner, and developed a generic interface for describing input features and training examples.

*The GKB Editor was developed by SRI under Contract F30602-94-C-0263 to Rome Laboratory, Generic Knowledge Base Browser and Editor.

This report describes the progress made on the development of KATY during the 3-year contract. Section 1 summarizes our accomplishments. Section 2 defines the problems we addressed and the rationale for our solutions. KATY and its components are described in Section 3. Section 4 describes the various simulators that we reviewed in developing the current demonstration scenario. Evaluation metrics are discussed in Section 5. Related work is surveyed in Section 6. Section 7 contains a list of publications written under this contract. In Section 8 we present our conclusions and describe future work. In Section 9 we list referenced documents.

1 SUMMARY OF ACCOMPLISHMENTS

Our accomplishments during this contract are listed below and are described in detail in the cited sections.

- We ported PAGODA, an inductive machine learning system [desJardins 1992], from ZetaLisp/Flavors* into Lucid Common Lisp/Common Lisp Object System (CLOS). The system now runs on any standard Common Lisp and/or CLOS platform. We also rewrote substantial parts of the system to increase its efficiency and generality (Subsection 3.2.5).
- We implemented an Operator Editor (Subsection 3.1) based on SRI's Act Editor [Wilkins et al. 1994]. Many of our extensions of the Operator Editor have been incorporated into the Act Editor.
- We implemented an Operator Learner that uses qualitative constraints on a partial operator to create a series of experiments, use those experiments to generate plans, evaluate the plans, extract training instances, and apply inductive methods to the instances in order to learn preconditions for operator application (Subsection 3.2). We extended this system to support learning from the user's choices, by creating training examples for each operator choice the user makes during planning (Subsection 3.2.5).
- We developed and implemented an abstraction language for qualitative constraints that allows the user to specify what information is relevant to the success of the operator, without actually writing specific preconditions (Subsection 3.2.1).
- We developed a generic input language for PAGODA (Subsection 3.2.5). This language enables PAGODA to accept training examples in a range of formats, and can also serve as a generic interface from the Operator Learner to other inductive learning systems.
- We created demonstration scenarios for the military and oil-spill application domains, and explored a number of simulators and evaluation tools in the military domain (Section 4).
- We ported SOCAP† and KATY to Allegro Common Lisp and CLIM 2.0.

*All project and company names mentioned in this document are the trademarks of their respective holders.
†SOCAP: System for Operations Crisis Action Planning [Bienkowski 1995].

- We created a World Wide Web home page for the project (http://www.erg.sri.com/people/marie/papers/ml-summary.html).
- We presented project-related talks at the 1994 IEEE Conference on Tools with AI [desJardins 1994b], the 1994 Fall Symposium on Planning and Learning [desJardins 1994c], the 1994 Fall Symposium on Relevance [desJardins 1994d], the 1995 AAAI Fall Symposium on Active Learning, Stanford University, Carnegie Mellon University, the University of Massachusetts, Rome Laboratory, and several ARPI workshops.

2 MOTIVATION

One of the most time-consuming and critical tasks in the development of crisis response planning systems is knowledge engineering. For example, a significant part of the development effort for SOCAP [Bienkowski 1995], a prototype military operations planning system based on the AI generative planning system, SIPE-2, consisted of writing and debugging planning operators. This process required an AI expert; it would have been difficult to teach an AI-naive domain expert how to write and debug planning operators, using the few tools that SIPE-2 provided [Desimone et al. 1993].

Our experience suggested, however, that given the appropriate tools, human planners who were not AI experts could construct knowledge bases for AI planning systems. These tools should guide the user through the operator development and debugging process, and should embody the expertise about the planning representation that currently must be provided by an AI knowledge engineer. These tools should also support both the construction of the initial knowledge base and the updating of the knowledge over the life of the planning system. Continuous updating of the knowledge base is essential to ensure that the planning system is not constrained by a pre-existing knowledge base and can be used for unforeseen types of operations and situations. The tools will be used for knowledge acquisition and also will enable the system to learn—that is, to improve its performance over time.

3 KATY

Each operator in SIPE-2 specifies a method for achieving a single mission or task, including the actions required, preconditions for using the specified method, temporal constraints among the actions, and expected effects of the actions. Previously, these operators had to be developed manually by AI planning experts who edited ASCII descriptions of the operators. This process was tedious, prone to errors, and difficult for users who were not AI experts.

KATY is a package of knowledge acquisition tools that simplifies this process and reduces the likelihood of errors. If extended through future development, these tools will eventually allow AI-naive domain experts to transfer their knowledge directly to SIPE-2.

KATY provides two types of tools: knowledge editors and knowledge refiners. *Knowledge editors* provide graphical editing functions for creating and modifying planning knowledge. KATY includes tools for editing operators (the Operator Editor), class hierarchies (the Object Editor), and knowledge about the state of the world (the Predicate Editor). *Knowledge refiners* automatically or interactively analyze and improve the knowledge base. KATY's Operator Learner is a knowledge refiner that learns the preconditions for applying planning operators by using evaluation feedback from a simulator, an automated evaluation tool, or the domain expert's planning choices.

Several critical issues are associated with knowledge acquisition tools in general, and tools based on automated learning in particular:

1. It is difficult to get domain experts to understand and use new tools.
2. Automated learning methods must rely on possibly inaccurate or incomplete data.
3. Training data may be difficult and expensive to collect, and training instances may be quite large in complex domains.

Consideration of these issues led us to define three desiderata for KATY:

1. Graphical interfaces and semiautomated verification techniques should be used to simplify user training.
2. Data from a variety of sources including multiple simulators and evaluation tools, as well as user behavior, should be incorporated into KATY, to reduce uncertainty in the acquired knowledge.
3. Partial knowledge and other guidance provided by the user should be used to reduce the size and number of required training instances.

In the first year of this effort, we implemented the Operator Editor. This graphical editing tool for planning operators enables users to develop new operators and edit existing ones. An intelligent interface guides users through the development process, ensuring that the knowledge is in the correct form. In the second year, we implemented the Operator Learner, an inductive learning module that tests and refines these partial operators. In the third and final year of the project, we extended the inductive learning system, improved the experiment generation and selection techniques of the Operator Learner, and extended the Operator Learner to observe and learn from expert planning behavior.

3.1 OPERATOR EDITOR

The Operator Editor provides a graphical interface (shown in Figure 1) for creating and modifying SIPE-2 planning operators. The *plot* of the operator is displayed as a graph, with nodes that correspond to actions and subgoals, and edges that indicate temporal relationships among these nodes. The other information associated with the operator can be displayed as buttons or text fields (like the preconditions in Figure 1).

The Operator Editor has four significant advantages over the methods previously provided in SIPE-2 for developing operators: first, the operators are easier to understand because they are displayed and edited graphically, rather than as ASCII text. Second, all editing is done via templates, so that the possibility of syntactic errors is avoided. Third, a data dictionary is maintained to ensure that all classes, variables, and predicates entered are in the correct format.

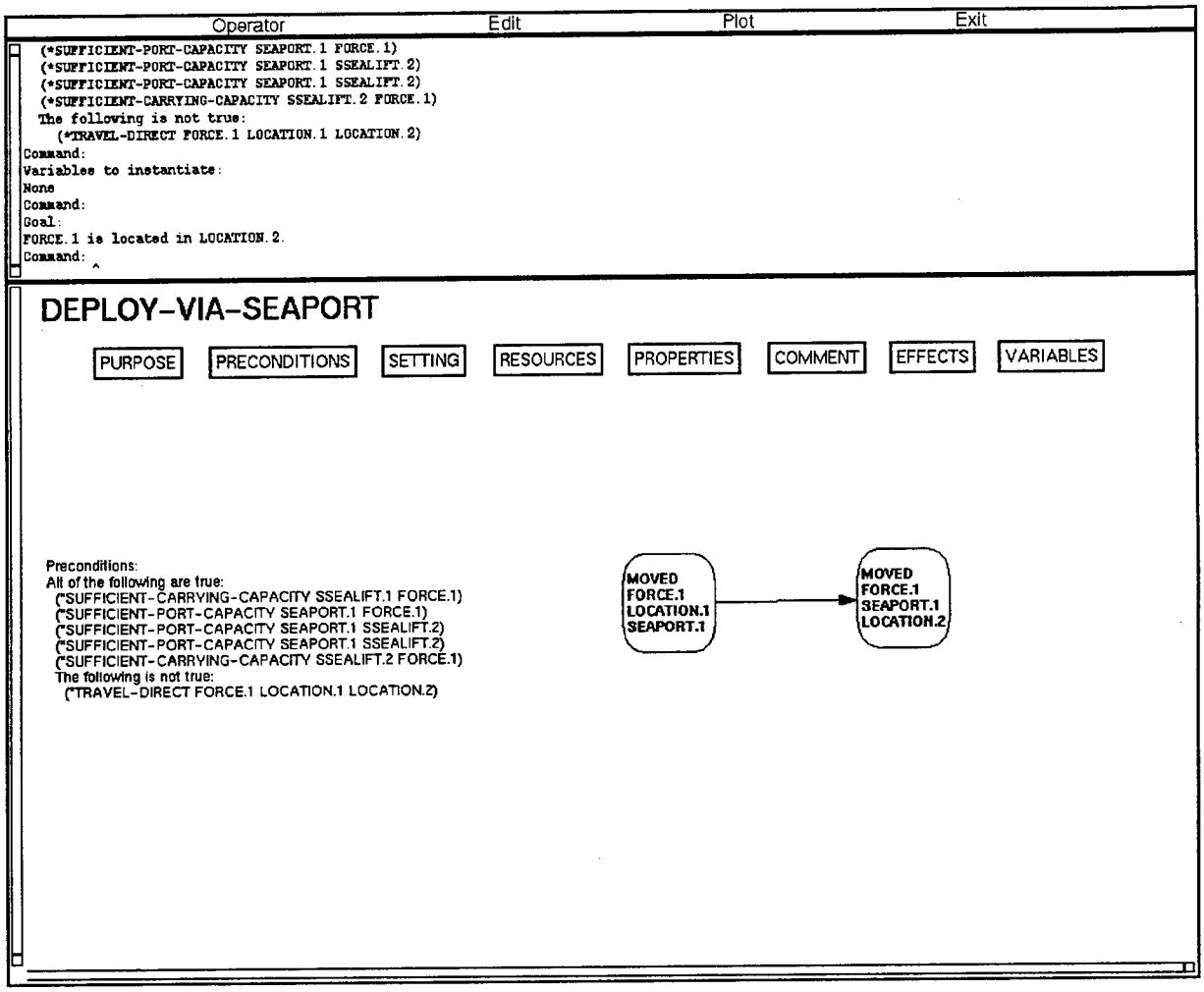

Figure 1. Operator Editor Graphical Display

Finally, developing the plot as a graph rather than as a text description of the branches in the plot is less prone to error, especially since the Operator Editor automatically maintains the parallel structure of the graph.

The method we have used to build the Operator Editor can best be described as *evolutionary development*. We used the Operator Editor at SRI to develop operators for the military planning domain and for a United States Coast Guard (USCG) project in which SIPE-2 was applied to oil-spill response planning [Desimone and Agosta 1993]. We also introduced additional functionality and updated the user interface, based on feedback from the ongoing usage of the editor. We plan to use the editor more extensively in a new NRaD*-funded project to apply SIPE-2 to maritime crisis action planning. The editor will thus continue to evolve, in response to requirements generated by its actual users.

*NRaD: Naval Command, Control, and Ocean Surveillance Center (NCCOSC) Research, Development, Test, and Evaluation Division.

The Operator Editor is based on SRI's Act Editor. The Act Editor uses the Act representation for procedural knowledge, which subsumes SIPE-2's operator representation. Because the Act representation used in the editor is different from SIPE-2's internal representation, operators must be translated back and forth between the two representations. (This translation is done automatically by the software.) We have tailored the Operator Editor to users who are familiar with SIPE-2's representation, but do not necessarily understand the details of the Act representation. For the remainder of this report we will refer to "operators," although in the editor they are internally represented as Acts.

The Operator Editor can be used during planning and replanning, as well as during knowledge development. In particular, whenever SIPE-2 fails to solve a problem, the user has the option of entering the Operator Editor. Any operators that the user adds or modifies during the Operator Editor session are made available to SIPE-2 when the editor is exited. For example, if SIPE-2 fails to solve a goal because no operator has a purpose that matches the goal, the user can enter the Operator Editor, build an operator that represents a subplan for solving that goal, and return to SIPE-2; the new operator will be used to expand the goal in the plan without any need for backtracking.

The Operator Editor provides consistency checking and type checking throughout operator development. The goal is to support the users by giving them as much assistance as possible, without constraining them to a particular model of operator development.

We developed an abstraction language for expressing qualitative constraints (QCs) (see Subsection 3.2.1) and used this language to extend the Operator Editor to support the writing of operators. Thus, users can develop partial operator descriptions that can be fed into the inductive learning tool. The QCs shown in Figure 2 are examples of abstract preconditions that must be instantiated by the machine learning system. For example, the second QC states that the sea-state (roughness of the water) in the sea sector where an operation is being performed (`sea-sector.1`), at the time of the operation (`latest.1`), is relevant to the success of the operation. The asterisk (`*`) indicates that the sea-state value is unknown (i.e., the range of acceptable values must be determined by the inductive learning system.) We will return to this example in Subsection 3.2.

All of the fields in an operator are displayed graphically on the screen. Normally, the plot nodes and edges are drawn as a graph, and the other fields (purpose, preconditions, etc.) appear as buttons. Left-clicking* on any button, node, or edge causes a description of the contents of that object to be printed in the interaction window. Middle-clicking on an object copies it into an edit buffer, which can then be pasted into another operator. Right-clicking edits the object.

We have developed a toolkit of CLIM-based *dialog boxes* that can be used to build templates for editing a variety of objects (an example is shown in Figure 3). Each field in a dialog box has an associated type that defines the set of legal completions, which the user can access as a menu of choices. If the user enters an object that is not in this set, the system either signals an error or defines a new object, depending on the context. Each domain's knowledge base has an associated *data dictionary*, which is built automatically by the Operator Editor and contains the set of known predicates (along with their arities and argument types), classes, and objects. This data dictionary is used to generate the completion sets in the dialog boxes.

*The phrases "left-clicking," "middle-clicking," and "right-clicking" mean clicking the left, middle, and right-hand mouse buttons.

```
OPERATOR: ml-get-boom1
ARGUMENTS: vessel1, boom-level1, numerical1, sea-sector1,
   sea-state1, boom1, numerical2 is (length-boom-ft boom1);
INSTANTIATE: boom1;
PURPOSE: (level>= vessel1 boom-level1 numerical1);
PRECONDITION: (in-service boom1);
Properties:
   NONLOCAL-VARS = (sea-state.1 sea-sector.1 latest.1),
   QC = ((%property max-sea-state boom.1 *)
      (sea-state sea-sector.1 latest.1 *));
PLOT:
PARALLEL
   BRANCH 1:
      GOAL
         GOALS: (located boom1 sea-sector1);
         RESOURCES: boom1;
      PROCESS
         ACTION: deploy-boom;
         ARGUMENTS: boom1, sea-sector1, vessel1, boom-level1,
            numerical2;
         RESOURCES: boom1;
         EFFECTS: (boom-deployed boom1 vessel1),
            (produce vessel1 boom-level1 numerical2);
   BRANCH 2:
      GOAL
         GOALS: (level>= vessel1 boom-level1 numerical1);
         ARGUMENTS: vessel1, boom-level1, numerical1, sea-sector1,
            sea-state1;
END PARALLEL
END PLOT END OPERATOR
```

Figure 2. ML-Get-Boom1 Operator

Dialog boxes are used during the execution of menu- and mouse-based editing commands, to add and delete nodes, edges, and values in the fields of the operator (e.g., preconditions and resources). Additionally, the user can reposition nodes in the graphical display and can toggle the appearance of operator fields (which can be viewed as buttons or as text boxes in the graphical display).

When the user adds or deletes an edge, the graph structure required by SIPE-2 is maintained automatically. In particular, no cycles are permitted, and whenever a node has two successors, there must be an intervening Split node and a corresponding Join node at the ends of the branches.

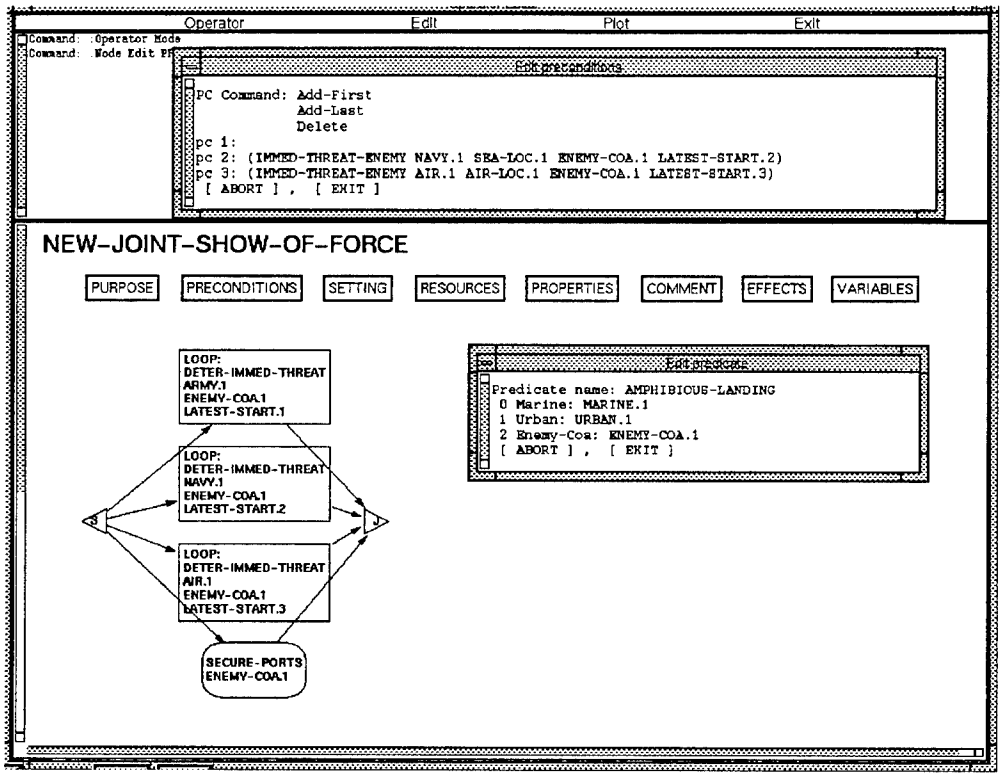

Figure 3. Dialog Box

3.2 OPERATOR LEARNER

The Operator Learner refines the partial operators created by the user in the Operator Editor by learning the preconditions that the planning system uses to determine when the operators can be applied.

The Operator Learner uses an operator's *qualitative constraints* (QCs): partial knowledge specified by the user) to generate a series of experiments, each of which specifies a set of constraints on the planning process. By varying these constraints, the Operator Learner tests the quality of the plans produced by using the operator under a range of conditions. An external module evaluates the plan, and the Operator Learner again uses the QCs to extract training instances describing the world state, plan, and evaluation result.

PAGODA analyzes these training instances to create a hypothesis of the conditions under which the operator is expected to succeed. The result is interpreted by the Operator Learner as a precondition for the operator, and is incorporated into the operator definition, subject to the expert user's approval.

The system learns preconditions via feedback from a simulator or by observing the user's planning choices. The simulator feedback tells the system what actions succeeded and how long it took to complete these actions. What will be learned in this case is a description of a particular operator's success in the simulator (i.e., the world state in which the operator succeeds); therefore, the learned knowledge is only as accurate and complete as the simulation. However, if multiple simulators are used, their "domains of expertise" could be combined for completeness.

The user's planning choices are indications of the user's belief that a particular operator will succeed. If the user chooses operator A over operator B in a particular context, it is assumed that operator A is "successful" in that context and operator B "fails." Typically, the user has knowledge that is not captured in the system (which is why the preferred mode of the planning system is interactive). Learning methods based on observing the user allow us to make explicit some of this previously unrepresented knowledge. This process and some problems that arise in its application are discussed in Subsection 3.2.5.

The architecture of the Operator Learner is shown in Figure 4. The Operator Learner is composed of four major processes (Operator Creation, Operator Refinement, Experiment Generation, and Data Generation), each with one or more subprocesses. These processes are described in the following subsections.

3.2.1 Operator Creation

The Operator Editor is used to create one or more operators with partial preconditions, represented as QCs, that represent user-provided guidance that may be partial or incomplete. The QCs are used to guide the learning process. By expressing QCs, users can intuitively specify abstract constraints on the operator (e.g., by specifying relevant properties for determining success), even when they cannot precisely and completely specify the actual constraints. Some examples of QCs follow:

- "Bad weather usually delays transportation actions, and air movements are more likely to be delayed than sea movements."
- "The equipment required to clean up an oil spill depends on the type and amount of oil, weather, water currents, and response time."
- "Republicans are less likely than Democrats to vote for social programs."

None of these QCs are precise enough to be used as preconditions by the planning system, but they significantly constrain the space of possible preconditions, making the automated learning of preconditions via refinement of the QCs computationally feasible. In the current KATY implementation, QCs represent generalizations of preconditions (predicates with underspecified arguments). The system fills these in by systematically varying the values of the arguments (see Subsection 3.2.2).

The QCs of an operator are specified on its properties slot, and can be created and modified in the Operator Editor. Each QC matches SIPE-2 predicates or properties of objects in the SIPE-2 sort hierarchy. Arguments to a QC can be a wild card (`*`) that matches anything, arguments (variables) of the operator that contains the QC, or objects or classes (the latter match any object of that class). Negated predicates are allowed.

For example, the QCs of the operator `m1-get-boom1`, shown in Figure 2, are `(%property max-sea-state boom.1 *)` and `(sea-state sea-sector.1 latest.1 *)`. The first QC refers to the `max-sea-state` property of the variable `boom.1` (the most severe level of ocean conditions under which the boom is effective). The user has indicated that the `max-sea-state` can take on any value with the asterisk (`*`) as an argument, so the Operator Learner will have to use its learning methods to identify the correct value(s). The second QC indicates that the `sea-state` predicate (ocean conditions) for `sea-sector.1` (the location of the operation) at time `latest.1` is relevant. Again, `*` indicates that the value has not been constrained, so the system must learn the correct value.

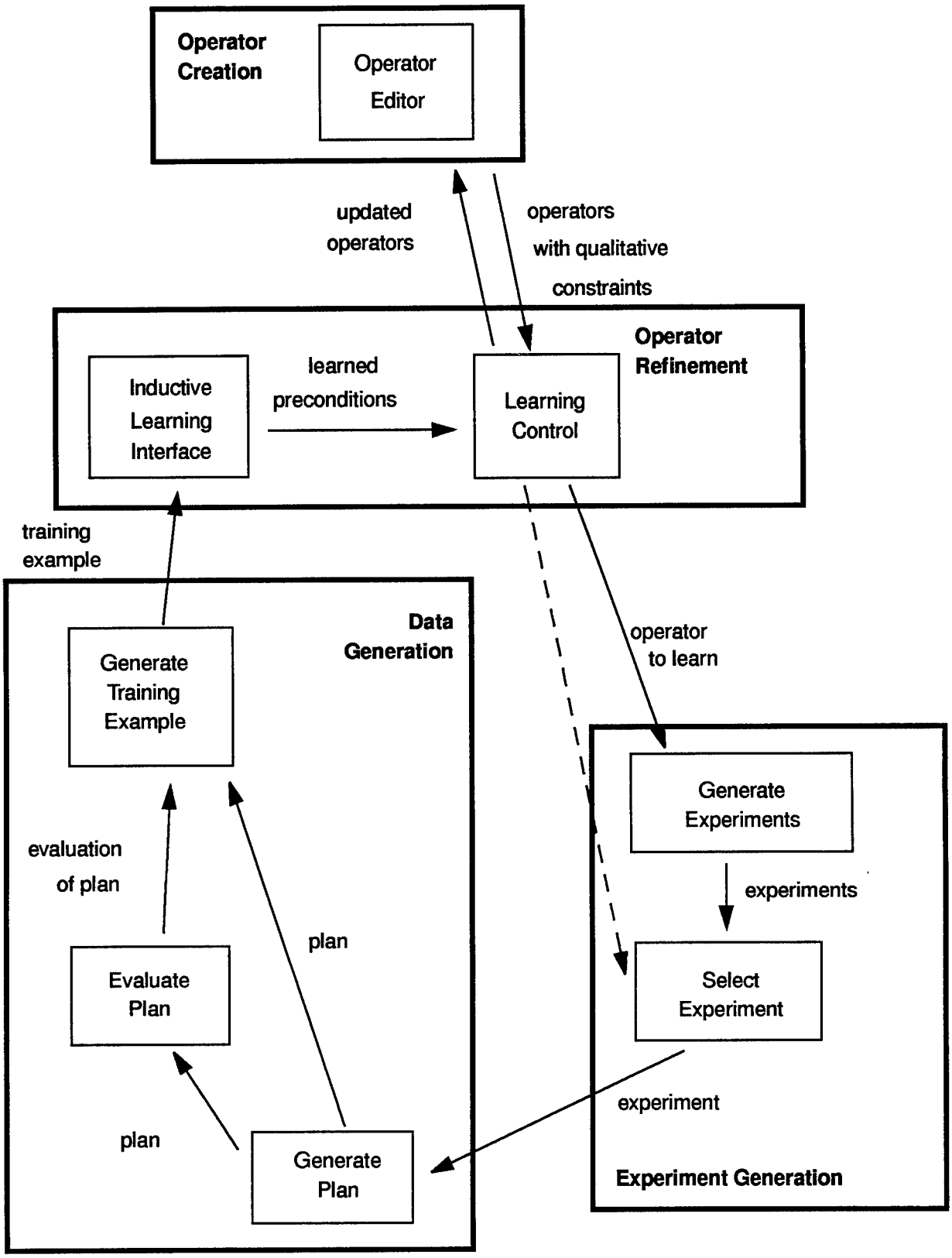

Figure 4. Operator Learner Architecture

3.2.2 Operator Refinement

The Operator Refinement process provides the overall control for the Operator Learner. The Learning Control subprocess decides which operator to learn (currently, this decision is made when the user explicitly invokes the learning system) and calls the Experiment Generation process. The Inductive Learning subprocess provides a generic interface to an external inductive learning module. We are currently using PAGODA (Subsection 3.2.5) as this module. However, this interface has been developed in a general, transparent way that permits the use of other inductive learning software.

After inductive learning takes place, the learning system must decide whether to continue generating experiments, or whether to stop learning. Learning should stop when one or more acceptable preconditions based on the QCs have been learned, or when none could be found. At present, the user enters this decision manually, after the system asks the user whether to continue. If more learning is required, another experiment is selected and the loop continues. If no more learning is to be done, the learned preconditions are passed back to the Learning Control process. The preconditions are then presented to the user, who indicates whether or not to add them to the operator and adjusts the operator's QCs. Again, this process is currently manual; an open problem is to develop methods for automating the process. Finally, the updated operators are stored in the domain knowledge base.

3.2.3 Experiment Generation

Using the QCs in the operator(s) to be learned, the Generate Experiments subprocess generates a list of experiments. Each experiment represents a set of constraints to be applied during the planning process, and consists of a problem to solve, operators to select, variable bindings to apply during operator expansion, additional world predicates to establish, and new objects and properties to define. Arguments to the QC that are filled with variables are either predefined (for nonlocal variables that are bound before this operator is even applied) or are to be selected by the QC (for local variables whose values need to be set in accordance with the constraints specified in the precondition). The two types of variables require different constraints on the planning process to establish their values.

The other arguments (`*` or class names) are values that provide constraints for determining whether or not to select this operator, and for instantiating the other variables within the operator. These values, however, cannot be set directly; they are dependent on the variable choices (e.g., in `ml-get-boom1`, the `max-sea-state value *` depends on the choice of `boom.1`). For example, in the `ml-get-boom1` operator in Figure 2, the first QC indicates that the resource `boom.1` should be selected, in part, on the basis of its `max-sea-state` value. That is, the variable `boom.1` should be bound to a boom with an *appropriate* value for `max-sea-state`, where the meaning of *appropriate* is yet to be defined. Moreover, selecting a boom with an inappropriate value for `max-sea-state` may cause the operator to fail. The second QC indicates that the `sea-state` at the place and time of the operation will be relevant for determining whether or not to select this operator. If the `sea-state` does not fall within an appropriate range (again, where the precise meaning of *appropriate* must be learned), the operation may fail.

The generated experiments should therefore vary the values of the wild cards (corresponding in this example to the `max-sea-state` value of `boom.1` and the `sea-state` in `sea-sector.1` at time `latest.1`), generate plans and plan evaluations to collect data about the success of the operation for different values of these wild cards, and construct hypotheses about the values required for the operator to succeed.

To select the possible values for each wild card, the system matches each QC against the initial world state. From the set of all possible matches (instantiations of the QC), the system then selects a representative subset by associating a value with each QC. This subset consists of a list of bindings for the wild cards in the QC. (Wild cards include the * argument and class names, but not variable names.) One instantiated QC is selected for each value that was seen.

For example, in `ml-get-boom1`, for the sea-state QC, the matches might be

```
(sea-state sf-bay 1 1)
(sea-state richards-bay 1 2)
(sea-state drakes-bay 1 1).
```

In this case, the values are 1, 2, and 1, respectively, and the first and second matches would be selected. For each of these matches, one or more constraints are identified that will result in this instantiation of the QC in a generated plan. Local variables (those whose value can be selected at the time the operator is applied) have variable binding constraints. For nonlocal variables, whose value has been determined in a previously applied operator, the process is more complicated, and requires that the initial world state be modified in such a way that the variable will be bound as desired. For example, `sea-sector.1` in `ml-get-boom1` is bound at a previous planning level to a location that is defined as a sensitive area in the planning scenario. We do not have a general solution for computing the required modifications, so the user or system designer must tell the system the correct predicate(s) to establish for each QC.

The cross-product of experiments for each QC is formed, resulting in a list of combined experiments. This is currently done explicitly, but in principle could be done dynamically as individual experiments are selected for each QC. The Select Experiment subprocess then selects an experiment from the list, by asking the user to chose one, or by selecting the first experiment on the list. An open issue that remains to be resolved is to develop methods for selecting an experiment automatically.

3.2.4 Data Generation

The Data Generation process uses the selected experiment to create the appropriate input data (training examples) for the Inductive Learning subprocess. This process incorporates three components, Generate Plan, Evaluate Plan, and Generate Training Example.

3.2.4.1 Generate Plan

A plan is generated interactively or automatically, using the constraints represented by the selected experiment. The constraints that can be specified in an experiment are as follows.

- **Variable Binding**: bind a given variable to a given value in the specified operator.
- **Operator Selection**: always select a specified operator when it is applicable.
- **Predicate Assertion**: assert one or more predicates in the initial world state.
- **Object Creation**: define a new object with given properties before planning.

The implementation of these constraints uses SIPE-2's built-in hooks for variable binding, operator selection, and predicate/object definition; therefore, we wrote relatively little code to implement this module.

If other plans have already been generated using similar experiments, a limited capability exists to reuse those plans. Specifically, the system can define new objects and rebind variable values in an existing plan. Therefore, if a new experiment differs from a previous one only in the variable-binding constraints, the system can reuse the plan, rather than constructing a new one from scratch. Additional development would also allow the system to use SIPE-2's replanning capabilities to apply different operator-selection and predicate-assertion constraints to previously generated plans.

3.2.4.2 Evaluate Plan

The plan is sent to the evaluation module, which returns an evaluation that indicates the success or failure of the overall plan and of individual actions or action sequences in the plan. The definition of "success" depends on the domain and on the purpose of the operator being learned (e.g., whether a unit arrived on time, or oil was successfully cleaned up).

Plan evaluation is a difficult problem in general. When plan evaluation methods are used in conjunction with other tools such as the Operator Learner, the inherent difficulty of plan evaluation is exacerbated by the need for the evaluator and learner to share a semantic representation of the plan. In the rest of this section, we discuss the problems that arose in this area during this project.

One factor that made it difficult to assess the success of an operator in the oil-spill domain is that there is not a direct mapping between the purpose of a given operator and the quantities returned by the evaluation model. There are three primary reasons for this lack of direct mapping: first, multiple operators are often applied to achieve a single shared goal, so credit assignment is a problem. For example, in the `ml-get-boom1` operator, the purpose is to get a certain quantity of boom to the location of the operation; this can be done by combining several operators that each bring a smaller quantity of boom to the final destination. Therefore, the actions in a given operator may succeed, although it fails to achieve its overall purpose.

Second, the purpose in the SIPE-2 sense may not match the conceptual purpose of the operator. In the `ml-get-boom1` operator, the purpose slot lists the boom level as the operator's purpose, but a planning expert would say that the actual purpose of this operator is to contain a certain quantity of oil. Although the operator clearly would fail if the boom did not arrive in the given location by the required time, the operator would also fail if the boom did not work properly in the prevailing ocean conditions; this conclusion is not explicitly stated in the purpose slot.

Finally, even if "containing the oil in the sector" were explicitly stated in the operator's purpose, that goal would not be explicitly represented in the evaluation model. Rather, the output of the evaluation model specifies the quantity of oil in each sector at each point in time, as well as the quantities that have washed up on shore, evaporated, or been transported out of the sector (e.g., via skimmer). "Containing the oil" thus must be translated into a quantitative measure of those values, which raises many questions: How much oil must be contained (and by what time), for the operator to be considered successful? When oil is removed by skimmer and barge, so that no oil is left in the sector and no oil has escaped, should the operator be considered to have succeeded? If the oil sinks before it is contained, so that it cannot be cleaned up although the ocean surface is clear, has the operator succeeded?

In the military transportation planning domain, there appears to be a more obvious mapping between operator purpose (in deployment planning, the purpose is usually to ensure that a given unit arrives at a location by a specified time) and simulator results for at least a subset of deployment operators (although this is not always true). In general, we believe that better ontologies and reasoning methods for mapping between plans (and planning knowledge) and evaluation criteria are needed, for these applications to interact effectively.

3.2.4.3 Generate Training Example

The Operator Learner next uses the qualitative constraints to generate a training example for the operator being learned, using the new plan and its evaluation. The training example consists of the relevant world state (i.e., any instantiations that match the qualitative constraints) at the point in the plan where the operator was applied, and a positive or negative label, depending on whether the operator succeeded or failed.

The world state at a specified node is generated by executing the plan up to that node. (The effects of this execution are undone after the training example is generated.) The qualitative constraints in the operator being learned determine which predicates and properties are extracted from the world state.

The training example is sent to the inductive learning system. This procedure currently uses PAGODA, but is implemented as a "black box" module that could be replaced by a different induction method, or by multiple competing methods.

3.2.5 Learning from an Expert User

In addition to learning from simulators and automated evaluation tools, the Operator Learner can learn from an expert's planning choices. Whenever the planner chooses an operator during planning, a set of training examples are generated that describe the planning context (i.e., the state of the world at the node where the operator was applied). A positive example is generated for the chosen operator, indicating that the operator is applicable in that context. Negative examples are generated for each operator that is not selected, indicating that the operator does not apply (or is less applicable) in the current context. For example, if the planner frequently picks a certain operator for actions in the Middle East, and a different operator for actions in the Pacific region, a geographic precondition would be learned for each operator.

In the process of learning from the user, the feedback consists of training examples generated by the users from the choices they make during planning. The user may guide the learning process by deliberately setting up planning problems on which to train the system, or may simply train the system "on the job" by allowing it to observe the expert's planning behavior during the actual planning process. The latter method is a particularly useful way to apply machine learning techniques during actual planning, since knowledge is continuously acquired throughout the life of the system. The incremental nature of PAGODA is an advantage for this type of learning, because the system improves its behavior incrementally as each new training example arrives.

Each training example consists of a description of the world state at a particular point in the training example, and an operator that the user did or did not choose; these operators correspond to positive and negative training examples, respectively. These examples are used to induce candidate preconditions for the operators. The same inductive learning techniques that are used for the feedback from the simulator are applied to learn these preconditions.

These examples are more likely to be "noisy" or incorrect than the examples received from the simulator, for the following reasons: (1) users sometimes make arbitrary or incorrect choices; (2) false negatives occur, since more than one operator may be applicable and the operators that are not selected are classified as negative examples; (3) the user may understand certain aspects of the situation that are not reflected in SIPE-2's world state; (4) users sometimes exhibit superstitious behavior (always preferring a particular operator even when another would be more appropriate); and (5) planning may later fail, indicating that the operator choice was incorrect. The last of these conditions can sometimes be detected by identifying cases in which the system backtracks and an alternative operator is applied during backtracking. To remedy the other incorrect examples, probabilistic learning methods are used, and learned knowledge is always confirmed with an expert user before being added to the system.

In essence, this process consists of acquiring "hidden" knowledge: that is, knowledge that the user has but that may not be explicitly represented in the system. In this case, nothing in the learning system's representation of the current world state enables it to distinguish between two situations in which the user makes different decisions. The system could infer that there should be an additional predicate to represent this distinction, and would then add the predicate to its representation for future use. This process would enable us not only to refine the operators, but to improve the representation of the domain.

3.3 PAGODA

PAGODA is a model for an intelligent autonomous agent that learns and plans in complex, nondeterministic domains [desJardins 1992]. The guiding principles behind PAGODA include probabilistic representation of knowledge, Bayesian evaluation techniques, and limited rationality as a normative behavioral goal.

We are using only the probabilistic inductive learning component of PAGODA for this project. The inductive hypotheses are represented as sets of conditional probabilities that specify the distribution of a predicted feature's value, given a set of input features. A probabilistic inference mechanism allows PAGODA to make predictions about the value of the output feature in a given world state by combining the relevant probabilities.

Theories are generated by means of a heuristic search process, guided by the training examples. The theories are evaluated by means of a Bayesian technique that provides a tradeoff between the accuracy and the simplicity of learned theories. The prior probability of a theory is a measure of its simplicity—shorter theories are more probable.

Since SIPE-2 cannot represent the probabilistic theories learned by PAGODA as preconditions, we use thresholding to create deterministic preconditions. Information is lost in this process: in general, the deterministic preconditions are overly strict (i.e., they sometimes rule out an operator in a case where it is, in fact, applicable). Each rule that PAGODA learns states that in situation S, an action or operator A succeeds with probability P. The learning system analyzes the theories (rule sets) to identify situations S such that if S is true, A succeeds with probability greater than some threshold $P_{success}$; if S is false, A fails with probability greater than another threshold $P_{failure}$. These situations are the discrimination conditions for A, and are added to the system as preconditions after they are confirmed by the user.

To give a very simple example of how inductive learning works, suppose that in `ml-get-boom1`, the result of evaluating the plan is such that regardless of the specific sea-sector and time, whenever the value of the `sea-state` predicate is 3 or less, the operator succeeds, and whenever it is 4 or more, the operator fails. The learning system would form the hypothesis

(sea-state sea-sector.1 latest.1 [1-3]) => success

(sea-state sea-sector.1 latest.1 [4-5]) => failure.

If there is noise or randomness (e.g., in some cases the operator fails, even though sea-state is 3 or less, or sometimes succeeds when sea-state is 4 or 5), the probabilistic hypothesis evaluation model built into PAGODA determines the most probable hypothesis. The hypothesis evaluation and inductive learning mechanisms of PAGODA are detailed elsewhere [ibid.].

3.4 PREDICATE EDITOR

The Predicate Editor is a revised and generalized version of what was originally called the Information Window in SOCAP. Previously, this display was tailored to a particular scenario, showing specific world predicates in each of six fixed panes. The generalized Information Window allows the user to tailor a presentation for different domains and for different views of a given domain, by specifying how many panes appear, and which predicates are displayed in each pane.

3.5 GKB EDITOR

Under a separate contract, SRI has developed the GKB Editor for editing class and object hierarchies [Karp, Myers, and Gruber 1995]. The GKB Editor provides a graphical interface for browsing and modifying classes and instances and their properties. The underlying representation is the Generic Frame Protocol (GFP), so object hierarchies created with the GKB Editor can be shared with any system that understands (or provides an interface to) the GFP. KATY provides an interface to this editor for creating and modifying the SIPE-2 sort hierarchy.

4 SIMULATORS

Our original proposal was to develop and apply knowledge acquisition tools in SOCAP's military transportation application domain. Due to representational inadequacies of the available transportation simulators, and the integration difficulties their use presented, we decided to use an oil-spill domain instead. In the following subsections, we discuss the simulators and evaluation tools that were available for the oil-spill and military deployment domains, and explain why we selected this oil-spill domain.

4.1 OIL-SPILL DOMAIN

The Operator Learner demonstrations used an oil-spill planning domain, as noted above. The Spill Response Configuration System (SRCS) plans responses to coastal oil spills and identifies equipment shortfalls. The SRCS incorporates a spreadsheet-based evaluation model that is used for the feedback required by the Operator Learner.

The oil-spill domain is a good analogue to the military transportation planning domain, so that our development work for the oil-spill-based demonstration can be applied directly to the military domain. Both domains are crisis response planning situations, where actions to respond to an emergency must be identified, along with the resources needed to perform the actions. In both cases, methods for moving the requisite equipment to the crisis site must also be identified.

The success of an action in a plan in the oil-spill domain translates to the percentage of oil cleaned up in the specified sector. In the demonstration scenario, positive instances for the learning system are those actions resulting in greater than $P\%$ of the oil being cleaned up (where P is a fixed value determined by a domain expert). A future research direction would be to explore methods of learning the degree of success of an action.

The success of an action in the oil-spill domain has a direct analogue in the military transportation planning domain, where the degree of success is determined by how much time a unit is delayed (i.e., whether and by how much time a unit arrives late) and how many resources (transportation assets, fuel, or personnel) are used.

We identified three alternative demonstration scenarios, in addition to the sea-state example described in Subsection 3.2. The first of these scenarios was used for the final project demonstration, which showed the system learning high-level strategies for oil-spill cleanup by observing the user's choices. The three scenarios are outlined below.

1. In the top-level operators, learn the conditions under which each type of operation is effective. This learning activity might involve identifying the priority of each targeted sea sector, based on the expected degree of effectiveness of the plan; or it might involve simply learning preconditions to identify the best single sea sector. The factors relevant to this prediction include
   - The proximity of cleanup site to spill
   - The depth of water (shallower water makes cleanup more effective)
   - The location of protected areas
   - The equipment available
   - Weather conditions.
2. At the middle level of planning, learn which type of skimmer works best under which conditions. The relevant factors include
   - The sea state (a number summarizing the severity of weather, which is in turn determined by tides, waves, and currents)
   - The oil thickness (available from the trajectory model)
   - The encounter rate (determined by sweep area, speed, and skim rate, which are properties of the boat and boom used)
   - The recovery rate (i.e., the percentage of the oil encountered that is successfully skimmed)
   - The efficiency
   - The pump rate
   - The storage capacity (on board, as well as in bladders or tugs available for offloading)
   - The personnel available
   - The time of day.

3. At the lowest level of planning, learn to predict the length of boom required for a given situation. Relevant factors include
   - The weather and ocean conditions
   - The angle of boom with respect to current (a shallow angle is more effective in deflecting, but requires more boom)
   - The ocean current
   - The boom height
   - The purpose of using boom (excluding oil from a protected area, versus deflecting it to a shore or water area where it will be cleaned up)
   - The type of boom
   - The depth of water
   - The leakage rate (determined by the above factors)
   - The number of booms (e.g., doubling booms reduces leakage but requires twice as many boom feet).

These scenarios show the wide range of applicability of the learning-based methods at all planning levels. Inductive learning methods enable the system to identify the factors relevant to the success of each planning operator, and to aggregate these factors to an appropriate level for each step in the decision-making process.

4.2 MILITARY DEPLOYMENT DOMAIN

In the deployment planning scenario we originally proposed, severe weather conditions would cause certain types of ports (e.g., those with unsheltered harbors) to become unavailable, and certain types of operations (e.g., transport, loading, and offloading) to take longer than usual. In addition, decisions about the allocation of resources such as ports, transportation assets, and personnel would be made by matching the capabilities of the resources to the requirements imposed by the plan. We decided to use the oil-spill domain instead of the transportation planning domain or another military planning domain (e.g., joint operations planning or air campaign planning) because of the availability and suitability of a plan evaluator for the oil-spill domain. In particular, the available simulators were unable to represent the factors that would impact the planning process, as listed above, and/or the cost of integration was too high. In this subsection, we briefly describe the simulators that were available for military planning domains.

4.2.1 PFE

The Prototype Feasibility Estimator (PFE) is a transportation simulator developed by Bolt Beranek and Newman Inc. (BBN). It uses a very simple model of port capability, and does not model weather or other aspects of the situation. To be usable in this project, PFE would have to be provided with a time-phased description of port availability. In addition, the computation of movement times would have to be modified to depend on a wider range of environment features, such as the current weather.

These revisions could be incorporated by using the time-phased port availability information to compute PFE's input (i.e., the set of available ports), and by reimplementing PFE's time-computation component, which would require substantial development effort. In addition, using PFE would require the use of FMERG* to expand the major force-level plans generated by SOCAP to the correct level for running the simulator. FMERG has not been maintained since the

Integrated Feasibility Demonstration 2 (IFD-2) demonstration; although we had installed it and spent some time working on its integration into SOCAP, we realized that it would be unrealistic to rely on FMERG for this project.

4.2.2 TransSim

We acquired the TransSim transportation scheduler/simulator from the University of Massachusetts (UM), and had a series of discussions with members of the TransSim development team about using their software. TransSim has some features that might make it a better choice than PFE for a military demonstration scenario. TransSim takes into account ship speeds, ship, port, and berth availability, and weather conditions; all of these are represented explicitly and are easy to vary programmatically. It also incorporates UM's CLIP/CLASP data collection package. On the other hand, TransSim expects Time-Phased Force Deployment Data (TPFDD) as input. FMERG would therefore be required for translation from the major force level of SOCAP's plans to the TPFDD level. As previously explained, the use of FMERG for the project was not feasible.

4.2.3 TACWAR

We examined the extensive documentation on the TACWAR wargaming system. TACWAR is used for wargaming at the U.S. Central Command and at other locations; it was developed and maintained by the Institute for Defense Analysis (IDA) in Washington, D.C. TACWAR was promising because it is the simulation system that most closely matches the type of scenario (joint military operations) that was encoded by the operators developed for SOCAP as part of IFD–2.

We concluded that TACWAR would be suitable for use as a simulator to execute high-level plans created by SOCAP. SOCAP could be used to determine when a unit arrives and where it is located, as well as its mission and posture; given this information, TACWAR would be run in batch mode to simulate the battle. The granularity of the representations of units, geography, and events used by TACWAR appears similar to that used by SOCAP.

However, the integration effort required to use TACWAR would have been large. It would have required obtaining a copy of TACWAR (which was previously part of the Common Prototyping Environment [CPE], but was no longer supported at the time we were evaluating simulators for this project); obtaining a suitable, unclassified TACWAR scenario that would obviate the creation of the voluminous input files required by TACWAR; implementing the scenario in SOCAP; extracting the plan from SOCAP in a form suitable for TACWAR; adding new operators to correspond to TACWAR's missions and postures; establishing scenario-controlling parameters that are independent of the plan generated by SOCAP; and extracting the results of the simulation for use by SOCAP. We concluded that the TACWAR integration effort was beyond the scope of this contract.

4.2.4 CTEM

We discussed with ISX Corporation the possible use of CTEM in this project. CTEM is used in the ACPT for air campaign plan evaluation. We determined that CTEM would be too difficult to acquire (because of security classification problems) and would not provide feedback at an appropriate level of detail. Also, at the time we were prepared to use it for this project, the knowledge acquisition process required for SIPE-2 to work in the ACP domain had not yet been completed.

*FMERG: Force Module Enhancer and Requirements Generator.

5 EVALUATION METRICS

We were unable to perform a formal evaluation of KATY under this contract, due to time and funding limitations. Our experience has shown the knowledge editing tools to be extremely useful, substantially reducing development time and the likelihood of errors. The Operator Learner performed well in the simple demonstration scenarios we developed, and we expect that the learning techniques used in the Operator Editor will scale up well.

We identified a number of operationally relevant evaluation metrics, both quantitative and qualitative, for future evaluations of KATY. These metrics include

- The quality of plans, measured in terms of probability of success (e.g., in a simulator), or subjectively by an expert user
- The time required for a user to create new planning operators
- The quantity of data needed for acquiring knowledge (e.g., the number of training examples required by the inductive learning system)
- The computational time and memory required to run simulations and the learning system.

In addition to these system evaluation metrics, our approach, acquiring planning knowledge from on-line simulators and evaluation tools, raises the issues of plan evaluation and measuring plan quality. These metrics are required by any system whose function is to improve planning performance, since is impossible to improve performance without some measurement of that performance.

Simulators are one obvious type of plan evaluation tool for the domains in which they exist and are considered to be reliable. Many military domains (including transportation planning) use simulators to verify plans generated by humans, so it seems reasonable that a learning system should consider these simulators to be reliable sources of knowledge.

In the oil-spill domain, oil-trajectory models and utility analysis of oil-spill damage are widely regarded by the community of domain experts as an effective tools for evaluating response contingency plans. The demonstration we have developed in this domain uses an oil-trajectory model and a spreadsheet-based utility analysis of the generated plan. These tools enable the system to gather feedback about the success of the operators being learned, and enable the user to assess the quality of the plans that are generated by the system.

6 RELATED WORK

The problems we have addressed in the work described here, and the methods we have used to solve the problems, are similar to problems and methods described in recent research on experiment generation, knowledge acquisition, and learning apprentices. In addition, several researchers are studying ways to improve the performance of planning systems via machine learning.

Gil [1992] describes research on experiment generation for knowledge acquisition in planners. The general approach she uses is to identify missing preconditions by observing when actions fail, and then to generate experiments to determine the correct precondition. Some of the methods described by Gil are applicable to the problem of experiment generation in KATY, but many problems remain to be solved (see Section 8).

Much of the research in the knowledge acquisition community has focused on structuring the global knowledge acquisition process. EXPECT is a knowledge acquisition architecture that dynamically forms expectations about the knowledge that a problem-solving system needs to acquire, and then uses these expectations to interactively guide the user through the knowledge acquisition process [Gil and Swartout 1994]. Davis [1993] describes the use of metalevel knowledge in TEIRESIAS, an expert system for stock market investment advising, to guide identify new rules to be added to the expert system. The metalevel knowledge allows the system to "know what it knows," and therefore to identify and repair bugs in its knowledge base (missing or incorrect knowledge). Eshelman et al. [1993] describe MOLE, a knowledge acquisition system for heuristic problem solving. MOLE generates an initial knowledge base interactively, and then detects and corrects problems by identifying "differentiating knowledge" that distinguishes among alternative hypotheses. Ginsberg, Weiss, and Politakis [1993] have developed SEEK, which performs knowledge base refinement by using a case base to generate plausible suggestions for rule refinement. These methods, which view the knowledge base as a whole, complement the Operator Learner's approach of focusing on refining individual operators.

Learning apprentices are a recent development in knowledge acquisition tools. Mitchell, Mahadevan, and Steinberg [1993] characterize a learning apprentice as an "interactive, knowledge-based consultant" that observes and analyzes the problem-solving behavior of users. One advantage of a learning apprentice is that it is running continuously as the system is used by a wide range of users; thus, the evolving knowledge base reflects a broad range of expertise. These researchers developed the LEAP apprentice, which uses explanation-based learning (EBL) techniques to explain and generalize cases (traces of the user's problem-solving behavior) in the domain of digital circuits. DISCIPLE [Kodratoff and Tecuci 1993] also uses EBL, as well as similarity-based learning, to acquire problem-solving knowledge in the domain of design for the manufacturing of loudspeakers. The Operator Learner is similar to a learning apprentice in its mode of learning from the user, but uses inductive methods rather than EBL, allowing the system to acquire a broader range of new knowledge without the need for domain theories. Since we also learn from external simulators, there is less burden on the user to provide a complete set of training examples from which the system learns.

Wang and Veloso [1994] have developed a system that inductively learns planning control knowledge. Their system makes some simplifying assumptions (e.g., that there is no randomness, and that the system has a complete domain representation) that limit the applicability of their approach to complex, real-world domains. KATY permits randomness, and allows the user to guide the learning process using QCs, which we believe to be critical for large-scale domains.

Calistri-Yeh and Segre [1994] describe an ARPI-sponsored adaptive learning and planning system (ALPS). The primary mechanism for learning within their system is the use of a set of speedup learning techniques to improve planning performance. Their Probabilistic Theory Revision mechanism refines incorrect or incomplete domain theories, and can be viewed as a type of learning or adaptation. Veloso and Borrajo [1994] use a combination of bounded explanation

and inductive generalization to learn control rules for planning systems. Learning control rules is a slightly different problem than that addressed by KATY—the former focuses on improving the efficiency of the planning process, whereas the latter is concerned with its correctness. These two methods could be combined in order to improve performance along both dimensions simultaneously.

7 PUBLICATIONS

The following publications were written during the contract. These publications are also included in the list of references in Section 9.

- desJardins, M. 1994a. "Evaluation of Learning Biases using Probabilistic Domain Knowledge," in *Computational Learning Theory and Natural Learning Systems*, Vol. 2, eds. S.J. Hanson et al., the MIT Press, Cambridge, Massachusetts.
- desJardins, M. 1994b. "Knowledge Acquisition Tools for a Military Planning System," presented at the 1994 IEEE Conference on Tools with AI, New Orleans, Louisiana (November); in *Proc. 1994 IEEE Conference on Tools with AI*, Morgan Kaufmann Publishers Inc., San Francisco, California.
- desJardins, M. 1994c. "Knowledge Development Methods for Planning Systems," presented at the AAAI Fall Symposium on Planning and Learning, New Orleans, Louisiana (November); in *Working Notes of the AAAI Fall Symposium on Planning and Learning*, AAAI Press, Menlo Park, California.
- desJardins, M. 1994d. "The Use of Relevance to Evaluate Learning Biases," presented at the AAAI Fall Symposium on Relevance, New Orleans, Louisiana (November); in *Working Notes of the AAAI Fall Symposium on Relevance*, AAAI Press, Menlo Park, California.
- Gordon, D.F., and M. desJardins. 1995. "Evaluation and Selection of Biases in Machine Learning," *Machine Learning* 20(1/2), pp. 5–22 (July/August).
- desJardins, M. 1995. "Goal-Directed Learning: A Decision-Theoretic Model for Deciding What to Learn Next," in *Goal-Driven Learning*, eds. A. Ram and D.B. Leake, the MIT Press pp. 241–250, Cambridge, Massachusetts.
- desJardins, M. 1996. "Knowledge Acquisition Tools for Planning Systems," in *Advanced Planning Technology: Technological Achievements of the ARPA/Rome Laboratory Planning Initiative*, ed. A. Tate, AAAI Press, Menlo Park, California.

8 CONCLUSIONS AND FUTURE WORK

Automated and semiautomated tools for knowledge acquisition will become increasingly essential as large-scale planning systems are developed and deployed. Our research in this area has led to initial prototypes of two types of tools: interactive graphical editors for developing planning knowledge, and an inductive learning system that uses simulator feedback and the user's planning choices to refine and verify partial operators.

On the basis of ongoing usage and evaluation of these prototypes, we believe that both the editor and the learning system form essential parts of a knowledge developer's toolkit for constructing large-scale planning applications. The key advantages these tools provide, which will enable the development, deployment, and ongoing maintenance of realistic planning applications, are

- Template-based editing methods for constructing individual planning operators, reducing the likelihood of errors in the development process, and reducing the tedium of operator construction
- A framework for acquiring knowledge from multiple sources (plan evaluation modules, simulators, and expert planner behavior), guided by initial approximations provided by a knowledge developer.

These tools focus on the development and refinement of individual planning operators. We also recognize the need for additional tools in KATY, particularly those that guide and manage the development of the knowledge base as a whole. Some such tools are being developed by others in the planning research community; for example, researchers at the Jet Propulsion Laboratory have developed specialized techniques for assessing the consistency and completeness of a knowledge base [Chien 1996]; and ISI's EXPECT project provides a framework for managing a structured knowledge acquisition process [Gil and Swartout 1994]. These methods could be generalized and applied within the SIPE-2 framework.

Future Work. Many of the problems raised during this work have not been addressed in depth by the machine learning research community. Most of the current research focuses on algorithms and methods for inductive or explanation-based learning. While developing good inductive learning methods is important (and we list some specific research directions in that area below), this work also points to the need for supporting technologies that will enable the effective application of inductive learning methods. These supporting technologies include representing and reasoning about bias, experiment generation, knowing when to learn and when to stop learning, and evaluation methods for complex forms of learned knowledge.

The qualitative constraints used by the operator learner provide a way for the knowledge developer to feed partial knowledge into the system, without having to specify all of the details of an operator's preconditions. In machine learning terminology, the developer is imposing a *bias* on the learning system. Generalizing the representation and implementation of qualitative constraints would broaden the types of bias that could be introduced into this process.

The experiment generation capabilities that we developed for this project were tailored for the small learning problems we examined. In large-scale domains, methods will be needed to select an appropriate subset of experiments to guide the learning process. Very little work has been done in this area; the techniques developed by Gil [1992] provide some interesting ideas for directions, but are limited to fairly simple domains.

The development of stopping criteria that enable the system to know when to stop learning is directly related to the problem of experiment generation: to construct an efficient sequence of experiments, one must know when enough experimentation has been performed (or, equally important, when experimentation is not yielding a useful answer, and other approaches should be tried).

Evaluating the learned knowledge is a nontrivial problem in general, and particularly difficult in a planning domain. Inductive learning methods are usually evaluated against a test set of examples that are drawn from the same sample population as the training examples. In the case of planning knowledge, it may be difficult to generate a test set, and, more importantly, the real effectiveness of a planning operator can be determined only by using it in the planning process. Therefore, evaluating the learning process corresponds to evaluating generated plans, which is an open problem for most application domains.

Finally, the inductive learning methods that have been developed by the machine learning research community have generally focused on a batch learning situation (where all training examples are available at the onset of learning), predicting discrete classes using a well-defined set of input features (e.g., predicting a disease type from a set of symptoms). Our observation is that for planning problems, and for a learning context where data is expensive to collect and may arrive continuously over the lifetime of the system, different methods are needed. In particular, incremental learning methods that continuously revise hypotheses as new data arrives are necessary. These learning methods must be able to predict numerical values (e.g., the expected degree of success of an operation, time to complete an activity, or amount of a resource required), and must be able to reason at a meta-level about the representation they use (e.g., they should be able to recognize when the domain representation should be extended, as when no good theory can be learned by using the current representation).

We believe that the prototypes we have developed demonstrate the utility and necessity of providing tools for knowledge acquisition in the development of planning applications. The tools in their current form, particularly the operator editor, already provide useful functionality, and are being used to support knowledge development in an ongoing process. However, this work has also highlighted the need for additional research directions in machine learning and knowledge acquisition.

9 REFERENCES

Bienkowski, M.A. 1995. *Decision Support for Transportation Planning in Joint COA Development*, ITAD-2062-FR-95-176, SRI International, Menlo Park, California.

Calistri-Yeh, R.J., and A.M. Segre. 1994. "The Design of ALPS: An Adaptive Learning and Planning System," in *Proc. 1994 Workshop on the ARPA/RL Knowledge-Based Planning and Scheduling Initiative*, Morgan Kaufmann Publishers Inc., San Francisco, California.

Chien, S.A. 1996. "Static and Completion Analysis for Planning Knowledge Base Development and Verification," presented at the Third International Conference on Artificial Intelligence Planning Systems, Edinburgh, UK (May); in *Proc. Third International Conference on Artificial Intelligence Planning Systems*, Morgan Kaufmann Publishers Inc., San Francisco, California.

Davis, R. 1993. "Interactive Transfer of Expertise: Acquisition of New Inference Rules," in *Readings in Knowledge Acquisition and Learning*, pp. 221–239, Morgan Kaufmann Publishers Inc., San Francisco, California.

Desimone, R., D.E. Wilkins, M. Bienkowski, and M. desJardins. 1993. "SOCAP: Lessons Learned in Automating Military Operations Planning," in *Sixth International Conference on Industrial and Engineering Applications of AI and Expert Systems* (June).

Desimone, R.V., and J.M. Agosta. 1993. "Oil Spill Response Simulation: The Application of Artificial Intelligence Planning Techniques," in *Proc. Simulation MultiConference* (April).

desJardins, M. 1992. *PAGODA: A Model for Autonomous Learning in Probabilistic Domains*. Ph.D. thesis, University of California at Berkeley, Berkeley, California (available as UCB CS Dept. Technical Report 92/678).

desJardins, M. 1994a. "Evaluation of Learning Biases using Probabilistic Domain Knowledge," in *Computational Learning Theory and Natural Learning Systems*, Vol. 2, eds. S.J. Hanson et al., the MIT Press, Cambridge, Massachusetts.

desJardins, M. 1994b. "Knowledge Acquisition Tools for a Military Planning System," presented at the 1994 IEEE Conference on Tools with AI, New Orleans, Louisiana (November); in *Proc. 1994 IEEE Conference on Tools with AI*, Morgan Kaufmann Publishers Inc., San Francisco, California.

desJardins, M. 1994c. "Knowledge Development Methods for Planning Systems." presented at the AAAI Fall Symposium on Planning and Learning, New Orleans, Louisiana (November); in *Working Notes of the AAAI Fall Symposium on Planning and Learning*, AAAI Press, Menlo Park, California.

desJardins, M. 1994d. "The Use of Relevance to Evaluate Learning Biases." Presented at the AAAI Fall Symposium on Relevance, New Orleans, Louisiana (November); in *Working Notes of the AAAI Fall Symposium on Relevance*, AAAI Press, Menlo Park, California.

desJardins, M. 1995. "Goal-Directed Learning: A Decision-Theoretic Model for Deciding What to Learn Next," in *Goal-Driven Learning*, eds. A. Ram and D.B. Leake, the MIT Press pp. 241–250.

desJardins, M. 1996. "Knowledge Acquisition Tools for Planning Systems," in *Advanced Planning Technology: Technological Achievements of the ARPA/Rome Laboratory Planning Initiative*, ed. A. Tate, AAAI Press, Menlo Park, California.

Eshelman, L., D. Ehret, J. McDermott, and M. Tan. 1993. "MOLE: A Tenacious Knowledge-Acquisition Tool," *Readings in Knowledge Acquisition and Learning*, pp. 253–259, Morgan Kaufmann Publishers Inc., San Francisco, California.

Gil, Y. 1992. *Acquiring Domain Knowledge for Planning by Experimentation*, Ph.D. thesis, Carnegie Mellon University, Pittsburgh, Pennsylvania (August).

Gil, Y., and B. Swartout. 1994. "EXPECT: A Reflective Architecture for Knowledge Acquisition," in *Proc. 1994 Workshop on the ARPA/RL Knowledge-Based Planning and Scheduling Initiative*, Morgan Kaufmann Publishers Inc., San Francisco, California.

Ginsberg, A., S.M. Weiss, and P. Politakis. 1993. "Automatic Knowledge Base Refinement for Classification Systems," *Readings in Knowledge Acquisition and Learning*, Morgan Kaufmann Publishers Inc., San Francisco, California, pp. 387–401.

Gordon, D.F., and M. desJardins. 1995. "Evaluation and Selection of Biases in Machine Learning," *Machine Learning* 20(1/2), pp. 5–22 (July/August).

Karp, P.D., K. Myers, and T. Gruber. 1995. "The Generic Frame Protocol," in *Proc. IJCAI*, Montreal, Canada, Morgan Kaufmann Publishers Inc., San Francisco, California.

Kodratoff, Y., and G. Tecuci. 1993. "Techniques of Design and DISCIPLE Learning Apprentice," in *Readings in Knowledge Acquisition and Learning*, Morgan Kaufmann Publishers Inc., San Francisco, California, pp. 655–668.

Mitchell, T.M., S. Mahadevan, and L.I. Steinberg. 1993. "LEAP: A Learning Apprentice for VLSI Design," *Readings in Knowledge Acquisition and Learning*, Morgan Kaufmann Publishers Inc., San Francisco, California, pp. 645–654.

Veloso, M., and D. Borrajo. 1994. "Learning Strategy Knowledge Incrementally," in *Proc. 1994 IEEE Conference on Tools with Artificial Intelligence*, IEEE Computer Society Press, pp. 484–490.

Wang, X., and M. Veloso. 1994. "Learning Planning Knowledge by Observation and Practice," in *Proc. 1994 Workshop on the ARPA/RL Knowledge-Based Planning and Scheduling Initiative*, Morgan Kaufmann Publishers Inc., San Francisco, California.

Wilkins, D.E., K.L. Myers, L.P. Wesley, and J.D. Lowrance. 1994. *Planning in Dynamic and Uncertain Environments*, final report for Project 1520, SRI International, Menlo Park, California (January).

DISTRIBUTION LIST

addresses	number of copies
LOUIS J HOEBEL RL/C3CA 525 BROOKS RD ROME NY 13441-4505	5
SRI INTERNATIONAL 333 RAVENSWOOD AVE MENLO PARK CA 94025	5
ROME LABORATORY/SUL TECHNICAL LIBRARY 26 ELECTRONIC PKY ROME NY 13441-4514	1
ATTENTION: DTIC-OCC DEFENSE TECHNICAL INFO CENTER 8725 JOHN J. KINGMAN ROAD, STE 0944 FT. BELVOIR, VA 22060-6218	2
ADVANCED RESEARCH PROJECTS AGENCY 3701 NORTH FAIRFAX DRIVE ARLINGTON VA 22203-1714	1
RELIABILITY ANALYSIS CENTER 201 MILL ST. ROME NY 13440-8200	1
ROME LABORATORY/C3AB 525 BROOKS RD ROME NY 13441-4505	1
ATTN: RAYMOND TADROS GIDEP P.O. BOX 8000 CORONA CA 91718-8000	1

AFIT ACADEMIC LIBRARY/LDEE 1
2950 P STREET
AREA B, BLDG 642
WRIGHT-PATTERSON AFB OH 45433-7765

OL AL HSC/HRG, BLDG. 190 1
2698 G STREET
WRIGHT-PATTERSON AFB OH 45433-7604

US ARMY STRATEGIC DEFENSE COMMAND 1
CSSD-IM-PA
P.O. BOX 1500
HUNTSVILLE AL 35807-3801

NAVAL AIR WARFARE CENTER 1
6000 E. 21ST STREET
INDIANAPOLIS IN 46219-2189

COMMANDER, TECHNICAL LIBRARY 1
474700D/C0223
NAVAIRWARCENWPNDIV
1 ADMINISTRATION CIRCLE
CHINA LAKE CA 93555-6001

SPACE & NAVAL WARFARE SYSTEMS 2
COMMAND (PMW 178-1)
2451 CRYSTAL DRIVE
ARLINGTON VA 22245-5200

COMMANDER, SPACE & NAVAL WARFARE 1
SYSTEMS COMMAND (CODE 32)
2451 CRYSTAL DRIVE
ARLINGTON VA 22245-5200

CDR, US ARMY MISSILE COMMAND 2
RSIC, BLDG. 4484
AMSMI-RD-CS-R, DOCS
REDSTONE ARSENAL AL 35898-5241

ADVISORY GROUP ON ELECTRON DEVICES 1
SUITE 500
1745 JEFFERSON DAVIS HIGHWAY
ARLINGTON VA 22202

REPORT COLLECTION, CIC-14 1
MS P364
LOS ALAMOS NATIONAL LABORATORY
LOS ALAMOS NM 87545

AEDC LIBRARY 1
TECHNICAL REPORTS FILE
100 KINDEL DRIVE, SUITE C211
ARNOLD AFB TN 37389-3211

COMMANDER 1
USAISC
ASHC-IMD-L, BLDG 61801
FT HUACHUCA AZ 85613-5000

US DEPT OF TRANSPORTATION LIBRARY 1
FB10A, M-457, RM 930
800 INDEPENDENCE AVE, SW
WASH DC 22591

AIR WEATHER SERVICE TECHNICAL 1
LIBRARY (FL 4414)
859 BUCHANAN STREET
SCOTT AFB IL 62225-5118

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

SOFTWARE ENGINEERING INSTITUTE 1
CARNEGIE MELLON UNIVERSITY
4500 FIFTH AVENUE
PITTSBURGH PA 15213

NSA/CSS 1
K1
FT MEADE MD 20755-6000

DCMAD/WICHITA/GKEP 1
SUITE B-34
401 N MARKET STREET
WICHITA KS 67202-2095

PHILLIPS LABORATORY 1
PL/TL (LIBRARY)
5 WRIGHT STREET
HANSCOM AFB MA 01731-3004

THE MITRE CORPORATION 1
ATTN: E. LADURE
D460
202 BURLINGTON RD
BEDFORD MA 01732

OUSD(P)/DTSA/DUTD 2
ATTN: PATRICK G. SULLIVAN, JR.
400 ARMY NAVY DRIVE
SUITE 300
ARLINGTON VA 22202

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

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

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

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

DR MARK BUPSTEIN 1
BBN SYSTEMS & TECHNOLOGIES
10 MOULTON STREET
CAMBRIDGE MA 02138

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

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

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

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

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

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

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

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

MR. GARY EDWARDS 1
ISX CORPORATION
2000 N 15TH ST, SUITE 1000
ARLINGTON, VA 22201

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

DR MICHAEL FEHLING
STANFORD UNIVERSITY
ENGINEERING ECO SYSTEMS
STANFORD CA 94305
1

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

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

DR JIM HENDLER
UNIV OF MARYLAND
DEPT OF COMPUTER SCIENCE
COLLEGE PARK MD 20742
1

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

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

DR RON LARSEN
NAVAL CMD, CONTROL & OCEAN SUR CTR
RESEARCH, DEVELOP, TEST & EVAL DIV
CODE 444
SAN DIEGO CA 92152-5000
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, SUITE 101
SAN DIEGO CA 92110
1

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

ALICE MULVEHILL 1
BBN
10 MOULTON STREET
CAMBRIDGE MA 02238

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

DR 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

DIRECTOR 1
DARPA/ITO
3701 N. FAIRFAX DR., 7TH FL
ARLINGTON VA 22209-1714

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

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

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

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

DR BILL SWARTOUT
USC/ISI
4676 ADMIRALTY WAY
MARINA DEL RAY CA 90292 1

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

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

DR JOHN P. SCHILL
ARPA/ISO
3701 N FAIRFAX DRIVE
ARLINGTON VA 22203-1714 1

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

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

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

DR MATTHEW L. GINSBERG
CIRL, 1269
UNIVERSITY OF OREGON
EUGENE OR 97403 5

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

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

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

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

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

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

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

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

DR CARLA GOMES 1
ROME LABORATORY/C3CA
525 BROOKS RD
ROME NY 13441-4505

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

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

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

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

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

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

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

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

OFFICE OF THE CHIEF OF NAVAL RSCH
ATTN: MR PAUL QUINN
CODE 311
800 N. QUINCY STREET
ARLINGTON VA 22217 1

DR GEORGE FERGUSON
UNIVERSITY OF ROCHESTER
COMPUTER STUDIES BLDG, RM 732
WILSON BLVD
ROCHESTER NY 14627 1

DR STEVE HANKS
DEPT OF COMPUTER SCIENCE & ENG'G
UNIVERSITY OF WASHINGTON
SEATTLE WA 98195 1

DR WILLIAM S. MARK
LOCKHEED PALO ALTO RSCH LAB
DEPT 9620, BLDG 254F
3251 HANOVER ST
PALO ALTO CA 94304-1187 1

DR ADNAN DARWICHE
INFORMATION & DECISION SCIENCES
ROCKWELL INT'L SCIENCE CENTER
1049 CAMINO DOS RIOS
THOUSAND OAKS CA 91360 1

DR JAMES CRAWFORD
CIRL, 1269
UNIVERSITY OF OREGON
EUGENE OR 97403 1

ROBERT J. KRUCHTEN
HQ AMC/SCA
203 W LOSEY ST, SUITE 1016
SCOTT AFB IL 62225-5223 1

*U.S. GOVERNMENT PRINTING OFFICE: 1997-509-127-61019

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 Material 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.