Volume 17 Number 4 December 1993 ISSN 0350-5596

Informatica

An International Journal of Computing and Informatics

Special Issue: Multistrategy Learning
Guest Editor: Gheorghe Tecuci

The Slovene Society Informatika, Ljubljana, Slovenia

Informatica **17** (1993) Number 4, pp. 325–420

1977

Informatica

An International Journal of Computing and Informatics

Subscription Information

Informatica (ISSN 0350-5596) is published four times a year in Spring, Summer, Autumn, and Winter (4 issues per year) by the Slovene Society Informatika, Vožarski pot 12, 61000 Ljubljana, Slovenia.

The subscription rate for 1993 (Volume 17) is

- DEM 50 (US$ 34) for institutions,
- DEM 25 (US$ 17) for individuals, and
- DEM 10 (US$ 4) for students

plus the mail charge DEM 10 (US$ 4).
Claims for missing issues will be honored free of charge within six months after the publication date of the issue.

&T$_E$X Technical Support: Borut Žnidar, DALCOM d.o.o.

Stegne 27, 61000 Ljubljana, Slovenia.

Lectorship: Fergus F. Smith, AMIDAS d.o.o., Cankarjevo nabrezje 11, Ljubljana, Slovenia.

Printed by Biro M, d.o.o., Zibertova 1, 61000 Ljubljana, Slovenia.

Orders for subscription may be placed by telephone or fax using any major credit card. Please call Mr. R. Murn, Department for Computer Science, Jožef Stefan Institute: Tel (+386) 61 1259 199, Fax (+386) 61 219 385, or use the bank account number 50100-620-133-27620-5159/4 (LB 50101-678-51841 for domestic subscribers only).

According to the opinion of the Ministry for Informing (number 23/216-92 of March 27, 1992), the scientific journal Informatica is a product of informative matter (point 13 of the tariff number 3), for which the tax of traffic amounts to 5%.

Informatica is published in cooperation with the following societies (and contact persons):
Robotics Society of Slovenia (Jadran Lenarčič)
Slovene Society for Pattern Recognition (Franjo Pernuš)
Slovenian Artificial Intelligence Society (Matjaž Gams)
Slovenian Society of Mathematicians, Physicists and Astronomers (Bojan Mohar)

Referees:
D. Abel, M. Bohanec, M. Bonač, M. Drobnič, B. Filipič, J. Fiirnkranz, G.K. Gupta, M.A. Jenkins, A. Karalič, M. Kljajič, M. Kovačič, J. Lenarčič, B. Likar, B. Litow, D. Mladenič, I. Mozetič, M. Prešern, M. Rihar, B. Robič, I. Savnik, I. Sega, T. Szuba, M. Tamiz, G. Trinidad, O. de Vel.

The issuing of the Informatica journal is financially supported by the Ministry for Science and Technology, Slovenska 50, 61000 Ljubljana, Slovenia.

PROFILE

The readers of *Informatica* will probably ascertain that the choice of profiles is quite diverse concerning the positions and careers of our editors. I must confess that my and the editors choice for certain profiles is intuitive and simultaneously spontaneous. We search for excellence in individuals. In this choice, the international (geographical), generational, and disciplinary-field balance may play a significant role.

Some of our editors are well-situated, but some are neither supplied with the basic research infrastructure nor living suitability. As researchers they persist in their profession irrespective of the obstacles and misunderstanding of their research communities. In Eastern Europe and also in the West, some of them are left to themselves, without public support. But, the only important issue is their scientific achievement which may already impact the present research and technology generations or will have an impact in the future. Regarding this point, the intuition of choosing individuals for this column of Informatica is very challenging.

Professor *Gheorghe Tecuci* belongs to the younger generation of researchers and his professional position is in harmonic accordance with his professional achievements. He is already a member of the national academy of sciences, cosmopolitan, and holds a leading position in the field of machine learning and knowledge acquisition. In this respect he might be a real pattern for the coming research generations—particularly in the fields which concern modern artificial intelligence, knowledge methodologies and technologies, and even more the postmodern science and technology of the informational.

It is a great pleasure to present the profile of professor Tecuci to the readers of Informatica together with his introductory article entitled "Multistrategy Approaches to Learning: Why and How" and a group of five articles being edited by him. In this manner, the readers of Informatica will obtain the best insight and comprehension of the field professor Tecuci is leading.

In this issue of Informatica, for the first time, the so-called thematic groups of articles are presented. In the next Volume 18 (1994), in Informatica, several other thematic groups will be published concerning modern fields of logic, computation, artificial intelligence and, hopefully, also the critics (and crisis) of computer science and informational treatment of knowledge. In this respect, I believe, the readers and contributors of Informatica will not be disappointed. Informatica will also proceed from its quarterly issuing of the journal to a bimonthly one. All this will require greater activity and help from our editors.

Gheorghe Tecuci

Gheorghe Tecuci, a full member of the Romanian Academy since September 1993, is Associate Professor of Computer Science at George Mason University in Fairfax, Virginia, U.S.A., and Director of the Romanian Academy Center for Machine Learning, Natural Language Processing and Conceptual Modeling.

He received the M.S. degree in Computer Science from the Polytechnic Institute of Bucharest in 1979, graduating first among all the Computer Science students at the Polytechnic Universities of Romania. He received two Ph.D. degrees in Computer Science, one from the University of Paris-South, in July 1988, and the other from the Polytechnic Institute of Bucharest, in December 1988.

Between 1979 and 1990 Tecuci was a researcher and project leader at the Research Institute for Informatics in Bucharest, Romania, where he developed many of the skills that are essential for a successful researcher. During the summers of 1986-1990 he worked at LRI, University of Paris-South, as a research director of a joint research program of the Romanian Academy and the French National Research Center, and in Fall 1990 he joined the faculty of the Computer Science Department of the George Mason University.

In March 1991, Tecuci was elected corresponding member of the Romanian Academy and became a full member in September 1993. Tecuci has published over 60 scientific papers. He is one the founders of the field of multistrategy learning which represents a new stage in the evolution of machine learning.

Tecuci has developed one of the first multistrategy learning systems, DISCIPLE, which synergistically integrates explanation-based learning, learning by analogy, empirical inductive learning, and learning by questioning the user. Many important concepts in machine learning and knowledge acquisition have originated from research on DISCIPLE [Ph.D. Thesis, University of Paris-South, 1988]. Besides multistrategy learning, one could also mention adaptive learning (which is illustrated by DISCIPLE's ability to adapt its behavior to the features of the learning task), the usefulness of plausible explanations and over-generalizations in learning and knowledge acquisition, and plausible version spaces as a symbolic representation of uncertainty. For his work on DISCIPLE, Tecuci received the prize "Traian Vuia" of the Romanian Academy, in 1987. A paper on DISCIPLE [International Journal of Expert Systems, 1 (1): 39-66, (1987)] has also been included in "Readings in Machine Learning and Knowledge Acquisition", a book from the well-known Morgan Kaufmann's series which collects the best papers from each field of artificial intelligence.

More recently, Tecuci has introduced the concept of a plausible justification tree as a generalization of the concept of explanation, and as a framework for the integration of different types of inference. He has also introduced the concept of inference-based generalization, showing that there is a special type of generalization associated with each type of inference. Based on these new concepts, Tecuci has developed a general framework for multistrategy learning [Machine Learning, 11 (2-3): 129-153, (1993)] which dynamically integrates the elementary building blocks of the single-strategy learning methods (i.e., deduction, analogy, abduction, generalization, specialization, abstraction, concretion, etc.). The learning method MTL-JT (Multistrategy Task-adaptive Learning by Justification Trees), developed in this framework, integrates deeply and dynamically explanation-based learning, determination-based analogy, empirical induction, constructive induction, and abduction, depending of the features of the learning task. It also behaves as any of these single-strategy learning methods whenever the applicability conditions of such a method are satisfied.

Tecuci has also made important contributions to the integration of machine learning and knowledge acquisition. He has developed NeoDISCIPLE which represents a general approach to the automation of knowledge acquisition. This approach is based on several powerful ideas like understanding-based knowledge extension, knowledge acquisition through plausible inference and multistrategy learning, consistency-driven concept elicitation, and plausible version spaces [IEEE Transactions on Systems, Man and Cybernetics, 22 (6): 1444-1460 (1992); Knowledge Acquisition 4 (4), (1993)]. Other research contributions of Tecuci are in the areas of expert systems, advanced robotics, compiler generation, and the application of graph theory to vision and to the analysis of electrical circuits.

Tecuci is the co-editor of "Machine Learning: A Multistrategy Approach", which is the first comprehensive book on this topic, as well as the forth volume in the series of classical books "Machine Learning". Previous volumes were co-edited by R.S. Michalski, J.G. Carbonell, T.M. Mitchell and Y. Kodratoff. Tecuci was the program chairman and a co-organizer of several workshops on new research directions in artificial intelligence: the First International Workshop on Multistrategy Learning (MSL-91, Harpers Ferry, West Virginia), the Second International Workshop on Multistrategy Learning (MSL-93, Harpers Ferry, West Virginia), and the IJCAI-93 workshop Machine Learning and Knowledge Acquisition: Common Issues, Contrasting Methods and Integrated Approaches (Chambery, France).

Tecuci was also a tutorial speaker on Multistrategy Learning at the two most important artificial intelligence conferences, the International Joint Conference on Artificial Intelligence (Chambery, France, 1993) and the National Conference of the American Association for Artificial Intelligence (Washington D.C., 1993). He is the guest editor of this special issue of Informatica on multistrategy learning, and of a special issue of the Knowledge Acquisition Journal on the integration of machine learning and knowledge acquisition. He served the scientific community as program chairman, program committee member, organizer, member of the editorial board, or reviewer of several conferences, journals, and publishers. He speaks four languages, Romanian, English, French, and Italian.

Edited by *A.P. Železnikar*

MULTISTRATEGY APPROACHES TO LEARNING: WHY AND HOW

Gheorghe Tecuci
Center for Artificial Intelligence, Department of Computer Science,
George Mason University, Fairfax, VA 22030, U.S.A.,
and
Center for Machine Learning, Natural Language Processing and Conceptual Modelling,
Romanian Academy, Bucharest, Romania
tecuciQcs.gmu.edu

Keywords: case-based reasoning, machine learning, multistrategy approach, single-strategy learning

Machine Learning is concerned with developing systems which are able to improve themselves by learning from an external source of information (e.g. a teacher or a collection of data) or from their own experience.

Research in this area can be traced back to the beginning of Artificial Intelligence. However, it is in the last several years that Machine Learning has greatly expanded and diversified.

Most machine learning research has been concerned with single strategy learning methods which illustrate basic forms of learning. Examples of single-strategy learning are:

— *empirical inductive learning from examples,* which consists of learning the definition of a concept by comparing positive and negative examples of the concept in terms of their similarities and differences, and inductively creating a generalized description of the similarities of the positive examples;

— *explanation-based learning,* which consists of learning an operational definition of a concept by proving that an example is an instance of the concept and by deductively generalizing the proof;

— *learning by analogy,* which consists of learning new knowledge about an entity by transferring it from a similar entity;

— *case-based reasoning and learning,* which consists of using past experiences to deal with new situations and then extending and improving the descriptions of these experiences;

— *abductive learning,* which consists of hypothesizing causes based on observed effects;

- *reinforcement learning,* which consists of updating the knowledge, based on feedback from the environment;

- *genetic algorithm-based learning,* which consists of evolving a population of individuals over a sequence of generations, based on models of heredity and evolution;

- neurai *network learning,* which consists of evolving a network of simple units to achieve an input-output behavior based on a simplified model of the brain's dendrites and axons.

Single-strategy learning methods have been successfully used to build adaptive intelligent agents. However, each such method, taken separately, has a limited applicability because it requires a special type of input and background knowledge, and it only learns a specific type of knowledge.

For instance, an empirical inductive learning system needs many input examples in order to learn, but does not require much prior knowledge. On the other hand, an explanation-based learning system can learn from only one example, but it needs complete prior knowledge. Many real-world learning situations are such that the learner has more than one example, but not enough to successfully apply an empirical inductive learning strategy. However, the learner usually has more prior knowledge than needed by empirical induction, but not enough for applying an explanation-based learning strategy.

This comparison can easily be extended to the other single-strategy approaches to learning. For instance, a learning by analogy system could learn additional features of some input entity only if it has a similar entity in the knowledge base. Similarly, an abductive learning system could learn new knowledge about an input situation only if it has causal knowledge that could explain the situation. Moreover, each of these single-strategy learners produces a different (and often complementary) result.

The complementary nature of the requirements and results of the single-strategy learning methods naturally suggests that by properly integrating them, one could obtain a synergistic effect in which different strategies mutually support each other, and compensate for each other's weaknesses. As a result, one may build a *multistrategy learning* system that is applicable to a wider spectrum of problems.

There are several general frameworks for the design of multistrategy learning systems.

One such framework consists of a global control module, and a toolbox of single strategy learning modules, all using the same knowledge base. The control module analyzes the relationship between the input and the knowledge base and decides which learning module to activate.

Another framework consists of a cascade of single strategy learning modules, in which the output of one module is an input to the next module.

Yet another framework consists of integrating the elementary inferences (like deduction, analogy, abduction, generalization, specialization, abstraction, concretion, etc.) that generate the individual learning strategies, and thus achieving a deep integration of these strategies.

Early multistrategy learning research has often been limited to the integration of empirical inductive learning and explanation-based learning. Besides integrating a greater range of learning strategies, current research in multistrategy learning addresses the utilization of learning goals and the adaptation of the learner to the learning task.

This special issue of Informatica contains updated versions of papers that have been selected from among those presented at the Second International Workshop on Multistrategy Learning, organized by George Mason University, in May 1993. The papers are representative of the current research by leading researchers in multistrategy learning.

Gordon and *Subramanian* present a multistrategy learning method for designing and refining knowledge for autonomous agents.

The method combines two complementary strategies, deductive concretion (a form of explanation-based learning) and inductive refinement (by a genetic algorithm).

First, high-level advice provided by a human is deductively operationalized in all possible ways, resulting in a set of operational rules for the autonomous agent.

Then these rules are inductively refined by a genetic algorithm, producing a set of better rules for the agent.

The two main advantages of this multistrategy learning method are a high convergence rate (due to the operationalized advice which biases the genetic algorithm into a favorable region of the search space) and robustness (due to the genetic algorithm-based learning).

Ram and *Santamaria* present a multistrategy learning system for robot navigation (called SINS), which combines case-based reasoning and reinforcement learning.

The case-based reasoning component of SINS perceives the robot's environment and retrieves an appropriate case which recommends parameter values for the control system of the robot.

The reinforcement learning component of SINS updates the content of a case to reflect the current experience.

Extensive experiments show that SINS is able to achieve the same level of performance as non-adaptive systems acting in environments for which they have been optimized. Moreover, SINS is able to achieve this level of performance autonomously, and can cope with sudden changes in the characteristics of the environment.

Widmer presents a multistrategy learning method for predicting numeric values. The method, which is implemented in the system IBL-SMART, combines multistrategy symbolic concept learning and numeric instance-based learning (which is a form of case-based reasoning and learning).

The multistrategy symbolic concept learner uses deductive and inductive specialization operators to learn rules for partitioning a set of training examples into different instance spaces.

The prediction of a numeric value associated with a new instance takes place in two stages. In the first stage, IBL-SMART determines the symbolic rule which covers the new instance and hence the corresponding instance space. In the second stage, the instance-based learner predicts the numeric value of the new instance based on the value of the nearest neighbor in the instance space.

The most important advantage of this multistrategy approach is that it provides a way of using qualitative domain knowledge for predicting precise numeric values.

Baffes and *Mooney* address the general problem of theory refinement which consists of improving an incomplete and/or incorrect rule base of a classification system to correctly classify a set of input training examples.

The system EITHER, which combines deduction, abduction and empirical induction, showed that multistrategy learning from theory (the rule base) and data (the input examples) gives better results than single strategy learning from data alone (i.e. learning the rule base only from examples).

This paper presents NEITHER which is a major revision of EITHER. NEITHER extends the representation of the rule base to include M-of-N rules, and also improves significantly the speed of learning.

Finally, *Hieb* and *Michalski* present a new knowledge representation formalism, dynamic interlaced hierarchies (DIH), that is specially developed to facilitate the performance of the basic inferential learning strategies, deduction, induction and analogy.

The idea is to represent the knowledge in the form of concept hierarchies and traces that link nodes of different hierarchies. Within this representation the basic inferential operations such as generalization, abstraction, similization, as well as their opposites, specialization, concretion and dissimilation, could be performed by modifying the traces in different ways.

The goal of DIH design is to provide a foundation for developing multistrategy learning systems that could adapt their learning strategy to the learning task characterized by a learner's input, knowledge and goal.

These five papers illustrate only a few representative approaches to multistrategy learning. Other approaches are presented in the papers from the following bibliography.

Besides developing multistrategy learning methods and systems, research in multistrategy learning is also concerned with the analysis and comparison of different learning strategies (which identify their capabilities and limits) and with human learning (which is intrinsically multistrategy).

Through its research goals, multistrategy learning contributes to the unification of the entire field of machine learning, and is one of the field's most significant new developments.

References

[1] Buchanan B., and Wilkins D. (Eds.), *Readings in Knowledge Acquisition and Learning: Automating the Construction and the Improvement of Programs.* Morgan Kaufmann, San Mateo, CA, (1992).

[2] Kodratoff Y., and Michalski R. S. (Eds.), *Machine Learning: An Artificial Intelligence Approach, III.* Morgan Kaufmann, San Mateo, CA, (1990).

[3] Michalski, R.S, and Tecuci, G. (Eds.), *Proceedings of the First International Workshop on Multistrategy Learning.* Organized by George Mason University, Harpers Ferry, WV, November 7-9, (1991).

[4] Michalski, R.S, and Tecuci, G. (Eds.), *Proceedings of the Second International Workshop on Multistrategy Learning.* Organized by George Mason University, Harpers Ferry, WV, May 26-29, (1993).

[5] Michalski R.S. and Tecuci G. (Eds.), *Machine Learning: A Multistrategy Approach, IV.* Morgan Kaufmann, San Mateo, CA, (1993).

[6] Shavlik J., and Dietterich T. (Eds.), *Readings in Machine Learning.* Morgan Kaufmann, San Mateo, CA, (1990).

[7] Tecuci G., *DISCIPLE: A Theory, Methodology and System for Learning Expert Knowledge.* These de Docteur en Science, Universite de Paris-Sud, France, (1988).

[8] Tecuci G., *Plausible Justification Trees: a Framework for the Deep and Dynamic Integration of Learning Strategies.* Machine Learning Journal, vol.11, pp.237-261, (1993).

A MULTISTRATEGY LEARNING SCHEME FOR AGENT KNOWLEDGE ACQUISITION

Diana Gordon
Naval Research Laboratory, Code 5514
Washington D.C. 20375
gordonQaic.nrl.navy.mil
AND
Devika Subramanian
Department of Computer Science
Cornell University
Ithaca, NY 14853
devikafics.Cornell.edu

Keywords: multistrategy learning, advice taking, compilation, operationalization, genetic algorithms

Edited by: Gheorghe Tecuci

Received: May 26, 1993 **Revised:** October 8, 1993 **Accepted:** October 15, 1993

The *problem of designing and refining task-level strategies in an embedded multiagent setting is an important unsolved question. To address this problem, we have developed a multistrategy system that combines two learning methods: operationalization of high-level advice provided by* a *human and incremental refinement by a genetic algorithm. The first method generates seed rules for finer-grained refinements by the genetic algorithm. Our multistrategy learning system is evaluated on two complex simulated domains as well as with a Nomad 200 robot.*

1 Introduction

The problem of designing and refining task-level strategies in an embedded multi-agent setting is an important unsolved question. To address this problem, we have developed a multistrategy learning system that combines two learning methods: operationalization of high-level advice provided by a human, and incremental refinement by a genetic algorithm (GA). We define advice as a recommendation to achieve a goal under certain conditions. Advice is considered to be operationalized when it is translated into stimulus-response rules directly usable by the agent. Operationalization generates seed rules for finer-grained refinements by a GA.

The long term goal of the work proposed here is to develop task-directed agents capable of acting, planning, and learning in worlds about which they do not possess complete information. These agents refine factual knowledge of the world they inhabit, as well as strategic knowledge for achieving their tasks, by interacting with the environment. *Agent knowledge acquisitionis* desirable for the same reasons that knowledge acquisition for expert systems is: *It* is easier for a user to provide Wgh-level knowledge about the world and the task than to provide knowledge at a lower level of detaiL The latter is well-known to be a costly, tedious, and error-prone process. Although agent knowledge acquisition is desirable, it is very difficult (for the agent)' as is knowledge acquisition for expert systems. The additional challenge for agent knowledge acquisition comes from the fact that the knowledge must be dynamically updated ^ the agent through its interactions with the environment.

There are two basic approaches to constructing agents for dynamic environments. The first decomposes the design into stages: a parametric design followed by refinement of the parameter values using feedback from the world in the context of the task. Several refinement strategies have been studied in the literature: GAs

[21], neural-net learning [3], statistical learning [13], and reinforcement learning [14]. The second, more ambitious, approach [7, 30] is to acquire the agent knowledge directly from example interactions with the environment. The success of this approach is tied to the efficacy of the credit assignment procedures, and whether or not it is possible to obtain good training runs with a knowledge-impoverished agent.

We have adopted the first approach. The direction we pursue is to compile an initial parametric agent using high-level strategic knowledge (e.g., advice) input by the user, as well as a body of general (not domain-specific) spatial knowledge in the form of a *Spatial Knowledge Base* (SKB). The SKB contains qualitative rules about movement in space. Example rules in our SKB are If something is on my side, and I turn to the other side, I will not be facing it and If I move toward something it will get closer. This SKB is portable because it is applicable to a variety of domains where qualitative spatial knowledge is important. A similar qualitative knowledge base was constructed by [18] for the task of pushing objects in a plane. Since the knowledge provided to our agent is imperfect (incomplete and incorrect), our agent refines the knowledge using a GA by directly interacting with the world.

First we describe our deductive advice operationalization process and the nature of the parameterization adopted for our agent. Then we describe the inductive (GA) refinement stage and compare our multistrategy approach with one that is purely inductive. Before we present the details of the method, we characterize the class of environments and tasks for which we have found this decomposition of an agent design into an initial parametric stage and subsequent refinement stage to be effective.

- *Environment characteristics:* Complete models of the dynamics of the environment in the form of differential equations or difference equations, or discrete models like STRIPS operators, are unavailable. An analytical design that maps the percepts of an agent to its actions (e.g., using differential game theory or control theory) in these domains is thus not possible. Even if a model were available, standard analytical methods for deriving agents are extensional and involve exploration of the entire state space. They are inapplicable here because the domains we consider have of the order of a hundred million states.

– *Task characteristics:* Tasks are sequential decision problems: payoff is obtained at the end of a sequence of actions and not after individual actions. Examples are pursuit-evasion in a single or multi-pursuer setting and navigating in a world with moving obstacles. The tasks are typically multi-objective in nature: for instance for pursuit-evasion, the agent needs to minimize energy consumption while maximizing the time till capture by the pursuers.

– Agent *characteristics:* The agent has imperfect sensors. Imperfections occur in the form of noise, as well as incompleteness (all aspects of the state of the world cannot be sensed by our agent, a problem called *perceptual aliasing* in [32]). Stochastic differential game theory has methods for deriving agents with noisy sensors, but it requires detailed models of the noise as well as a detailed model of the environment and agent dynamics.

The action set of the agents and the values taken on by sensors are discrete and can be grouped into equivalence classes. This is the basis for the design of the parametric agent. A similar intuition underlies the design of fuzzy controllers that divide the space of sensor values into a small set of classes described by linguistic variables.

In domains with characteristics such as those just described, human designers typically derive an initial solution by hand and use numerical methods (usually very dependent on the initial solution) to refine their solution. Our ultimate objective is to automate the derivation of good initial solutions by using general knowledge about the environment, task, and agent characteristics and thus provide a better starting point for the refinement process. We begin with the SKB and advice.

2 Compiling Advice

Our operationalization method compiles high-level domain-specific knowledge (e.g., advice) and spatial knowledge (SKB) into low-level reactive rules directly usable by the agent. The compilation performs *deductive concretion* [16] because it deductively converts abstract goals and other knowledge into concrete actions. An important question is why we adopt a deductive procedure for the operationalization of advice. At this time, we are able to generate good parametric designs with deductive inference alone. We expect that as we expand our experimental studies to cover more domains, the incompleteness of the SKB will force us to adopt more powerful operationalization methods.

The advice compilation problem can be stated as follows:

Given:

- Strategic advice of the form: *Set of States* —> *achieve(Goal),* which recommends that *Goal* be achieved (be made to hold in the world) whenever the agent finds itself in a state that matches the antecedent of the advice. Note that there is no requirement that the set of states in the antecedent be directly perceivable. The agent may have to plan to acquire the information needed to determine whether or not it is in the specified set of states.

- Qualitative factual knowledge about the domain in the form of a description of a *Set of States.*

- A qualitative theory with information of the following types:

  (1) Domain-specific rules. a. A set of domain-specific terminological mappings that define terms needed for interpreting the advice: these mappings are of the form *Set of states —y Set of states* and are vocabulary conversion rules, *b.* A set of state transition rules that represent the dynamics of the world: these mappings are of the form *Set of states* x *A* —• *Set of states,* where *A* is the set of actions, c. A set of rules for predicting other agent's actions: these mappings are of the form *Set of States* x *A* —» *A.* These rules are special to multi-agent domains.

  (2) A general qualitative theory of movement (SKB). This is a common subpart of the domains that we consider here. Therefore it need not be reintroduced by the user for each new domain. The SKB is of the form *Set of states* x *AM —* Set of states,* where *AM* is the subset of actions *A* that involve movement.

- The operational sensors *P* and actions *A* of the agent. A detailed agent design involves finding a mapping from the sense history *P** of the agent to *A.* This is usually simplified to be of the form 2^P —> *A* so that only current percepts determine the next action. The map need not be deterministic. Indeed, in the GA architecture, which is our target architecture, the agent map is non-deterministic.

Find: An operational mapping 2^P —> *A* from the sensors to the effectors of the agent such that the map implements the provided strategic advice. By this we mean that the map tests the conditions under which the recommended goal is to be achieved, and the action sequence it generates in the world places the agent in a state in which the goal is achieved.

In summary, advice tells the agent to achieve a goal under specified conditions. Facts, domain-specific rules, a general theory of movement, and a list of what is operational are all required for interpreting this advice and compiling it into an operational form.

As an example, consider the problem of an agent that tries to evade a pursuing adversary, (For a more detailed description of this problem, see Section 4.3.) We assume the adversary is initially faster than the agent, though it is less maneuverable. Both the adversary and the agent lose speed as they turn, and the adversary's speed loss is permanent. The agent and the adversary can sense each other's instantaneous position and velocity to within some specified accuracy. Although these are facts of the problem, they are initially unknown to the agent.

The strategic advice offered by a human expert for this problem is the following: if the adversary's speed is high, then try to slow down the adversary; if the adversary's speed is low, then avoid the adversary. In this example the preconditions of the advice involve predicates that the agent can sense; however it needs to pin down the semantics of *high* and *low*. The goals to be achieved are: slow down the adversary in the one case, and avoid the adversary in the other. These are not directly implement able using a single action from *A*. We need to devise a plan to realize these goals using knowledge about the domain, including knowledge about movements. These plans form the initial parametric design with *high* and *low* being two of the parameters whose ranges need to be learned by interaction with the environment. An example SKB rule that is used to compile our advice is:

```
IF bearing(agent,adversary) = right AND
   turn(agent) = left
THEN heading(adversary,agent) ^ headon.
```

Heading *(X ,Y)* refers to the direction of motion of *Y* relative to *X*, and bearing (X.Y) refers to the direction of *Y* relative to *X*. This SKB rule states that an agent no longer faces an adversary when it turns away from it. Turn is an operational action.

To compile the advice, we also need terminological mappings that define terms such as *avoid*. An example definition needed to compile our advice is:

```
IF range(X.Y) # close AND
heading(Y,X) ^ headon
THEN avoids{X, Y).
```

Avoids(X, *Y)* means *X* avoids *Y*. Factual knowledge is also required for compilation, e.g.,

```
Moving(agent).
Moving(adversary).
```

From our advice, the set of operational sensors and effectors, and our qualitative theory, compilation results in a set of operational sensor-effector mappings. The set of all these operational sensor-effector mappings is what we consider to be a *reactive plan* (agent map). A portion of the reactive plan that implements our advice, and is the product of our compilation procedure, is shown below.

```
IF speed(adversary) = high AND
   range(agent.adversary) = close
THEN turn(agent) = hard
IF speed(adversary) = high AND
   range(agent,adversary) -£ close AND
   bearing(agent,adversary) = left
THEN turn(agent) = left
IF speed(adversary) = high AND
   range(agent,adversary) ^ close AND
   bearing(agent,adversary) = right
THEN turn(agent) = right
IF speed(adversary) = low AND
   bearing(agent,adversary) = right
THEN turn(agent) = left
IF speed(adversary) = low AND
   bearing(agent,adversary) = left
THEN turn(agent) = right
```

This is a fairly complex plan. It implements the strategic advice in the following manner: when the adversary's speed is high, the agent moves toward the adversary and makes a hard turn when it is close enough. Since the adversary is chasing the agent, this plan will cause the adversary to turn hard, which then causes the adversary to permanently lose speed. If the adversary is moving slowly, the agent simply stays out of its way. In this section we present an algorithm that generates plans of this form starting from high level advice and a qualitative domain theory.

Our compilation algorithm is tailored for situations in which the execution of plans, but not the learning of plans, is highly time-critical. We assume the advice provided is operationalized immediately, though it need not be applied immediately. We precompile all high-level knowledge into an operational form because the agent will apply it in a time-critical situation. In our approach, advice and SKB rules have nonoperational elements, and the compilation process results in rules that are fully operational.

The compilation algorithm, shown in Figure 1, uses two stacks: a GoalStack and an (operational condition) OpCondStack. Four types of knowledge are initially given to the compiler: advice, facts, nonoperational rules (abbreviated *nonop rules),* which include domain-specific rules and the SKB, and the set of operational sensors and actions. The output from compilation is a set of *op rules* directly usable by the agent, i.e., operational. Advice has the form:

<IF cond AND ... AND cond THEN> ACHIEVE goal.

Facts have the form:

Predicate(X_1, ... X_n)

Nonop rules have the form:

IF cond AND ··· AND cond <AND action>
THEN goal.

The set of operational sensors and actions are presented to the agent as a list.

Anything in angle brackets is optional. The portion preceding the THEN is the rule *antecedent*, and the portion following the THEN is the rule *consequent*. A nonop rule consequent is a single goal. The syntax for a goal is function(X_1,...,X_n) =/≠ value or predicate(X_1,...,X_n). Each X_i is an object (e.g., an agent). The syntax for a cond (condition) or action in the rule antecedent is the same as for goals.

Although advice has a similar syntax to nonop rules, its interpretation differs. Advice recommends achieving the given goal under the given conds. A nonop rule, which is needed to compile this advice, states that the given goal will be achieved if the given conds (and action) occur. Compilation results in stimulus-response op rules of the form:

IF cond AND ... AND cond THEN action.

The conditions of an op rule are sensor values detectable by the agent. The action can be performed by the agent.

Our compilation algorithm in Figure 1 takes advice and backchains through the SKB and user-provided nonop rules until an operational action is found. Once an operational action is found, it pops back up the levels of recursion, attaching operational conditions along the way, to form a new reactive agent op rule. To prevent cycles, the last nonop rule used in step 3 is marked as "used" so that it will not be used again.

Let us examine a simple example of how this algorithm operates, as shown in Figures 2 and 3. Consider the advice which advocates avoiding the adversary when the adversary's speed is low.

IF speed(adversary) = low THEN
ACHIEVE heading(adversary,agent) ≠ headon

Figure 2 shows how SKB nonop rules match this advice for backchaining, thereby creating an "and" tree. Anything preceded by a "*" is operational.

```
Push advice on GoalStack: goal followed by
     conditions.
Initialize OpCondStack to be empty and invoke
     Compile(GoalStack,OpCondStack).
Procedure Compile (GoalStack,
     OpCondStack)
     if GoalStack is not empty then
        g<—pop(GoalStack);
        case g:
           1. g is an operational condition:
                 Push(g, OpCondStack);
                 Compile(GoalStack,
                 OpCondStack);
           2. g matches a fact:
                 Compile(GoalStack,
                 OpCondStack);
           3. g is nonoperational:
                 foreach nonop rule R_i whose
                 consequent matches g do
                     Push(antecedent(R_i),
                     GoalStack);
                     Compile(GoalStack,
                     OpCondStack);
           4. g is an operational action
                 Form a new op rule from the
                 contents of
                 OpCondStack and g;
                 Clear OpCondStack;
     else Clear OpCondStack;
```

Figure 1: Algorithm for operationalizing advice

Figure 3 shows this algorithm in operation. Note that stacks grow downward. The algorithm begins by pushing the advice goal, followed by the advice condition, on the GoalStack. It then calls procedure Compile, which moves the advice condition to the OpCondStack because it is operational. The advice goal is not operational. In our example, the advice goal can be unified with the goal of SKB RULE1, which states

IF bearing(agent.adversary) = right AND
  turn(agent) = left
THEN heading(adversary,agent) ≠ headon.

The condition and action of RULE1 are pushed on the GoalStack. Because the condition of RULE1 is operational, it is moved to the OpCondStack. Note that if this condition had not been oper-

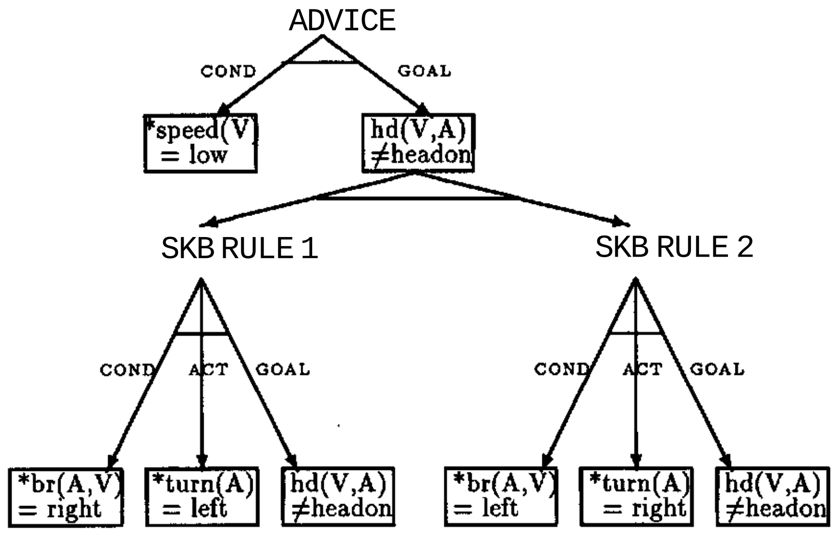


Figure 2: Graph of example. A is short for agent, V for adversary, br for bearing and hd for heading.

ational, our compilation algorithm would have searched for all nonop rules whose consequents unify with this nonoperational condition. Further backchaining would have continued until operational conditions were found.

At this point, the action of RTJLE1 is at the top of the GoalStack, and it is operational, so we can create an op rule. The conditions from the OpCondStack are added to the action. This creates an op rule that states

```
IF speed(adversary) = low AND
   bearing(agent,adversary) = right
THEN turn(agent) = left
```

Both stacks are cleared. The algorithm (see Figure 2) continues similarly to generate a second op rule from SKB RULE2 that states

```
IF speed(adversary) = low AND
   bearing(agent, adversary) = left
THEN turn(agent) = right
```

Next, we apply a conversion from qualitative to quantitative op rules. The rules are given default quantitative ranges. For example, if speed has two values, slow and fast, we bisect the range of all possible values into two subranges. Then, we allow the system to improve this initial choice of quantitative ranges by using a GA to refine the initial ranges while interacting with the environment.

3 Executing and Refining Advice

The system we use to refine and apply the op rules derived from our compiled advice is the SAMUEL reactive planner [7]. We have chosen SAMUEL because this system has already proven to be highly effective for refining rules on complex domains [7, 25]. SAMUEL adopts the role of an agent in a multiagent environment in which it senses and acts. This system has two major components: a performance module and a learning module. Section 4.2 explains how performance interleaves with learning in our experiments.

The performance module, called the Competitive Production System (CPS), interacts with a simulated or real world by reading sensors, setting effector values, and receiving payoff from a critic. CPS performs matching and conflict resolution on the set of op rules. This performance module follows the match/conflict-resolution/act cycle of traditional production systems. Time is divided into *episodes:* the choice of what constitutes an episode is domain-specific. Episodes begin with random initialization and end when a critic provides payoff. At each time step within an episode, CPS selects an action using a probabilistic voting scheme based on rule strengths. All rules that match (or partially match - see [7]) the current state bid to have their actions fire. The actions of rules with higher strengths are more likely to

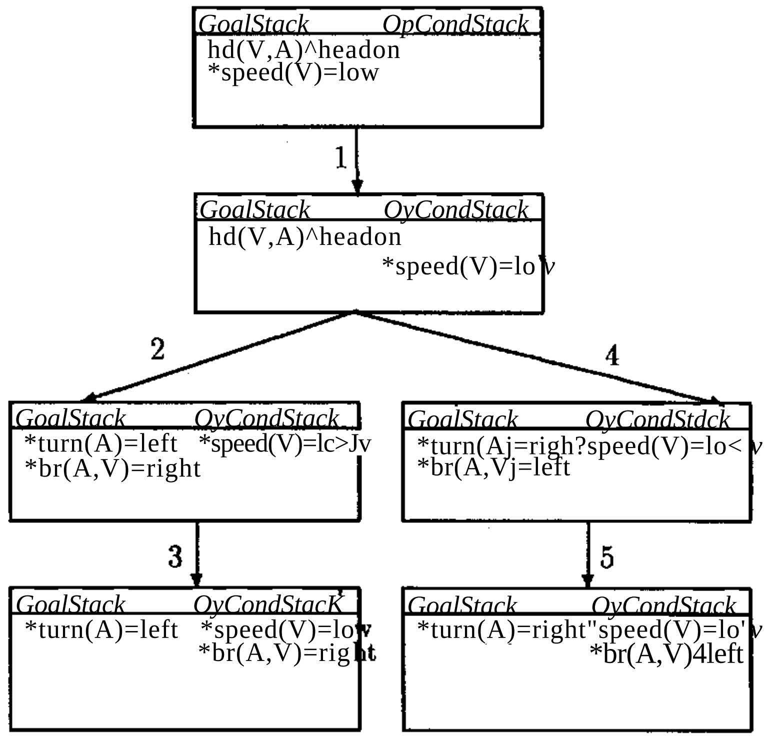


Figure 3: Example of compilation algorithm

fire. If the world is being simulated, then after an action fires, the world model is advanced one simulation step and sensor readings are updated.

CPS assigns credit to individual rules based on feedback from the critic. At the end of each episode, all rules that suggested actions taken during this episode have their strengths incrementally adjusted to reflect the current payoff. Over time, rule strengths reflect the degree of usefulness of the rules.

SAMUEL's learning module is a genetic algorithm. GAs are motivated by simplified models of heredity and evolution in the field of population genetics [8]. GAs evolve a population of individuals over a sequence of *generations*. Each individual acts as an alternative solution to the problem at hand, and its *fitness* (i.e., potential worth as a solution) is regularly evaluated. During a generation, individuals create *offspring* (new individuals). The fitness of an individual probabilistically determines how many offspring it can have. Genetic operators, such as *crossover* and *mutation,* are applied to produce offspring. Crossover combines elements of two individuals to form two new individuals; mutation randomly alters elements of a single individual. In SAMUEL, an individual is a set of op rules, i.e., a reactive plan. In addition to genetic operators, this system also applies non-genetic knowledge refinement operators, such as *generalize* and *specialize,* to op rules within a rule set.

The interface between our compilation algorithm and the SAMUEL system is straightforward. The output of our compilation algorithm is a set of op rules for the SAMUEL agent. Because the op rules may be incomplete, a *random rule* is added to this rule set. The random rule recommends performing a random action under any conditions. The final rule set, along with CPS and the GA learning component for improving the rules, is our initial agent.

4 Evaluation

We have not yet analyzed the cost of our compilation algorithm. The worst case cost appears to be exponential in the sizes of the inputs to the algorithm because the STRIPS planning problem can be reduced to it. In the future, we plan to investigate methods to reduce this cost for complex realistic problems. Potential methods include: (1)

attaching a likelihood of occurrence onto advice, which enables the agent to prioritize which advice to compile first if time is limited, (2) tailoring the levels of generality and abstraction of the advice to suit the time available for compilation (e.g., less abstract advice is closer to being operational and therefore requires less compilation time), and (3) generating a parallel version of the algorithm.

We have evaluated our multistrategy approach empirically. We focus on answering the following questions:

1. Will our advice compilation method be effective for a reactive agent on complex domains?

2. Will the coordination of multiple learning techniques lead to improved performance over using any one learning method? In particular, we want the GA to increase the success rate of the compiled advice, and the advice to reduce the convergence rate of the GA. Success and failure have domain-specific definitions (see Sections 4.3 and 4.4). The convergence rate is defined as the time taken to ascend to a given success rate. A reduced convergence rate is useful when learning time is limited.

3. Can we construct a portable SKB?

4.1 Domain characteristics

To address our questions, we have run experiments on two complex problems: Evasion and Navigation. Our choice of domains is motivated by the results of Schultz and Grefenstette, who have obtained large performance improvements by initializing the GA component of SAMUEL with hand-coded op rules in these domains [25]. Their success has inspired this work: our objective is to automate their tedious manual task, and the work described here is one step toward the goal.

Both problems are two-dimensional simulations of realistic tactical problems. However, our simulations include several features that make these two problems sufficiently complex to cause difficulties for more traditional control-theoretic or game-theoretic approaches [7]:

— A weaA' *domain model:* The learner has no initial model of other agents or objects in the domain. Most control-theoretic and game-theoretic models make worst case assumptions about adversaries. This yields poor designs in the worlds we consider because we have statistical rather than worst case adversaries.

— *Incomplete* state *information:* The sensors are discrete, which causes perceptual aliasing: a many-to-one mapping from actual states to perceived states.

- A large state space: The discretization of the state space makes the learning problem combinatorial. In the Evasion domain, for instance, over 25 million distinct feature vectors are observed.

— *Delayed payoff:* The critic only provides payoff at the end of an episode. Therefore a credit assignment scheme is required.

- *Noisy sensors:* Gaussian noise is added to all sensor readings. Noise consists of a random draw from a normal distribution with mean 0.0 and standard deviation equal to 5% of the legal range for the corresponding sensor. The value that results is discretized according to the defined granularity of the sensor. A 5% noise level is sufficient to slightly degrade SAMUEL's performance.

4.2 Experimental design

Two sets of experiments are performed on each of the two domains. Perception is noise-free for the first set, but noisy for the second. The primary purpose of the first set is to address our question about the effectiveness of our advice compilation method alone, without GA refinement. Facts, nonop rules, advice, and the set of operational sensors and actions are given to the compiler and the output is a set of op rules. The random rule is added to the op rules and this rule set is given to SAMUEL's CPS module to be applied within the simulated world model. The baseline performance with which these rules are compared is the random rule alone. These experiments measure how the success rate of the compiled rules compares with that of the baseline as problem complexity increases, where the success rate is an average over 1000 episodes. Statistical significance of the differences between the curves with and without advice are presented. Significance is measured using the large-sample test for the differences between two means.

The primary purpose of the second set of experiments is to address our question about the effectiveness of the multistrategy approach (compilation followed by GA refinement). We used the same set of op rules (i.e., the output of the compiler) for this second set of experiments as was used for the first set of experiments. This rule set, plus the random rule, becomes every individual in SAMUEL's initial GA population, i.e., it seeds the GA with initial knowledge. The baseline performance with which these rules are compared is SAMUEL initialized with every individual equal to just the random rule. In either case, GA learning evolves this initial population. In other words, we compare the performance of advice seeding the GA with GA learning alone (i.e., random seeding). Random seeding produces an initially unbiased GA search; advice initially biases the GA search - hopefully into favorable regions of the search space.

In this second set of experiments, performance of the agent in the environment interleaves with GA refinement. SAMUEL runs for 100 generations using a population size of 100 rule sets. Every 5 generations, the *best* (in terms of success rate) 10% of the current population are evaluated over 100 episodes to choose a single plan to represent the population. This plan is evaluated on 1000 randomly chosen episodes and the success rate is calculated. This entire process is repeated 10 times and the final success rate, averaged over all 10 trials, is found. The curves in our graphs plot these averages. For this set of experiments, statistical significance is measured using the two-sample £-test, with adjustments as required whenever the *F* statistic indicates unequal variances.

We add sensor noise, as defined in Section 4.1, for this second set of experiments because GAs can learn robustly in the presence of noise [7]. Two performance measures are used: the success rate and the convergence rate. The convergence rate is defined as the number of GA generations required to achieve and maintain an *n%* success rate, where *n* is different for each of the two domains. The value of *n* is set empirically.

4.3 Evaluation on the Evasion problem

Our simulation of the Evasion problem is partially inspired by [4]. This problem was introduced in Section 2. The problem consists of an agent that moves in a two-dimensional world with a single adversary pursuing the agent. The agent is controlled by SAMUEL, and the adversary is controlled by a simple set of rules. The agent's objective is to avoid contact with the adversary for a bounded length of time. Contact implies the agent is captured by the adversary. The problem is divided into episodes that begin with the adversary approaching the agent from a random direction. The adversary initially travels faster than the agent, but is less maneuverable (i.e., it has a greater turning radius). Both the agent and the adversary gradually lose speed when maneuvering, but only the adversary's loss is permanent. An episode ends when either the adversary captures the agent (failure) or the the agent evades the adversary (success). At the end of each episode, a critic provides full payoff for successful evasion and partial payoff otherwise, proportional to the amount of time before the agent is captured. The strengths of op rules that fired are updated in proportion to the payoff.

The agent has the following operational sensors: time, last agent turning rate, adversary speed, adversary range, adversary bearing, and adversary heading. The agent has one operational action: it can control its own turning rate. For further detail, see [7].

In our experiments, we provide the following domain-specific knowledge to the agent:

FACTS:
```
Chased-by(agent,adversary).
Moving(agent).
Moving(advers ary).
```

```
DOMAIN-SPECIFIC NQNOP RULES:
IF chased-by(X,Y) AND
   range(X.y) = close AND
   turn(X)=Z
THEN turn(F) = Z.
IF range(X.y) ^ close AND
   heading(Y,X) ^ headon
THEN avoids(X,Y).
IF turn(adversary) = hard
THEN decelerates(adversary).
```

ADVICE:

```
IF speed(adversary) = high THEN
ACHIEVE decelerates(adversary).
IF speed(adversary) = low THEN
ACHIEVE avoids(agent,adversary).
```

We also include knowledge of the agent's operational sensors and actions as facts. Ten SKB nonop rules are used (they are instantiations of the two rules described in English in the introduction of this paper). Although room does not permit listing them all, some examples are:

```
IF bearing(X.Y) = right AND
   turn(X) = left
THEN heading(Y,X) ± headon.
IF bearing(X,Y) = left AND
   moving(X) AND
   turn(X) = left
THEN range(X.F) = close.
```

From our input and our SKB rules, the compilation method of Section 2 generates op rules. The sensor values of these rules are translated from qualitative values to default quantitative ranges. For example, bearing = left is translated into bearing = [6..12], where the numbers correspond to a clock, e.g., 6 means 6 o'clock. Every new rule is given an initial strength of 1.0 (the maximum). The final op rule set includes rules such as[1]

```
IF speed(adversary) = [700..1000] AND
   range(agent,adversary) = [0..750]
THEN turn(agent) = hard-left.
```

The total number of op rules generated from our advice is 16.

We begin our experiments by addressing the first question, which concerns the effectiveness of our advice-taking method. We do not use the GA. Problem difficulty is varied by adjusting a *safety* envelope around the agent. The safety envelope is the distance at which the adversary can be from the agent before the agent is considered captured by the adversary.

Figure 4 shows how the performance (averaged over 1000 episodes) of these op rules compares with that of just the random rule. All of the differences between the means are statistically significant (using significance level a = 0.05). From Figure 4 we see that from difficulty levels 80 to 120, the agent is approximately twice as successful with advice than without it. This is a 100% performance advantage. Furthermore, for levels 130 to 160, the agent is about four times more effective with advice. For levels 160 to 200, the agent is an order of magnitude more effective with advice. We conclude that as the difficulty of this problem increases, the advice becomes more helpful relative to the baseline. These results answer our first question: our advice compilation method is effective on this domain.

[1]To generate a few of these rules, we used a variant of our compilation algorithm. We omitted a description of this variation for the sake of clarity. See [6] for details.

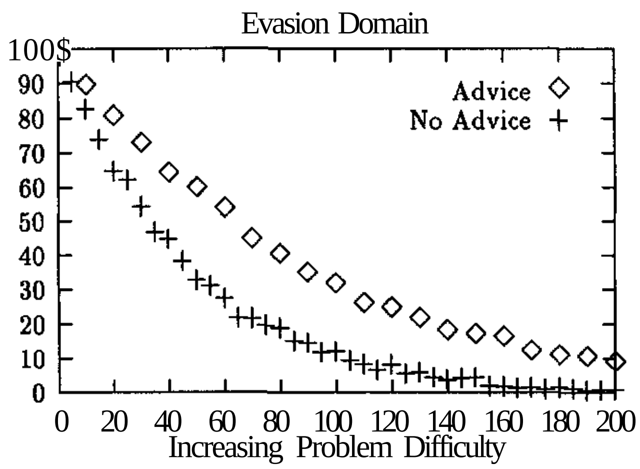


Figure 4: Results on the Evasion domain without GA refinement. Y axis denotes success rate.

We address the second question about multistrategy effectiveness by combining the compiled advice with GA refinement. Figure 5 shows the results of comparing the performance of the GA with and without advice. The safety envelope is fixed at 100 (chosen arbitrarily) and noise is added to the sensors. For this domain, the convergence rate is the number of GA generations required to maintain a 60% success rate.

Figure 5 shows that in a small amount of time (approximately 10 generations), the GA doubles the success rate of the advice rules. Furthermore, the addition of advice produces a 3.5-fold improvement in the convergence rate over using a GA with random initialization. The differences between the means are statistically significant for the first 65 generations, but not afterwards *(a* =

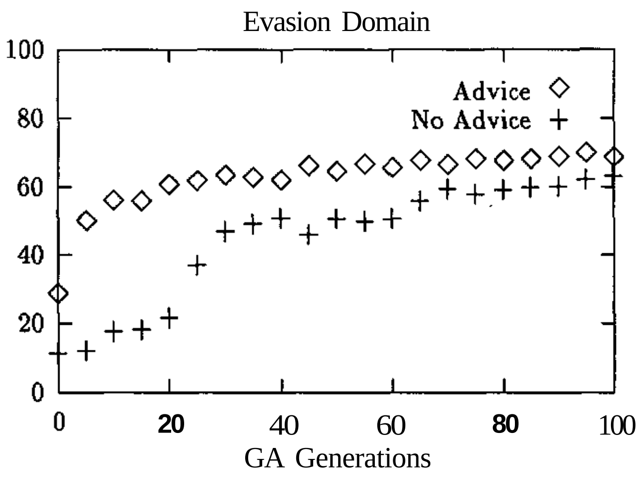


Figure 5: Results on the Evasion domain with the GA. Y axis denotes success rate.

0.05).[2] Apparently, the advice initially biases the GA into a favorable region of the search space, which improves the convergence rate over random initialization. The convergence value, however, does not appear to be higher with advice than without it because eventually the performance of the randomly initialized GA is roughly equivalent to that of the GA seeded with advice.

4.4 Evaluation on the Navigation problem

In the Navigation domain, our agent is again controlled by SAMUEL in a two-dimensional simulated world. The agent's objective is to avoid obstacles and navigate to a stationary target with which it must rendezvous before exhausting its fuel (implemented as a bounded length of time for motion). Each episode begins with the agent centered in front of a randomly generated field of obstacles with a specified density. An episode ends with either a rendezvous at the target location (success) or the exhaustion of the agent's fuel or a collision with an obstacle (failure). At the end of an episode, a critic provides full payoff if the agent reaches the target, and partial payoff otherwise, depending on the agent's distance to the goal.

The agent has the following operational sensors: time, the bearing of the target, the bearing and range of an obstacle, and the range of the target. The agent has two operational actions: it can control its own turning rate and its speed. For further detail, see [24].

We provide the following domain-specific knowledge (in addition to a list of operational sensors and actions):

```
FACTS:
Moving(agent).

DOMAIN-SPECIFIC NONOP RULES:
IF range(X,Y) ^ close AND
   heading(Y,X) ^ headon
THEN avoids(X,Y).

ADVICE:
IF range(agent,obstacle) ^ close THEN
ACHIEVE range(agent.target) = close.
IF range(agent,obstacle) = close THEN
ACHIEVE avoids(agent,obstacle).
ACHIEVE speed(agent) = high.
```

The *same* 10 SKB nonop rules used for Evasion are again used for this domain and are successful in the compilation procedure. This confirms the portability of our SKB for these two domains, thus addressing our third question. Also, although the definition of *avoids* was intended to be domain-specific, it actually applies to both of our domains. A total of 42 op rules are generated.[3]

Again, we address the first question by using SAMUEL without the GA. We increase the problem difficulty on this domain by increasing the number of obstacles. The success rate is an average over 1000 episodes. Without advice, the success rate is 0% because this is a very difficult domain. Figure 6 shows how we improve the success rate to as much as 90% by using our advice on this domain. We increase problem difficulty by increasing the number of obstacles. At all but the last few points, the differences between the means are statistically significant ($a = 0.05$). When we

[2]These results differ from the results presented in [6]. This discrepancy results from an improvement in the initial rule strengths. Future work will focus on studies analyzing system sensitivity to parameter variations. Our initial studies indicate that performance is more robust with respect to changes in some parameters, e.g., variations in the qualitative to quantitative mappings, than to changes in others, e.g., the initial rule strengths.

[3]We were able to decrease the number of op rules to 9 by making one careful qualitative to quantitative mapping choice.

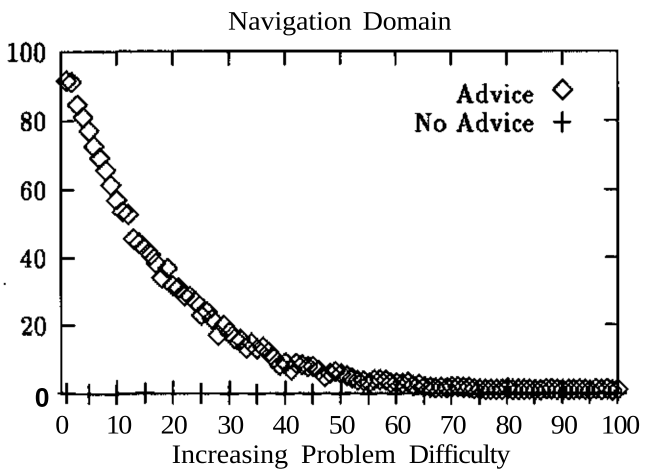


Figure 6: Results on the Navigation domain without GA refinement. Y axis denotes success rate.

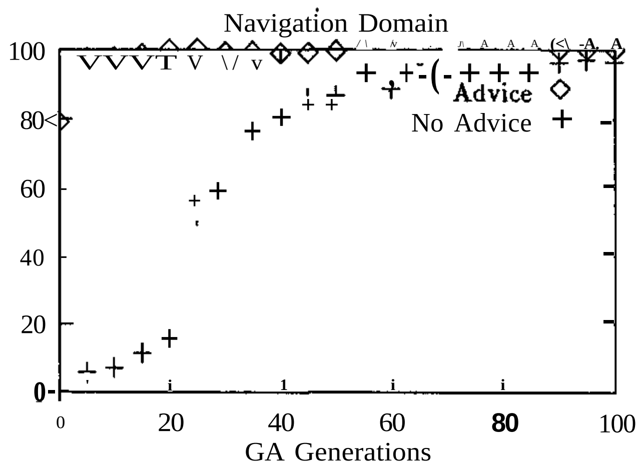


Figure 7: Results on the Navigation domain with the GA. Y axis denotes success rate.

vary the number of obstacles, performance follows a different trend than for the Evasion domain. By far the greatest benefit of the advice occurs when there are few obstacles. The success rate is 90% with advice and 0% without advice when there is only one obstacle, for example. The advantage drops as the problem complexity increases. After difficulty level 80, the particular advice we gave the agent no longer offers any benefit. For this level of problem difficulty, different advice is probably more appropriate.

Our experiments on both domains confirm that our advice compiler can be effective, however, they also indicate that the usefulness of advice may be restricted to a particular range of situations. Another learning task, which we are currently exploring, would be to identify this range and add additional conditions to the advice.

We address the second question by comparing the performance of the GA with and without advice. Noise is added. Figure 7 shows the results. All differences between the means are statistically significant (a = 0.05). Here, the number of obstacles is fixed at five (chosen arbitrarily). For this domain, the convergence rate is the number of GA generations required to maintain a 95% success rate.

Figure 7 shows that the addition of advice yields an enormous performance advantage on this domain. Figure 7 also shows that given a moderate amount of time (10 generations), the GA provides a 10% increase in the success rate. Furthermore, the addition of advice produces an 18-fold improvement in the convergence rate over using GAs alone. Not only does advice improve the convergence *rate,* but it also appears to improve the *level* of convergence: after 80 generations, the GA with advice holds a 99% or above success rate whereas after all 100 generations the GA without advice still cannot get above a 97% success rate.[4]

To further test our compilation method, we have recompiled our Navigation advice into op rules for a Nomad 200 mobile robot that is equipped with very noisy sonar and infrared sensors and can adjust its turning rate and speed. The sensors are so noisy that the robot sometimes mistakes two boxes four feet apart for a wall. The op rules that result from compilation have not been refined by the GA to develop a tolerance to noise; therefore, this noise poses a severe challenge.

The op rules are linked to a vendor-provided interface that translates the language of the SAMUEL rules (e.g., IF asonar4 [17..85] THEN SET turn -400) into joint velocity and servo motor commands. From high-level advice to avoid obstacles and rendezvous with a goal point, our method has compiled rules that enable the

[4] On both this domain and the one described in the previous section, our results are for 100 GA generations. In future work we plan to run the GA longer to determine whether our conclusions hold over a longer time period.

robot to succeed approximately a third of the time in avoiding three large boxes and reaching a goal point on the other side of a room. The same SKB rules are used for compilation. With the random rule alone, it is extremely unlikely to successfully complete this task. Our next step will be to refine these robot rules using GAs within a simulation of this actual domain.[5]

In conclusion, our multistrategy system offers two advantages. First, it provides an initial boost from seeding with high-level knowledge. On both simulated domains we see a significant improvement in the convergence rate - an order of magnitude on the Navigation domain. Second, the multistrategy system provides the robustness and improvement gained from GA refinement. Refinement yields a 10% increase in the success rate on Navigation and a 100% increase on Evasion. The fact that on both domains the advice biases the GA into a favorable region and significantly improves its convergence rate is very important in complex domains where a GA approach alone may not be feasible or may be very inefficient. Finally, the effectiveness of our advice-taking method has been confirmed on a robot with poor sensory capabilities.

To determine whether the presence of noise influences our conclusions, we have repeated our experiments on the two simulated domains with sensor readings that are not noisy. Although the success rates differ slightly, our conclusions still hold.

5 Related Work

This work relates most strongly to the following topics in machine learning: advice taking, combining projective and reactive planning, methods for compiling high-level goals into reactive rules, learning in fuzzy controllers, and multistrategy learning. This work also relates to research in differential game theory. We discuss each in turn.

5.1 Machine Learning

Advice taking has been considered by McCarthy as early as 1958 [15] and later by Mostow [19] and others. To date, research on assimilating advice in embedded agents has been limited but encouraging. Previous research has focused mainly on providing low-level knowledge. For example, Laird *et al.* [11] and Clouse and Utgoff [3] have had good success providing agents with information about which action to take. Chapman gives his agent high-level advice [2] . Our advice taker differs from Chapman's because it can operationalize advice long before the advice is applied and because it refines the advice with a GA. Most important of all, our advice taking method is unique because it involves a multistrategy approach that couples a knowledge-intensive *deductive* precompilation phase with an empirical *inductive* refinement phase.

We assume that high-level knowledge is operationalized but not applied immediately. Methods for operationalizing advice that will be applied immediately include STRIPS-like planners [20] and explanation-based learning (EBL) planners e.g., [26]. A closely related system is Mitchell's [17]. This system combines EBL projective planning with reactive planning. Our method for compiling goals is similar to that of EBL because it uses the notion of operationally. It differs because we do not assume that the advice will be applied immediately, and therefore our compilation method has no current state on which to focus plan generation. All of the above-mentioned methods create a projective plan to achieve a goal from the current state. We precompile advice for multiple possible states.

Because our method precompiles plans from possible states rather than from a current state, it is very similar to the methods of Schoppers [23] and Kaelbling [9] for compiling high-level goals into low-level reactive rules. Our method differs from those of Schoppers and Kaelbling because it includes the EBL notion of operationally. Also unlike Schoppers and Kaelbling, we use a refinement method following compilation.

Considerable prior work has focused on knowledge refinement. Others have used GAs to refine qualitative to quantitative mappings. For example, Karr uses GAs to select fuzzy membership functions for a fuzzy controller [10]. Lin, Mahadevan and Connell, and Singh initialize their systems with modular agent architectures then refine them with reinforcement learning [12], [14], and

[5]We wish to use SAMUEL both to handle the noise and because we had to manually refine the qualitative to quantitative mappings somewhat - SAMUEL could automate this.

[27]. Lin trains a robot by giving it advice in the form of a sequence of desired actions. Mahadevan and Connell initialize their reinforcement learner with a prespecified subsumption architecture, and Singh guides his reinforcement learner by giving it abstract actions to decompose.

One of the most similar approaches to ours is that of Towell and Shavlik [31]. They also couple rule-based input with a refinement method; however, their refinement method is neural networks. This multistrategy system converts rules into a network topology. The content of each rule is preserved; therefore, the transformation is syntactic. Our multistrategy system, on the other hand, focuses primarily on semantic transformations that use qualitative knowledge about movements in space to convert abstract goals into concrete actions. Ram and Santamaria [22] present a multistrategy refinement scheme for a parametric agent performing a navigation task. In contrast with our genetic algorithms based approach, their system acquires values for the defined parameters by using reinforcement learning. Different parameter value combinations are acquired, for distinct environment subclasses. The interesting feature of their system is the explicit identification of different environmental configurations using case-based reasoning. The identification of regularities in the environment occurs implicitly in the genetic algorithms approach. The deductive compilation scheme (but not the refinement) is in common with Mitchell *et* a/.'s derivation of a strategy for pushing objects in a tray using a qualitative theory of the process [18].

5.2 Differential game theory

Differential game theory is a branch of mathematical optimal control theory. It assumes that the behavior of the controlled system can be modeled as a system of ordinary differential equations (ODEs). The evasion problem considered in this paper is a typical example of a differential game. In particular, the problem is two-person zero-sum differential game with a *constant terminal time*. Both the pursuer and the evader move in a bounded rectangle in two dimensions. The evader has to avoid getting to within a certain distance of the pursuer for a certain length of time. In the minimax formulation of the problem, the optimal strategy of the evader is one that achieves its objective under the least favorable assumptions on the motion of the pursuer.

Differential games are formulated mathematically by specifying the motion equations of the pursuer and evader, the class of admissible controls for both systems (which identifies the way in which the pursuer and evader can change their motions), and the target or goal functional. A classic reference for this is [1]. The focus of work in differential game theory is to identify conditions under which optimal strategies for the evader can be derived. This assumes complete knowledge of the dynamics of the evader and pursuer, both of which are unavailable to us. The theory would be more useful to us if it had a qualitative counterpart which allowed us to determine the existence of solutions to the evader's problem from partial knowledge of the evader and pursuer's dynamics.

6 Discussion

We have presented a novel multistrategy learning method for operationalizing and refining high-level advice into low-level rules to be used by a reactive agent. Operationalization uses a portable SKB. An implementation of this method has been tested on two complex domains and a Nomad 200 robot.

We have learned the following lessons:

1. Our advice compiler can be effective on complex domains, and it will be important to identify the regions of greatest effectiveness for advice,

2. A portable SKB appears feasible, and

3. Coordinating a deductive learning strategy (advice compilation) with an inductive learning strategy (GA refinement) can lead to a substantial performance improvement over either method alone.

This success, however, depends on the how the advice biases the GA search. Future work will focus on identifying those characteristics of advice that bias this search favorably. We will also focus on further addressing our questions about performance using different advice and alternative domains (e.g., [28]).

Many other interesting directions are suggested by our experimental results. At present we do not consider the cost of incorporating advice. For larger scale problems and situations where advice is provided more frequently, the agent has to reason about the costs and benefits of compiling advice at a given point in time. Classical issues in trading off deliberation time for action time are relevant here.

Another issue for future study is the problem of operationalization theory incompleteness. For example, our SKB was sufficiently complete for both domains but had it not been, we would have been faced with the problem of supplementing this knowledge base with additional rules. We are considering an approach like that used in the DISCIPLE system [29] to elicit advice and learn rules for operationalizing the advice.

We have chosen the GA method for refinement because it was readily available to us. A comparison of a neural network and other approaches with our current GA approach, on the problems studied here, will provide valuable insights into the tradeoffs between different refinement strategies. We believe that multistrategy learning systems of the future must have a bank of operationalization and refinement methods at their disposal and have fast methods for selecting them. We have chosen a specific breakdown of effort between the advice compilation and refinement phases. How this coordinates with our choice of problem domains and refinement schemes is another question for future study.

Acknowledgements

We appreciate the helpful comments and suggestions provided by Bill Spears, Alan Schultz, Connie Ramsey, John Grefenstette, Alan Meyrowitz. We would especially like to thank John Greffenstette for generously allowing us to use the SAMUEL system upon which our system is built.

References

[1] Basar, T. and G. Olsder, Dynamic Noncooperative Game Theory. New York: Academic Press, 1982.

[2] Chapman, D., Vision, instruction and action. Ph.D. thesis. MIT, 1990.

[3] Clouse, J. and P. Utgoff, A teaching method for reinforcement learning. In *Proc. of the Ninth International Workshop on Machine Learning,* 1992.

[4] Erickson, M. and J. Zytkow, Utilizing experience for improving the tactical manager. In *Proc. of the Fifth International Workshop on Machine Learning,* 1988.

[5] Gordon, D. and D. Subramanian, A multistrategy learning scheme for assimilating advice in embedded agents. In *Proc. of the Second International Workshop on Multistrategy Learning,* 1993.

[6] Gordon, D. and D. Subramanian, Assimilating advice in embedded agents. Unpublished manuscript, 1993.

[7] Grefenstette, J., Ramsey, C, and A. Schultz, Learning sequential decision rules using simulation models and competition. *Machine Learning,* Volume 5, Number 4, 1990.

[8] Holland, J. *Adaptation in Natural and Artificial Systems.* University of Michigan Press: Ann Arbor, 1975.

[9] Kaelbling, L., Goals as parallel program specifications. In *Proc. of the Seventh National Conference on Artificial Intelligence,* 1988.

[10] Karr, C, Design of an adaptive fuzzy logic controller using a genetic algorithm. In *Proc. of the Ninth International Conference on Genetic Algorithms,* 1991.

[11] Laird, J., Hucka, M., Yager, E., and C. Tuck, Correcting and extending domain knowledge using outside guidance. In *Proc. of the Seventh International Conference on Machine Learning,* 1990.

[12] Lin, L., Programming robots using reinforcement learning and teaching. In *Proc. of the Ninth National Conference on Artificial Intelligence,* 1991.

[13] Maes, P. and R. Brooks, Learning to coordinate behaviors. In *Proc. of the Eighth National Conference on Artificial Intelligence,* 1990.

[14] Mahadevan, S. and J. Connell, Automatic programming of behavior-based robots using reinforcement learning. In *Proc. of the Ninth National Conference on Artificial Intelligence,* 1991.

[15] McCarthy, J., Mechanisation of thought processes. In *Proc. Symposium,* Volume 1, 1958.

[16] Michalski, R., Inferential learning theory as a basis for multistrategy task-adaptive learning. In *Proc. of the Eighth International Workshop on Multistrategy Learning,* 1991.

[17] Mitchell, T., Becoming increasingly reactive. In *Proc. of the Eighth National Conference on Artificial Intelligence,* 1990.

[18] Mitchell, T., Mason, M., and A. Christiansen, Toward a learning robot. CMU Technical Report, 1987.

[19] Mostow, D. J., Machine transformation of advice into a heuristic search procedure. In R. Michalski, J. Carbonell, and T. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach* (Vol. 1). Tioga Publishing Co., Palo Alto, CA., 1983.

[20] Nilsson, N., *Principles of Artificial Intelligence.* Tioga: Palo Alto, 1980.

[21] Odetayo, M. and D. McGregor, Genetic algorithm for inducing control rules for a dynamic system. In *Proc. of the Third International Conference on Genetic Algorithms,* 1989.

[22] Ram, A. and J.C. Santamaria, A multistrategy case-based and reinforcement learning approach to self-improving reactive control systems for autonomous robotic navigation. In *Proc. of the Second International Workshop on MultiStrategy Learning,* 1993.

[23] Schoppers, M., Universal plans for reactive robots in unpredictable environments. In *Proc. of the Sixth National Conference on Artificial Intelligence,* 1987.

[24] Schultz, A. and J. Grefenstette, Using a genetic algorithm to learn behaviors for autonomous vehicles. In *Proc. of the Navigation and Control Conference,* 1992.

[25] Schultz, A. and J. Grefenstette, Improving tactical plans with genetic algorithms. In *Proc. of the IEEE Conference on Tools for AI,* 1990.

[26] Segre, A., *Machine Learning of Robot Assembly Plans.* Kluwer: Boston, 1988.

[27] Singh, S., Scaling reinforcement learning algorithms by learning variable temporal resolution models. In *Proc. of the Ninth International Workshop on Machine Learning,* 1992.

[28] Subramanian, D. and S. Hunter, Some preliminary studies in agent design in simulated environments. Unpublished manuscript, 1993.

[29] Tecuci, G. Cooperation in knowledge base refinement. In *Proc. of the Ninth International Workshop on Machine Learning,* 1992.

[30] Tesauro, G., Temporal difference learning of backgammon strategy. In *Proc. of the Ninth International Workshop on Machine Learning,* 1992.

[31] Towell, G. and J. Shavlik, Refining symbolic knowledge using neural networks. In *Proc. of the Eighth Workshop on Machine Learning,* 1991.

[32] Whitehead, S. and D. Ballard, Learning to perceive and act by trial and error. *Machine Learning,* Volume 7, Number 1, 1991.

MULTISTRATEGY LEARNING IN REACTIVE CONTROL SYSTEMS FOR AUTONOMOUS ROBOTIC NAVIGATION

Ashwin Ram and Juan Carlos Santamarfa
College of Computing, Georgia Institute of Technology,
Atlanta, Georgia 30332-0280, U.S.A.

Keywords: Robot navigation, reactive control, case-based reasoning, reinforcement learning, adaptive control

Edited by: Gheorghe Tecuci

Received: October 15, 1993 **Revised:** November 6, 1993 **Accepted:** November 14, 1993

This paper presents a self-improving reactive control system for autonomous robotic navigation. The navigation module uses a schema-based reactive control system to perform the navigation task. The learning module combines case-based reasoning and reinforcement learning to continuously tune the navigation system through experience. The case-based reasoning component perceives and characterizes the system's environment, retrieves an appropriate case, and uses the recommendations of the case to tune the parameters of the reactive control system. The reinforcement learning component refines the content of the cases based on the current experience. Together, the learning components perform on-line adaptation, resulting in improved performance as *the reactive control system tunes itself to the environment, as well as on-line case learning, resulting in an improved library of cases that capture environmental regularities necessary to perform on-line adaptation. The system is extensively evaluated through simulation studies using several performance metrics and system configurations.*

1 Introduction

Autonomous robotic navigation is defined as the task of finding a path along which a robot can move safely from a source point to a destination point in an obstacle-ridden terrain, and executing the actions to carry out the movement in a real or simulated world. Several methods have been proposed for this task, ranging from high-level planning methods to reactive control methods.

High-level planning methods use extensive world knowledge and inferences about the environment they interact with (Fikes, Hart & Nilsson, 1972; Sacerdoti, 1975; Georgeff, 1987; Maes, 1990). Knowledge about available actions and their consequences is used to formulate a detailed plan before the actions are actually executed in the world. Such systems can successfully perform the path-finding required by the navigation task, but only if an accurate and complete representation of the world is available to the system. Considerable high-level knowledge is also needed to learn from planning experiences (e.g., Mostow & Bhatnagar, 1987; Minton, 1988; Segre, 1988; Hammond, 1989). Such a representation is usually not available in real-world environments, which are complex and dynamic in nature. To build the necessary representations, a fast and accurate perception process is required to reliably map sensory inputs to high-level representations of the world. A second problem with high-level planning is the large amount of processing time required, resulting in significant slowdown and the inability to respond immediately to unexpected situations.

Situated or reactive control methods have been proposed as an alternative to high-level planning methods (Brooks, 1986; Kaelbling, 1986; Payton, 1986; Arkin, 1989). In these methods, no planning is performed; instead, a simple sensory representation of the environment is used to select the next action that should be performed. Actions are represented as simple behaviors, which can be selected and executed rapidly, often in real-time. These methods can cope with unknown and dynamic environmental configurations, but only those that lie within the scope of predetermined behaviors. Furthermore, such methods cannot modify or improve their behaviors through experience, since they do not have any predictive capability that could account for future consequences of their actions, nor a higher-level formalism in which to represent and reason about the knowledge necessary for such analysis.

We propose a self-improving navigation system that uses reactive control for fast performance, augmented with multistrategy learning methods that allow the system to adapt to novel environments and to learn from its experiences. The system autonomously and progressively constructs representational structures that aid the navigation task by supplying the predictive capability that standard reactive systems lack. The representations are constructed using a hybrid case-based and reinforcement learning method without extensive high-level reasoning. The system is very robust and can perform successfully in (and learn from) novel environments, yet it compares favorably with traditional reactive methods in terms of speed and performance. A further advantage of the method is that the system designers do not need to foresee and represent all the possibilities that might occur since the system develops its own "understanding" of the world and its actions. Through experience, the system is able to adapt to, and perform well in, a wide range of environments without any user intervention or supervisory input. This is a primary characteristic that autonomous agents must have to interact with real-world environments.

This paper is organized as follows. Section 2 presents a technical description of the system, including the schema-based reactive control component, the case-based and reinforcement learning methods, and the system-environment model representations, and places it in the context of related work in the area. Section 3 presents several experiments that evaluate the system. The results shown provide empirical validation of our approach. Section 4 concludes with a discussion of the lessons learned from this research and suggests directions for future research.

2 Technical Details

2.1 System Description

The Self-Improving Navigation System (SINS) consists of a navigation module, which uses schema-based reactive control methods, and an on-line adaptation and learning module, which uses case-based reasoning and reinforcement learning methods. The navigation module is responsible for moving the robot through the environment from the starting location to the desired goal location while avoiding obstacles along the way. The adaptation and learning module has two responsibilities. The adaptation sub-module performs on-line adaptation of the reactive control parameters being used in the navigation module to get the best performance. The adaptation is based on recommendations from cases that capture and model the interaction of the system with its environment. With such a model, SINS is able to predict future consequences of its actions and act accordingly. The learning sub-module monitors the progress of the system and incrementally modifies the case representations through experience. Figure 1 shows the SINS functional architecture.

The main objective of the learning module is to construct a model of the continuous sensorimotor interaction of the system with its environment, that is, a mapping from sensory inputs to appropriate behavioral (schema) parameters used for reactive control. This model allows the adaptation module to control the behavior of the navigation module by selecting and adapting schema parameters in different environments. To learn a mapping in this context is to discover environment configurations that are relevant to the navigation task and corresponding schema parameters that improve the navigational performance of the system. The learning method is unsupervised and uses a reward that depends on the similarity of the observed mapping in the current environment to the mapping represented in the model combined with a more traditional punishment signal. This causes the system to converge towards those mappings that are consistent and beneficial over a set of experiences.

The representations used by SINS to model its interaction with the environment are initially under-constrained and generic; they contain very

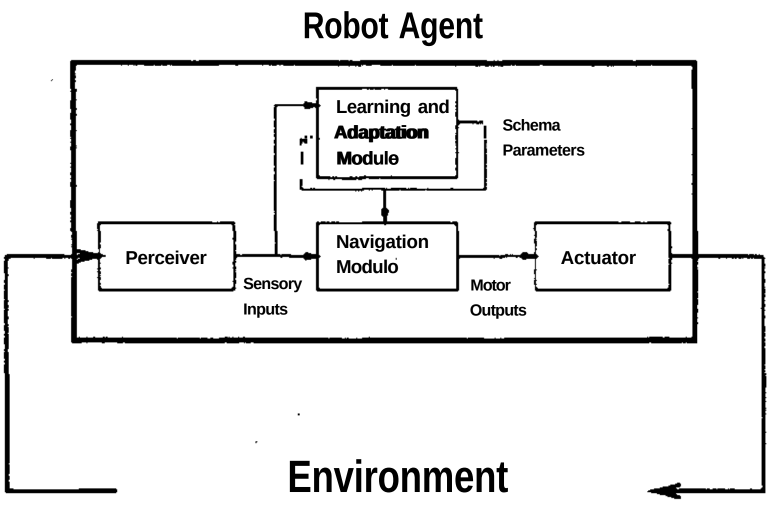


Figure 1: System architecture

little useful information for the navigation task. As the system interacts with the environment, the learning module gradually modifies the content of the representations until they become useful and provide reliable information for adapting the navigation system to the particular environment at hand.

The learning and navigation modules function in an integrated manner. The learning module is always trying to find a better model of the interaction of the system with its environment so that it can tune the navigation module to perform its function better. The navigation module provides feedback to the learning module so it can build a better model of this interaction. The behavior of the system is then the result of an equilibrium point established by the learning module which is trying to refine the model and the environment which is complex and dynamic in nature. This equilibrium may shift and need to be re-established if the environment changes drastically; however, the model is generic enough at any point to be able to deal with a very wide range of environments.

We now present the reactive module, the representations used by the system, and the methods used by the learning module in more detail.

2.2 The Navigation Module

The navigation module, which uses schema-based reactive control methods, is based on the AuRA architecture (Arkin, 1989). The module consists of a set of motor schemas that represent the individual motor behaviors available to the system. Each schema reacts to sensory information from the environment, and produces a velocity vector representing the direction and speed at which the robot is to move given current environmental conditions. The velocity vectors produced by all the schemas are then combined to produce a potential field that directs the actual movement of the robot. Simple behaviors, such as wandering, obstacle avoidance, and goal following, can combine to produce complex emergent behaviors in a particular environment. Different emergent behaviors can be obtained by modifying the simple behaviors. This allows the system to interact successfully in different environmental configurations requiring different navigational "strategies" (Ram, Arkin, Moorman, *h* Clark, 1992).

A detailed description of schema-based reactive control methods can be found in Arkin (1989). In this research, we used three motor schemas: AVOID-STATIC-OBSTACLE, MOVE-TO-GOAL, and NOISE. AVOID-STATIC-OBSTACLE directs the system to move itself away from detected obstacles. MOVE-TO-GOAL schema directs the system to move towards a particular point in the terrain. The NOISE schema makes the system move in a random direction; it is used to escape from local minima and, in conjunction with other schemas, to produce wandering behaviors. Each motor schema has a set of parameters that control the potential field generated by the motor schema. In this research, we used the following parameters: **Obstacle-Gain,** associated with AVOID-STATIC-OBSTACLE, determines the magnitude of the repulsive potential field generated by the obstacles perceived by the system; **Goal-Gain,** associated with MOVE-TO-GOAL, determines the magnitude of the attractive potential field generated by the goal; **Noise-Gain,** associated with NOISE, determines the magnitude of the noise; and **Noise-Persistence,** also associated with NOISE, determines the duration for which a noise value is allowed to persist.

Different combinations of schema parameters cause different .behaviors to be exhibited by the system (see figure 2). Traditionally, parameters are fixed and determined ahead of time by the system designer. However, on-line selection and modification of the appropriate parameters based on the current environment can enhance navigational performance, as in the ACBARR system (Ram, Arkin, Moorman & Clark, 1992). SINS adopts this approach by allowing schema parameters to be modified dynamically. However, in ACBARR, schema modification information is supplied by the designer using hand-coded cases. Our system, in contrast, can learn and modify its own cases through experience. The representation of our cases is also considerably different and is designed to support reinforcement learning.

2.3 The System-Environment Model Representation

The navigation module in SINS can be adapted to exhibit many different behaviors. SINS improves its performance by learning how and when to tune the navigation module. In this way, the system can use the appropriate behavior in each environmental configuration encountered. The learning module, therefore, must learn about and discriminate between different environments, and associate with each the appropriate adaptations to be performed on the motor schemas. This requires a representational scheme to model, not just the environment, but the interaction between the system and the environment. However, to ensure that the system does not get bogged down in extensive high-level reasoning, the knowledge represented in the model must be based on perceptual and motor information easily available at the reactive level.

SINS uses a model consisting of associations between sensory inputs and schema parameters values. Each set of associations is represented as a case. Sensory inputs provide information about the configuration of the environment, and schema parameter information specifies how to adapt the navigation module in the environments to which the case is applicable. Each type of information is represented as a vector of analog values. Each analog value corresponds to an estimate of a quantitative variable (a sensory input or a schema parameter) over a window of time. A vector represents the trend or recent history of such estimates of a variable. A case models an association between sensory inputs and schema parameters by grouping their respective vectors together. Figure 3 shows an example of this representation.

This representation has three essential properties. First, the representation is capable of capturing a wide range of possible associations between.sensory inputs and schema parameters. Second, it permits continuous progressive refinement of the associations. Finally, the representation captures trends or patterns of input and output values over time. This allows the system to detect patterns over larger time windows rather than having to make a decision based only on instantaneous values of perceptual inputs.

In this research, we used four input vectors to characterize the environmental and discriminate among different environment configurations: **Obstacle-Density** provides a measure of the occupied areas that impede navigation; **Absolute-Motion** measures the activity of the system; **Relative-Motion** represents the change in motion activity; and **Motion-Towards-Goal** specifies how much progress the system has actually made towards the goal. These input vectors are constantly updated with the information received from the sensors.

We also used four output vectors to represent the schema parameter values used to adapt the navigation module, one for each of the schema parameters **(Obstacle-Gain, Goal-Gain, Noise-**

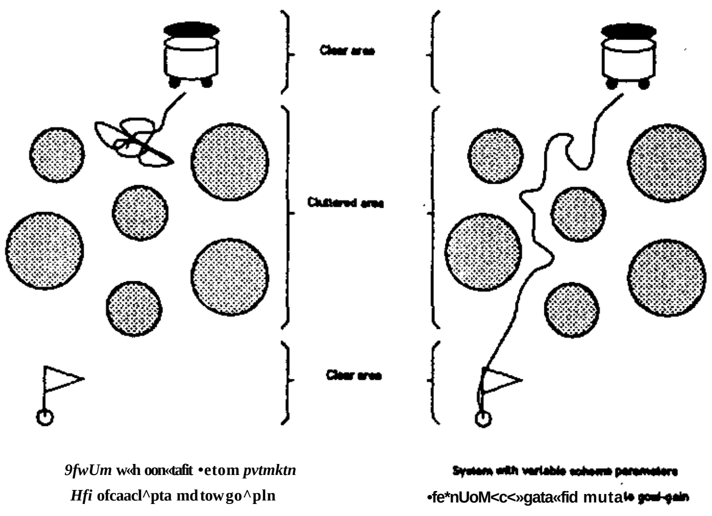

Clear area
Cluttered area
Clear area
9fwUm w«h oon«tafit •etom pvtmktn
Hfi ofcaacl^pta mdtowgo^pln
System with variable schema parameters
•fe*nUoM<c<»gata«fid muta goal-gain

Figure 2: Typical navigational behaviors of different tunings of the reactive control module. The figure on the left shows the non-learning system with high obstacle avoidance and low goal attraction. On the right, the learning system has lowered obstacle avoidance and increased goal attraction, allowing it to "squeeze" through the obstacles and then take a relatively direct path to the goal.

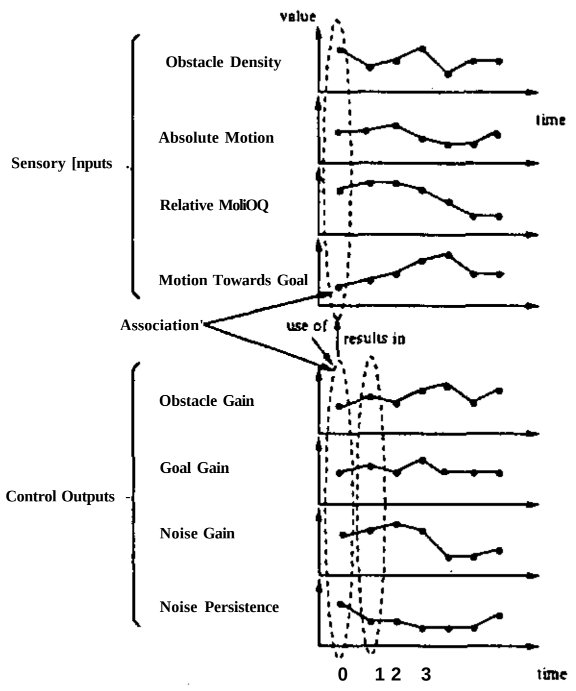


Figure 3: Sample representations showing the time history of analog values representing sensory inputs and control outputs. Associations between sensory inputs and control outputs are arranged vertically, and the sequence of associations over time is arranged horizontally. Each case in the system is represented in this manner, as is the current on-going navigational experience of the system.

Gain, and **Noise-Persistence)** discussed earlier. The values are set periodically according to the recommendations of the case that best matches the current environment. The new values remain constant for a "control interval" until the next setting period.

The choice of input and output vectors was based on the complexity of their calculation and their relevance to the navigation task. The input vectors were chosen to represent environment configurations in a generic manner but taking into account the processing required to produce those vectors (e.g., obstacle density is more generic than obstacle position, and can be obtained easily from the robot's ultrasonic sensors). The output vectors were chosen to represent directly the actions that the learning module uses to tune the navigation module, that is, the schema parameter values themselves.

2.4 The On-Line Adaptation And Learning Module

This module creates, maintains, and applies the case representations used for on-line adaptation of the reactive module. The objective of the learning method is to detect and discriminate among different environment configurations, and to identify the appropriate schema parameter values to be used by the navigation module, in a dynamic and an on-line manner. This means that, as the system is navigating, the learning module is perceiving the environment, detecting an environment configuration, and modifying the schema parameters of the navigation module accordingly, while simultaneously updating its own cases to reflect the observed results of the system's actions in various situations.

The method is based on a combination of ideas from case-based reasoning and learning, which deals with the issue of using past experiences to deal with and learn from novel situations (e.g., see Hammond, 1989; Kolodner, in press), and from reinforcement learning, which deals with the issue of updating the content of system's knowledge based on feedback from the environment (e.g., see Sutton, 1992). However, in traditional case-based planning systems (e.g., Hammond, 1989) learning and adaptation requires a detailed model of the domain. This is exactly what reactive planning systems are trying to avoid. Earlier attempts to combine reactive control with classical planning systems (e.g., Chien, Gervasio, & DeJong, 1991) or explanation-based learning systems (e.g., Mitchell, 1990) also relied on deep reasoning and were typically too slow for the fast, reflexive behavior required in reactive control systems. Unlike these approaches, our method does not fall back on slow non-reactive techniques for improving reactive control.

To effectively improve the performance of the navigation task., the learning module must find a consistent mapping from environment configurations to control parameters. The learning module captures this mapping in the learned cases, each case representing a portion of the mapping localized in a specific environment configuration. The set of cases represents the system's model of its interactions with the environment, which is adapted through experience using the case-based and reinforcement learning methods. The case-based method selects the case best suited for a particular environment configuration. The reinforcement learning method updates the content of a case to reflect the current experience, such that those aspects of the mapping that are consistent over time tend to be reinforced. Since the navigation module implicitly provides the bias to move to the goal while avoiding obstacles, mappings that are consistently observed are those that tend to produce this behavior. As the system gains experience, therefore, it improves its own performance at the navigation task. Additionally, the reinforcement learning method uses a punishment signal to reject mappings that are not beneficial to the system.

Each case represents an observed regularity between a particular environmental configuration and the effects of different actions, and prescribes the values of the schema parameters that are appropriate (as far as the system knows based on its previous experience) for that environment. The learning and adaptation module performs the following tasks in a cyclic manner: (1) *perceive* and represent the current environment; (2) *retrieve* a case whose input vector represents an environment most similar to the current environment; (3) *adapt* the schema parameter values in use by the reactive control module by installing the values recommended by the output vectors of the case; and (4) *learn* new associations and/or adapt

```
do
{
  /* PERCEIVE: Update input vectors */
  current.environment = perceiveQ;

  if ( end of control interval )
    then {

      /* LEARN: Decide whether to reinforce or explore */
      if ( outcome was good )
        then {
          reinforcejschemas (previous-case,
currentjenvironment);
          /* LEARN: Decide if case should be extended */
          if ( prediction is good and at end of sequence )
            then
              extend_case (previous_case);
        }
        else
          explore_schemas (previousjcase);

      /• RETRIEVE: Retrieve best case */
      best_case = retrieve_bestjcase(current-environment);

      /* LEARN: Decide if a new case should be created */
      if ( bestjcase is not a good match )
        then
          bestjcase = create-case (currentjenvironment);

      /* ADAPT: Modify current set of schema parameters
*/
      adapt_schemas (best_case);

      /* Wait until next cycle */
      previousjcase = best.case;
    }

  /* Move robot using the current set of schema parameters */
  execute();
}
while (not (goal reached or maximum number of steps
exceeded))
```

Table 1: SINS algorithm

existing associations represented in the case to reflect any new information gained through the use of the case in the new situation to enhance the reliability of its predictions.

The overall algorithm is shown in table 1.

The **perceive** function constructs and maintains a representation of the current environmental situation by reading the robot's sensors and updating the input and output vectors accordingly. This results in a set of $J = 4$ input vectors E_{input_j}, one for each sensory input j, and $K = 4$ output vectors $E_{outputJc}$, one for each output vector k as described earlier. Then, every control interval T, the learning and adaptation module performs two main functions: It adapts the schema parameters currently in use by the reactive control module so that it performs better in the new environment, and it learns useful sequences of associations between environment situations and schema parameters.

Schema parameters are adapted using the **retrieve_best_case** and **adapt_schemas** functions. In the **retrieve_best_case** function, the case most similar to the current environment situation is selected by matching the environment's input and output vectors $E_{input>}$, $E_{outputfc}$ from the **perceive** step against the corresponding input and output vectors $C''_{input j}$, $C?_{utputfc}$ of the cases C^n in the system's memory (see figure 4). The best matching case C''^{be3t} and the position of the best match p_{best} are handed to the **adapt_schemas** function, which modifies the schema parameter values currently in use based on the recommendations $C''^{e}_{o u,pi tj.}(p_{bes}, + 1)$ from the output vectors of the case.

Finally, the learning and adaptation module decides how to utilize information from the current experience with the best case in order to improve its case library. The system learns in three different ways: by improving the content of the case that was just used in order to make it more reliable, by creating a new case whenever the best case retrieved is not good enough, or by extending the length of the case in order to build up longer sequences of associations. The contents of a case are improved by the **reinforce_schemas** function, which reinforces the suggestions of the case if these suggestions led to a favorable outcome over the last control interval, and by the **explore_schemas** function, which uses random exploration to try out other schema parameter values if the suggested set of values did not prove useful. The outcome is evaluated by monitoring the behavior of the robot over the last control interval; collisions are undesirable, as is lack of movement.

One difference between our methods and traditional reinforcement learning is that SINS is trying to maximize consistency in "useful" behaviors as determined by a reward signal, whereas traditional reinforcement learning tries to maximize the expected utility the system is going to receive in the future as determined by the reward signal (cf. Watkins, 1989; Whitehead *k* Ballard, 1990). In schema-based reactive control navigation, it is inherently a good idea to modify schema parame-

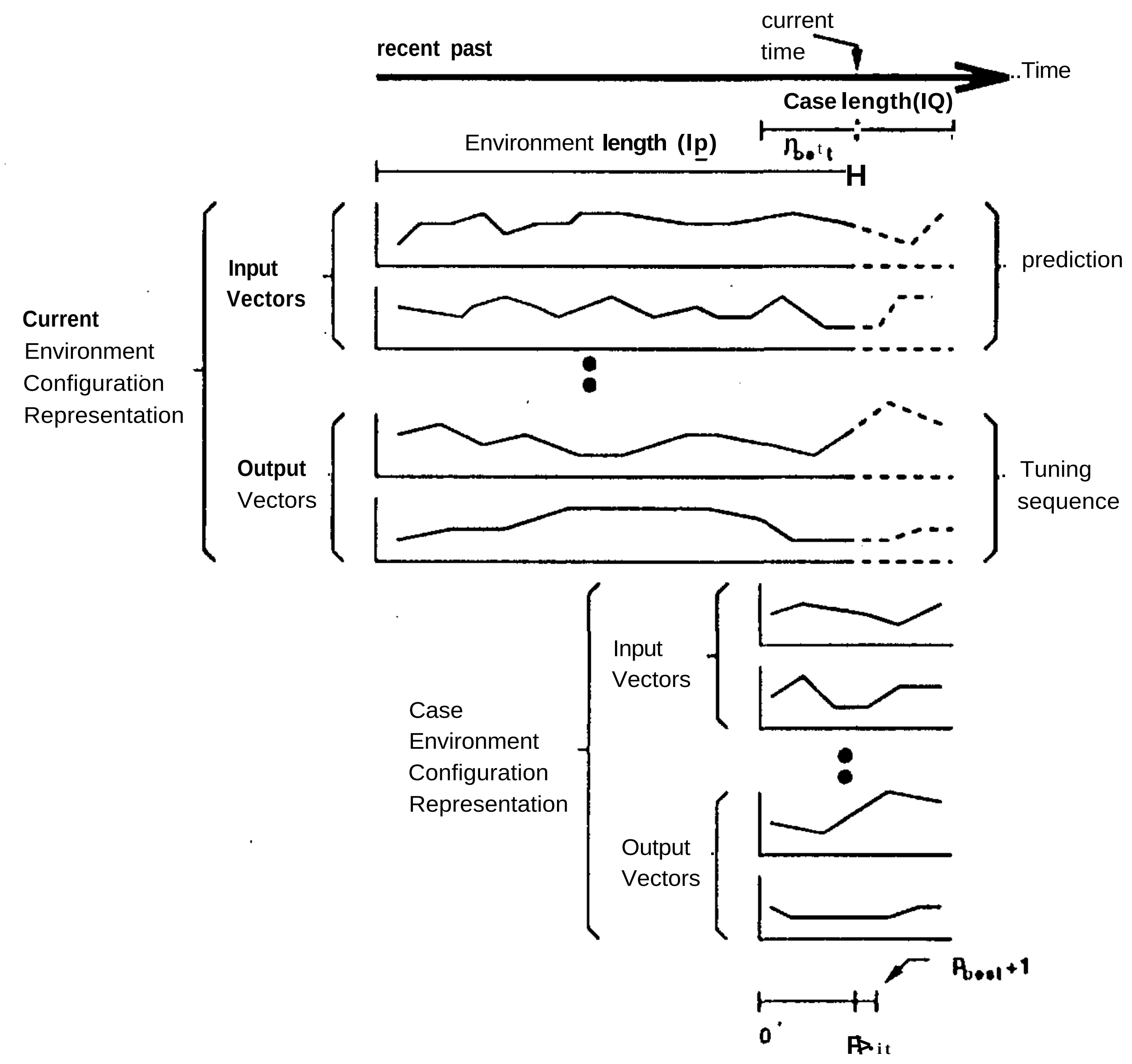


Figure 4: Schematic representation of the match process. Each graph in the case (below) is matched against the corresponding graph in the current environment (above) to determine the best match, after which the remaining part of the case is used to guide navigation (shown as dashed lines).

ters in an on-line fashion; however, not all modifications are equally good since some may cause the robot to collide with obstacles or not to move at all. SINS uses the reward signal to decide whether to reinforce a behavior or to explore alternative behaviors; reinforcement, when chosen, is used to reinforce behaviors that are consistent across experiences. Thus, in addition to external outcome, consistency is used as an "internal" reward signal for the reinforcement learning method.

Furthermore, traditional reinforcement learning assumes that the outcomes of the system's actions are known; it learns which actions to execute to maximize a reward. In SINS, the outcomes of what corresponds to "actions" (the adaptations to be performed on the navigation module) are not known; part of the learning task is to discover the sequences of environmental situations are likely to result from a given sequence of adaptations and, in turn, which adaptations are appropriate in different situations. Thus, SINS is learning a model of its sensorimotor interaction with the environment (represented as a set of cases) at the same time as it is learning to improve its navigational performance through on-line adaptation of its reactive control schemas.

In addition to modifying its cases, SINS can also extend its cases and learn hew cases. In order to decide which kind of learning to perform in a given situation, SINS uses a relative similarity criterion to judge the appropriateness of the best matching case in the current situation. This determination is based on statistical information about the quality of match in prior applications of the case as compared to the quality of match in the current situation. If the best matching case is not as similar to the current environment situation as it has been in previous situations, the case is probably inappropriate for this situation; thus, it is better to learn a new case to represent what is probably a new class of situations. If this occurs, SINS uses the **create_new_case** function to create a new case based on the current experience and add it to the case library. To determine whether to create a new case, SINS compares the current match with the mean match plus the standard deviation of the matches over the past utilizations of the case. This ensures that new sequences of associations are created only when the available sequences of associations already captured in the case library do not fit the current environment.

The third kind of learning is carried out by the **extend_case_size** function, which extends the length of a case whenever the best case makes an accurate prediction of the next environment situation and there are no more associations in the sequence. This allows the system to increase the length of the sequence of associations only when it is confident that the sequence of the case accurately predicts how the environment changes if the suggested schema parameters are used. To estimate this confidence, the predicted values are matched with the actual environmental parameters that result; if this match-is better than the mean match, the case is extended. Intuitively (as before), if the case predicts the current situation better than it predicted the previous situations that it was used in, it is likely that the current situation involves the very regularities that the case is beginning to capture; thus, it is worthwhile extending the case so as to incorporate the current situation. Alternatively, if the match is not quite as good, the case should not be modified because doing so would take it away from the regularities it has been converging towards.

Since the reinforcement formulae are based on a relative similarity criterion, the overall effect of the learning process is to cause the cases to converge on stable associations between environment configurations and schema parameters. Stable associations represent regularities in the world that have been identified by the system through its experience, and provide the predictive power necessary to navigate in future situations. The assumption behind this method is that the interaction between the system and the environment can be characterized by a finite set of causal patterns or associations between the sensory inputs and the actions performed by the system. The method allows the system to learn only the causal patterns that the reward utility identifies as useful and to use them to modify its actions by updating its schema parameters as appropriate. Useful causal patterns are those that do not cause robot collisions or do not cause the robot to stop.

Genetic algorithms may also be used to modify schema parameters in a given environment (Pearce, Arkin, *k* Ram, 1992). However, while this approach is useful in the initial design of the navigation system, it cannot change schema parameters during navigation when the system faces environments that are significantly different from the environments used in the training phase of the genetic algorithm. Another approach to self-organizing adaptive control is that of Verschure, Kröse, and Pfeifer (1992), in which a neural network is used to learn how to associate conditional stimulus to unconditional responses. Although their system and ours are both self-improving navigation systems, there is a fundamental difference on how the performance of the navigation task is improved. Their system improves its navigation performance by learning how to incorporate new input data (i.e., conditional stimuli) into an already working navigation system, while SINS improves its navigation performance by learning how to adapt the system (i.e., the navigation module) itself. Our system does not rely on new sensory input, but on patterns or regularities detected in perceived environment. Our learning methods are also similar to Sutton (1990), whose system uses a trial-and-error reinforcement learning strategy to develop a world model and to plan optimal routes using the evolving world model. Unlike this system, however, SINS does not need to be trained on the same world many times, nor are the results of its learning specific to a particular world, initial location, or destination location.

We now present a detailed description of and the mathematical formulas used in the perception, matching, adaptation, and learning tasks.

2.4.1 Perception

The objective of the **perceive** function is to generate an accurate description of the current environment situation. It performs this task by shifting the previous values in each input and output vector one position back in time[1] and then calculating the current values for each input vector $E_{inputj}.(0), j = 1,\ldots,J$ and output vector $E_{ou,putfc}(0)$,A: = *l,...,K,* where 0 is the current position in time. The current values for the input vectors are based on the robot's sensors, and the current values for the output vectors are just the respective values of the schema parameters suggested in the previous control interval. The vectors are updated at the end of each control interval of time *T*.

To update the input vectors, the system monitors the values of the robot's sensors **Sensory** corresponding to each input vector $E_{input>}$. The sensors are monitored at each time step over the past control interval; these sensor readings are then averaged to yield the new value for the corresponding input vectors. Thus, the input vectors in the environment representation are updated using the following formula:[2]

$$E_{inputy}(i) = \begin{cases} E_{inputj}(i-1) & \text{if } i > 0 \\ \dfrac{\wedge = ^{\circ}_{=} \_{r}\ \text{Sensor}\wedge)}{T} & \text{if } i = 0 \end{cases}$$

where Sensorj(ra) is the sensory input that corresponds to the input vector $E_{inpu,j}$ (sensed obstacles for **Obtacle-Density,** distance traveled for **Absolute-Motion,** relative position for **Relative-Motion,** and normal relative position for **Motion-Towards-Goal),** and *t* ranges over each robot step since the last control interval.

2.4.2 Retrieval and Matching

The function **retrieve_best_case** is responsible for selecting a case from the case library that best matches the current environment situation. The case similarity metric is based on the mean squared difference between each of the vector values of the case over a trending window, and the vector values of the environment. The best match window is calculated using a reverse sweep over the time axis *p* similar to a convolution process to find the relative position that matches best. Each case C^n in the case library is matched against the current environment using exhaustive search, which returns the best matching case $C^{be''}$ along with the relative position p_{be8}, of the match (see figure 4). After retrieving the best case, the mean and variance of the case's statistical match history are updated; these will be used later to calculate the relative similarity criterion during learning.

The case similarity metric *SM* of a case C at position *p* relative to the environment E is a value that indicates the similarity between the sequence of associations encoded in the case to the sequence of associations in the current environment situation starting at position *p.* The lower the value

[1] This is implemented using a circular buffer which does not requires copying each of the values from one cell to the next.

[2]Note that *i* counts back in time (i.e., $t' = 0$ is the current time and $t' > 0$ is the recent past.)

of the case similarity metric, the more similar the sequences of associations. The case similarity metric formula calculates a weighted sum of the squared difference between the corresponding vectors of the case and the environment. For the *SM* to be valid, *p* must lie between 0 and l_C.[3]

$$SM(E,C,p) = \sum_{j=1}^{J} w_j \sum_{t=0}^{\min(p,l_E)} \frac{(E_{input_j}(t) - C_{input_j}(p-t))^2}{p+1} + \sum_{k=1}^{K} w_k \sum_{t=0}^{\min(p,l_E)} \frac{(E_{output_k}(t) - C_{output_k}(p-t))^2}{p+1}$$

The best case is obtained by matching each case C^n in the case library at all the positions *p* and selecting the pair (n_{best}, p_{best}) that yields the lowest *SM*. Formally, this can be expressed as:

$$(n_{best}, p_{best}) \mid \min(SM(E,C^n,p)), \; \forall n, \; 0 \le p \le l_{C^n}$$

Each case C maintains a statistical record of the similarity metrics it has produced in the past, which is updated every time the case is retrieved as the best case. The mean $(C_{SM_{mean}})$ and variance $(C_{SM_{var}})$ of the case similarity metric as well as the number of times the case has been used (C_{used}) are updated using standard formulae in descriptive statistics:

$$\begin{aligned}
\text{new } C_{SM_{mean}} &= \frac{C_{used} C_{SM_{mean}} + SM}{C_{used}+1} \\
\text{new } C_{SM_{var}} &= \frac{C_{used}-1}{C_{used}} C_{SM_{var}} \\
&\quad + (\text{new } C_{SM_{mean}} - C_{SM_{mean}})^2 \\
&\quad + \frac{(SM - \text{new } C_{SM_{mean}})^2}{C_{used}} \\
\text{new } C_{used} &= C_{used} + 1
\end{aligned}$$

2.4.3 Adaptation

The best matching case $C^{n_{best}}$ is used to adapt the schema parameter values currently in use by the reactive control module. The values of output vectors for the next association $C^{n_{best}}_{output_k}$ after position p_{best} are used to determine the new set of schema parameters values $\textbf{Parameter}_k$ until the next control interval. Since learning tends to reinforce those associations that are consistently observed over several experiences, the new set of schema parameters can be expected to cause the robot to move safely and the next environment configuration that results from the movement can be expected to be the one predicted by the association. Since output vectors directly represent schema parameters, adaptation is a straightforward operation:

$$\begin{aligned}
\textbf{Parameter}_k &= C^{n_{best}}_{output_k}(p_{best}+1), \\
&\quad \forall k = 1, \ldots, K
\end{aligned}$$

2.4.4 Learning

In addition to perceiving the environment, retrieving the best matching case, and adapting the schema parameters being used by the reactive control module, SINS must also learn by updating its case library based on its current experience. Three types of learning are possible: modification of the associations contained in a case, creation of a new case based on the current experience, and extension of the size of a case to yield associations over larger time windows. Modification of case contents, in turn, can be of two types: reinforcement of the associations contained in the case based on a successful experience, and exploration of alternative associations based on an unsuccessful experience.

SINS decides which kind of learning to perform using a relative similarity criterion which determines the quality of the best match. The match value of the best case, based on the case similarity metric, is compared with the match values of the case in previous situations in which it was used. If the current match is worse than the mean match value by more than a standard deviation, the case (although still the best match) is considered to be too different from the current situation, since it has been a better match to other situations in the past. In this case, the `create_case` function is invoked to create a new case containing a sequence of associations formed by copying the values of the sequence of associations in the current environmental representation:

$$\begin{aligned}
C^{n_{best}}_{input_j}(0) &= E^{n_{best}}_{input_j}(0), \quad \forall j = 1, \ldots, J \\
C^{n_{best}}_{output_k}(0) &= E^{n_{best}}_{output_k}(0), \quad \forall k = 1, \ldots, K
\end{aligned}$$

If, on the other hand, the best case matches the current situation well, it is likely that the current situation is representative of the class of situations that that case is beginning to converge

[3]We used $w_j = w_k = 1.0$ (i.e., input and output vectors all contribute equally in the similarity metric.)

towards. If the case provides good recommendations for action, its recommendations should be reinforced; if not, its recommendations should be modified. In SINS, collisions with obstacles and lack of movement are undesirable by definition of the navigation task. A set of schema parameters is considered beneficial if using it does not lead to an undesirable outcome. The objective of the learning functions is to improve the accuracy of prediction of the system's cases and, in turn, to discover those schema parameter values that result in environmental situations that are beneficial for the robot.

If the best case recommends a set of schema parameters that are not beneficial to the robot, the **explore_schemas** function is used to modify the case such that it suggests a different set of schema parameters in similar circumstances in the future. Specifically, the output vectors *C^'^ik(pbe!lt* + 1) associated with the environment situation following the best match position p_{beat} are modified in a random manner since the current values are not useful to the system. The small random changes allow the system to explore the space of possible schema parameters in a controlled manner. These changes are defined by the following formula:

$$p = \min(1, a \text{ collisions} + \beta \frac{\textbf{velocity}}{\textbf{max\_velocity}})$$

c^Ow +1) =

+ p random(min C^,k,maxC^'St$_k$),
VA; = *1,...,K*

where p is a "reject" value that determines the extent to which the current recommendations should be taken into account when determining the modified values. A value of p = 0 specifies that the value of the output vector should be left unchanged, and a value of p = 1 specifies that the value of output vector should be replaced completely by a new random value in the allowable range. In any given learning cycle, the value of p depends on a and /?, which represent the importance of avoiding collisions and moving, respectively. In this paper, we used a = 0.5 and /9 = 1.0.

If, on the other hand, the schema parameters suggested by the best matching case produce desirable results, the **reinforce_schemas** function is invoked. This function updates the case by making it more like the current environmental situation, so as to produce the same recommendations in similar situations in the future. This reinforcement is done using the following formulae:

Vi = 0,...,p$_{best}$

$$\mathbf{C}^{n_{best}}_{input_j}(i) = \frac{\lambda}{i+1}(\mathbf{E}_{input_j}(p_{best} - i) - \mathbf{C}^{n_{best}}_{input_j}(i)),\quad \forall j = 1, ..., J$$

Vi = 0,...,jw

$\mathbf{C}^{n_{best}}_{output_k}(i)$ = $\frac{\lambda}{t'+T}$(V^output/t (Pbest − $\dot{i}$)−
/-i"besl (j\\
^outputfc^jj)
VA; = *1,...,K*

where A determines the learning rate (0.9 in the cuurent version of SINS).

Finally, the **extend_case** function extends the sequence of associations contained in a case. The decision to extend a case is also based on a statistical relative similarity criterion. If the case's predictions C'i'b"inpaV(j»be8, + 1) are similar to the resulting environment situation within a standard deviation from the mean predictive similarity, and the case does not have more associations in the sequence (that is, it cannot provide a next set of schema parameters), then the case is extended by duplicating the last association of the case:

$$\mathbf{C}^{n_{best}}_{input_j}(p_{best} + 2) = \mathbf{C}^{n_{best}}_{input_j}(p_{b...} + 1),\quad \forall j = 1, \ldots, J$$

^-"outputjj(Pbeat "T" 2) = $\mathbf{C}^{n_{best}}_{output_k}(p_{best} \frac{1}{T} 1)$, $\forall k = 1, \ldots, K$

The net result of these learning procedures is to cause the cases in the system's case library to converge towards regularities in the system's interactions with its environment. The system learns useful sequences of schema parameters for different environment situations; these are used to guide navigation and, in turn, are updated based on navigational outcomes so as to improve the reliability of their predictions in similar situations in the future.

3 Evaluation

We evaluated the methods presented above using extensive simulations across a variety of different types of environment, performance criteria, and system configurations. The objective of these experiments was to measure the qualitative and quantitative improvement in the navigation performance of SINS (denoted "sins" in the figures), and to compare this performance against several non-learning schema-based reactive systems (the "static" systems) that do not change schema parameters and a system that changes its schema parameter values randomly after every control interval (the "random" system). Furthermore, rather than simply measure improvement in performance in SINS by some given metric such as "speedup," we were interested in systematically evaluating the effects of various design decisions on the performance of the system using several different metrics. To achieve this, we designed several experiments, which can be grouped into four sets as discussed below.

3.1 Experiment Design

The systems were tested on randomly generated environments consisting of rectangular bounded worlds. Each environment contains circular obstacles, a start location, and a destination location, as shown in figure 2. Figure 5

shows an example runs of two of the static systems, the random system, and the SINS system on a typical randomly generated world. The location, number, and radius of the obstacles were randomly determined to create environments of varying amounts of **clutter,** defined as the ratio of free space to occupied space. 15% clutter corresponded to relatively easy worlds and 50% clutter to very difficult worlds. We tested the effect of three different design parameters in the SINS system: **max-cases,** the maximum number of cases that SINS is allowed to create; **max-size,** *lc,* representing the maximum number of associations in a case; and **control-interval,** T, which determines how often the schema, parameters in the reactive control module are adapted.

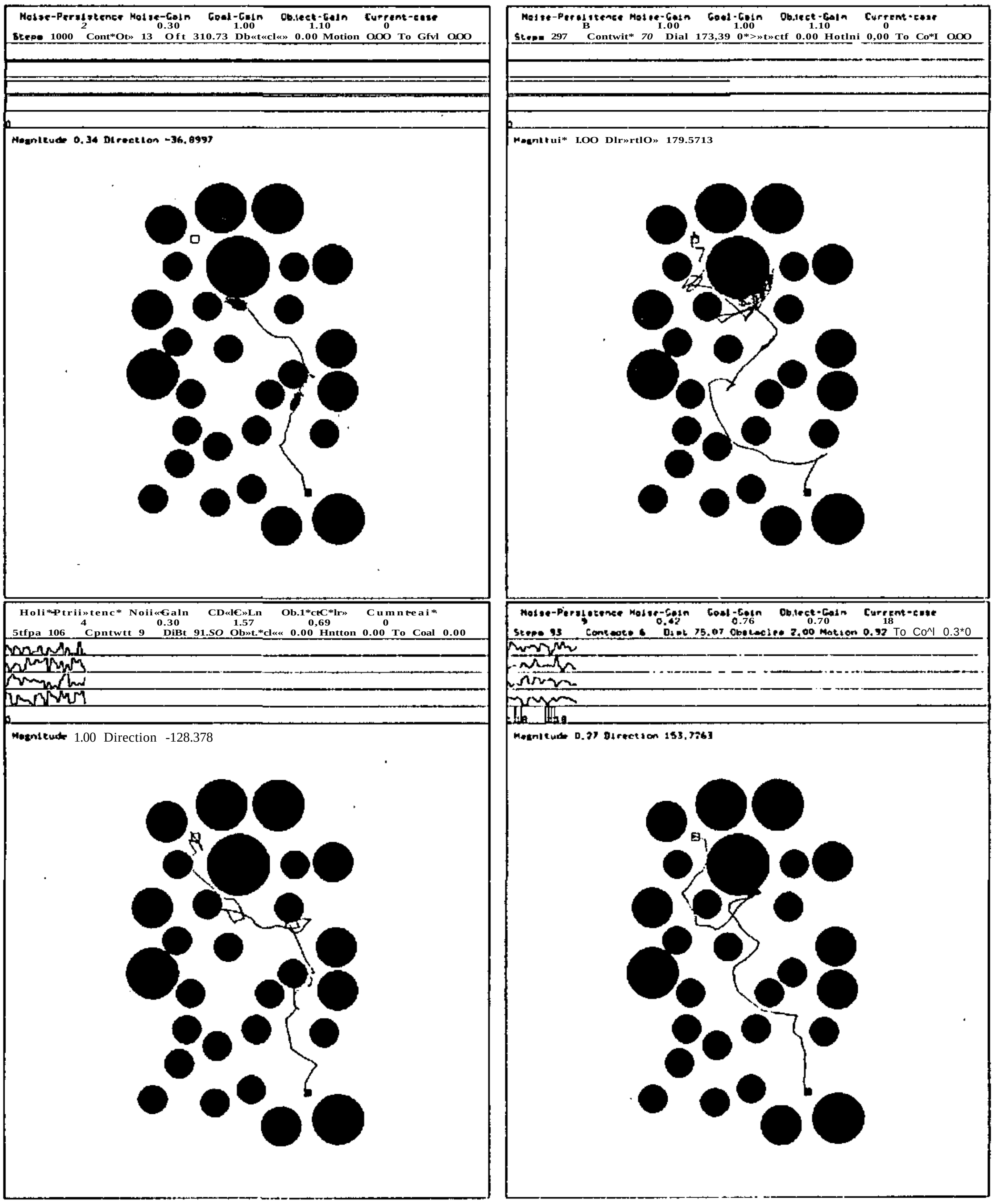
We used three non-adaptive systems for comparison. These systems were identical to the underlying navigation module in SINS, but used different fixed sets of schema parameter values. The "staticH" system used a hand-coded set that we designed manually by watching the behavior of the system on both 15% and 50% cluttered worlds. The "staticl5" and "static50" systems used the schema parameter values reported in Ram, Arkin, Moorman, and Clark (1992), which were hand-optimized through extensive trial and error to 15% and 50% cluttered worlds, respectively. As discussed below, SINS performed as well as the staticl5 system on 15% cluttered worlds and the static50 system on the 50% cluttered worlds; furthermore, SINS reached this level of performance autonomously and, unlike the static systems which were optimized only for a specific clutter, was robust across sudden changes in clutter.

We used six estimators to evaluate the navigation performance of the systems. These metrics were computed using a cumulative average over several hundred test worlds to factor out the intrinsic differences in difficulty of different worlds. *Percentage of worlds solved* indicates in how many of the worlds posed the system actually found a path to the goal location. The optimum value is 100% since this would indicate that every world presented was successfully solved. *Average steps per world* indicates the average of number of steps that the robot takes to terminate each world; smaller values indicate better performance. *Average distance per world* indicates the total distance traveled per world on average; again, smaller values indicate better performance. *Average* $\underline{^{ac}\}V\text{-}^{a}\dot{K}}$ *opumal distance per world* indicates the ratio of the total distance traveled and the Euclidean distance between the start and end points, averaged over the solved worlds. The optimal value is 1, but this is only possible in a world without obstacles. *Average virtual collisions per world* indicates the total number of times the robot came within a pre-defined distance of an obstacle. Finally, *average time per world* indicates the total time the system takes to execute a world on average.

The data for the estimators was obtained after the systems terminated each world. This was to ensure that we were consistently measuring the effect of learning across experiences rather than within a single experience (which is less significant on worlds of this size anyway). The execution is terminated when the navigation system reaches its destination or when the number of steps reaches an upper limit (1000 in the current evaluation). The latter condition guarantees termination since some worlds are unsolvable by one or both systems.

In this paper, we discuss the results from the following sets of experiments:

— Experiment set 1: Evaluation of the effect of our multistrategy case-based and reinforce-



```
Noise-Persistence Noise-Gain   Goal-Gain   Object-Gain   Current-case
        2           0.30         1.00         1.10            0
Steps 1000   Contacts 13   Dist 310.73  Obstacles 0.00 Motion 0.00 To Goal 0.00
Magnitude 0.34 Direction -36.8997
```

```
Noise-Persistence Noise-Gain   Goal-Gain   Object-Gain   Current-case
        8           1.00         1.00         1.10            0
Steps 297   Contacts 70   Dist 173.39  Obstacles 0.00 Motion 0.00 To Goal 0.00
Magnitude 1.00 Direction 179.5713
```

```
Noise-Persistence Noise-Gain   Goal-Gain   Object-Gain   Current-case
        4           0.30         1.57         0.69            0
Steps 106   Contacts 9   Dist 91.50  Obstacles 0.00 Motion 0.00 To Goal 0.00
Magnitude 1.00 Direction -128.378
```

```
Noise-Persistence Noise-Gain   Goal-Gain   Object-Gain   Current-case
        9           0.42         0.76         0.70           18
Steps 93   Contacts 6   Dist 75.07  Obstacles 2.00 Motion 0.92 To Goal 0.30
Magnitude 0.27 Direction 153.7763
```

Figure 5: Sample runs of the static15 (top left), static50 (top right), random (bottom left), and SINS (bottom right) systems on a randomly generated world. The robot starts at the black box (towards the lower right side of the world) and tries to navigate to the white box. The graphs near the top of each figure show the values of each schema parameter over time.

ment learning method by comparing the performance of the SINS system against the static and random systems.

- Experiment set 2: Evaluation of the effect of the parameters of the case-based reasoning component of the multistrategy learning system.

- Experiment set 3: Evaluation the effect of the **control-interval** parameter, which determines how often the adaptation and learning module modifies the schema parameters of the reactive control module.

- Experiment set 4: Evaluation of the effect of changing environmental characteristics, and evaluation of the ability of the systems to adapt to new environments and learn new regularities.

3.2 Discussion of Experimental Results

The results in figures 6 through 10 show that SINS does indeed perform significantly better than its non-learning counterparts. To obtain a more detailed insight into the nature of the improvement, let us discuss the experimental results in more detail.

Experiment set 1: Effect of the multistrategy learning method. We first evaluated the effect of our multistrategy case-based and reinforcement learning method by comparing the performance of the SINS system against the static and random systems. SINS was allowed to learn up to 10 cases (**max-cases** = 10), each of **max-size** = 10. Adaptation occurred every **control-interval** = 4 steps.

Figure 6 shows the results obtained for each estimator over 200 randomly generated worlds. Each graph compares the performance on one estimator of each of the five systems, SINS, staticH, staticl5, static50, and random, discussed above. Figure 6 shows the results obtained for each estimator over the 200 worlds with 50% clutter. The best configuration of SINS, which could learn up to 10 cases of maximum size 10 and with a control interval of 4, was selected to do the comparison. As shown in the graphs, SINS performed as well as or better than the other systems with respect to five out of the six estimators. Table 2 and 3 show the final improvement in the system after all the worlds with 15% and 50% clutter, respectively. For example, table 3 shows that SINS successfully navigates 100% of the worlds, the same as the static50 system optimized for 50% cluttered worlds, with 27% fewer virtual collisions. Although the non-learning system was 85% faster in time, the paths it found required the same number of steps. On average, SINS' solution paths were about the same length as those of the static50 system; however, it should be noted that the static50 system was customized for 50% cluttered worlds, whereas SINS improved its performance regardless what type of environment was dealing with. The static50 system did not perform as well in 15% cluttered worlds; although the staticl5 system did better, SINS compared favorably with that system in those worlds. Another important result is that SINS improved the performance independently of the initial values for the schema parameters; for example, when initialized with the same schema parameters as the staticH system, it was able to achieve performance far superior to the staticH system, and comparable with or better than the staticl5 system in 15% cluttered worlds and the static50 system in 50% cluttered worlds.

The average time per world was the only estimator in which the self-improving system performed worse. The reason for this behavior is that the case retrieval process in SINS is very time consuming. However, since in the physical world the time required for physical execution of a motor action outweighs the time required to select the action, the time estimator is less critical than the distance, steps, and solved worlds estimators. Furthermore, as discussed below, better case organization methods should reduce the time overhead significantly.

The experiments also demonstrate also our assumption about the utility of adapting schema parameter values in reactive systems: the number of worlds solved by the navigation system is increased by changing the values of the schema parameters even in a random fashion, although the random changes lead to greater distances traveled. This may be due to the fact that random changes can get the system out of "local minima" situations in which the current settings of

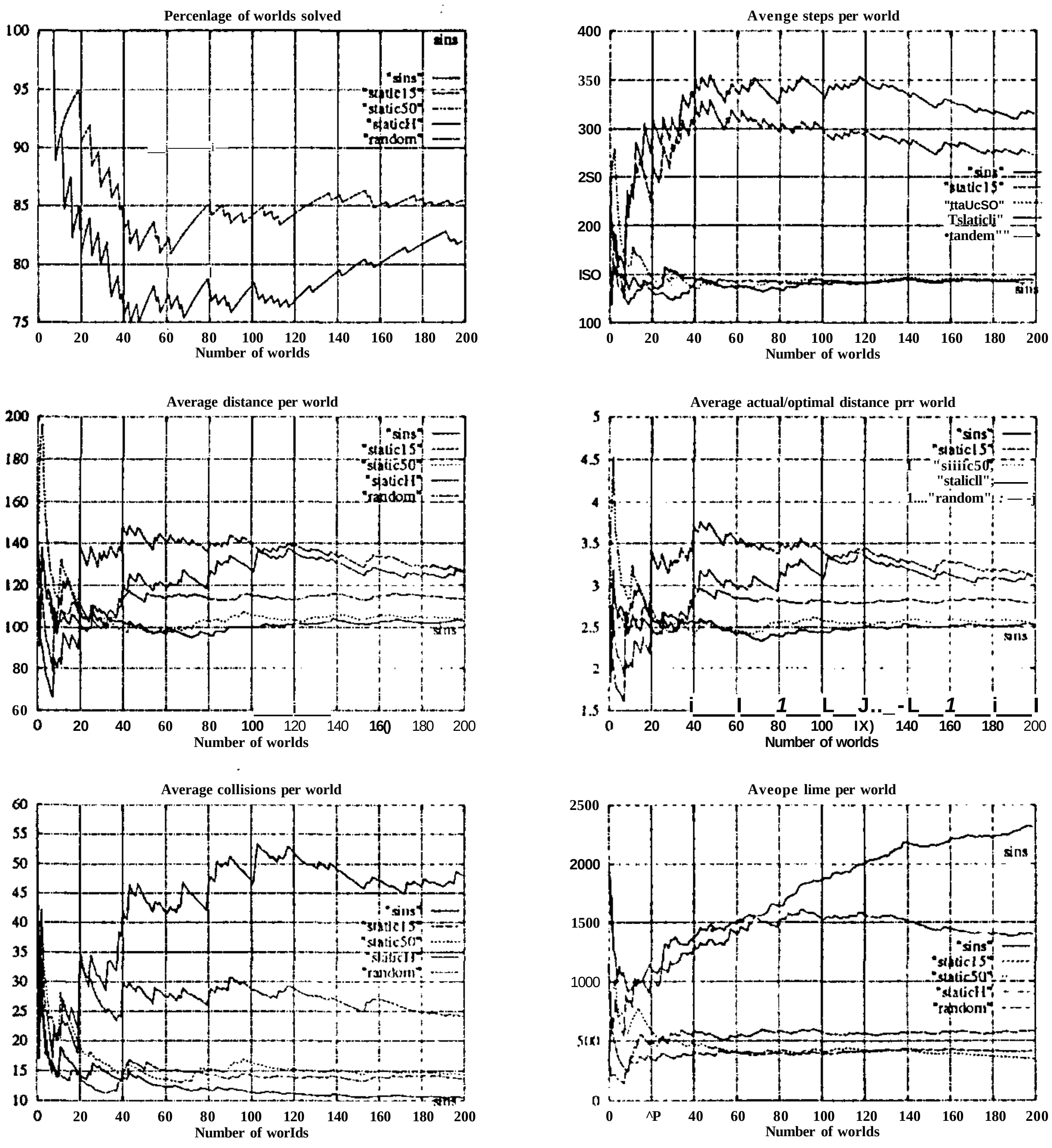


Figure 6: Cumulative performance results on 50% cluttered worlds.

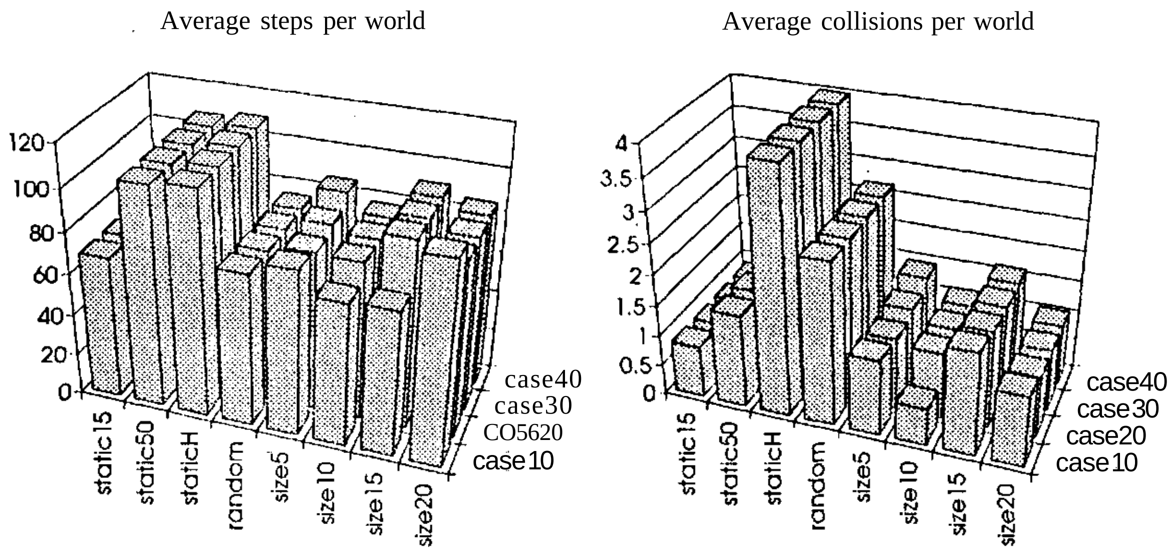


Figure 7: Effect of **max-cases** and **max-size** on 15% cluttered worlds.

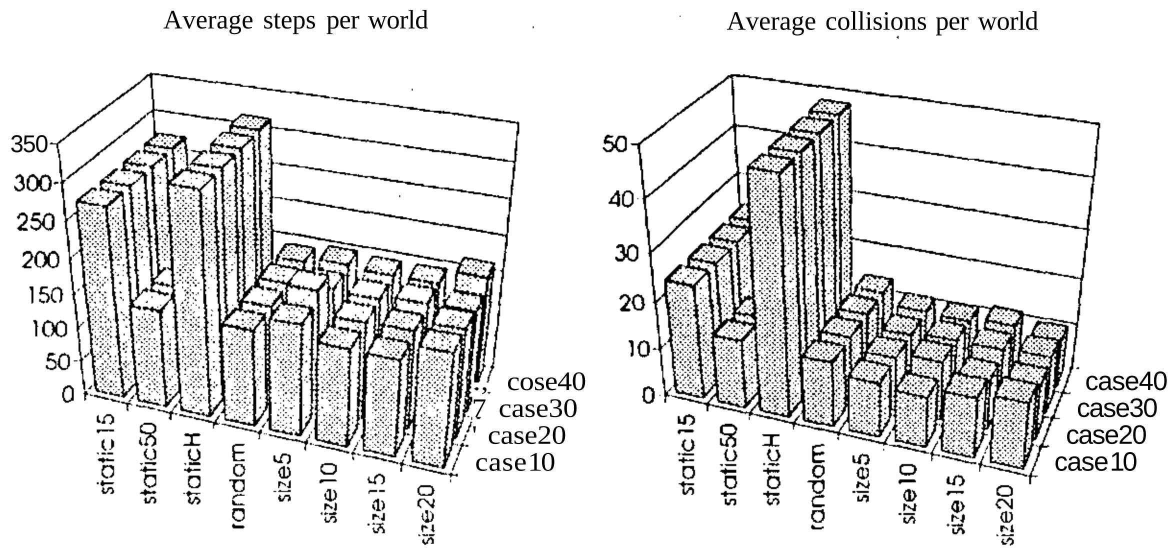


Figure 8: Effect of **max-cases** and **max-size** on 50% cluttered worlds.

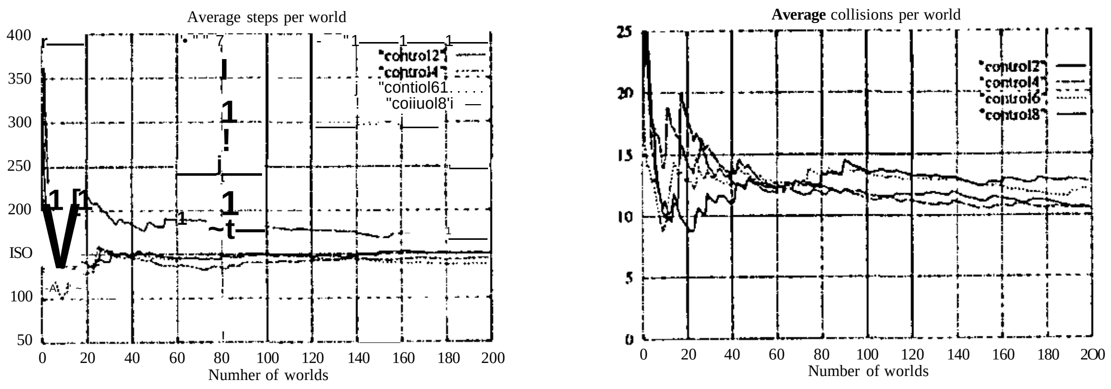


Figure 9: Effect of control-interval on 50% cluttered world with max-cases=20 and max-size=15.

	static15	*static50*	*static H*	*random*	*SINS*
Percentage of worlds solved	99.5%	100%	98.5%	99%	99.5%
Average steps per world	69.1	106.4	108.5	73.8	68.9
Average distance per world	49.9	71.8	49.8	56.5	51.5
Average $\frac{actual}{optimal}$ distance	1.23	1.76	1.22	1.39	1.27
Average virtual collisions	0.85	1.53	3.99	2.66	0.62
Average time per world, ms	57	90	210	94	513

Table 2: Final performance results for 15% cluttered worlds.

its parameters are inadequate. However, consistent changes (i.e., those that follow the regularities captured by our method) lead to better performance than random changes alone.

Experiment set 2: Effect of case parameters. This set of experiments evaluated the effect of two parameters of the case-based reasoning component of the multistrategy learning system, that is, max-cases and max-size. The parameter control-interval was held constant at 4, while max-cases was set to 10, 20, 30 and 40, and max-size was set to 5, 10, 15 and 20. All these configurations of SINS, and the static and random systems, were evaluated using all six estimators on 200 randomly generated worlds of 15% and 50% clutter. Figures 7 and 8 show the results for two of the estimators, *average steps* and *average virtual collisions.*

AH configurations of the SINS system navigated successfully in a comparable or larger percentage of the test worlds than the static systems. Regardless of the max-cases and max-size parameters, SINS could solve most of the 15% and 50% cluttered worlds. As before, the graphs show that SINS' performance was comparable to that of the static15 system in 15% cluttered worlds, and to the static50 system in 50% cluttered worlds. Thus, even if SINS is initialized with a poor set of schema parameter values, it can discover a good set of values and improve upon its initial performance.

Our experiments revealed that, in both 15% and 50% cluttered worlds, SINS needed about 40 worlds to learn enough to be able to perform successfully thereafter using 10 or 20 cases. However, with higher numbers of cases (30 and 40), it took more trials to learn the regularities in the environment. It appears that larger numbers of cases require more trials to train through trial-and-error reinforcement learning methods, and furthermore there is no appreciable improvement in later performance. The max-size parameter has an ap-

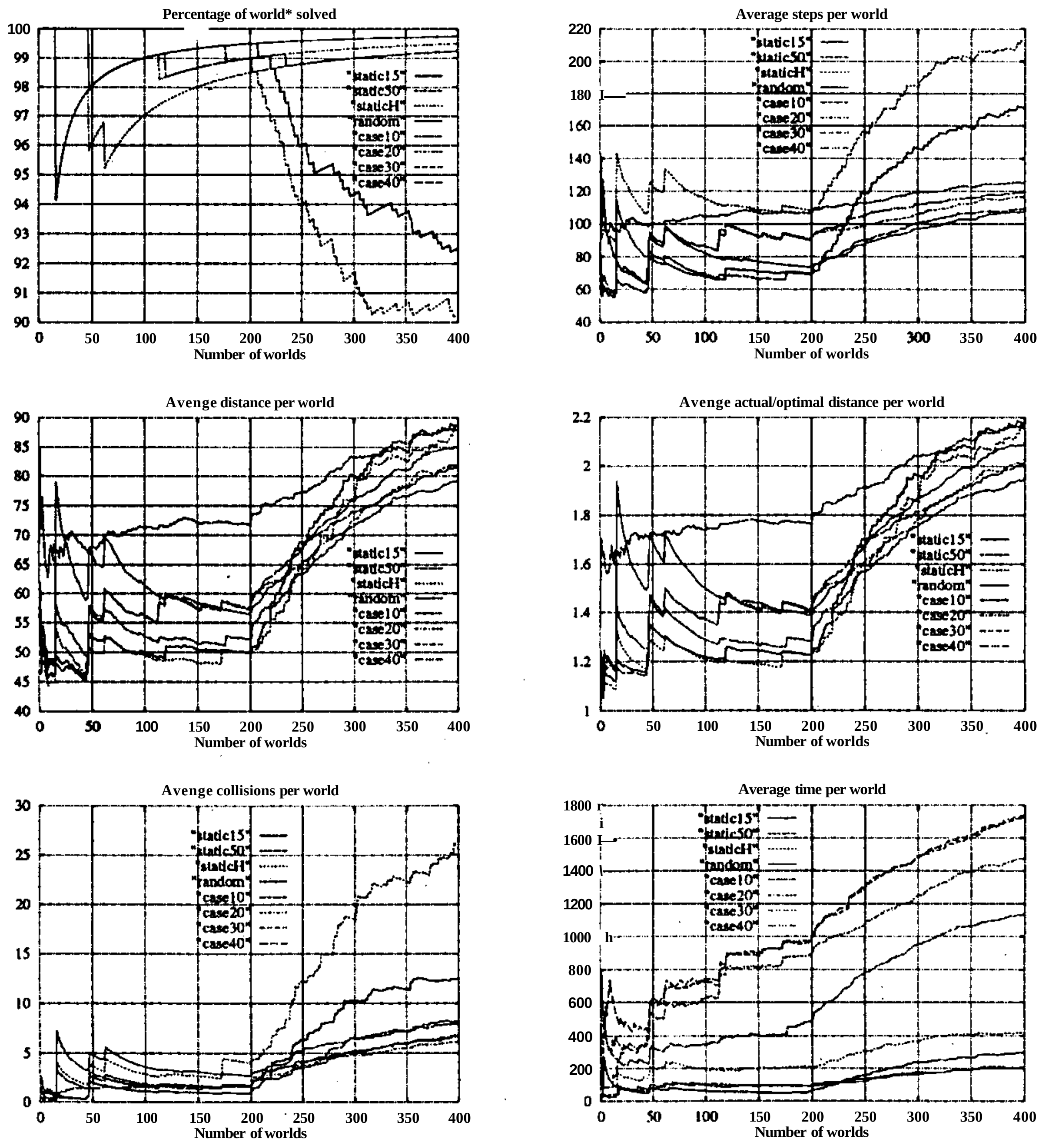


Figure 10: Effect of a sudden change in environment (after the 200th world).

	staticlS	*static50*	*staticH*	*random*	*SINS*
Percentage of worlds solved	85.5%	100%	82%	100%	100%
Average steps per world	272.7	143.6	315.5	141	141.9
Average distance per world	127.1	103.8	126.2	113.7	107.2
Average $\frac{\text{ac <;ual}}{\text{optimal}}$ distance	3.12	2.55	3.10	2.79	2.63
Average virtual collisions	24.1	14.4	48.1	13.7	10.7
Average time per world, ms	584	350	1400	413	1375

Table 3: Final performance results for 50% cluttered worlds.

preciable effect on the performance for different environments. While in the 15% cluttered worlds, the best performance is obtained with cases of size 5, in 50% cluttered worlds, the best performance occurs with cases of size 15. The reason for this is that complex worlds (i.e., 50% cluttered) require longer sequences of associations to ensure real progress. With shorter sequences of associations, there is a high probability that the system will get into a cycle that consists of going in and out of a local minimum point.

As observed earlier in experiment set 1, SINS requires a time overhead for case-based reasoning and thus loses out on the *average time* estimator. Due to the nature of our current case retrieval algorithm, the time required for case retrieval increases linearly with **max-cases** and with **max-size.**

Experiment set 3: Effect of control interval. This set of experiments evaluated the effect of the **control-interval** parameter, which determines how often the adaptation and learning module modifies the schema parameters of the reactive control module, **max-cases** and **max-size** were held constant at 20 and 15, respectively, while **control-interval** was set to 2, 4, 6 and 8. All systems were evaluated using all six estimators on 200 randomly generated worlds of 50% clutter. The results are shown in figure 9.

Although all settings of control interval resulted in improved performance through experience, the best and worst performance in terms of *percentage of worlds solved* was obtained with **control-interval** set to 4 and 8, respectively. For low **control-interval** values, we expect poorer performance because environment classification cannot occur reliably. We also expect poorer performance for very high values because the system cannot adapt its schema parameters quickly enough to respond to changes in the environment. Other performance estimators also show that **control-interval** = 4 is a good setting. Larger **control-intervals** require less case retrievals and thus improve *average time per world;* however, this gets compensated by poorer performance on other estimators.

Experiment set 4: Effect of environmental change. This set of experiments was designed to evaluate the effect of changing environmental characteristics, and to evaluate the ability of the systems to adapt to new environments and learn new regularities. With **max-cases** set to 10, 20, 30 and 40, **max-size** set to 15, and **control-interval** set to 4, we presented the systems with 200 randomly generated worlds of 15% clutter followed by 200 randomly generated worlds of 50% clutter. The results for *average steps* and *average virtual collisions* are shown in figure 10.

The results from these experiments demonstrate the flexibility and adaptiveness of the learning methods used in SINS. Regardless of parameter settings, SINS was very robust and continued to be able to navigate successfully despite a sudden change in environmental clutter. After the 200th world, it continued to solve 100% of the worlds presented to it, with only modest deterioration in steps, distance, virtual collisions, and time in the more cluttered environments. The performance of the static systems, in contrast, deteriorated in the more cluttered environments, with the exception of static50 which was designed for such environments. The static50 started to improve in 50% worlds as compared to the 15% worlds where it was not performing as well as staticl5. As can be seen from the graphs, SINS performed as well as staticl5 in the first 200 worlds, and then as well as static50 in the next 200 worlds; furthermore, it achieved this level of performance even when it was initialized with non-optimized schema parameters. This result also suggests that the regularities that SINS captures in its cases encode strategic knowledge for navigation that is generally applicable across different types of environments. The performance of the random system does not deteriorate too badly but it was not the best in either of the two types of worlds.

Summary: These and other experiments show the efficacy of the multistrategy adaptation and learning methods used in SINS across a wide range of qualitative metrics, such as flexibility of the system, and quantitative metrics that measure performance. The results also indicate that a good configuration for practical applications is **max-cases** = 20, **max-size = 15,** and **control-interval** = 4, although other settings might be chosen to optimize particular performance estimators of interest. These values have been determined empirically. Although the empirical results can be explained intuitively, more theoretical research is needed to analyze why these particular values worked best.

4 Conclusions

We have presented a novel method for augmenting the performance of a reactive control system that combines case-based reasoning for on-line parameter adaptation and reinforcement learning for on-line case learning and adaptation. The method is fully implemented in the SINS program, which has been evaluated through extensive simulations.

The power of the method derives from its ability to capture common environmental configurations, and regularities in the interaction between the environment and the system, through an on-line, adaptive process. The method adds considerably to the performance and flexibility of the underlying reactive control system because it allows the system to select and utilize different behaviors (i.e., different sets of schema parameter values) as appropriate for the particular situation at hand. SINS can be characterized as performing a kind of constructive representational change in which it constructs higher-level representations (cases) from low-level sensorimotor representations (Ram, 1993).

In SINS, the perception-action task and the adaptation-learning task are integrated in a tightly knit cycle, similar to the "anytime learning" approach of Grefenstette & Ramsey (1992). Perception and action are required so that the system can explore its environment and detect regularities; they also, of course, form the basis of the underlying performance task, that of navigation. Adaptation and learning are required to generalize these regularities and provide predictive suggestions based on prior experience. Both tasks occur simultaneously, progressively improving the performance of the system while allowing it to carry out its performance task without needing to "stop and think."

Although SINS integrates reinforcement learning and case-based reasoning, its algorithms are somewhat different from the standard algorithms used in these areas. One of the main differences between traditional reinforcement learning and SINS is with respect to the action model used by the system. While in reinforcement learning the action model is given along with the definition of the system, in SINS it is learned through experience. One open issue in this respect is how to use this action model once it has been learned or while it is in the process of being learned. In SINS, the action model is represented as sequences of associations. In our current implementation, the system always uses the sequence of associations most similar to the current environment, but other implementations may select different sequences according to other criteria. One such criterion we are currently exploring is selecting the sequence of associations that is likely to result in a desired environmental configuration. This would enable SINS to behave in a goal-oriented fashion while still staying within a reactive framework (see also Maes, 1990).

In contrast to traditional case-based reasoning methods which perform high-level reasoning in discrete, symbolic problem domains, SINS is based on a new method for "continuous case-based reasoning" in problem domains that involve continuous information, such as sensorimotor information for robot navigation (Ram & Santamari'a, 1993). There are still several unresolved issues in this research. The case retrieval process is very expensive and limits the number of cases that the system can handle without deteriorating the overall navigational performance, leading to a kind of utility problem (Minton, 1988). Our current solution to this problem is to place an upper bound on the number of cases allowed in the system. A better solution would be to develop a method for organization of cases in memory; however, conventional memory organization schemes used in case-based reasoning systems (see Kolodner, in press) assume structured, nominal information rather than continuous, time-varying, analog information of the kind used in our cases.

Another open issue is that of the nature of the regularities captured in the system's cases. While SINS' cases do enhance its performance, they are not easy to interpret. Interpretation is desirable, not only for the purpose of obtaining of a deeper understanding of the methods, but also for possible integration of higher-level reasoning and learning methods into the system.

Despite these limitations, SINS is a complete and autonomous self-improving navigation system, which can interact with its environment without user input and without any preprogrammed "domain knowledge" other than that implicit in its reactive control schemas. As it performs its task, it builds a library of experiences that help it enhance its performance. Since the system is always learning, it can cope with major environmental changes as well as fine tune its navigation module in static and specific environment situations. The system benefits from the tight integration of multiple learning strategies that support and complement each other, and the on-line use of the learned knowledge in guiding the performance task.

Acknowledgements

We would like to thank Ron Arkin for his contributions to this research and for providing the AuRA robot simulator, Paul Rowland for his help with replicating the results presented in this paper, and George Tecuci for his comments on an early draft of this paper.

References

[1] R.C. Arkin, Motor *Schema-Based Mobile Robot Navigation.* The International Journal of Robotics Research, 8(4), 92-112, (1989).

[2] R. Brooks, A *Robust Layered Control System for a Mobile Robot.* IEEE Journal of Robotics and Automation, RA-2(1), 14-23, (1986).

[3] S.A. Chien, M.T. Gervasio, and G.F. DeJong, *On Becoming Decreasingly Reactive: Learning to Deliberate Minimally.* In Birnbaum, L. & Collins, G. (editors), Proceedings of the Eighth International Workshop on Machine Learning, 288-292, Chicago, IL, (1991).

[4] R.E. Fikes, P.E. Hart, and N.J. Nilsson, *Learning and Executing Generalized Robot Plans.* Artificial Intelligence, 3, 251-288, (1972).

[5] M. Georgeff, *Planning.* Annual Review of Computer Science, 2, 359-400, (1987).

[6] J.J. Grefenstette and C.L. Ramsey, *An Approach to Anytime Learning.* In Sleeman, D. and Edwards, P. (editors), Machine Learning: Proceedings of the Ninth International Conference, 189-195, Aberdeen, Scotland, (1992).

[7] K.J. Hammond, *Case-Based Planning: Viewing Planning as* a *Memory Task.* Academic Press, Boston, MA, (1989).

[8] L. Kaelbling, An *Architecture for Intelligent Reactive Systems,* Technical Note 400, SRI International, (1986).

[9] J.L. Kolodner, *Case-Based Reasoning,* Morgan Kaufmann, San Mateo, CA, (in press).

[10] P. Maes, *Situated Agents can have Goals.* Robotics and Autonomous Systems, 6, 49-70, (1990).

[11] S. Minton, *Learning Effective Search Control Knowledge: An Explanation-Based Approach,* PhD thesis, Technical Report CMU-CS-88-133, Carnegie-Mellon University, Computer Science Department, Pittsburgh, PA, (1988).

[12] T.M. Mitchell, *Becoming Increasingly Reactive.* In Proceedings of the Eighth National Conference on Artificial Intelligence, 1051-1058, Boston, MA, (1990).

[13] J. Mostow and N. Bhatnagar, *FAILSAFE — A Floor Planner that uses EBG to Learn from its Failures.* In Proceedings of the Tenth International Joint Conference on Artificial Intelligence, 249-255, Milan, Italy, (1987).

[14] D. Payton, An *Architecture for Reflexive Autonomous Vehicle Control.* In Proceedings of the IEEE Conference on Robotics and Automation, 1838-1845, (1986).

[15] M. Pearce, R. Arkin, and A. Ram, *The Learning of Reactive Control Parameters through Genetic Algorithms.* In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 130-137, Raleigh, NC, (1992).

[16] A. Ram, *Creative Conceptual Change.* In Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society, 17-26, Boulder, CO, (1993).

[17] A. Ram, R.C. Arkin, K.. Moorman, and R.J. Clark, *Case-Based Reactive Navigation: A Case-Based Method for On-Line Selection and Adaptation or Reactive Control Parameters in Autonomous Robotic Systems.* Technical Report GIT-CC-92/57, College of Computing, Georgia Institute of Technology, Atlanta, Georgia, (1992).

[18] A. Ram and J.C. Santamaría, *Continuous Case-Based Reasoning.* In Leake, D.B. (editor), Proceedings of the AAAI Workshop on Case-Based Reasoning, AAAI Press Technical Report WS-93-01, 86-93, Washington, DC, (1993).

[19] E.D. Sacerdoti, A Structure for Plans and Behavior, Technical Note 109, Stanford Research Institute, Artificial Intelligence Center. Summarized in P.R. Cohen and E.A. Feigenbaum's Handbook of AI, Volume III, 541-550,(1975).

[20] A.M. Segre, *Machine Learning of Robot Assembly Plans,* Kluwer Academic Publishers, Norwell, MA, (1988).

[21] R.S. Sutton, *Integrated Architectures for Learning, Planning, and Reacting based on Approximating Dynamic Programming.* In Proceedings of the Seventh International Conference on Machine Learning, 216-224, Austin, TX, (1990).

[22] R.S. Sutton (editor), *Special Issue on Reinforcement Learning.* Machine Learning, 8(3/4), (1992).

[23] P.F.M.J. Verschure, B.J.A. Kröse and R. Pfeifer, *Distributed Adaptive Control: The Self-Organization of Structured Behavior.* Robotics and Autonomous Systems, 9, 181-196,(1992).

[24] C.J.C.H. Watkins, *Learning from Delayed Rewards,* PhD thesis, University of Cambridge, England, (1989).

[25] S.D. Whitehead and D.H. Ballard, Active *Perception and Reinforcement Learning.* In Proceedings of the Seventh International Conference on Machine Learning, 179-188, Austin, TX, (1990).

COMBINING KNOWLEDGE-BASED AND INSTANCE-BASED LEARNING TO EXPLOIT QUALITATIVE KNOWLEDGE

Gerhard Widmer
Department of Medical Cybernetics and Artificial Intelligence, University of Vienna, and
Austrian Research Institute for Artificial Intelligence,
Schottengasse 3, A-1010 Vienna, Austria

Keywords: artificial intelligence, multistrategy learning, instance-based learning, knowledge-based learning, qualitative knowledge, music

Edited by: Gheorghe Tecuci

Received: May 26, 1993 **Revised:** October 15, 1993 **Accepted:** October 28, 1993

The paper presents a. general learning method that integrates knowledge-based symbolic learning with instance-based numeric learning. This combination is motivated by a class of learning problems where the task is to predict a numeric target variable, and where a qualitative domain theory is available. The method has been implemented in a learning program named IBL-SMART. *Its symbolic learning component is a multi-instance plausible explanation algorithm that can use the qualitative domain knowledge to guide its search, and the numeric component performs instance-based prediction of the numeric target variables. The system has been applied to a complex problem from the domain of tonal music. The application domain is briefly described, and some experiments are presented to illustrate the power of the learning method.*

1 Introduction

The advent of *Explanation-Based Learning* (EBL) [11, 22] marked a great advance in machine learning, as it stressed the importance of knowledge for learning and provided a clear framework for how knowledge could be used to improve learning. At the same time, it was clear that many interesting learning tasks are not amenable to straight EBL, as it is not always possible to come up with domain theories that are complete, consistent, and also tractable.

Subsequently, a lot of research concentrated on the *imperfect theory problems* that were already identified by Mitchell *et al.* [22]: various approaches to learning from incorrect, incomplete, and/or intractable domain theories were investigated [3, 7, 8, 12, 14, 16, 18, 19, 23, 25, 27, 31, 33, 36]. Also, methods for inductively refining or extending incorrect and incomplete theories were developed [2, 24, 34, 40].

Most of these approaches assumed that the domain theory, whether incomplete or inconsistent or otherwise deficient, was in the form *of if-then-rules* as in classical EBL. It has become increasingly clear, however, that in many domains, the knowledge that is readily available is naturally in *qualitative, abstract form.* Consider, for instance, numeric learning tasks such as stock market prediction or the prediction of the energy demand in some city. While it seems impossible to come up with a precise model of the underlying processes that would enable one to correctly predict (or explain in an EBL sense) each observation, one can easily conceive of partial qualitative models of these domains that would capture some important underlying trends, biases, or dependencies between relevant parameters,

This insight has led to a number of approaches to learning with qualitative domain theories—ones that could best be characterized as 'Plausible Explanation-Based Learning' [9, 10, 30, 32, 38].

Generally, research on such methods is guided by the view that for any learning system, learning entails trying to understand its inputs in terms of its current knowledge [21, 29]. Trying to understand, in this context, means trying to relate new incoming information (the training instances) to the knowledge (domain theory) that the learner already possesses. The result of such an attempt at understanding is usually some kind of *explanation* which shows how the training input is consistent with the system's general knowledge about the domain. In classical Explanation-Based Learning [22], explanations are deductive proofs of an instance being a member of the target concept. In domains with qualitative background knowledge, *plausible inference methods* [6] can be used to construct *plausible explanations.* Such explanation may be incomplete or more or less uncertain, depending on the available knowledge, but if they are at all sensible, they will allow the learner to prefer certain generalizations over others on the grounds of relative plausibility.

Most of the research dealing with qualitative background knowledge and plausible explanations so far has concentrated on single-instance explanations, where one example is explained at a time, and new examples may lead to incremental modification of learned concepts (e.g., [9, 10, 30, 38], although the method presented in [30] can also use several examples at once for inductive generalization). The problem with this kind of approach is obvious: if the available background knowledge is very weak (i.e., incomplete or extremely abstract), so will be the explanations based on it; in the extreme case, one can 'explain' almost anything if the plausible inference methods used are sufficiently permissive. The weakness of plausible explanations per *se* must be compensated by additional empirical support from the training data, which means that plausible explanations should also take into account the numbers and distribution of training examples satisfying various conditions.

The particular learning problem that motivated our research, which happens to come from the realm of tonal music (see section 5), is a typical case of a task for which we can relatively easily formulate a qualitative domain theory, but for which no precise theory exists. In addition, the chosen learning task presents another challenge, which necessitated a new approach: the target concepts to be learned are *numeric,* that is, the task is to predict precise numeric values for some target variables. The problem is that the abstract, qualitative background knowledge cannot be used to directly explain precise numeric observations. Still, we want a learner that can use the knowledge to maximum effect.

This paper will present a general learning model, and its realization in a system named IBL-SMART, that was designed to deal with these types of problems. The model consists of two components, a symbolic and a numeric learner. The symbolic learner, a non-incremental search-based method, is itself a multistrategy learner that integrates various forms of inference. It produces search trees that can be viewed as plausible explanations of sets of training examples. The search method that constructs these trees is based on a well-known deductive-inductive learning algorithm [3]. For our system, the method has been extended to utilize *qualitative background knowledge.* Complementing this knowledge-based learning algorithm, the second part of the model is an empirical, *instance-based* component that takes care of the numeric aspects of the learning task.

The model provides a clear division between a symbolic learning sub-task where qualitative knowledge can naturally be incorporated, and a numeric prediction task that is most naturally handled by a numeric, interpolation-based method. The main effect of the model is to provide a way of utilizing abstract background knowledge in numeric domains where the knowledge cannot be directly related to the precise numeric observations. The main contributions of this work, then, are (1) a multi-instance plausible explanation algorithm using qualitative background knowledge; (2) a learning method that integrates such a plausible explanation component with an instance-based learner; and (3) a working system that learns complex concepts in a specific domain and has already produced results that are of interest to researchers in music theory and musicology [39].

2 The Learning Problem

The general scenario is supervised learning of numeric target concepts. More precisely, the class of learning problems we are interested in can be defined as follows:

Given: a set E of training examples, described in terms of a set of operational predicates P, where we distinguish symbolic predicates PS and numeric predicates (attributes) PN. Thus, $PS \cup PN = P$. Also attached to each example $e \in E$ is a numeric attribute $T(e, v)$ (the *target attribute)* with known value v. (This replaces the *classification* in symbolic supervised concept learning.) As v is a function of (the description of) the instance, we will also write $v = T(e)$. Note that so far, there are no negative instances in this scenario.

Find: a set of general rules that predict, for any given object o described by predicates $\in P$, a numeric value $v = T(o)$, based on the description of o. (As in symbolic concept learning, we might require these rules to be *complete* (predict a value for every example) and correct (predict the correct value for each example) with respect to the training data (cf. [20]). However, this may not be 100% desirable or feasible in every application domain.)

In addition, we assume that there is domain-specific *background knowledge (BK)* relating the target concept $T(X, V)$ (or some abstractions of T - see below) to some of the operational predicates P in specific ways, possibly via some intermediate non-operational predicates. This knowledge might be in the form of rules, as in standard EBG domain theories [22] or in the form of less precise, qualitative knowledge items as in [38]. The knowledge need not be correct or complete, nor need it be quantitative and precise. An additional constraint then is to find solutions (rules) that conform as closely as possible to BK while also consistently describing the training data E.

The learning method we are going to introduce in the next section includes a symbolic component that can utilize qualitative background knowledge for generating plausible explanations. For that method to be applicable, we need to make the following *assumptions:*

1. We assume that there are some discrete, qualitative sub-concepts $T_i(X)$ of the target concept $T(X, V)$ that can naturally be distinguished; a sub-concept is defined by a more or less clearly distinguished subrange (interval) of the function value V.

2. We further assume that it is these discrete sub-concepts that are related to operational predicates P by the available background knowledge BK.

3. Finally, we assume that examples of the discrete sub-concepts T_i[1] can be distinguished using the operational predicates P.

For example, in the energy demand prediction task mentioned above, such subconcepts might be extremely_low(Demand) or higher_than_capacity (Demand); in our musical domain (see section 5), there are natural qualitative subconcepts such as crescendo(Note) and decrescendo(Note) (increase or decrease, respectively, in loudness relative to the current level) or accelerando(Note) and ritardando(Note) (increase or decrease in tempo). [1]

The motivation for the first assumption is that these discrete, qualitative, symbolic sub-concepts will be the target concepts for the symbolic learning component. Each of the original training instances will be assigned to one of the sub-concepts T_i, depending on its value $v = T(e)$, and the symbolic learner will learn general rules for each sub-concept. Note that in this way we also introduce negative instances for each target concept T_i, namely, all examples assigned to some T_j where $j \neq i$. Assumptions (2) and (3) are needed to guarantee that the background knowledge is applicable to the symbolic target concepts, and that the symbolic learning task is solvable at all.

3 The General Learning Model

One simple way to approach the problem would be to do instance-based learning [1] in the entire description space spanned by all the available attributes P, symbolic and numeric. That is, training instances would be stored along with their complete descriptions. Of course, the symbolic and numeric attributes would have to be normalized to some standard scale so as to give equal

[1]The boundaries between these subconcepts will sometimes have to be defined somewhat arbitrarily. This is not necessarily a problem, as the results of the symbolic learning component are not used for prediction, but only to find sets of 'similar' instances for focussed interpolation. Section 3 will make that clearer.

importance to all attributes. The value $v = T(o)$ for some new object o would be predicted by some nearest neighbor method in the space of stored instances, possibly with some numeric interpolation. (Various possible interpolation methods are discussed in [17].) There are several problems with this approach. The main one concerns the general sensitivity of IBL algorithms to irrelevant attributes (of which there are many in our application). Second, the only way to integrate available background knowledge into the learning process is via the similarity metrics. This is certainly not the most natural way to express one's domain knowledge. Moreover, instance-based approaches suffer from the problem that they do not produce comprehensible concept descriptions. On the other hand, it is clear that either some kind of instance-based interpolation component or some numeric regression method is needed to predict precise values for the continuous target variable.

The solution adopted here is a general learning model composed of two parts: a *symbolic learning component* that learns to distinguish different types of situations and can utilize all the available domain knowledge, and an instance-based component which stores the instances with their precise numeric attribute values and can predict the target value for some new object by numeric interpolation over known instances. The connection between these two components is as follows: each rule (conjunctive hypothesis) learned by the symbolic learning component describes a subset of the instances; these are assumed to represent one particular subtype of the concept to be learned. All the instances covered by a rule are given to the instance-based learner to be stored together in a separate instance space. P-redicting the target value for some new object then involves matching the object against the symbolic rules and using only those numeric instance spaces (interpolation tables) for prediction whose associated rules are satisfied by the object. In this way, the system learns several distinct instance spaces where different laws and regularities may apply. In fact, different instance spaces may contain examples with conflicting values. These spaces are thus very specialized predictors, conditioned for particular types of situations.

More precisely, the target concepts for the *symbolic learning component* are the discrete, qualitative sub-concepts $T\{$ mentioned in section 2. The symbolic learner learns general rules that characterize or discriminate between these discrete classes. These rules may refer to both symbolic and numeric predicates. The symbolic learner tries to use all the available qualitative background knowledge. The result produced by this component is a set of general rules that group the examples into clusters by assigning them to different sub-classes of the target concept.

The *numeric, instance-based component* takes the original training instances E as clustered by the symbolic learner, and creates a separate instance space from each cluster. Instances are stored with all their numeric attributes and with their precise numeric target values. The dimensions of such an instance space are thus defined by the numeric attributes PN. For some new object o, the target value $v = T(o)$ can then be predicted by selecting the appropriate instance space (by using the generated symbolic rules as filters), and applying some numeric interpolation method over the stored instances. As only the numeric attributes are used by the instance-based component (the rules learned by the symbolic learning component are assumed to contain all the relevant symbolic information), some straightforward interpolation scheme can be used.

The most important feature of the model as a whole is that it provides a natural entry point for the available domain knowledge. The assumptions spelled out in section 2 naturally define a symbolic learning task, to which we can apply any learning algorithm that can utilize the available knowledge. An additional advantage is that the symbolic component produces explicit symbolic rules, which can be inspected and interpreted by humans.

4 The system IBL-SMART

The general method has been implemented in a system named IBL-SMART. In accordance with the model, IBL-SMART consists of two components. The system has been tested with a class of complex musical problems (see section 5). As it turned out, there is a lot of general knowledge (derived from various theories of tonal music) that is relevant to the learning task and can readily be provided, but large parts of this knowledge are only *qualitative* and *approximate.* Thus, IBL-SMART'S symbolic learning component has been specifically designed to be able to use qualitative background knowledge.

The name IBL-SMART derives from the two main components: IBL stands for *Instance-Based Learning* [1] and characterizes the numeric component, and SMART is a tribute to the ML-SMART algorithm [3], which provided some of the ideas for the main search strategy of the symbolic learner.

4.1 The qualitative domain theory

A domain theory to be used by IBL-SMART is similar in structure to the 'classical' domain theories as used in *Explanation-Based Generalization (EBG)* [22]. It is a hierarchy of statements relating non-operational predicates (including the target concepts) to more operational, specific conditions. However, these statements may describe relations of various strength and specificity:

Strict (deductive) rules: As in EBG, the domain theory may contain strict deductive rules of the form Q : -Pi,P2,... that specify sufficient conditions (Pi,P2, • • •) for some (non-operational) predicate Q to be true.

Directed qualitative dependencies: A statement of the form **q+(A,B)** can be paraphrased as "the values of attributes A and B are positively proportionally related" or "high (or low) values of A tend to produce high (or low) values of B, all other things being equal". Negative dependency q-(A,B) is defined analogously. Such statements are, of course, restricted to functional predicates (i.e., attributes) that assign values to objects. Obviously, this type of knowledge is less precise and logically weaker than strict rules. In particular, it does not permit deductive reasoning. Similar types of knowledge items have been proposed in [20] and [6] for plausible reasoning and learning.

Undirected qualitative dependencies: A statement depends_on(Q,[Pi,P2,...]) denotes an unspecific, undirected relation between the set of predicates Pi and the (non-operational) predicate Q. Basically, it says that the value (or truth value) of Q depends somehow on the values (or truth values) of the Pi, but we do not know the exact function that defines this dependency. A similar type of general knowledge items was described in [4]. Also, such dependencies are similar, but not identical to S. Russell's *determinations* [28] (they need not exhaustively list all the relevant factors that determine Q). In the learning algorithm, they will be used to focus the learner on sets of relevant predicates or attributes.

In our musical application, by far the most knowledge items in the domain theory are of the qualitative type. Clearly, EBG is not possible with such a weak theory. But the symbolic learner can use the theory to guide the search for plausible generalizations via a kind of *plausible reasoning* [6].

4.2 The symbolic learning component

IBL-SMART'S symbolic learning component is a non-incremental discrimination algorithm that learns explicit symbolic rules in disjunctive normal form (DNF). Its basic search strategy is inspired by the ML-SMART framework [3]. The algorithm performs top-down discrimination, integrating and interleaving deductive and inductive (plausible) operationalization steps. More specifically, the learner starts with a nonoperational definition of the target concept (some discrete subconcept *Ti)* and performs stepwise operationalization (specialization) by growing a heuristic best-first search tree. Expressions (nodes of the tree) are refined by operationalizing non-operational predicates or by inductively adding new conditions. A node becomes a leaf when it covers only positive training instances; it then represents one conjunct (rule) in the final learned DNF hypothesis. The search terminates when a certain percentage of the positive examples are covered.

The behavior of the best-first search algorithm is determined by two components: the available *search operators* and the *search strategy,* including the heuristic evaluation *function* that guides the search. These are now described in turn.

The specialization operators

IBL-SMART makes use of a number of different specialization operators to exploit the various types of knowledge items in the domain theory:

Deductive operationalization: If the expression to be refined contains a non-operational predicate Q and there is a rule Q : -Pi,P2, • • • in the domain theory, deductive operationalization creates a new expression by replacing Q with its sufficient conditions Pi,P2,... This is the standard EBG explanation operator [22].

Plausible discrimination based on q+ or q-: Qualitative dependencies, while not permitting deductive explanation steps, do provide information about the influence of one attribute to another, more abstract one, and about the direction of that influence. Given a non-operational expression B(X,b), where b is a specific (qualitative) value, and knowing that q+(A,B), a *plausible operationalization step* is to replace (or 'explain') B with A. The algorithm creates successor nodes by replacing B(X,b) with A(X,ai) for all values ai appearing in positive instances covered by the current node. Which of these 'explanations' is most plausible and will be most likely be expanded further is then determined by the evaiuation *function* (see below). Basically, the evaluation function rates the degree to which the particular values involved match the direction of the underlying dependency (+ or -): knowing that q+(A,B), an operationalization B(.)=b because A(.)=a will be regarded the more plausible the more the relative positions of b and a in their respective domains agree: B(.)=high *because* A(.)=high is rated as more plausible than B(.)=high *because* A(.)=low (see also [38])*?*

Discrimination with general dependencies: A statement depends_on(Q,[Pi,P2,...]) indicates that an entire set of predicates (Pl,P2,...) should be used in one operationalization step: successors of a node containing Q are created for *all possible combinations* of instantiations for the predicates Pi occurring in some positive instances covered by the node. Which of these refinements is most plausible will then be determined empirically by the evaluation function, which takes into account the numbers of positive and negative instances covered by an expression. Basically, adding several predicates in one discrimination step permits IBL-SMART to perform strictly constrained forms of look-ahead, and helps overcome blindness effects that would arise in purely step-wise specialization.

Empirical discrimination: When no knowledge-based specialization step is possible for a given expression, but the expression still covers some negative instances, IBL-SMART empirically adds discriminating conditions—one at a time—to the expression to exclude some of the negative examples. For *numeric* attributes, the system computes some best split point and adds an inequality test (as it is typically done in decision-tree learners [13]).

The search strategy

Generally, given a some expression to refine or specialize, IBL-SMART prefers those operators that are based on the most specific knowledge. Deductive operationalization is performed whenever possible, plausible discrimination steps are considered next, and inductive specialization is attempted only when no knowledge (rule or dependency) is applicable to the current node.

Which node in the search tree is considered most promising and is to be refined next is determined by the heuristic *evaluation function H* that measures the relative 'goodness' of nodes. It takes into account (1) the coverage of the current node, i.e., the ratio positive / negative instances covered by the expression; (2) the absolute number of positive instances covered (expressions covering more positive examples are slightly preferred); and (3) in the case of nodes derived via some directed qualitative dependency, the *degree of fit* between the values used in the discrimination step and the direction of the influence (see above).

By taking into account both such inference-dependent plausibility measures and information about the numbers of instances covered by a node,

[2]In principle, q+(A,B) could represent any monotonic function. However, we assume that the relation is roughly *linear.* As the features involved are only qualitative (usually with small domains like {extremelyJLOB,.. . ,extremely_high}), that seems sufficient. Assuming more complex relationships between such crude features would seem to be missing the point.

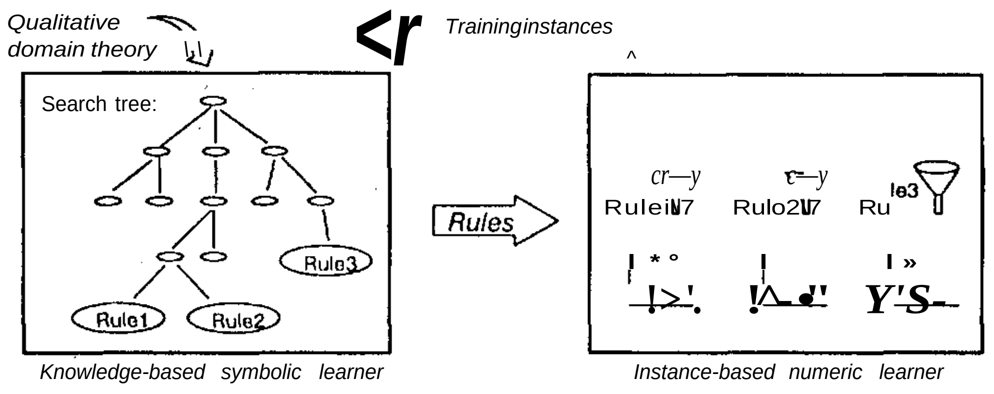

Qualitative domain theory
Training instances
Search tree:
Rule1
Rule2
Rule3
Rules
Rule1
Rule2
Rule3
Knowledge-based symbolic learner
Instance-based numeric learner

Figure 1: Integration of knowledge-based symbolic learning and instance-based numeric learning in IBL-SMART.

the search heuristic combines weak, imprecise background knowledge with empirical information from the training data, producing hypotheses that tend to correspond to the background knowledge as much as the data permits and overriding the background knowledge if the data is in conflict with the knowledge. The search algorithm thus realizes a tight and fine-grained integration of deduction and more or less plausible inference.

In a sense, the search tree can be regarded as a *plausible explanation* of the training instances: it relates the training examples to the target concept via the system's domain knowledge and, where that is missing or inappropriate, via empirical arguments.

4.3 The numeric instance-based component

The symbolic learning component produces a concept hypothesis for a discrete, qualitative subconcept T_i' in the form of a DNF expression, where each conjunct describes one particular subtype of the sub-concept. For each of these conjuncts, i.e., for each leaf of the search tree, the instance-based learner collects all the training instances covered by the conjunct and builds an instance store in the form of an *interpolation table*, using these examples. The output dimension (the dependent variable) is the value of the numeric target function $V = T(X)$. The input dimensions (the independent variables) are chosen to be all the numeric attributes ($\subseteq PN$) that are shared by all instances assigned to the instance store. (The reason for this is that in our musical application, not all instances may have defined values for all numeric attributes).

When given a new instance for which to predict the value of the target variable, the system matches the instance against all learned rules, retrieves those instance spaces whose associated rules are matched, and computes a value for the instance's target value by linear interpolation between the two nearest neighbors in each of the retrieved spaces. As all the dimensions of the tables are numeric, Euclidean distance is used as the similarity measure. If the instance matches more than one rule, and thus target values are computed in several spaces, the target values are simply averaged. Figure 1 summarizes the basic structure of IBL-SMART.

5 An Application of IBL-SMART: Learning Expressive Interpretation

Expressive interpretation is the art of shaping a piece of music by playing it not exactly as given in the written score, but continuously varying tempo, loudness, and other parameters during a performance. Actually, when played exactly as written, most pieces of music would sound utterly mechanical and lifeless.

When viewed as a learning problem, the task is to learn, from listening to recordings of known pieces as played by some performer and comparing them to the score of the pieces as actually written, general rules that allow one to decide how to play new melodies, i.e., when to increase or decrease the loudness, when to speed up or slow down, etc.

IBL-SMART was applied to two dimensions of this learning problem, namely, *dynamics* (loudness variations) and *rubato* (tempo variations). In the following presentation, we will concentrate on the dimension of dynamics.

The level on which examples and target concepts are defined in our application is the level of the individual note: each note in a piece is an example, and the goal is to learn general rules to decide how loud to play any note in some new piece. Obviously, this concept is inherently numeric, as the task is to decide not just whether or not to play some note louder or softer, but exactly by *how much.* But there are two discrete, qualitative sub-concepts that can naturally be distinguished: crescendo (Note) and decrescendo(Note), which are defined here to mean that a note is to be played louder or softer, respectively, than some standard level.[3] These are the target concepts *T{* for the symbolic learning component. The precise amounts by which the loudness is to be varied are numeric multiplication factors that are to be learned by the instance-based component.

Training instances are derived from actual performances of piano pieces recorded on an electronic piano via a MIDI interface. At the moment, we are restricting ourselves to single line melodies (with additional information about the underlying harmonic structure of the piece). That is, the input is a sequence of notes, described in terms of various predicates, symbolic and numeric, and accompanied by explicit information about the degree of crescendo or decrescendo that was applied to it by the performer (the target variable).

The *description language* consists of predicates that describe various features of a note and structural features of its surroundings. There are currently 41 operational predicates, of which 21 are symbolic (like followed_by_rest(Note)) and 20 are numeric (like duration(Note.X)). Some of these predicates are computed by a preprocessing component which performs a music-theoretic analysis of the given piece in terms of some relevant musical structures (e.g., phrases and various types of patterns such as linear melodic lines (ascending or descending), rhythmic patterns, etc.). Many numeric attributes then describe the relative position of a note in a phrase or in other types of patterns. Note that the number of attributes defined for a given note varies: some notes occur in many patterns, others only in some. So not all numeric attributes are defined for every note.

The *domain theory* we have developed for this problem is based in part on accepted theories of music, and in part represents our personal hypothesis concerning the relation between the structure of music and its plausible interpretations.[4] For our presentation here, it suffices to say that the domain theory is mainly in the form of directed and undirected qualitative dependency statements. It is a hierarchy of such dependencies and some crisp rules. The top level of the theory relates the phenomenon of loudness variations to some abstract musical notions by a set of dependencies like

```
depends_on(crescendo(Note,X),
        [stability(Note.Y)]).
depends_on(crescendo(Note,X),
        [goal_directedness(Note,Y)]).
depends_on(crescendo(Note,X),
        [closure(Note.Y)]).
```

The first of these can be paraphrased as "Whether *crescendo should be applied to a note (and it so, the* exact *amount X) depends, among other things, on the musical stability Y of the note,"* and analogously for the other ones.

Abstract notions like stability, goal, directedness, and closure are then again related to lower-level musical effects, all the way down to some surface features of training instances, for example:

```
q+( metrical_strength(Note,X),
    stability(Note.Y)).
q+( harmonic_stability(Note,X),
    stability(Note,Y)).
```

"The degree of stability Y of a note is positively proportionally related to

[3] In standard music terminology, crescendo means an increase of loudness with respect to the previous note. For various reasons, we have adopted a somewhat different definition for our musical system. To the learner this should make no difference.

[4] This project also has a strong music-theoretic component which cannot be discussed here. Readers interested in the musicological aspects are referred to [39].

(among other things) the metrical strength X of the note" etc.

where **metrical_strength** is a numeric and harmonicstability is a symbolic attribute (with a discrete, ordered domain of qualitative values). Both are defined as operational.

Given this domain theory and some played pieces,
the plausible explanation component learns mixed symbolic/numeric rules that discriminate various types of situations where a crescendo or a decrescendo occurs. Each rule describes a particular class of crescendo/decrescendo situations. The IBL component then creates a numeric interpolation table for each such rule and stores the instances associated with the rule. The set of all numeric attributes shared by all the instances covered by a rule defines the dimensions of the respective interpolation table.

6 Two experiments

The system has been tested with pieces from various musical epochs and styles (Bach minuets, Chopin waltzes, even jazz standards). The following sections present two small experiments that show the typical behavior of IBL-SMART.

6.1 Bach minuets

In this experiment, we chose three well-known minuets from J.S.Bach's *Notenbiichlein fiir Anna, Ma.gda.lena Bach* as training and test pieces. The beginnings of the three minuets are shown in Figure 2. All three pieces consist of two parts. The second part of each piece was used for training: they were played on an electronic piano by the author, and recorded through a MIDI interface. After learning, the system was tested on the first parts of the same pieces. In this way, we combined some variation in the training data (three different pieces) with some uniformity in style (three pieces from the same period and with similar characteristics; test data from the same pieces as training data, though different).

The training input consisted in 212 examples (notes), of which 79 were examples of crescendo, and 120 were examples of decrescendo (the rest were played in a neutral way). The system learned 14 rules and, correspondingly, 14 interpolation tables characterizing crescendo situations, and 15 rules for decrescendo. Quite a number of instances were covered by more than one rule. For illustration, here is a simple rule for crescendo:

```
crescendo(Note,X) :-
    metrical_strength(Note,S),
    S > 4.0,
    harmony_stability(Note,high) ,
    previous_interval(Note,Il),
    direction(Il,up),
    next_interval(Note,I2),
    direction(I2,down).
```

"Apply some crescendo to the current note if
the metrical strength of the note is > 4
and the underlying harmony is stable
and the direction of the melody from the previous to the current note is up
and the direction of the melody from the current note to the next is down"

The quality of the learning results is not easy to measure, as there is no precise criterion to decide whether some performance is right or wrong. Judging the correctness is a matter of listening. Unfortunately, we cannot attach a recording to this article so that the reader can appreciate the results. Instead, Figure 3 depicts a part of one of the training pieces (the second part of the first minuet in G major as played by the author), and also shows the performance created by the system for a test piece (the first part of the same minuet) after learning. The figures plot the relative loudness with which the individual notes were played. A level of 1.0 would be neutral, values above 1.0 represent crescendo (increased loudness), values below 1.0 decrescendo.

The reader familiar with standard music notation may appreciate that there are strong similarities in the way similar types of phrases are played by the human teacher and the learner. Note, for instance, the crescendo in lines rising by stepwise motion, and the decrescendo patterns in measures with three quarter notes. Note also the consistent pattern of accents (loud notes) at the beginnings of measures. Given the limited amount of training data, the degree of generalization achieved is quite remarkable. In addition, an inspection of

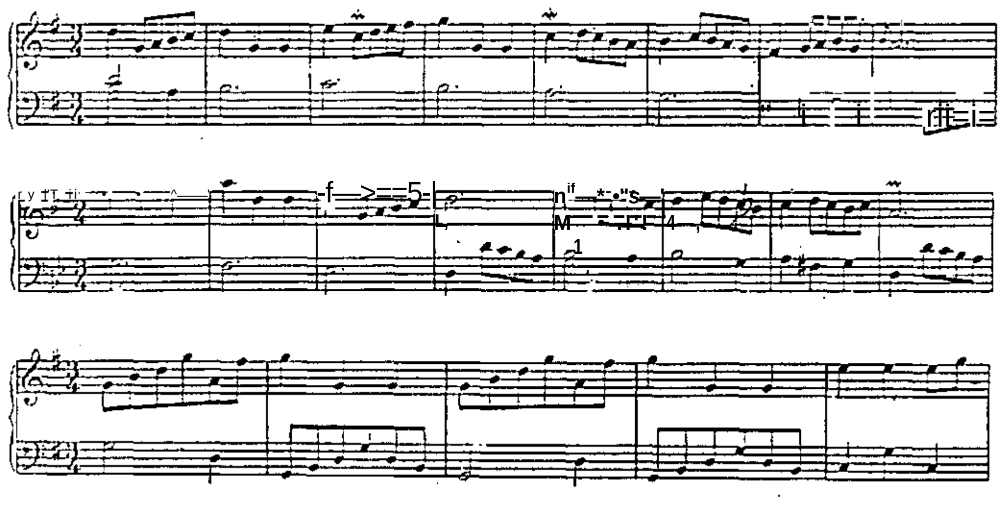


Figure 2: Beginnings of three little minuets by J.S.Bach.

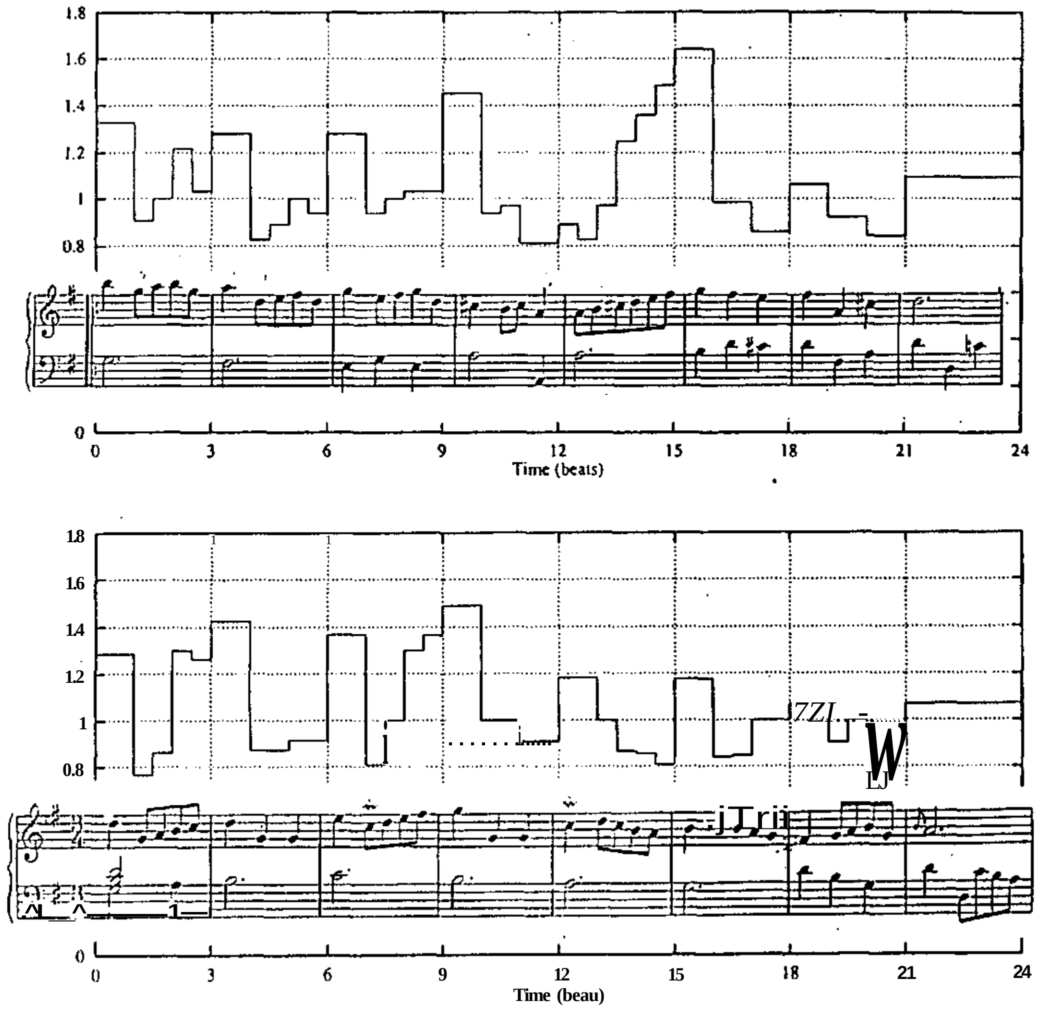


Figure 3: Parts of a training piece as played by teacher (top) and test piece as played by learner after learning (bottom).

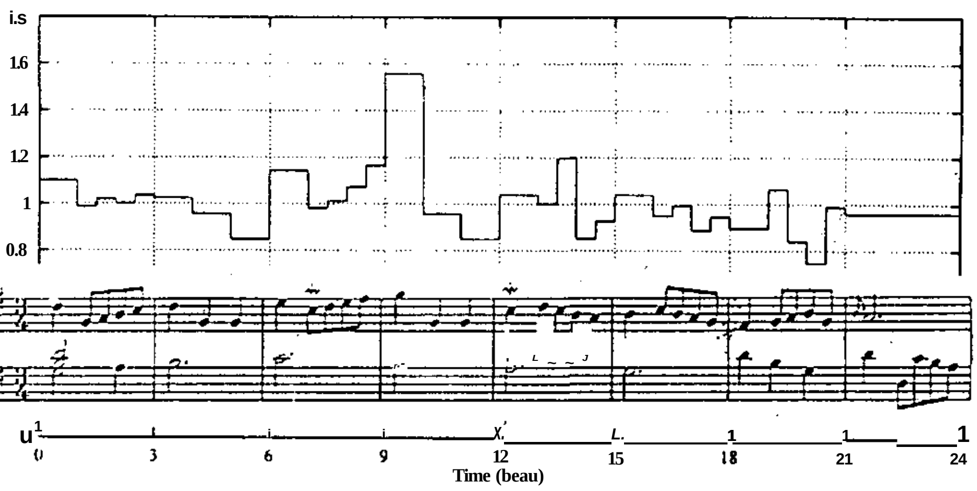


Figure 4: Part of test piece as played after instance-based learning only.

the symbolic rules learned in this experiment revealed that the system had re-discovered some expression principles that had been formulated years ago by music theorists (for more details, see [39]). Results of similar quality were obtained in experiments with music from other eras and styles (again, see [39]).

6.2 The effect of knowledge

In order to verify the importance of the domain theory and generally the superiority of the complete IBL-SMART system, experiments on the same data were also performed with a system restricted to instance-based learning *only*, i.e., without the symbolic explanation component. In these experiments, the system simply used all available training examples with all their attributes (numeric and symbolic) to predict values (decide how to play new melodies) by nearest-neighbor interpolation. The similarity metric mapped all the numeric attributes to a uniform scale between 0 and 1, and for symbolic attributes, the distance was defined to be 0 in the case of a match and 1 otherwise. As not all training instances share all numeric dimensions, the system learned as many interpolation tables as there were combinations of numeric attributes occurring in the training data.

Figure 4 shows the result (the system's performance on the same test piece) after learning from the Bach minuets in this way. In this case, 18 rules for crescendo and 12 for decrescendo were learned. There is a marked deterioration in the resulting performance from learning *with* knowledge (figure 3) to learning *without* knowledge (figure 4). The differences are easily audible (the performance sounds irregular and unfocused, with no clear stress patterns except in measures 3 and 4), but can also be appreciated by looking at the plots: the performance pattern of figure 4 is rather blurred, it lacks the regularity of figure 3. Obviously, the pure IBL learner does not distinguish as well between relevant types of situations, and generalization does not always seem to focus on musically sensible patterns. Similar experiences were made in experiments with other musical data sets.

These results highlight the role of the domain knowledge: the learned symbolic rules have a focusing effect; they cluster those examples (predictors) together in distinct instance spaces that are not just coincidentally similar, but can be explained in similar ways and thus seem to represent significant sub-classes. The effect is more specialized prediction, structured according to domain principles.

7 Discussion and Related Work

In this paper, we have described a general class of numeric learning problems and a learning model that deals with these problems. The model is composed of two parts - a symbolic, knowledge-intensive search algorithm and a numeric, instance-based prediction method. Applying the model to a learning task requires defining abstract sub-concepts T_i, which introduces a natural distinction between a symbolic and a numeric learning problem, and also produces negative instances for the symbolic learner. All the available (qualitative) background knowledge can then be brought to bear on the symbolic learning problem.

The most important advantages of our method are (1) that it can effectively use qualitative domain knowledge for learning precise numeric target functions; (2) that learning symbolic rules for discrete sub-concepts partitions the space for the instance-based prediction component; this results in specialized instance stores and very specific prediction behavior; and (3) as a side-effect, that learning rules for discrete sub-concepts clusters the examples around meaningful abstractions, which may be useful for other tasks.

There is an interesting parallel between our method and the symbolic-numeric clustering algorithm CSNAP that has recently been proposed by Whitehall and Sirag [35]. CSNAP also learns to predict numeric target attributes. It first uses statistical criteria to cluster the examples into disjoint groups, and then searches for *explanations* of these clusters in terms of symbolic descriptions. In that process, instances may be moved between clusters so as to permit more concise or more plausible descriptions. The CSNAP approach is somewhat orthogonal to ours, but is guided by the same motivation: to group the learning examples into clusters that are specialized predictors and at the same time are similar with respect to given background knowledge (or, more generally, amenability to concise symbolic description).

The IBL-SMART learning method is related to a number of other systems described in the literature. An important source of influence was the work of G. DeJong [9, 10] on learning and plausible reasoning. He had presented a system that combined a very weak notion of plausible inference over single cases with interpolation over numeric target variables. Our approach departs from his, among other things, in the variety of types of background knowledge and in the use of a heuristically guided, search-based, multi-instance explanation algorithm that permits much more control over the learning process. Not only does this search introduce a strong notion of *empirical plausibility* by taking into account the distribution of instances; the use of an explicit search heuristic also makes it possible to exploit the qualitative knowledge contained in qualitative dependencies (q+, q-) to compute the relative plausibility of arguments. The best-first search is likely to find explanations that are most plausible overall, both with respect to the knowledge and the data. DeJong's system, on the other hand, simply assumed that the syntactically simplest explanation was also the most plausible one.

Another way of exploiting qualitative relationships similar to q+ and q- (though in a more process-oriented framework) has recently been proposed in [5], where the known direction of the qualitative dependencies is used to severely constrain the search space for an inductive learner.

There is also a close relation between IBL-SMART and the work on plausible justification trees by Tecuci [30]. In both cases, the goal is a fine-grained integration of various types of inference in the framework of creating plausible explanations. IBL-SMART takes a more data-driven approach, in that it learns from all the instances at once and uses the statistical information contained in the data (in addition to the domain knowledge) to guide a heuristic search. As noted, the basic search framework is related to Bergadano and Giordana's method [3].

Generally, we see two main advantages in constructing plausible explanations from whole sets of examples at once. The first is, of course, the *empirical support* afforded by the data. All plausible arguments have an implicit inductive component, so it is really the actual observations that lend more or less support to such arguments and the generalizations that are drawn from them.

A second advantage of the multi-instance explanation approach is that it suggests a natural way to deal with certain types of *noise* in the training data. IBL-SMART implements a simple noise handling mechanism. The search algorithm incorporates two thresholds: the evaluation function accepts only nodes (conjuncts) that cover some minimum number of positive instances, and the termination criterion allows the search to halt when a certain percentage (usually less than 100%) of positive instances are covered. Thus, the system can ignore rare instances that look like exceptions, but are really the result of noise. By varying these thresholds, the system can be tuned to the characteristics of different application domains.[5]

The main disadvantage of the multi-instance explanation approach is its *non-incrementality.* Whether that is a problem depends on the particular application. Also, it is possible that, using techniques similar to those in [37], IBL-SMART could be made to learn incrementally without losing too much in effectiveness.

Finally, remember that the system IBL-SMART presented here is just one particular incarnation of a more general approach. We have found it convenient to use a best-first search algorithm as the basis for the plausible explanation component, as it constructs an explicit a search tree and allows us to integrate various sources of knowledge into the learning process via the search heuristic. However, with appropriate modifications and extensions, other symbolic learners capable of utilizing incomplete and inconsistent knowledge - for instance, FOCL [26] - might be used just as well in this framework.

Similarly, more elaborate strategies could be used in the instance-based component. Aha *et al.* [1, 17] have described a number of instance-based learning methods, both for symbolic classification and for numeric prediction tasks, that could be applied within a framework such as ours. Also, available domain knowledge about the relative degree of relevance of numeric attributes or about the domains and typical values of numeric variables could be used to devise more sophisticated similarity metrics, tailored to the particular application. Given this generality, we may expect the IBL-SMART approach to be applicable to a wide variety of domains.

[5] In fact, the musical experiments described in the previous section were characterized very strongly by noise in the data, originating from the author's imperfect piano technique, from the imprecise boundaries between the abstract sub-concepts crescendo and decrescendoi and from imprecision inherent in the domain itself (there are simply no 100% laws as to how some passage must and will be played; variation will invariably happen). The parameter settings in the experiments reported in section 6 were as follows: each leaf of the tree had to cover at least 5% of the positive training examples, and the search terminated when 80% of the examples were covered.

Acknowledgments

I would like to thank Robert Trappl for his support of this work. This research is sponsored in part by the Austrian *Fonds znr Förderung der Wissenschaftlichen Forschung (FWF)* under grant P8756-TEC. Financial support for the Austrian Research Institute for Artificial Intelligence is provided by the Austrian Federal Ministry for Science and Research.

References

[1] D.W. Aha, D. Kibler and M.K. Albert, Instance-Based Learning Algorithms. *Machine Learning 6(1),* pp. 37-66 (1991).

[2] P.T. Baffes and R.J. Mooney, Symbolic Revision of Theories with M-of-N Rules. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93),* Chambery, France, pp.1135-1142 (1993).

[3] F. Bergadano and A. Giordana, A Knowledge Intensive Approach to Concept Induction. In *Proceedings of the Fifth International Conference on Machine Learning,* Ann Arbor, MI, pp. 305-317 (1988).

[4] F. Bergadano, A. Giordana and S. Ponsero, Deduction in Top-Down Inductive Learning. In *Proceedings of the Sixth International Workshop on Machine Learning,* Ithaca, N.Y., pp. 23-25 (1989).

[5] P. Clark and S. Matwin, Using Qualitative Models to Guide Inductive Learning. In *Proceedings of the 10th International Conference on Machine Learning,* Amherst, MA (1993).

[6] A. Collins and R.S. Michalski, The Logic of Plausible Reasoning: A Core Theory. *Cognitive Science 13(1),* pp. 1-49 (1989).

[7] A.P. Danyluk, Finding New Rules for Incomplete Theories: Explicit Biases for Induction with Contextual Information. In Proceedings *of the Sixth International Workshop on Machine Learning,* Ithaca, N.Y., pp.34-36 (1989).

[8] A.P. Danyluk, An Integration of Analytical and Empirical Learning. In *Proceedings of the First International Workshop on Multistrategy Learning (MSL-91),* Harpers Ferry, W.VA., pp.191-206 (1991).

[9] G. DeJong, Explanation-Based Learning with Plausible Inferencing. In *Proceedings of the Fourth European Working Session on Learning (EWSL-89),* Montpellier, France, pp. 1-10 (1989).

[10] G. DeJong, Explanation-Based Control: An Approach to Reactive Planning in Continuous Domains. In *Proceedings of the DARPA Workshop on Innovative Approaches to Planning, Scheduling, and Control,* San Diego, CA, pp. 325-336 (1990).

[11] G. DeJong and R.J. Mooney, Explanation-based Learning: An Alternative View. *Machine Learning 1,* pp. 145-176 (1986).

[12] T.E. Fawcett, Learning from Plausible Explanations. In *Proceedings of the Sixth International Workshop on Machine Learning,* Ithaca, N.Y., pp. 37-39 (1989).

[13] U. Fayyad and K. Irani, On the Handling of Continuous-Valued Attributes in Decision Tree Generation. Machine *Learning 8(1),* pp. 87-102 (1992).

[14] N.S. Flann and T.G. Dietterich, A Study of Explanation-Based Methods for Inductive Learning. Machine Learning *4(2),* pp. 187-226 (1989).

[15] K.D. Forbus, Qualitative Process Theory. *Artificial Intelligence 24(1-3),* pp. 85-169 (1984).

[16] J. Genest, S. Matwin, and B.Plante, Explanation-Based Learning with Incomplete Theories: A Three-Step Approach. In *Proceedings of the Seventh International Conference on Machine Learning,* Austin, Texas, pp. 286-294 (1990).

[17] D. Kibler, D.W. Aha and M.K. Albert, Instance-based Prediction of Real-valued Attributes. *Computational Intelligence 5,* pp. 51-57 (1989).

[18] S. Mahadevan, Using Determinations in EBL: A Solution to the Incomplete Theory Problem. In *Proceedings of the Sixth International Workshop on Machine Learning,* Ithaca, N.Y., pp. 320-325 (1989).

[19] S. Mahadevan and P. Tadepalli, On the Tractability of Learning from Incomplete Theories. In *Proceedings of the Fifth International Conference on Machine Learning,* Ann Arbor, MI, pp. 235-241 (1988).

[20] R.S. Michalski, A Theory and Methodology of Inductive Learning. In R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds.), Machine *Learning: An Artificial Intelligence Approach, vol. I.* Palo Alto, CA: Tioga (1983).

[21] R.S. Michalski, Inferential Learning Theory as a Conceptual Basis for Multistrategy Learning. Machine *Learning 11(2/3),* Special Issue on Multistrategy Learning, pp. 111-151 (1993).

[22] T.M. Mitchell, R.M. Keller and S.T. Kedar-Cabelli, Explanation-Based Generalization: A Unifying View. Machine *Learning 1(1),* pp. 47-80 (1986).

[23] R.J. Mooney, Induction over the Unexplained: Using Overly-General Domain Theories to Aid Concept Learning. *Machine Learning 10(1),* pp. 79-110 (1993).

[24] R.J. Mooney and D. Ourston, A Multistrategy Approach to Theory Refinement. In *Proceedings of the First International Workshop on Multistrategy Learning (MSL-91),* Harpers Ferry, W.VA., pp. 115-130 (1991).

[25] M.J. Pazzani, Integrated Learning with Incorrect and Incomplete Theories. In *Proceedings of the Fifth International Conference on Machine Learning,* Ann Arbor, MI, pp. 291-297 (1988).

[26] M. Pazzani and D. Kibler, The Utility of Knowledge in Inductive Learning. Machine *Learning 9(1),* pp. 57-94 (1992).

[27] S. Rajamoney S. and G.F. DeJong, The Classification, Detection and Handling of Imperfect Theory Problems. In *Proceedings of the*

10th International Joint Conference on Artificial Intelligence (IJCAI-87), Milano, Italy, pp. 205-207 (1987).

[28] S.J. Russell, *Analogical and Inductive Reasoning.* Ph.D. thesis, Report STAN-CS-87-1150, Stanford University, Stanford, CA (1987).

[29] G. Tecuci, Learning as Understanding the External World. In *Proceedings of the First International Workshop on Multistrategy Learning,* Harper's Ferry, W.VA, pp. 49-64 (1991).

[30] G. Tecuci, Plausible Justification Trees: A Framework for Deep and Dynamics Integration of Learning Strategies. *Machine Learning 11(2/3),* Special Issue on Multistrategy Learning, pp. 237-261 (1993).

[31] G. Tecuci and Y. Kodratoff, Apprenticeship Learning in Imperfect Domain Theories, In Y. Kodratoff and R.S. Michalski (eds.), *Machine Learning: An Artificial Intelligence Approach, vol. Ill,* San Mateo, CA: Morgan Kaufmann, pp. 514-552 (1990).

[32] G. Tecuci and R.S. Michalski, A Method for Multistrategy Task-adaptive Learning Based on Plausible Justifications. In *Proceedings of the Eighth International Workshop on Machine Learning,* Evanston, HI., pp. 549-553 (1991).

[33] K. VanLehn, Learning One Subprocedure per Lesson. *Artificial Intelligence 31(1),* pp. 1-41 (1987).

[34] B.L. Whitehall, S.C. Lu and R.E. Stepp, Theory Completion Using Knowledge-Based Learning. In *Proceedings of the First International Workshop on Multistrategy Learning (MSL-91),* Harpers Ferry, W.VA., pp. 144-159 (1991).

[35] B.L. Whitehall and D.J. Sirag, Conceptual Clustering of Events Using Statistical Split Criteria. In *Proceedings of the Second International Workshop on Multistrategy Learning,* Harper's Ferry, W.VA., pp. 166-179 (1993).

[36] G. Widmer, A Tight Integration of Deductive and Inductive Learning. In *Proceedings of the Sixth International Workshop on Machine Learning,* Ithaca, N.Y., pp. 11-13 (1989).

[37] G. Widmer, An Incremental Version of Bergadano & Giordana's Integrated Learning Strategy. In *Proceedings of the Fourth European Working Session on Learning (EWSL-89),* Montpellier, France, pp. 227-238 (1989).

[38] G. Widmer, Learning with a Qualitative Domain Theory by Means of Plausible Explanations. In R.S. Michalski and G. Tecuci, eds., *Machine Learning: A Multistrategy Approach, vol. IV.* Los Altos, CA: Morgan Kaufmann (1993).

[39] G. Widmer, Modeling the Rational Basis of Musical Expression. *Computer Music Journal* (in press).

[40] D.C. Wilkins, Knowledge Base Refinement as Improving ah Incorrect and Incomplete Domain Theory. In Y. Kodratoff and R.S. Michalski (eds.), *Machine Learning: An Artificial Intelligence Approach, vol. Ill,* San Mateo, CA: Morgan Kaufmann, pp. 493-513 (1990).

EXTENDING THEORY REFINEMENT TO M-of-N RULES

Paul T. Baffes and Raymond J. Mooney
Department of Computer Sciences
University of Texas
Austin, Texas 78712-1188 USA

Keywords: artificial intelligence, multistrategy learning, theory refinement

Edited by: Gheorghe Tecuci

Received: May 26, 1993 **Revised:** October 15, 1993 **Accepted:** October 15, 1993

In recent years, machine learning research has started addressing a problem known as theory refinement. *The goal of a theory refinement learner is to modify an incomplete or incorrect rule base, representing a domain theory, to make it consistent with a set of input training examples. This paper presents a major revision of the* EITHER *propositional theory refinement system. Two issues are discussed. First, we show how run time efficiency can be greatly improved by changing from a exhaustive scheme for computing repairs to* an iterative *greedy method. Second, we show how to extend* EITHER *to refine M-of-N rules. The resulting algorithm,* NEITHER *(New* EITHER^, *is more than an order of magnitude faster and produces significantly more accurate results with theories that fit the M-of-N format. To demonstrate the advantages of* NEITHER, we *present experimental results from two real-world domains.*

1 Introduction

Recently, a number of machine learning systems have been developed that use examples to revise an approximate (incomplete and/or incorrect) domain theory [4, 11, 18, 3, 21, 7]. Most of these systems revise theories composed of strict if-then rules (Horn clauses). However, many concepts are best represented using some form of partial matching or evidence summing, such as M-of-N concepts, which are true if at least M of a set of N specified features are present in an exampie. There has been some work on the indue-tion of M-of-N rules demonstrating the advantages of this representation [17, 9]. Other work has focused on revising rules that have real-valued weights [19, 6]. However, revising theories with simple M-of-N rules has not previously been addressed. Since M-of-N rules are more constrained than rules with real-valued weights, they provide a stronger bias and are easier to comprehend.

This paper presents a major revision of the EITHER propositional theory refinement system a1, 12] that is significantly more efficient and is also capable of revising theories with M-of-N rules. EITHER is inefficient because it computes a potentially exponential number of repairs for each failing example. The new version, NEITHER (New EITHER), computes only the single best repair for each example, and is therefore much more efficient,

Also, because it was restricted to strict Horn-clause theories, the old EITHER algorithm could not produce as accurate results as a neural-network revision system called KBANN on a domain known as the DNA promoter problem [18, 19]. Essentially, this is because some aspects of the promoter concept fit the M-of-N format, Specifically, there are several potential sites where hydrogen bonds can form between the DNA and a protein; if enough of these bonds form, promoter activity can occur. EITHER attempts to learn this concept by forming a separate rule for each poten-' tial configuration by deleting different combinations of antecedents from the initial rules. Since a combinatoric number of such rules is needed to accurately model an M-of-N concept, the generality of the resulting theory is impaired. The new NEITHER algorithm, however, includes the abil-

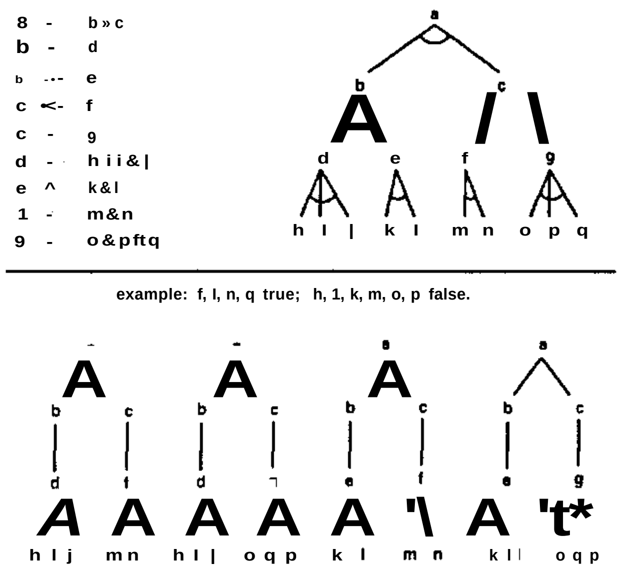


Figure 1: Partial proofs for unprovable positive example. Unprovable antecedents are shown with dotted lines.

ity to modify a theory by changing thresholds of M-of-N rules. Including threshold changes as an alternative method for covering misclassified examples was easily incorporated within the basic EITHER framework.

To demonstrate the advantages of NEITHER, we present experimental results comparing it to EITHER and various other systems on refining the DNA promoter domain theory. NEITHER runs more than an order of magnitude faster than EITHER and produces a significantly more accurate theory with minor revisions that are easy to understand. We also present results showing NEITHER'S ability to repair faults in a theory used to teach the diagnosis of shock to novice nursing students. We show that NEITHER is able to restore significantly damaged theories to near perfect accuracy.

2 Theory Revision Algorithm

2.1 The EITHER Algorithm

The original EITHER theory refinement algorithm has been presented in various levels of detail in [11, 12, 10]. It was designed to revise propositional Horn-clause theories. For EITHER, a *theory* is a set of prepositional Horn-clause rules such as those shown in the top half of Figure 1. Each theory is assumed to function as a classification system whereby *examples* are labeled as belonging to one of a given set of categories. Examples are vectors of feature-value pairs listing the value corresponding to each feature, as well as the category into which the example should be classified. As an illustration, one might imagine a diagnostic theory for determining whether or not a given product coming off an assembly line passes inspection. The categories for such a rule base might be "pass" and "fail" and the examples would consist of whatever measurements could be made on the product as part of the inspection test.

In revising an incorrect theory, note that EITHER can fix either overly-general or overly-specific rules through a combination of deductive, abductive and inductive techniques as shown in Figure 2. An overly-general theory is one that causes an example (called a *failing negative)* to be classified in categories other than its own. EITHER specializes existing antecedents, adds new antecedents, and retracts rules to fix these problems. An overly-specific theory causes an example (called *a, failing positive)* not to be classified in its own category. EITHER retracts and generalizes existing antecedents and learns new rules to fix these problems. Unlike other theory revision systems that perform hill-climbing (and are therefore subject to local maxima), EITHER is guaranteed to fix any arbitrarily incorrect propositional Horn-clause theory [10].

The basic algorithm used by EITHER for both generalization and specialization is shown in the top half of Figure 3. There are three steps. First, *all* possible repairs for each failing example are computed. Next, EITHER enters a loop to compute a subset of these repairs that can be applied to the theory to fix all of the failing examples. This subset is called a *cover.* Repairs are ranked according to a benefit-to-cost ratio that trades off the number of examples covered against the size of the repair and the number of new failing examples it creates. The best repair is added to the cover on each iteration. Lastly, the repairs in the cover are applied to the theory. If the application of a repair over-compensates by creating new failing examples, EITHER passes the covered examples and the new failing examples to an in-

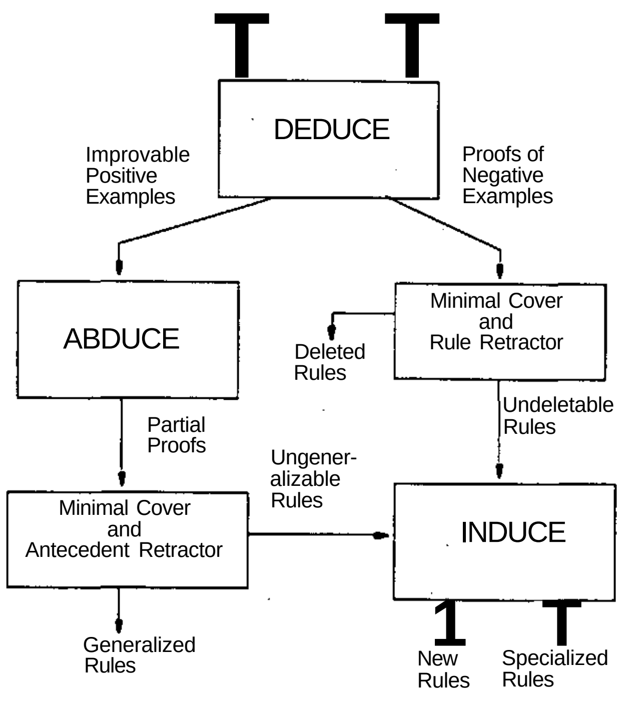


Figure 2: Block diagram of EITHER.

```
EITHER Main Loop

Compute all repairs for each example
While some examples remain uncovered
    Add best repair to cover set
    Remove examples covered by repair
end
Apply repairs in cover set to theory


NEITHER Main Loop

While some examples remain
    Compute a single repair for each
      example
    Apply best repair to theory
    Remove examples fixed by repair
end
```

Figure 3: Comparison of EITHER and NEITHER algorithms.

duction component.[1] The results of the induction are added as a new rule when generalizing or as additional antecedents when specializing.

The time consuming part of this algorithm is the first step where all repairs for a given failing example are found. Figure 1 illustrates this process for theory generalization where EITHER is searching for leaf-rule antecedent deletions to correct failing positive examples. A leaf rule is a rule whose antecedents include an observable or an intermediate concept that is not the consequent of any existing rule. The upper half of the diagram shows an input theory both as rules (on the left) and as an AND-OR graph. The lower half of the diagram shows a hypothetical failing positive example and its partial proofs. A partial proof is one in which some antecedents cannot be satisfied. From these proofs there are four possible repairs which will fix the example, corresponding to the four partial proofs. In each repair, the dotted lines represent antecedents which cannot be proved and must therefore be removed from the given rule(s). Thus, for the leftmost partial proof, h and j cannot be proved in the rule d <— h & **i** & j, and m cannot be proved for rule f <— m & n, so the repair for this partial proof is: delete (h,j ,m) from their respective rules. Likewise, the three other repairs are: delete (h.j.o.p), delete (k,m) and delete (k,o,p). Theory specialization follows a similar process to return sets of leaf-rule *rule* deletions which fix individual failing negative examples.

2.2 Speeding Up EITHER

We have recently implemented a new version of EITHER (NEITHER) that takes a different approach, as shown in the bottom half of Figure 3. Two new algorithms form the basis for the difference between EITHER and NEITHER. First, calculation of repairs is now achieved in linear time. Second, all searches through the theory (for deduction, antecedent deletion and rule deletion) are optimized in NEITHER to operate in linear time by marking the theory to avoid redundant subproofs. NEITHER abandons the notion of searching for all partial proofs in favor of a greedy approach which rapidly selects a *single* best repair for each example. The three steps of the old

[1] EITHER uses a version of ID3 [13] for its induction.

EITHER algorithm can then be integrated into a single loop (see Figure 3).

Rather than computing all partial proofs, NEITHER works bottom-up, constructing a single set of deletions. When multiple options exist, NEITHER alternates between returning the smallest option and returning the union of the options, depending whether the choice involves an AND or OR node. For generalization, deletions are unioned at AND nodes because all unprovable antecedents must be removed to make the rule provable. At OR nodes, only the smallest set of deletions is kept since only one rule need be provable. For specialization, these choices are reversed. Results are unioned at OR nodes to disable all rules which fire for a faulty concept. At AND nodes, the smallest set of rule deletions is selected since any single failure will disable a rule.

To illustrate how repairs are computed in linear time, refer again to Figure 1. The antecedent deletion calculations for this example would begin at the root of the graph, recursively calling nodes b and c. Deletion for node b would then recurse on nodes d and e. Since h, j and k are false, node d returns (h,j) and node e returns (k). When the recursion returns back to node b a choice must be made between the results from nodes d and e because the theory is being generalized and node b is an OR node. Since node e requires fewer deletions, its deletions are chosen as the return value for node b. Recursion for node c follows a similar pattern: node f returns (m), node g returns (o,p) and node c chooses the smaller results from node f as its return value. Finally, nodes b and c return their values to node a. Now, since node a is an AND node and the theory is being generalized, the results from b and c are combined. The final repair returned from node a is delete (k,m). Thus the rule e <— k & 1 is generalized to e «— 1, and the rule f <— m & n is generalized to f <— n.

Note that this algorithm is linear in the size of the theory. No node is visited more than once, and the computation for choosing among potential deletions must traverse the length of each rule at most once. The final repair is also minimum with respect to the various choices made along the way; it is not possible to find a smaller repair that will satisfy the example with the given set of rules. Of course, once a repair is applied to the theory it will effect subsequent repair calculations because the theory will change. Thus although each repair is minimum with respect to the state of the theory from which is was calculated, the total sum of all repairs may not be minimum due to ordering effects. The only way to reach such a global minimum is to do an exhaustive search which is exponential in the size of the theory. This new algorithm trades the complete information available in the partial proofs for speed in computation.

2.3 Adding M-of-N Rules to NEITHER

Expanding NEITHER to handle M-of-N rules involves both a change in the syntax and interpretation of rules as well as a modification to the types of revisions which can be made to a given theory. With M-of-N rules, there are six types of revisions. As before, antecedents may be deleted or rules may be added to generalize the theory, and antecedents may be added or rules deleted to specialize the theory. The two new revisions are to increase or decrease the threshold: decreasing generalizes a rule and increasing specializes it.

To incorporate these two new revisions, NEITHER must be changed in four places. First, the computation of a repair for each failing example must take thresholds into account. For generalization, one need only delete enough antecedents to make the rule provable; there is no need to delete all false antecedents if the rule has a threshold. For example, if the rule for e in Figure 1 had a threshold of 1 there would be no need to delete k to prove this rule. A similar accounting for thresholds is required for computing rule deletions for specialization. Note that during generalization the threshold of each rule from which antecedents are deleted must be decreased by the number of antecedents deleted to account for the smaller size of the rule.

Second, NEITHER must compute threshold repairs. Calculating threshold changes can be done in conjunction with the computation of antecedent and rule deletion repairs since it is directly related to how many of antecedents of a rule are provable. For generalization, we change the threshold to the number of antecedents which are provable. In specialization, we set the threshold to one more than the number of provable antecedents.

Third, a mechanism must be provided for se-

Generalization					Specialization				
change	*resulting rule*	b	c	be	*change*	*resulting rule*	b	c	be
orig. rule	a <- 2 of (b.c)	N	N	Y	orig. rule	a *- 1 of (b,c)	Y	Y	Y
threshold -1	a <- 1 of (b.c)	Y	Y	Y	threshold +1	a <- 2 of (b,c)	N	N	Y
delete b	a ⟵ c	N	Y	Y	delete rule	none	N	N	N

Table 1: Comparison of Revisions.

lecting between a threshold change and a deletion. Effectively, this amounts to deciding which type of revision to try first. The philosophy used in NEITHER is to try the most aggressive changes initially in the hopes that the resulting repair will cover more examples. If the repair creates new failing examples, the less ambitious repairs are tried in turn with induction used as a last resort. During generalization, more radical repairs are those which create more general rules (i.e., rules which can prove more examples). In specialization, the opposite is true. As with EITHER, if all changes result in new failing examples, the algorithm falls back to induction to learn new rules or add new antecedents.

Table 1 compares equivalent threshold and deletion changes for generalization and specialization. The columns labeled with b, c and be indicate whether the corresponding rule will conclude a when just b, just c or both b and c are true. Note that in both cases, the threshold change results in a more general rule. This means that threshold changes should be tried before antecedent deletions during generalization, but tried after rule deletions during specialization. This is because during generalization, the most aggressive changes are those which generalize the most, but during specialization, the most aggressive changes are those which generalize the least.

Fourth and finally, the induction component of NEITHER must be altered slightly to accommodate threshold rules. When the application of a repair causes new failing examples to occur, NEITHER resorts to induction as did EITHER. The result of the induction cannot, however, simply be added to the theory as before. Table 2 illustrates the problem. The original rule shown can be used to prove both the positive and negative examples, and deleting this rule or incrementing its threshold only prevents the positive example from being proved. Assume that induction returns a new feature, d, which can be used to distinguish the two examples (i.e., d is true for the positive example but false for the negative example). Because the original rule has a threshold, adding d directly will still allow both examples to prove the rule. This problem remains even if one tries to increment the threshold in addition to adding d. Instead, the rule must be *split* by renaming the consequent of the original rule, and creating a new rule with the renamed consequent and the results of induction as the new rule's antecedent list.

3 Experimental Results

In the experiments which follow, there are two versions of the NEITHER algorithm. The first has only the speedup changes and is termed simply NEITHER. The second includes M-of-N refinements and is termed NEITHER-MOFN.

3.1 The DNA Promoter Domain

3.1.1 Experimental Design

We tested both NEITHER and NEITHER-MOFN against other classification algorithms using the DNA promoter sequences data set [20]. This data set involves 57 features, 106 examples, and 2 categories. The theory provided with the data set is supposed to recognize *promoters* in strings of nucleotides. A promoter is a genetic region which initiates the first step in the expression of an adjacent gene *transcription.* However, the original theory has an initial classification accuracy of only 50%. We selected this particular data set because the original EITHER algorithm was outperformed by other systems on this data due to the M-of-N qualities required to reason correctly in this domain. In addition to testing EITHER, NEITHER and NEITHER-MOFN, we ran experiments using ID3 [13], backpropagation [16] and RAPTURE [6] (a revision system based on certainty factors). We

example features	*pos. example*	*neg. example*
b, c, d	*b*, -ic, d	***b*, C, -id**
orig. rule	*pos. example*	*neg. example*
a <- 1 of (b,c)	*Y*	*Y*
add to rule	*pos. example*	*neg. example*
a «- 1 of (b.c.d)	*Y*	L **Y**
split rule	*pos. example*	*neg. example*
X «- 1 of (b,c)	Y	*Y*
a <- X,d	Y	N

Table 2: Induced Antecedent Addition.

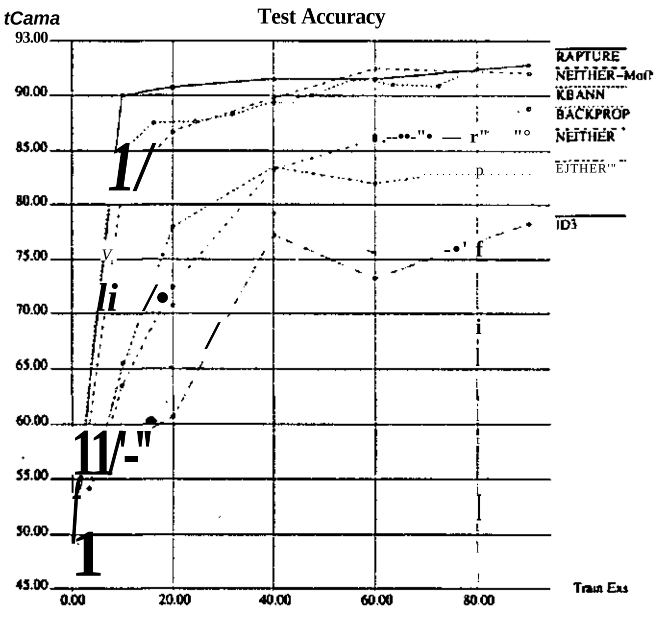


Figure 4: DNA Test Set Accuracy.

also included data on the performance of KBANN on this data set as reported in [20].

The experiments proceeded as follows. Each data set was divided into training and test sets. Training sets were further divided into subsets, so that the algorithms could be evaluated with varying amounts of training data. After training, each system's accuracy was recorded on the test set. To reduce statistical fluctuations, the results of this process of dividing the examples, training, and testing were averaged over 25 runs. The random seeds for the backpropagation algorithm were reset for each run. Training time, and test set accuracy were recorded for each run. Statistical significance was measured using a Student t-test for paired difference of means at the 0.05 level of confidence (i.e., 95% certainty that the differences are not due to random chance).

3.1.2 Results

The results of our experiments are shown in the three graphs of Figures 4, 5 and 6. Figure 4 compares the learning curves of the systems tested, showing how predictive accuracy on the test set changes as a function of the number of training examples. As can be seen NEITHER-MOFN'S performance was significantly better than all other systems except RAPTURE and KBANN.[2] RAPTURE out-performed NEITHER-MOFN with small numbers of training examples but their accuracy was comparable with larger inputs. NEITHER'S accuracy was on par with backpropagation, but was lower than EITHER for small training sets and higher than EITHER for large training sets. Note, that Figure 4 is not direct comparison of NEITHER and KBANN since the results reported were compiled from different subsets of the DNA promoter sequences data set. ID3 had significantly lower accuracy than the other systems.

Figure 5 shows a comparison of training times. Both NEITHER-MOFN and NEITHER were more than an order of magnitude faster than backpropagation and EITHER. Only ID3 ran faster than NEITHER-MOFN.

We also collected data on the average complexity of the revised theories produced by both NEITHER and NEITHER-MOFN. Complexity was measured as the total size; i.e., the total number of all literals in the theory. The results are shown in Figure 6. As can be seen from this graph, NEITHER-MOFN not only produces less complex resulting theories but also produces theories closer in size to the original.

technically, the last difference between backpropagation and NEITHER-MOFN was only significant at the 0.1 level.

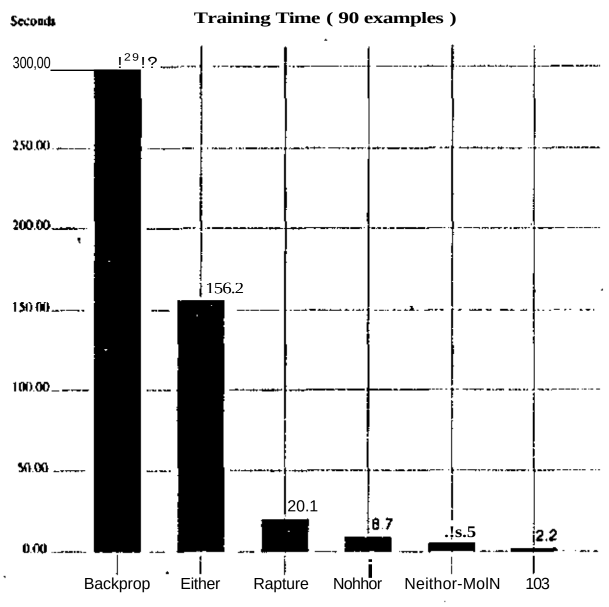


Figure 5: DNA Training Time Comparison.

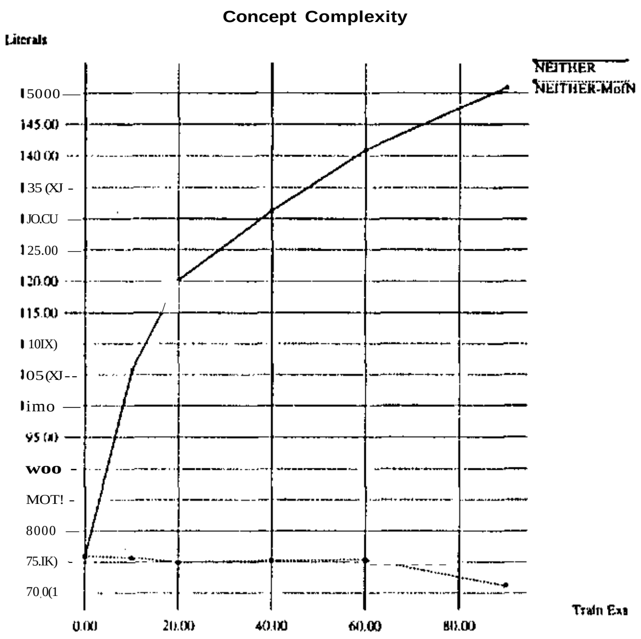


Figure 6: DNA Concept Complexity.

3.1.3 Discussion

Many of our expectations were borne out by the experimental results. Both NEITHER and NEITHER-MOFN ran more than an order of magnitude faster than EITHER due to the optimized algorithms discussed in section 2. NEITHER-MOFN'S increase in accuracy was also expected since the new algorithm is able to concentrate on making M-of-N revisions directly. Also, the fact that NEITHER-MOFN generates less complex theories is not surprising, again because it can directly modify threshold values rather than create new rules. In short, by adding one more operator to the generalization and specialization processes, NEITHER-MOFN is able to accurately revise a theory known to be difficult for symbolic systems, without having to sacrifice the efficiency of a symbolic approach. Finally, the most comparable learning-curve results from [20] would indicate that KBANN'S accuracy in the promoter domain is about the same as NEITHER-MOFN'S.

The most surprising result of the experiments was the difference in accuracy between the original EITHER algorithm and NEITHER. As stated above, EITHER was more accurate with fewer training examples, but its accuracy dropped off relative to NEITHER as the number of examples increased. One possible explanation for this behavior lies in the difference between how the two systems compare potential revisions (see Figure 3). Recall that EITHER computes multiple repairs for each example, but does so only once. NEITHER, by contrast, computes one repair per example each time through its main loop. As a result, with fewer training examples, EITHER has more potential revisions to examine, apparently giving it an edge over NEITHER. Even though NEITHER computes new repairs each time it iterates, there may not be enough iterations to generate as rich a set of deletions as is done in one step by EITHER. On the other hand, as the number of training examples grows, NEITHER undergoes many more iterations, each computing new repairs *in light of any previous revisions.* By contrast, EITHER computes its repairs for each example independently, missing out on any interactions which might occur when the revisions are applied to the theory in a particular order. Capturing these interactions may be one reason NEITHER out-performs EITHER with large numbers of examples.

3.2 The Shock Diagnosis Domain

A second set of experiments was run to test NEITHER'S ability to repair faulty theories. The data for this experiment was borrowed from a separate research project designed to test nursing students retention of concepts for determining if a patient is suffering from shock [8]. Each patient

```
hypovolemic <— shock disrupted-blood-volume
cardiogenic <— shock ineffective-pumping-action
vascular-tone <— shock disrupted-vascular-tone
shock <— 3 of ((pulse rising) (respiration rising)
          (blood-pressure falling) (urine-output low)
          (mental-status strained) skin-abnormal)
shock <— (patient pregnant) (symptoms blood-loss)
          (mental-status strained) skin-abnormal
skin-abnormal <— (skin cool-clamy)
skin-abnormal <— (skin hot-flushed)
disrupted-blood-volume <— (symptoms fluid-loss)
disrupted-blood-volume <— (symptoms blood-loss)
ineffective-pumping-action <— (symptoms cardiac)
disrupted-vascular-tone <— (symptoms infection)
disrupted-vascular-tone <— (symptoms allergy)
disrupted-vascular-tone <— (symptoms neural)
```

Figure 7: Shock Domain Theory.

can be labeled in one of four ways: as suffering from hypovolemic, cardiogenic, or vascular tone shock, or as not in shock. A theory for diagnosing shock was written using the definitions and examples presented to the students and consultations with a medical expert. The final theory is shown in Figure 7.

This data set was chosen for two reasons. First, it represents another real-world domain which has an M-of-N flavor (the "shock" concept in the theory is represented using a threshold rule). Second, NEITHER'S ability to refine theories in this domain is the centerpiece of another of our research efforts in *student modeling* [1]. In short, a student modeling algorithm must be able to recover from a variety of deviations from the correct theory in order to be useful to a variety of students.

3.2.1 Experimental Design

The basic design of this experiment was to introduce faults into the correct shock theory and test how well NEITHER-MOFN could refine the result. The following five alterations were injected into the theory with equal probability: antecedent deletions, antecedent additions, rule deletions, rule additions, and threshold changes. A modification factor was passed to the algorithm which made the alterations indicating the percentage of antecedents in the theory to be changed. Consequently, a modification factor of 0.1 indicated that roughly 10% of the antecedents in the theory would be modified.

Data for training and testing was drawn from a pool of 150 examples equally representative of the three categories of shock. These 150 examples were randomly generated using the correct theory. For each category, 40 positive examples and 10 near-miss negative examples were created. Thus, of the 150 total cases, 120 examples belonged to one of the three categories of shock and 30 were non-shock examples.

A three-phased experiment was run. In each experiment, the 150 examples were first split randomly into 100 training examples and 50 test examples. The original theory was then subjected to three independent modifications of 0.1, 0.2 and 0.3. Each resulting theory was refined by NEITHER-MOFN using the same 100 training examples. The theories were tested both before and after refinement using the same 50 test examples. This entire process was repeated 10 times, and the results averaged. For comparison purposes, we also ran the same training data through a propositional version of the FOIL inductive learner [14] and tested the results using the same test data, averaging the results of the 10 trials.

3.2.2 Results

Table 3 shows the results of the recovery experiments. Note that the results for FOIL are identical for each experiment since induction does not make use of an input theory. In each of the three experiments, NEITHER-MOFN was able to reconstruct a theory to perfect or near perfect accuracy on the test data. NEITHER-MOFN was also able to create more accurate theories than induction alone. Tests were also run using the non-threshold version of NEITHER but the results were nearly identical to those reported for NEITHER-MOFN (there was no statistically significant difference between NEITHER and NEITHER-MOFN on this data).

3.2.3 Discussion

The results of Table 3 are largely what one would expect for a good theory refinement algorithm. As the theory deviated more and more from the original, the performance of the altered theory on the test data continued to decline. Likewise, the ability of NEITHER-MOFN to repair the theory declined with increasingly altered theories, though

	0.1 modified theory	*0.2 modified theory*	*0.3 modified theory*
before refinement	68.25	50.25	43.5
after refinement	100.0	94.0	94.0
induction	80.4	80.4	80.4

Table 3: Shock Test Set Accuracy.

only slightly. Yet in all cases, NEITHER-MOFN was able to refine a wide variety of damaged theories to a high level of performance on novel test data. Additionally, NEITHER-MOFN was able to create more accurate theories than induction alone by taking advantage of input theories which were at least partly correct. Finally, though NEITHER-MOFN was unable to exactly duplicate the original theory in all cases, the refinements made seemed reasonable in light of the alterations made in the modified theories.

4 Related Work

Several researchers have developed methods for inducing M-of-N concepts from scratch. CRLS [17] learns M-of-N rules and out-performed standard rule induction in several medical domains. ID-2-of-3 [9] incorporates M-of-N tests in decision-tree learning and out-performed standard decision-tree induction in a number of domains. Both projects clearly demonstrate the advantages of M-of-N rules.

SEEK2 [5] includes operators for refining M-of-N rules; however, its revision process is heuristic and it is not guaranteed to produce a revised theory that is consistent with all of the training examples. NEITHER uses a greedy covering approach to guarantee that it finds a set of revisions that fix all of the misclassified examples in the training set. Also, unlike NEITHER, SEEK2 cannot learn new rules or add new antecedents to existing rules.

KBANN [19] revises a theory by translating it into a neural network, using backpropagation to refine the weights, and then retranslating the result back into symbolic rules. NEITHER'S symbolic revision process is much more direct and, from all indications, significantly faster. Although KBANN'S results are referred to as M-of-N rules, they actually contain real-valued antecedent weights and therefore are not strictly M-of-N. In addition, KBANN'S revised theories for the promoter problem are also more complex in terms of number of antecedents than the initial theory [20], while NEITHER actually produces a slight reduction. Therefore, NEITHER'S revised theories are less complex and presumably easier to understand. Finally, unlike KBANN, NEITHER is guaranteed to converge to 100% accuracy on the training data.

RAPTURE [6] uses a combination of symbolic and neural-network learning methods to revise a certainty-factor rule base [2]. Consequently, it lies somewhere between NEITHER and KBANN on the symbolic-connectionist dimension. As illustrated in the results, its accuracy on the promoter problem is only slightly superior to NEITHER'S. However, its real-valued certainty factors make its rules more complex.

5 Future Work

The current version of NEITHER needs to be enhanced to handle a number of issues. We need to incorporate a number of advanced features from the original EITHER algorithm, such as constructive induction, modification of higher-level rules, and the ability to handle numerical features and noisy data. Also, we could extend our methods to handle negation as failure and incorporate the ability to handle M-of-N rules into first-order theory revision [15]. The inductive component of NEITHER should be modified to produce threshold rules directly, rather than symbolic rules. Finally, we need to perform a more comprehensive experimental evaluation of the system.

6 Conclusions

This paper has presented an efficient propositional theory refinement system that is capable of revising M-of-N rules. The basic framework is a modification of EITHER [11]; however, the construction of partial proofs has been reduced from exponential to linear time and a method for revising the thresholds of M-of-N rules has been incorporated. The resulting system runs more than an order of magnitude faster and produces significantly more accurate results in domains requiring partial matching, such as the problem of recognizing promoters in DNA.

Acknowledgments

This research was supported by the NASA Graduate Student Researchers Program under grant number NGT-50732, the National Science Foundation under grant IRI-9102926, and a grant from the Texas Advanced Research Program under grant 003658144. We would like to thank Jude Shavlik, Geoff Towell, and Marylin Murphy for generously providing the DNA and Shock data sets. Special thanks also to Chris Whatley for his help implementing NEITHER and to Dr. Thomas Baffes for his invaluable help in developing the rules for the shock domain.

References

[1] P. Baffes and R. J. Mooney. Using theory revision to model students and acquire stereotypical errors. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society,* pages 617-622, Bloomington,IN,1992.

[2] G.G. Buchanan and eds. E.H. Shortliffe. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project.* Addison-Wesley Publishing Co., Reading, MA, 1984.

[3] A. D. Danyluk. Gemini: An integration of analytical and empirical learning. In *Proceedings of the International Workshop on Multistrategy Learning,* pages 191-206, Harper's Ferry, W.Va., Nov. 1991.

[4] A. Ginsberg. Theory reduction, theory revision, and retranslation. In *Proceedings of the Eighth National Conference on Artificial Intelligence,* pages 777-782, Detroit, MI, July 1990.

[5] A. Ginsberg, S. M. Weiss, and P. Politakis. Automatic knowledge based refinement for classification systems. *Artificial Intelligence,* 35:197-226, 1988.

[6] J. J. Mahoney and R. J. Mooney. Combining neural and symbolic learning to revise probabilistic rule bases. In S.J. Hanson, J.C. Cowan, and C.L. Giles, editors, *Advances in Neural Information Processing Systems, Vol. 5,* pages 107-114, San Mateo, CA, 1993. Morgan Kaufman.

[7] S. Matwin and B. Plante. A deductive-inductive method for theory revision. In *Proceedings of the International Workshop on Multistrategy Learning,* pages 160-174, Harper's Ferry, W.Va., Nov. 1991.

[8] Marilyn A. Murphy and Gayle V. Davidson. Computer-based adaptive instruction: Effects of learner control on concept learning. *Journal of Computer-Based Instruction,* 18(2):51-56, 1991.

[9] P. M. Murphy and M. J. Pazzani. ID2-of-3: Constructive induction of M-of-N concepts for discriminators in decision trees. In *Proceedings of the Eighth International Workshop on Machine Learning,* pages 183-187, Evanston, IL, June 1991.

[10] D. Ourston. *Using Explanation-Based and Empirical Methods in Theory Revision.* PhD thesis, University of Texas, Austin, TX, August 1991. Also appears as Artificial Intelligence Laboratory Technical Report AI 91-164.

[11] D. Ourston and R. Mooney. Changing the rules: A comprehensive approach to theory refinement. In *Proceedings of the Eighth National Conference on Artificial Intelligence,* pages 815-820, Detroit, MI, July 1990.

[12] D. Ourston and R. J. Mooney. Theory refinement combining analytical and empirical methods. *Artificial Intelligence,* in press.

[13] J. R. Quinlan. Induction of decision trees. *Machine Learning,* 1(1):81-106, 1986.

[14] J.R. Quinlan. Learning logical definitions from relations. *Machine Learning,* 5(3):239-266, 1990.

[15] B. Richards and R. Mooney. First-order theory revision. In *Proceedings of the Eighth International Workshop on Machine Learning,* pages 447-451, Evanston, IL, June 1991.

[16] D. E. Rumelhart, G. E. Hinton, and J. R. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing, Vol. I,* pages 318-362. MIT Press, Cambridge, MA, 1986.

[17] K. A. Spackman. Learning categorical decision criteria in biomedical domains. In *Proceedings of the Fifth International Conference on Machine Learning,* pages 36-46, Ann Arbor, MI, June 1988.

[18] G. Towell and J. Shavlik. Refining symbolic knowledge using neural networks. In *Proceedings of the International Workshop on Multistrategy Learning,* pages 257-272, Harper's Ferry, W.Va., Nov. 1991.

[19] G. Towell and J. Shavlik. Interpretation of artificial neural networks: Mapping knowledge-based neural networks into rules. In R. Lippmann, J. Moody, and D. Touretzky, editors, *Advances in Neural Information Processing Systems,* volume 4. Morgan Kaufmann, 1992.

[20] G. G. Towell. *Symbolic Knowledge and Neural Networks: Insertion, Refinement, and Extraction.* PhD thesis, University of Wisconsin, Madison, WI, 1991.

[21] B. L. Whitehall, S. C. Lu, and R. E. Stepp. Theory completion using knowledge-based learning. In *Proceedings of the International Workshop on Multistrategy Learning,* pages 144-159, Harper's Ferry, W.Va., Nov. 1991.

MULTITYPE INFERENCE IN MULTISTRATEGY TASK-ADAPTIVE LEARNING: DYNAMIC INTERLACED HIERARCHIES

Michael R. Hieb and Ryszard S. Michalski
Center for Artificial Intelligence
George Mason University, Fairfax, VA
hieb@gmu.edu and michalski@gmu.edu

Keywords: multistrategy learning, inferential theory of learning, knowledge transmutation, generalization, abstraction, similization

Edited by: Gheorghe Tecuci

Received: April 10, 1993 **Revised:** May 26, 1993 **Accepted:** October 16, 1993

Research on multistrategy task-adaptive learning aims at integrating all basic inferential learning strategies—learning by deduction, induction and analogy. The implementation of such a learning system requires a knowledge representation that facilitates performing a multitype inference in a seamlessly integrated fashion. This paper presents an *approach to implementing such multitype inference based on a novel knowledge representation, called Dynamic Interlaced Hierarchies (DIE). DIH integrates ideas from our research on cognitive modeling of human plausible reasoning, the Inferential Theory of Learning, and knowledge visualization. In DIH, knowledge is partitioned into* a "static" *part that represents relatively stable knowledge, and a "dynamic" part that represents knowledge that changes relatively frequently. The static part is organized into type, part, or precedence hierarchies, while the dynamic part consists of traces that link nodes of different hierarchies. By modifying traces in different ways, the system can perform different knowledge transmutations (patterns of inference), such as generalization, abstraction, similization, and their opposites, specialization, concretion and dissimilization, respectively.*

1 Introduction

The development of multistrategy learning systems requires a powerful and easily modifiable knowledge representation that facilitates multitype inference. This is particularly true in the case of *multistrategy task-adaptive learning* (MTL) systems that integrate a whole range of inferential strategies, such as empirical indue-tion, abduction, deduction, plausible deduction, abstraction, and analogy [11, 12, 16, 17]. A MTL system adapts a strategy or a combination of strategies to the *learning task,* defined by the available input knowledge, the learner's background knowledge and the learning goal. A theoretical framework for the development of MTL systems has been presented in [13].

This paper presents basic ideas underlying a knowledge representation proposed for the implementation of a MTL system and its use for implementing multitype inference. This representation, called *Dynamic Interlaced Hierarchies* (DIH), integrates ideas from our research on modeling human plausible inference, the Inferential Theory of Learning and the visualization of knowledge. DIH encompasses many different forms of knowledge p factSj ruleS) dependencies, etc., and facilitates knowledge transmutations, described in the Inferential Theory of Learning (ITL) [13]. This paper shows how Dffl supports several basic patterns of knowledge change (transmutations), such as generalization, abstraction, similization, and their oppositeS) specialization, concretion and dissimilizatioI1) respectively. These operations are performed on DIH traces, which correspond to well-formed predicate logic expressions associated with a degree of belief.

While our previous work has focused on the visualization of attribute-based representations for empirical induction [19], DIH allows the visualization of structural (attributional and relational) representations. The underlying assumption is that the syntactic structure for representing any knowledge should reflect as closely as possible the semantic relationships among the knowledge components, and facilitate knowledge modifications that correspond to the most frequently performed inferences. An early implementation of this idea was in the ADVISE system, which used three forms of knowledge representation: relational tables, networks and rules [14].

The DIH approach assumes that a large part of human conceptual knowledge is organized into varjous hierarchies, primarily type, part and precedence hierarchies (see Section 3 for an explanation). Such hierarchies reflect frequently occurring relationships among knowledge components, and make it easy to perform basic forms of inference.

The initial idea for DIH stems from the core theory of human plausible reasoning [5, 4] The theory presents a formal representation of various plausible inference patterns observed in human reasoning.

DIH is more fully described in [8].

2 Relevant Research

The core theory of Plausible Reasoning presents a system that formalizes various plausible inference patterns and "merit parameters" that affect the certainty of these inferences. This system combines structural aspects of reasoning (determined by knowledge structures) with parametric aspects that represent quantitative belief and other measures affecting the reasoning process.

Various components of the "Logic of Plausible Reasoning" have been implemented in several systems [1, 7, 9]. These implementations used various subsets of the inferences ("statement transforms") described in the core theory to investigate the parametric aspects of the theory. The implementations demonstrated how the core theory of plausible reasoning can be applied to various domains. DIH specifies a broader set of knowledge transmutations in a general and well-defined knowledge representation. These transmutations are part of a framework for both reasoning and learning.

The organization of concepts into various hierarchies has been proposed as a plausible structure for human semantic memory quite early [6]. The WordNet project at Princeton University, directed by George Miller, concerns the implementation of an electronic thesaurus using such a memory structure [3]. WordNet is a very large lexical database with approximately 50,000 different word forms. WordNet divides the lexicon into various categories including nouns, verbs, and modifiers (adjectives and adverbs). Significantly, the nouns are stored in topical hierarchies (both type and part hierarchies), lending support to the DIH representation. However, while WordNet can be used as a source of DIH hierarchies, it does not provide any inferential facilities.

Other relevant research includes the development of the Common Knowledge Representation Language (CKRL), done as part of an ESPRIT project [15]. CKRL offers a language in which knowledge can be exchanged between machine learning tools and it uses the set of most common representation structures and operators. While CKRL's representation for multistrategy learning seeks to integrate the various representations employed by several different learning programs for communication of knowledge between the machine learning tools, our aim is to develop a representation that facilitates an integration of learning and inference processes.

Semantic network knowledge representation systems, such as the KL-ONE family [2], utilize a large network of relationships between concepts, intermixing different relationships. The hierarchies they use are tangled, in which a concept can have more than one parent. As a consequence, implementing knowledge transmutations, e.g., generalization, is not as easy as in DIH. DIH facilitates such transmutations because it uses only single-parent hierarchies, representing a structuring of a set of entities from a certain viewpoint. In DIH, a concept can belong to different hierarchies, reflecting the fact that a given concept (or object) can usually be classified from several different viewpoints.

The design of semantic networks is primarily oriented toward facilitating deductive inference, and is not usually concerned with knowledge visualization. The design of DIH is oriented toward facilitating multitype inference and providing a basis for the visual presentation of knowledge. DIH also utilizes a hierarchy of merit parameters to represent probabilistic factors associated with plausible reasoning.

3 Basic Components of DIH

The theory of plausible reasoning postulates that there are recurring patterns of human plausible inference. To adequately represent these patterns, one needs a proper knowledge representation. The DIH approach partitions knowledge into a "static" part and a "dynamic" part. The static part represents knowledge that is relatively stable (such as established hierarchies of concepts), and the "dynamic" part represents knowledge that changes relatively frequently (such as statements representing new observations or results of reasoning). The static part is organized into *type* hierarchies (TH), *part* hierarchies (PH) and *precedence* hierarchies. Precedence hierarchies include several subclasses, specifically, *measure* hierarchies (MH), *quantification* hierarchies (QH) and *schema* hierarchies (SH). The dynamic part consists of *traces* that represent knowledge involving concepts from different hierarchies. Each trace links nodes of two or more hierarchies and is assigned a degree of belief.

These hierarchies are composed of nodes representing abstract or physical entities, and links representing certain basic relationships among the entities, such as "type-of", "part-of" or "precedes". In the "pure" form, these hierarchies are single parent, that is, no node can have more than one parent. The root node is assigned the name of the class of entities that are organized into the hierarchy from a given viewpoint.

A type (or generalization) hierarchy organizes concepts in a given class according to the "type-of" relation (also called a "generalization" or "kind-of" relation). For example, different types of "animals" can be organized into a "type" hierarchy.

A part hierarchy organizes entities according to a "part-of" relationship. For example, the world, viewed as a system of continents, geographical regions, countries, etc., can be organized into a part hierarchy. While properties of a parent node in the type hierarchy are inherited by children nodes, this does not necessarily hold for a part hierarchy. There are several different part relationships, which include part-component, part-member, part-location and part-substance [18].

To represent relationships among elements of ordered or partially ordered sets, a class of *precedence* hierarchies is introduced. Hierarchies in this class represent hierarchical structures of concepts ordered according to some precedence relation, such as "A precedes B", "A is greater than B", "A has higher rank than B", etc.

There are several subclasses of precedence hierarchies. One subclass is a *measure* hierarchy, in which leafs stand for values of some physical measurement, for example, weight, length, width, etc., and the parent nodes are symbolic labels characterizing ranges of these values, such as "low", "medium", "high", etc. Figure 1 shows a measure hierarchy of possible values of people's height. Dotted lines indicate a continuity of values between nodes. Arrows indicate the precedence order of the nodes. Another subclass hierarchy is a *belief* hierarchy, in which nodes represent degrees of an agent's beliefs in some knowledge represented by a trace.

Other subclasses of precedence hierarchies include a *rank* hierarchy and a *quantification* hierarchy. A rank hierarchy consists of values representing the "rank" of an entity in some structure, e.g., an administrative hierarchy or military hierarchy. A quantification hierarchy consists of nodes that represent different quantifiers for a set (An example is shown in Figure 2). A quantification hierarchy that is frequently used in commonsense reasoning includes such nodes as "one", "some" (both corresponding to the existential quantifier), "most", and "all" (corresponding to the universal quantifier).

Each hierarchy has a heading that specifies its kind (TH, PH, MH, QH or SH) and the underlying concept (or viewpoint) used for the creation of the hierarchy. In addition, the type and part hierarchies also have a *top* node that in the type hierarchies stands for the class of all entities in the hierarchical structure, and in the part hierarchies for the complete object.

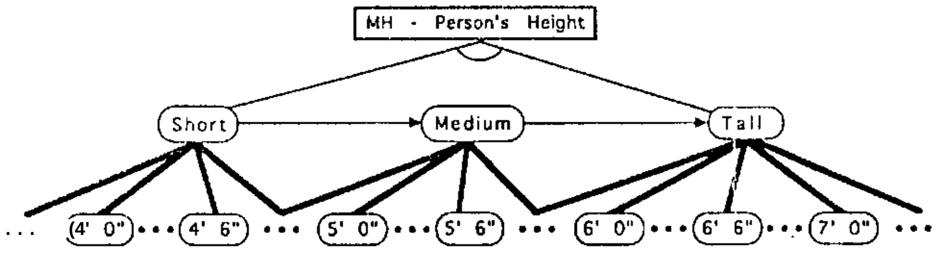


Figure 1: A measure hierarchy of values characterizing people's height.

Schema hierarchies (or schema) are structures that indicate which hierarchies are connected in order to express multi-argument concepts or relationships. For example, the schema hierarchy for the concept of "physical-object" can be <shape, size>. This states that an attribute "shape" applies to any object that is a "physical-object" (a node in the "physical-object" hierarchy), and produces a shape value, which is a node in the "shape" hierarchy. The schema hierarchy for the concept of "giving" may be <giver, receiver, object, time> that states that this concept involves an agent that gives, an agent that receives, an object that is being given, and the time when the "giving" occurs. The agents, object and time are elements of their respective hierarchies.

DIH also makes a distinction between structural and parametric knowledge. The structural knowledge is represented by hierarchies and traces that link nodes of different hierarchies. Parametric knowledge consists of numeric quantities characterizing structural elements of knowledge. In DIH, this knowledge is represented via precedence hierarchies of *merit parameters*. The basic merit parameter is a belief measure that characterizes the "truth" relationship of a given component of knowledge representation (a trace), as estimated by the reasoning agent. Other merit parameters include the *forward* and *backward* strength of a dependency, *frequency, dominance,* etc. [5, 13]. In this paper, we will consider only one merit parameter, namely, the belief measure.

The theory of human plausible reasoning [5] postulates that people rely primarily on the structural knowledge, and resort to parametric knowledge when the "structural" reasoning does not produce a unique result. They resist performing uncertain inferences based on only parametric knowledge, and they are not good at assigning a degree of certainty to a statement based only on the combination of the certainties of its constituents, without taking into consideration the meaning of the whole sentence. A reason for this may be that there does not exist a normative model for reasoning under uncertainty that is independent of the structural aspects of knowledge, i.e., its meaning. Plausible reasoning about a problem or question typically involves both structural and parametric knowledge components.

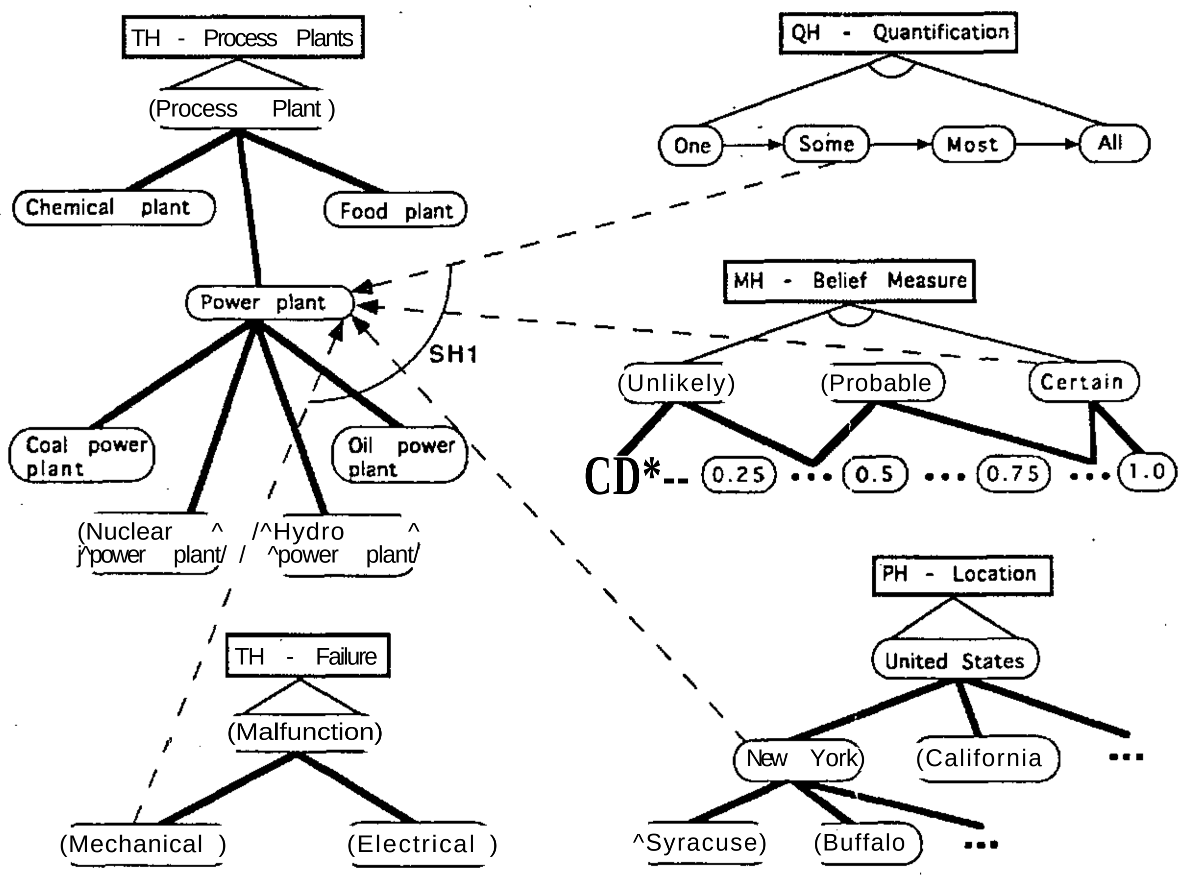
Nodes of a hierarchy are elementary units of the DIH representation. Each node represents some real or abstract entity—a concept, an object, a process, etc. A given entity can be a node in multiple hierarchies, where each hierarchy structures a set of entities from a different viewpoint. The relevant viewpoint is determined by the context of the discourse.

As mentioned earlier, the basic structures in the DIH representation are hierarchies, nodes, traces and schema. Our research on DIH demonstrates that these structures provide a very natural environment for performing basic types of inference on statements. The subsequent sections show how these inferences are performed using the DIH representation.

4 DIH Traces

To describe the DIH knowledge representation, let us start by representing the following statement: "It is certain that some power plants in New York have mechanical failures." Figure 2 presents this statement as a trace connecting nodes of five hierarchies: "Process plants" and "Failure", both type hierarchies; "Quantification", the quantification hierarchy; "Location", a part hierarchy; and "Belief measure" a measure hierarchy.

The interpretation of the trace is done on the basis of the schema hierarchy shown in Figure 3. The schema defines the universe of sentences that can be generated using concepts of these hierarchies, ordered according to the schema.



The trace representing the sentence consists of nodes linked by dotted lines. The arrows in the trace indicate the argument (reference set) that is being described by the sentence. The interpretation of the trace is given by schema hierarchy SH1 in Figure 3.

Figure 2: A DIH trace representing the sentence "It is certain that some power plants in New York have mechanical failures."

The convention for the direction of arrows in a trace is that they point from the nodes denoting descriptive concepts to the *argument* node that stands for the set (or individual) being described, called a *reference set.* In this example, the set being described is "Power plant" in the hierarchy of Process Plants, thus the node representing it is the argument node. Other nodes linked by the trace represent descriptive concepts for the argument node. The belief measure takes values from a belief hierarchy, and refers to the entire trace rather than a single node, which is indicated by the schema.

Using the formalism of the annotated predicate logic [9], this trace can be interpreted as: "(Some)x, [type(x) = Power plant) & [location(x) = New York] & [failure(x) = mechanical]: Belief = 1.0." This statement is a quantified conjunction of several *elementary* statements. An elementary statement expresses one property of the reference node (set), for example, "Location(Power plant) = New York."

In a formal expression of an elementary statement, the reference set ("Power plant") is called an *argument,* the predicate ("Location") is called a *descriptor,* and the value of the descriptor ("New York") is called the *referent.* Thus, an elementary statement is formally expressed in the form "descriptor(argument) = referent".

In Figure 2, the square boxes contain the heading of the hierarchy. The concept specified in the heading is the general descriptor for the hierarchy. The nodes in the hierarchy are possible values of this descriptor.

The schema hierarchy, SHI, in Figure 3 is used for the interpretation of the trace represented in Figure 2. The heading indicates the type of hierarchy (SH: Schema Hierarchy) and the reference

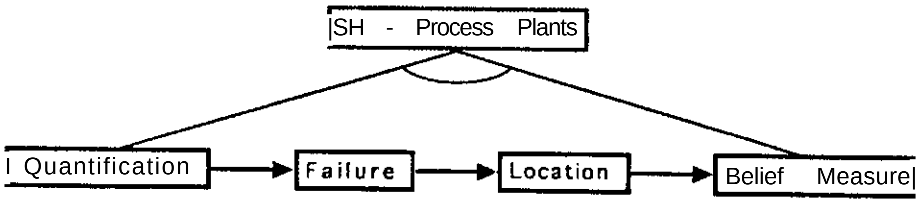


Figure 3: Schema hierarchy SHI.

set of the trace. Since the schema hierarchy is a precedence hierarchy, a valid interpretation of the schema requires each of the descriptors in order. Thus the first element of the trace must be from the quantification hierarchy, the second from the failure hierarchy, the third from the location hierarchy and the last from the hierarchy of belief measures. This schema hierarchy is also utilized for examples in Section 4.

Adding knowledge to the DIH representation is done by creating hierarchies and specifying traces that express statements involving nodes of different hierarchies. To allow proper interpretation of a trace, the schema is also specified by indicating relevant descriptors and their order.

DIH allows one to represent complex forms of knowledge, involving different kinds of quantifiers, multi-argument predicates, different types of logical operations on them, and to associate degrees of belief with individual statements. A more complete description of the DIH representation system is given in [8].

5 Multitype Inference in DIH

The core theory of plausible reasoning introduced in [5] gives four knowledge transmutation operators (also called transforms) P generalization, specialization, similization and dissimilization. The Inferential Theory of Learning [13] specifies several additional operators, of which abstraction and concretion are incorporated into DIH. (In [5] the abstraction and concretion transmutations were called referent generalization and referent specialization, respectively.)

Generalization (specialization) transmutations extend (contract) the reference set. They are done either by argument generalization (specialization) or by quantification generalization (specialization). Argument generalization is accomplished by moving above the node representing the reference set in a type hierarchy. Quantification generalization is accomplished by moving up the quantification hierarchy.

Abstraction (concretion) transmutations decrease (increase) the amount of information about the reference set. A way to accomplish such a transmutation is by moving above the node in the type or part hierarchy that corresponds to a value of some descriptor in the sentence represented by the trace.

Similization (dissimilization) transmutation is done by replacing a node corresponding to the reference set (argument) or a descriptor value (referent) by a node at the same level of hierarchy, which corresponds to a similar (dissimilar) concept within the context of the given hierarchy. In the case of dissimilization, the resulting trace is linked with a negation node, because the generated inference is a negation of the original sentence [13].

These transmutations can be given a simple conceptual interpretation, if one assumes that nodes at each level of hierarchy are ordered by the relation of similarity, that is, nodes that correspond to similar concepts (in the context of the given hierarchy) are located near each other, and nodes that correspond to dissimilar concepts are placed far away from each other. Such an arrangement is natural for precedence hierarchies. In sum, similization and dissimilization transmutations are performed by sideways node movements, while generalization (specialization) and abstraction (concretion) are performed by upward (downward) node movements.

Table 1 lists all the above knowledge transmutations, specifying their abbreviated name, the relevant hierarchies, and the underlying inference type. The relevant hierarchies are the kinds of hierarchies for which the transmutations are valid. The various kinds of part hierarchies are not shown, but are distinguished in DIH. Additional constraints are necessary in some kinds of part hierarchies to maintain the validity of the

Transmutation	Symbol	Relevant Hierarchies	Inference Type
Argument Generalization	AGen	Type, Part	Deductive
Argument Specialization	ASpec	Type, Part	Inductive
Quantification Generalization	QGen	Quantification	Inductive
Quantification Specialization	QSpec	Quantification	Deductive
Abstraction	Abs	Type, Part, Precedence	Deductive
Concretion	Con	Type, Part, Precedence	Inductive
Argument Similization	ASim	Type, Part	Analogical
Argument Dissimilization	ADis	Type, Part	Analogical
Referent Similization	RSim	Type, Part, Precedence	Analogical
Referent Dissimilization	RDis	Type, Part, Precedence	Analogical

Table 1: Basic knowledge generation transmutations.

transmutation.

Figure 4 presents a schematic diagram illustrating how knowledge transmutations modify a trace. A dotted line represents a link in a trace. An arrow means that the trace is moving to a new node in the indicated direction by performing the indicated transmutation. The quantification transmutations operate over the entire trace, rather than on a single node, as do the transformations involving the merit parameters.

One form of generalization transmutation moves a node in the quantification hierarchy upward, another form moves a node (argument) in the type hierarchy upward. The "+" indicates a strengthening of a merit parameter, or the movement of the link to a node that is "higher" in the particular merit parameter measure hierarchy. The "P" indicates a weakening of the merit parameter, or the movement of the link down in the hierarchy.

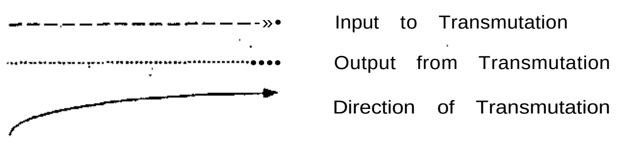
Moving a node in a trace in a manner that corresponds to a deductive inference (Table 1) produces a new trace (statement) with the same truth status as the original trace. In the case of node movement that corresponds to inductive or analogical inference, the smaller the node movement ("perturbation"), the more plausible the resulting inference.

The Argument Generalization transmutation represents a deductive inference. The abstraction operation is also deductive. In contrast, Argument Specialization, Quantification generalization and Concretion are inductive, because they produce traces (statements) that logically entail the original traces (statements).

The above transmutations can be usually done in a number of different ways, by moving to different alternative nodes. The plausibility of the generated statements depends on additional merit parameters, such as dominance, typicality, multiplicity, similarity, frequency, etc. [5]. These issues will be the subject of future research.

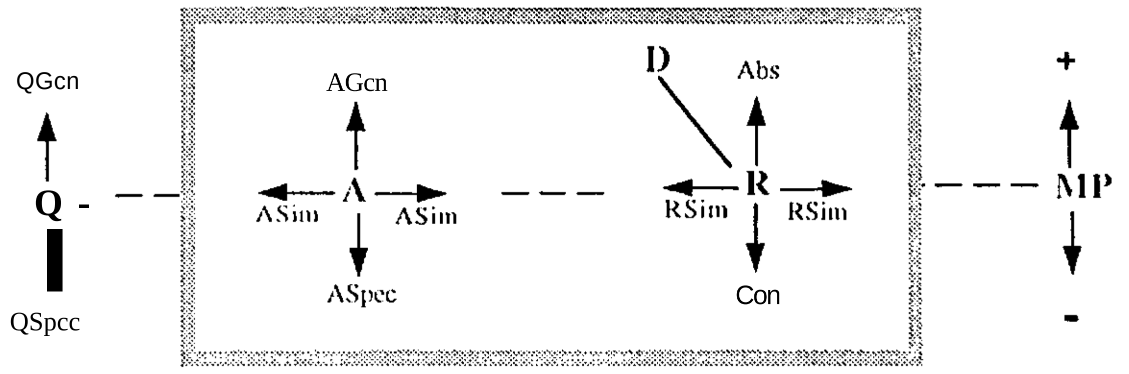
6 Visualizing DIH-Based Inference

This section illustrates several basic transmutations through a series of self-explanatory examples. These examples involve the same original statement, represented as a trace in Figure 2. Given the original statement, these transmutations generate new statements illustrated by DIH traces in Figures 5 through 12.



The legend above is used for interpreting the following figures. The input statement is the same as that of Figure 2, without the belief measure hierarchy. All of the examples are interpreted according to the schema SHI shown in Figure 3.

There are two referents in the input statement. The resulting statements (output) show the results of the given transmutation assuming that there are no merit parameters that assist in the



A	argument (standing for a set that is being described; the reference sel)
R	referent (a value of the descriptor that characterizes the argument)
D	descriptor
Q	quantification
MP	merit parameter (one or more merit parameters)
— —	a link in a trace
⟶	moving a node in the direction of the arrow performs the indicated transmutation.

Figure 4: Diagram of knowledge transmutations in DIH.

specialization or concretion and that the similization operator finds a single "most similar" node using the descriptors given. The Background Knowledge (BK) is the learner's prior knowledge that is relevant to the learning process.

7 MTL-DIH System

The research on DIH aims at developing a representation that will facilitate all basic inferential strategies and knowledge transmutations to be implemented in the multistrategy task-adaptive learning system MTL-DIH.

Although issues related to the implementation of an MTL system are beyond the scope of this paper, we will briefly outline the basic ideas. We have been pursuing two approaches, MTL-JT, which builds a plausible justification tree to "understand" a user's input [17], and a second one, MTL-DIH, based on DIH.

In the MTL-DIH approach, a learning strategy is determined by analyzing the learning task. This analysis relates the input information to the learner's background knowledge and the learning goal. The input information to the system is assumed to be given in the form of logic statements. It can be concept examples, concept descriptions, rules or a combination of the above. The system re- represents the input as a trace, or set of traces. Background knowledge is the part of the learner's prior knowledge that is relevant to the input and the learning goal.

The learning goal specifies criteria characterizing knowledge to be learned. There are different kinds of learning goals, such as to predict new information, to explain the input, to classify a fact or concept instance, to create an abstract description from an operational one or conversely, to create a problem solution or a plan. It is assumed that the learning goal is determined by a teacher or by the control module of the system.

The learning process involves determining the type of relationship between the given input and the background knowledge, and performing a sequence of knowledge transmutations, involving input and background knowledge, to produce knowledge satisfying the learning goal.

8 Summary and Future Research

The DIH knowledge representation presented serves as the basis for implementing multistrategy task-adaptive learning. It builds upon ideas of the Inferential Theory of Learning and the core theory of plausible reasoning. Although it is closely

Input: Siiine power piants in New York have mechanical failures
BK: Indicated hierarchies
Output: All power plants in New York have mechanical failures

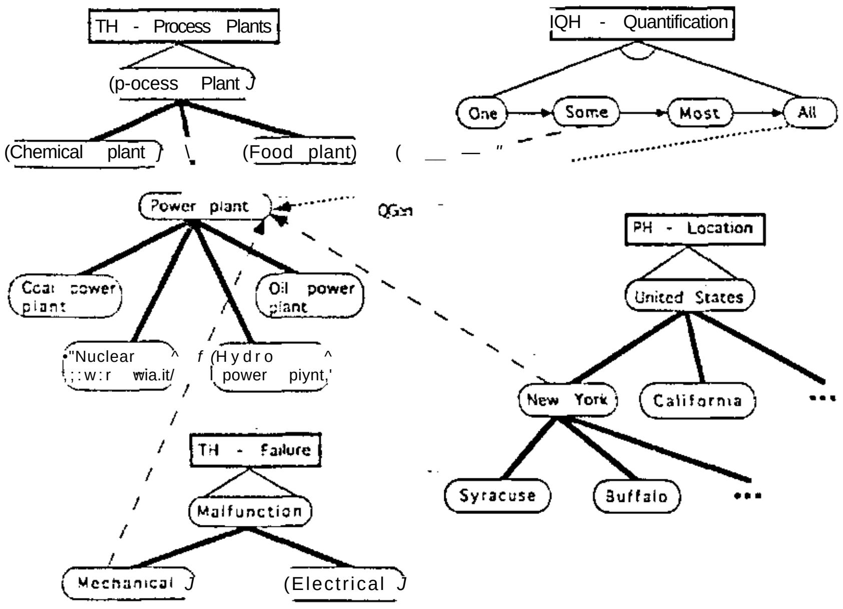


Figure 5: Inductive generalization based on quantification.

Input: Some power plants in New York have mechanical failures
BK: Indicated hierarchies
Output : One power plant in New York has a mechanical failure

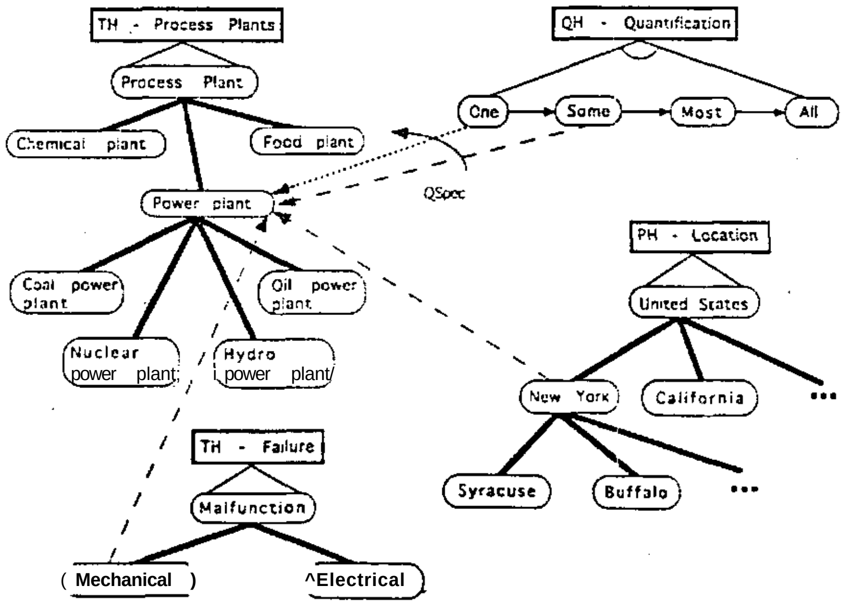


Figure 6: Deductive specialization based on quantification.

Input: Some power plants in New York have mechanical failures
BK: Indicated hierarchies
Output: Some process plants in New York have mechanical failures

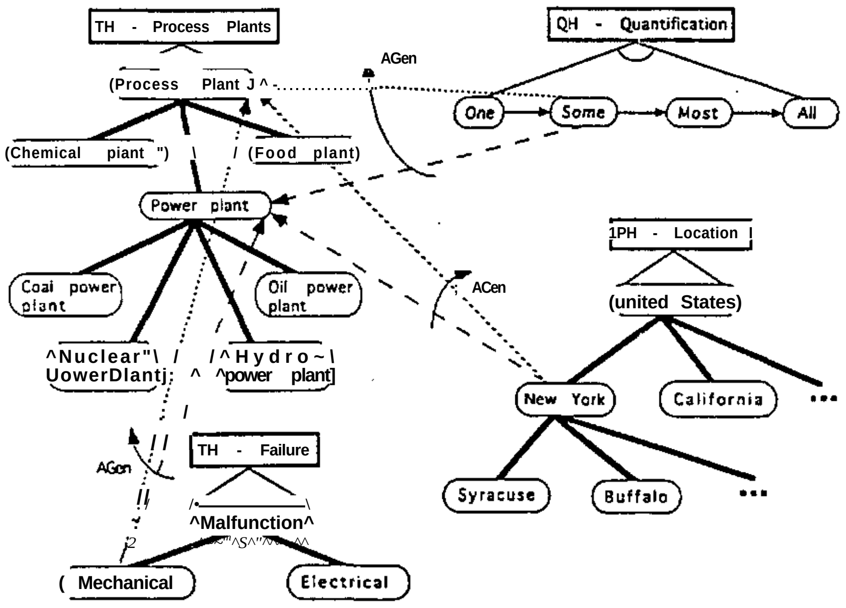


Figure 7: Deductive generalization based on the argument.

Input: Some power plants in New York have mechanical failures
BK: Indicated hierarchies
Output: Some Nuclear power plants in New York have mechanical failures

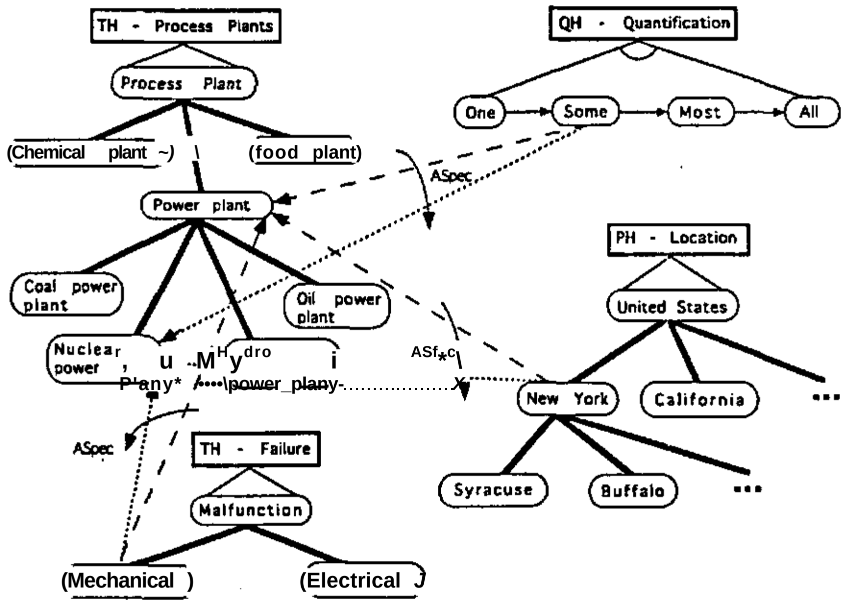


Figure 8: Inductive specialization based on the argument.

Input: Some power plants in New York have mechanical failures
BK: Indicated hierarchies
Output: Some power plants in New York have failures

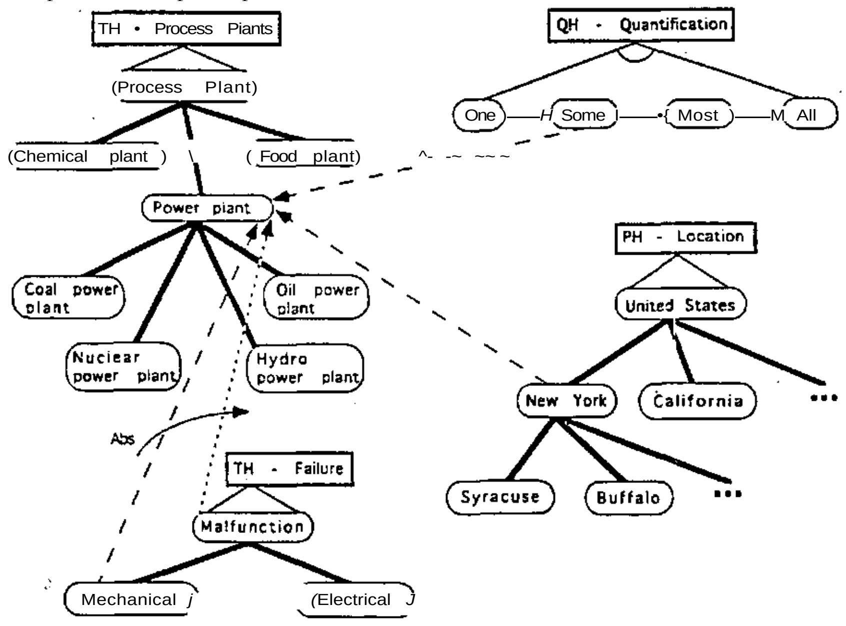


Figure 9: Abstraction transmutation.

Input: Some power plants in New York have mechanical failures
BK: Indicated hierarchies
Output: Some power plants in New York have component failures

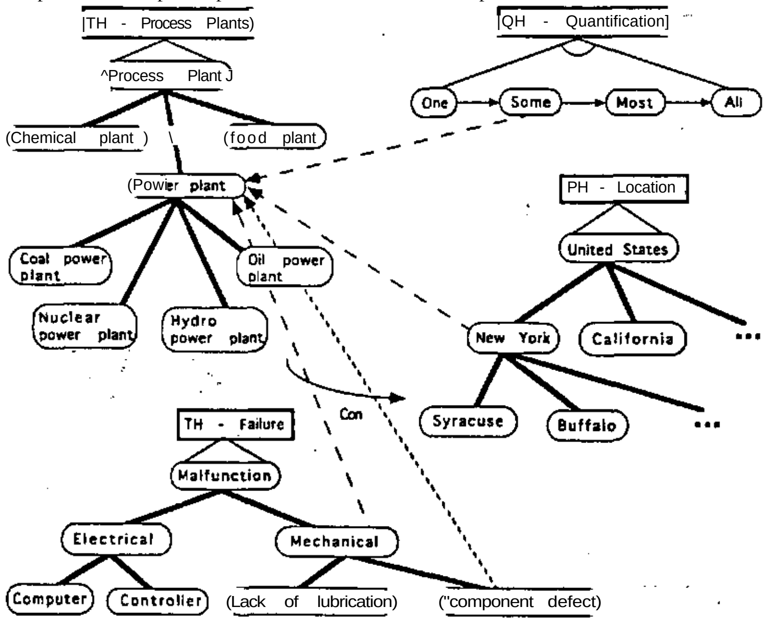


Figure 10: Concretion transmutation.

Input: Some power plants in New York have mechanical failures
BK: Indicated hierarchies

Output: Some chemical plants in New York have mechanical failures

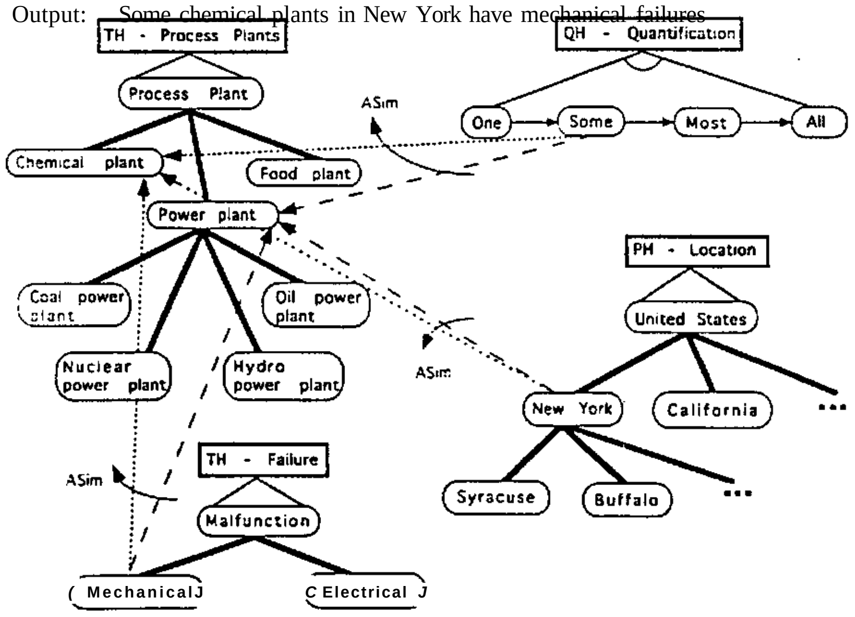


Figure 11: Argument similization transmutation.

Input: Some power plants in New York have mechanical failures
DK: Indicated hierarchies

Output: Some power plants in California have mechanical failures

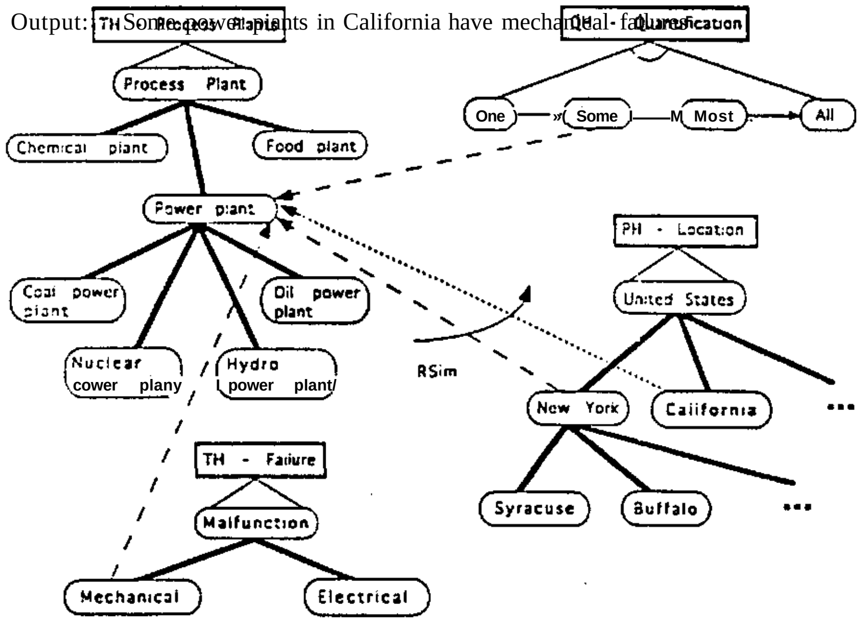


Figure 12: Referent similization transmutation.

related to the semantic network representation, it represents a significantly different approach, and contains many new ideas that make it particularly useful for representing multitype inference. These include the idea of dividing the knowledge representation into a static part and a dynamic part, the organization of knowledge in which basic forms of inference can be performed via simple trace perturbations, and the introduction of various precedence hierarchies, such as the schema hierarchy, the measure hierarchy, and the quantification hierarchy.

The primary purpose of this paper was to demonstrate how DIH supports several basic knowledge generation transmutations, specifically, generalization, specialization, abstraction, concretion, similization and dissimilization. The first version of DIH has been implemented in Smalltalk, and used as a tool for investigating the interactive display and modification of traces in hierarchies. The visual display of inference is particularly useful in situations that involve traces connecting only a few hierarchies (that is, representing short sentences). To facilitate knowledge visualization, the system has an option to present traces with only a limited number of neighboring nodes, rather then connecting complete hierarchies.

In DIH, the more knowledge structures there are *in* background knowledge, the easier it is to assimilate new knowledge, or to plausibly explain input statements. DIH is an efficient, representation, because most knowledge modifications consist of forming or changing traces, without affecting the established hierarchies.

Many issues remain to be addressed in future research. Among these issues are the representation of more complex forms of knowledgeQmutual implications, various types of dependencies, temporal and spatial knowledge, and the development of methods for determining the affect of merit parameters on the reasoning process.

Acknowledgments

The authors thank Eric Bloedorn, Mark Burnstein, David Hille and Ken Kaufman for their thoughtful comments on this paper.

This research was conducted in the Center for Artificial Intelligence at George Mason University. The Center's research is supported in part by the National Science Foundation under grant No. IRI-9020266, in part by the Advanced Research Projects Agency under the grant No. N00014-91- J-1854, administered by the Office of Naval Research and the grant No. F49620-92-J-0549, administered by the Air Force Office of Scientific Research, and in part by the Office of Naval Research under grant No. N00014-91-J-1351.

References

[1] M. Baker, M.H. Burstein, and A.M. Collins, *Implementing a Model of Human Plausible Reasoning.* In Proceedings of the Tenth International Joint Conference of Artificial Intelligence, pp. 185-188, Morgan Kaufman, Los Altos, CA (1987).

[2] R.J. Brachman, D.L. McGuinness, P.F. Patel-Schneider, L.A. Resnick, and A. Borgida, *Living with CLASSIC: When and How to Use* a *KL-ONE-Like Language. In Principles of Semantic Networks - Explorations in the Representation of Knowledge,* edited J.F. Sowa, Morgan Kaufmann, Los Altos, CA (1991).

[3] R. Beckwith, C. Fellbaum, D. Gross, and G.A. Miller, *WordNet: A Lexical Database Organized on Psycholingistic Principles. Using On-line Resources to Build a Lexicon.* edited U. Zernick, Erlbaum, Hillsdale, NJ (1991).

[4] D. Boehm-Davis, K. Dontas, and R.S. Michalski, A *Validation and Exploration of the Collins-Michalski Theory of Plausible Reasoning.* Reports of the Machine Learning and Inference Laboratory, MLI 90-5, Center for Artificial Intelligence, George Mason University, Fairfax, VA (1990).

[5] A. Collins, and R.S. Michalski, The *Logic of Plausible Reasoning: A Core Theory.* Cognitive Science, 13, pp. 1-49 (1989).

[6] A. Collins, and M.R. Quillian, *How To Make A Language User.* In Organization of Memory, edited E. Tulving and W. Donaldson, Academic, New York (1972).

[7] K. Dontas and M. Zemakova, *APPLAUSE: An Implementation of the Collins-Michalski Theory of Plausible Reasoning.* In Proceedings of the Third International Symposium on Methodologies for Intelligent Systems, Torino, Italy (1988).

[8] M.R. Hieb, and R.S. Michalski, A *Knowledge Representation System Based on Dynamic Interlaced Hierarchies: Basic Ideas and Examples.* Reports of the Machine Learning and Inference Laboratory, MLI 93-5, Center for Artificial Intelligence, George Mason University, Fairfax, VA (1993).

[9] J. Kelly, *PRS: A System for Plausible Reasoning.* M.S. Thesis, Department of Computer Science, University of Illinois, Urbana (1988).

[10] R.S. Michalski, A *Theory and Methodology of Inductive Learning.* In Machine Learning: An Artificial Intelligence Approach, edited R. S. Michalski, J. Carbonell and T. Mitchell, TIOGA Publishing Co., Palo Alto, CA, pp. 83-134 (1983).

[11] R.S. Michalski, Towards a *Unified Theory of Learning: Multistrategy Task-adaptive Learning.* Reports of the Machine Learning and Inference Laboratory, MLI 90-1, Center for Artificial Intelligence, George Mason University, Fairfax, VA (1990).

[12] R.S. Michalski, *Inferential Learning Theory as a Basis for Multistrategy Task-Adaptive Learning.* First International Workshop on Multistrategy Learning, pp. 3-18, Harpers Ferry, West Virginia (1991).

[13] R.S. Michalski, *Inferential Theory of Learning: Developing Foundations for Multistrategy Learning.* In Machine Learning: A Multistrategy Approach, Volume 4, edited R.S. Michalski and G. Tecuci, Morgan Kaufmann, Los Altos, CA (1993).

[14] R.S. Michalski, A.B. Baskin, C. Uhrik, T. Channic, S. Borodkin, A.G. Boulanger, L. Rodewald and R.E. Reinke, A *Technical Description of the ADVISE.l Meta-Expert System that Integrates Multiple Knowledge Representations and Learning Capabilities.* ISG 86-8, Department of Computer Science, University of Illinois, Urbana (1986).

[15] K. Morik, K. Causse, and R. Boswell, A *Common Knowledge Representation Integrating Learning Tools.* First International Workshop on Multistrategy Learning, pp. 81-96, Harpers Ferry, West Virginia (1991).

[16] G. Tecuci and R.S. Michalski, A *Method for Multistrategy Task-adaptive Learning Based on Plausible Justifications.* In Machine Learning: Proceedings of the Eighth International Workshop, edited L. Birnbaum and G. Collins, Morgan Kaufmann, Los Altos, CA (1991).

[17] G. Tecuci, *An Inference-Based Framework for Multistrategy Learning. In Machine Learning: A Multistrategy Approach,* Volume 4, edited R.S. Michalski and G. Tecuci, Morgan Kaufmann, Los Altos, CA (1993).

[18] M.E. Winston, R. Chaffin and D. Herrmann, A *Taxonomy of Part- Whole Relations.* Cognitive Science, 11, pp. 417-444 (1987).

[19] J. Wnek and R.S. Michalski, An *Experimental Comparison of Symbolic and Subsymbolic Learning Paradigms: Phase I - Learning Logic-style Concepts.* First International Workshop on Multistrategy Learning, pp. 324-339, Harpers Ferry, West Virginia (1991).

Call for Papers
Distributed and Parallel Real Time Systems
Special Issue of INFORMATICA

Guest Editors:
Marcin Paprzycki, Janusz Zalewski
University of Texas-Perian Basin

We would like to invite papers for the Special Issue of INFORMATICA, An International Journal of Computing and Informatics published in English by the Slovene Society Informatika and the Josef Stefan Institute in Ljubjana, Slovenia.

The scope of the volume will encompass a variety of issues associated with the recent developments in the area of distributed and parallel real-time computing. Papers related to both hardware and software aspects of cuncurrency will be considered. Their focus should be on the timeliness and responsiveness aspects (bounded response time) of respective solutions. Sample topics may include:

- multiprocessor buses and architectures
- real time aspects of local area networks
- message scheduling in distributed systems
- distributed and parallel operating systems
- task allocation and load balancing in real time
- interprocess synchronization and communication for real time
- specification and programming languages
- formal methods in specification and design
- debugging of distributed real-time systems
- designing parallel and distributed applications
- distributed real-time databases
- dependability, realiability and safety in distributed real-time systems
- standardization.

Only previously unpublished work will be accepted for the volume. All papers will be refereed.
Due dates:

*	February 15, 1994	Submission deadline
*	May 1, 1994	Notification of the authors
*	June 1, 1994	Camera-ready versions due

All correspondence and requests for sample copies of INFORMATICA should be addressed to the Guest Editors at the following address:

Dr. Marcin Paprzycki and Dr. Janusz Zalewski
Dept. of Computer Science
University of Texas-Permian Basin
4901 E. University Blvd
Odessa, TX 79762-0001
USA
Phone: (915)367-2310
Fax: (915)367-2115
Email: paprzycki_m®gusher.pb.utexas.edu
zalewski.j(Dutpb.pb.utexas.edu

Call for Papers

18th German Annual Conference on Artificial Intelligence (KI-94)

Saarbriicken, September, 18-23, 1994

General Chairs:
Jörg Siekmann, DFKI, Univ. Saarbriicken
Hans-Jiirgen Biirckert, DFKI Saarbriicken

The scientific, as well as the economic importance of Artificial Intelligence research has steadily grown, even though in some degree, partly slower than had been hoped for. AI research in Germany had a belated start, but steadily improved in recent years. Today the field is well established and some areas even play an outstanding role internationally. This positive development is accounted for in the forthcoming German Conference on AI, which is especially broadly based, aiming at researchers, university students, users and practitioners in industry alike.

The German Conference on Artificial Intelligence is traditionally organised once a year by the Al-section of the German Society for Computer Science (GI). This is one of the four subdivisions of the GI and today the AI section has more than four thousand members. In 1994, the conference will be carried out at the University of Saarland in Saarbriicken by the German Research Center of AI (DFKI), the Computer Science Department (FBI), the Institute of Computer Linguistics (CoLi) and the Max Planck Institute for Computer Science (MPI). It comprises a Scientific Conference, from the 18th to 22nd of September 1994, and an Industrial Congress on the 22nd and 23rd of September 1994. Tutorials will be offered on Sunday the 18th, scientific talks form Tuesday to Thursday in the morning and workshops in the afternoon.

On Monday a symposium will be held, covering a selected topic, the "International Symposium on Logics in Artificial Intelligence", chaired by Dov Gabbay (Imperial College London) and Jörg Siekmann (DFKI, University of Saarland). Invited speakers will give lectures on current issues pertaining to this context. This is prompted by the publication of the multivolume "Handbook of Logics in Artificial Intelligence and Logic Programming".

The Industrial Congress offers a multifaceted programme which demonstrates how AI is put into industrial practice: talks on areas of applications of AI, consulting service offers from the German AI-Institutes' association (AKI) and an AI Exhibition.

Contributions to the Scientific Conference and the Industrial Congress will be published in a set of Springer- Verlag conference publications. KI-94 will be supported by an advisory board consisting of the previous and future chair persons, as well as representatives from Industry and University.

Local Arrangement Committee

Susanne Biundo, DFKI (chair)
Michael Kohlhase, FBI
Brigitte Krenn, CoLi
Christoph Weidenbach, MPI

Conference Office

Coordinator: Reinhard Karger
DFKI Saarbriicken
Stuhlsatzenhausweg 3
D-66123 Saarbriicken
Phone +49 681 302-4444, Fax +49 681 302-5341
e-mail: ki-94<Bdfki.tuii-sb.de

Further information about the KI-94 can be obtained from the conference office.

KI-94 Scientific Conference

Chairs:

Leonie Dreschler-Fischer, Universität Hamburg
Bernhard Nebel, Universität Ulm

The scientific programme consists of invited talks, reviewed paper presentations, tutorials, workshops, poster sessions and system demonstrations. Conference languages are German and English, (simultaneous translation is not available.)

Submissions

Contributions from all aspects of AI are welcome, especially those (but not exclusively) which address the subjects:

— knowledge representation

— knowledge acquisition

— deduction, inference systems, logic programming

— machine learning

—cognition

— architectures, methods, tools

— natural language processing

—robotics

— image processing and image understanding

— intelligent interfaces

—diagnosis

—planning

— configuration

— qualitative reasoning

— neural networks and connectionism

— social impacts

Papers are also welcome, which investigate the principles and problems involved when AI is put into practice. Contributions though, which describe commercial AI applications should be submitted to the Industrial Congress.

Submissions to the scientific program must be original, neither, published or accepted for publication elsewhere, nor currently going through a review process at another conference (with exception of specialised workshops). Accepted submissions will appear in the Springer Lecture Notes in AI and therefore must be written in English.

Posters and System Demos at the Scientific Conference

Research groups and scientists will be given the opportunity through posters and system demos to present current Al-projects and Al-systems. These contributions will be reviewed by the program committee as well, and will appear along with the papers in the LNAI.

Submission Details for the Scientific Conference

Papers and proposals for posters should be sent in six (6) copies to one of the two program chairs (s.f.) till April 8th, 1994. Papers received after that date cannot be considered. Fax or e-mail submissions are also unacceptable.

The papers should be clearly legible and written in 12 point type. The length of a paper should not exceed 16 pages with roughly 38 lines/page and 75 characters per line (corresponding to the standard LaTex article-style), including bibliography and appendix. The title page is separate.

Proposals for posters and system demos require the submision of a one- to three-page summary of the presentation (adding diagramms if necessary).

The title page should show the title of the contribution, the catagory (paper, poster or system demo), the names of all authors with postal and e-mail addresses. In addition, the title page should contain an abstract, of not more than 200 words, supplemented by keywords which characterise the contribution's content.

In order to facilitate the review process, the contents of the title page should be sent till April 8th via e-mail to
ki-94-abstract(8dfki. uni-sb. de
using only plain ASCII text.

Review Process

All submitted papers, proposals for posters and system demos will be reviewed by at least two members of the program committee.

The main criteria underlying the judgment of papers are the achieved results, originality, technical quality and presentation. Proposals for posters and system demos are judged against their Al-context and quality of presentation. The authors will be informed of the program committee's decision by May 20th.

Publication

Accepted papers will be allocated 12 pages in the conference proceedings , which will be published in the Springer LNAI series. Each accepted poster and system demo will have one page reserved in the conference proceedings for a summary of its presentation. A camera-ready copy must be submitted to one of the program chairs by June 17th, 1994.

Deadlines

8/4/94 Submission of contributions
20/5/94 Notification of acceptance
17/6/94 Submission of final, revised copy

Addresses for Submission of Contributions

Prof. Dr. Leoni Dreschler-Fischer Universität Hamburg FB Informatik Bodenstedtstrasse 16 D-22765 Hamburg Phone: + 49 40 4123-6132 (-6128) e-mail: dreschlerfirz.informatik.uni-hamburg.d400.de

KI-94 Workshops

Chairs:

Jiirgen Kunze, HU Berlin
Herbert Stoyan, Universität Erlangen

The Workshops take place in the afternoon on September 20th and 22th. A workshop should discuss current problems in applications of AI and basic research. Its participants will be working on these problems and will share a common interest in an interchange with equally inclined persons or those active in similar areas.

Call for Papers

IEA/AIE-94

The Seventh International Conference on Industrial & Engineering Applications of Artificial Intelligence &: Expert Systems

May 31 - June 3, 1994, The Hyatt Regency on Town Lake, Austin, Texas 78704, USA

General Chair:
Moonis AH, Southwest Texas State University
Program Chair:
Frank Anger, University of West Florida
Program Co-Chair:
Bernard Widrow, Stanford University
Sponsored **by:**
The International Society of Applied Intelligence
Organized in Cooperation with:
ACM/SIGART, American Association for Artificial Intelligence, Institution of Electrical Engineers, IEEE Computer Society, INNS/SIG, Canadian Society for Computational Studies of Intelligence, Institute of Measurement and Control, Japanese Society of Applied Intelligence, Southwest Texas State Univ, European Coordinating Committee for Artificial Intelligence

IEA/AIE-94 continues the tradition of emphasizing applications of artificial intelligence and expert/knowledge-based systems to engineering and industrial problems. Topics of interest include, but are not limited to:

Computer Aided Design/Manufacturing, Dependability & AI/ES, Distributed AI Architectures, Expert & Diagnostic Systems, Intelligent Databases, Intelligent Interfaces, Intelligent Tutoring, Knowledge Acquisition, Knowledge Representation, Machine Learning, Machine Vision, Model-Based *k* Qualitative Reasoning, Natural Language Processing, Neural Networks, Pattern Recognition, Planning *k* Scheduling, Practical Applications, Reasoning Under Uncertainty, Robotics, Sensor Fusion, Intelligent Software Development Tools, System Dependability, Temporal and Spatial Reasoning, Verification & Validation.

Authors are invited to submit four copies of papers, written in English, of up to 10 single-spaced pages, presenting the results of original research or innovative practical applications relevant to one or more of the listed areas of interest. Practical experiences with state-of-the-art AI methodologies are also acceptable when they reflect lessons of unique value to the conference attendees. Shorter works, up to 6 pages, to be presented in 10 minutes, may be submitted as short papers representing work in progress or suggesting possible research directions. (Please indicate "short paper" in the submission letter in this case.) Submissions should be received by the Program Chair by November 5, 1993. Notification of the review process will be made by January 22, 1994, and final copies of papers will be due for inclusion in the conference proceedings by February 22, 1994.

Dr. Moonis Ali
General Chair
Dept. of Computer Science, SW Texas State University, San Marcos, TX 78666-4616, USA
Tele: (+1) 512 245-3409, FAX: (+1) 512 245-3804,
e-mail: maO4<8admin. swt. edu

Dr. Frank D. Anger
Program Chair
Dept. of Comp. Sci., The University of W. Florida, Pensacola, FL 32514, USA
Tele: (+1) 904 474-3022, FAX:(+l)904 474-3129,
email: faQcis.ufl.edu

Dr. Bernard Widrow
Program Co-Chair
Dept. of Elect. Engin.(ISL), Stanford University, Stanford, CA 94305-4055, USA
Tele: (+1) 415 723-4949,
email: widrowQisl. Stanford. edu.

The proceedings will be published and will be available at the conference. Copies of the proceedings of earlier conferences are available - contact:
Gordon and Breach Science Publishers
Customer Service
P.O. Box 786, Cooper Station, New York, NY 10276;
Tel.: 1-800-545-8398 (in USA only), (+1)212-206-8900 Ext. 246, Fax:(+l)212-645-2459

Tutorial Chair:
R. Rodriguez, U W FL
Local Chair:
K. Kaikhah, SW Texas State U
Publicity Chair:
S. Stoecklin, FAMU
Exhibition Chair:
W. Peng, SW Texas State U
Registration Chair:
C. Morriss, SW Texas State U

Announcement and First Call for Papers

The Fourth International Workshop on Inductive Logic Programming (ILP94)

September 12 - 14, 1994
Bad Honnef/Bonn, Germany

General Information

Originating from the intersection of Machine Learning and Logic Programming, Inductive Logic Programming (ILP) is an important and rapidly developing field that focuses on theory, methods, and applications of learning in relational, first-order logic formalisms. ILP94 is the fourth in a series of international workshops designed to bring together developers and users of ILP in a format that allows a detailed exchange of ideas and discussions. Reflecting the growing maturity of the field, ILP94 for the first time will offer a systems and application exhibit as an opportunity to demonstrate the practical results and capabilities of ILP.

Submission of papers

Reflecting the broadening scope of the field, ILP94 invites papers covering on the three main aspects of ILP, namely inductive data analysis and learning in first-order formalisms, inductive synthesis of non-trivial logic programs from examples, and inductive tools for software engineering. Possible topics include, but are not restricted to:

- complexity of learning in logical formalisms
- relationships between ILP and neighboring areas
- higher-order learning
- predicate invention
- learning of integrity constraints
- theory revision and restructuring
- multiple predicate learning
- learning in relational formalisms
- handling of noise
- declarative bias
- architectures for ILP
- comparative analyses of ILP methods
- application discussions

Ideally, papers should fit into one of the following categories:

Theory. Theory papers prove results about a new or known ILP problem or method, discuss the relationship with neighboring fields, or present a unified analysis of several methods.

Methods. Method papers present details of new algorithms, ideally including theoretical and complexity analysis, and empirical results on important applications. Ideally, a method paper would be accompanied by a system demo.

Applications. Application papers describe one or more real-life ILP applications in detail, justifying the use of ILP techniques, and giving a reproducible presentation of experiments and results. Ideally, an application paper would be accompanied by an application demo.

Please submit *four paper copies* of your paper to the workshop chair

Stefan Wrobel
GMD, I3.KI
Schlofi Birlinghoven
53757 Sankt Augustin, Germany.
E-Mail: ilp-94@gmd.de
Fax: +49/2241/14-2889 Tel: +49/2241/14-2670

to be received on or before **May 31, 1994.** There is no fixed page limit on submissions, but length should be reasonable and adequate for the topic. Please use LaTeX if at all possible. Authors will be notified of acceptance or rejection until **July 15, 1994,** and camery-ready copy will be due on **August 9, 1994.**

Program Committee

Francesco Bergadano (Italy)
Ivan Bratko (Slovenia)
Wray Buntine (USA)
William W. Cohen (USA).
Luc de Raedt (Belgium)
Koichi Furukawa (Japan)
Jörg-Uwe Kietz (Germany)
Nada Lavrac (Slovenia)
Stan Matwin (Canada)
Stephen Muggleton (UK)
Celine Rouveirol (France)
Claude Sammut (Australia)

Proceedings

To keep submission dates close to the workshop, accepted papers will be published as a GMD technical report to be distributed at the workshop and officially available to others from GMD afterwards. Publication of an edited book is planned for after the workshop.

Systems and Applications Exhibition

ILP94 offers participants an opportunity to demonstrate their systems and/or applications. Please announce your intention to demo to the conference office until **August 1, 1994,** specifying precisely what type of hardware and software you need.

Location

ILP94 will take place in Bad Honnef, a small resort town close to Bonn in the Rhine valley and adjacent to the Siebengebirge nature park. Participants will be able to take advantage of Bad Honnef's vicinity to medieval castles and of the new wine season that starts at the time of the workshop.

Registration and Conference Office

Please address all correspondence regarding registration to:

Christine Harms
ILP94
c/o GMD
SchloB Birlinghoven
53757 Sankt Augustin, Germany
Tel. +49/2241 14-2473, Fax +49/2241 14-2472 or 2618
E-Mail ilp-94@gmd.de

If you send (preferably by E-Mail) the following information to Christine Harms, you **will** be sent a complete registration brochure as soon as it is available:

Last name:
First name:
Institution:
Zip code, city:
Country:
E-Mail:
Fax:
Intend to submit a paper?

Important Dates

Paper submission deadline:	**May 31, 1994**
Notification of acceptance:	**July 15, 1994**
Demo requests:	**August 1, 1994**
Camera-ready copy due:	**August 9, 1994**
Early registration:	**August 9, 1994**

Workshop: **September 12 - 14, 1994**

THE MINISTRY OF SCIENCE AND TECHNOLOGY OF THE REPUBLIC OF SLOVENIA

The Ministry of Science and Technology also includes the Standards and Metrology Institute of the Republic of Slovenia, and the Industrial Property Protection Office of the Republic of Slovenia.

Scientific Research and Development Potential

The statistical data for 1991 showed that there were 230 research and development institutions, organizations or organizational units in Slovenia, of which 73 were independent, 32 were at the universities, and 23 at medical institutions. The remainder were for the most part departments in industry. Altogether, they employed 13,000 people, of whom 5500 were researchers and 4900 expert or technical staff.

In the past 10 years, the number of researchers has almost doubled: the number of Ph.D. graduates increased from 1100 to 1484, while the number of M.Sc.'s rose from 650 to 1121. The 'Young Researchers' (i.e. postgraduate students) programme has greatly helped towards revitalizing research. The average age of researchers has been brought down to 40, with one-fifth of them being younger than 29.

The table below shows the distribution of researchers according to educational level and fields of research:

	Ph.D.	M.Sc.
Natural Sciences	315	217
Engineering-Technology	308	406
Medical Sciences	262	174
Agricultural Sciences	122	69
Social Sciences	278	187
Humanities	199	68
Total	1484	1121

Financing Research and Development

Statistical estimates indicate that US$ 260 million (1.7% of GNP) was spent on research and development in Slovenia in 1991. Half of this comes from public expenditure, mainly the state budget. In the last three years, R&D expenditure by business organizations has stagnated, a result of the current economic crisis. This crisis has led to the financial decline and increased insolvency of firms and companies. These cannot be replaced by the growing number of mainly small businesses. The shortfall was addressed by increased public-sector R&D spending: its share of GNP doubled from the mid-seventies to 0.86% in 1993.

Overall, public funds available for Research & Development are distributed in the following proportions: basic research (35%), applied research (20%), R&D infrastructure (facilities) (20%) and education (25%).

Research Planning

The Science and Technology Council of the Republic of Slovenia, considering initiatives and suggestions from researchers, research organizations, professional associations and government organizations, is preparing the draft of a national research program (NRP). This includes priority topics for the national research policy in basic and applied research, education of expert staff and equipping institutions with research facilities. The NRP also defines the mechanisms for accelerating scientific, technological and similar development in Slovenia. The government will harmonize the NRP with its general development policy, and submit it first to the parliamentary Committee for Science, Technology and Development and after that to parliament as a whole. Parliament approves the NRP each year, thus setting the basis for deciding the level of public support for RfcD.

The Ministry of Science and Technology provides organizational support for the NRP, but it is mainly a government institution responsible for controlling expenditure of the R&D budget, in compliance with the NRP and the criteria provided by the Law on Research Activities: International quality standards of groups and projects, relevance to social development, economic efficiency and rationality of the project. The Ministry finances research or co-finances development projects through public bidding and partly finances infrastructure research institutions (national institutes), while it directly finances management and top-level science.

The focal points of R&D policy in Slovenia are:

- maintaining the high level and quality of research activities,
- stimulating cooperation between research and industrial institutions,
- (co)financing and tax assistance for companies engaged in technical development and other applied research projects,
- research training and professional development of leading experts,
- close involvement in international research and development projects,
- establishing and operating facilities for the transfer of technology and experience.

In evaluating the programs and projects, and in deciding on financing, the Ministry works closely with expert organizations and Slovene and foreign experts. In doing this, it takes into consideration mainly the opinions of the research leaders and of expert councils consisting of national research coordinators and recognized experts.

The Ministry of Science and Technology of the Republic of Slovenia. Address: Slovenska c. 50, 61000 Ljubljana. Tel. +386 61 131 11 07, Fax +38 61 132 41 40.

JOŽEF STEFAN INSTITUTE

Jožef Stefan (1835-1893) was one of the most prominent physicists of the 19th century. Born to Slovene parents, he obtained his Ph.D. at Vienna University, where he was later Director of the Physics Institute, Vice-President of the Vienna Academy of Sciences and a member of several scientific institutions in Europe. Stefan explored many areas in hydrodynamics, optics, acoustics, electricity, magnetism and the kinetic theory of gases. Among other things, he originated the law that the total radiation from a black body is proportional to the 4th power of its absolute temperature, known as the Stefan-Boltzmann law.

The Jožef Stefan Institute (JSI) is a research organisation for pure and applied research in the natural sciences and technology. Both are closely interconnected in research departments composed of different task teams. Emphasis in basic research is given to the development and education of young scientists, while applied research and development serve for the transfer of advanced knowledge, contributing to the development of the national economy and society in general.

At present the Institute, with a total of about 800 staff, has 500 researchers, about 250 of whom are postgraduates, over 200 of whom have doctorates (Ph.D.), and around 150 of whom have permanent professorships or temporary teaching assignments at the Universities.

In view of its activities and status, the JSI plays the role of a national institute, complementing the role of the universities and bridging the gap between basic science and applications.

Research at the JSI includes the following major fields: physics; chemistry; electronics, informatics and computer sciences; biochemistry; ecology; reactor technology; applied mathematics. Most of the activities are more or less closely connected to information sciences, in particular computer sciences, artificial intelligence, language and speech technologies, computer-aided design, computer architectures, biocybernetics and robotics, computer automation and control, professional electronics, digital communications and networks, and applied mathematics.

The Institute is located in Ljubljana, the capital of the independent state of Slovenia (or SQnia). The capital today is considered a crossroad between East, West and Mediterranean Europe, offering excellent productive capabilities and solid business opportunities, with strong international connections. Ljubljana is connected to important centers such as Prague, Budapest, Vienna, Zagreb, Milan, Rome, Monaco, Nice, Bern and Munich, all within a radius of 600 km.

In the last year on the site of the Jožef Stefan Institute, the Technology park "Ljubljana" has been proposed as part of the national strategy for technological development to foster synergies between research and industry, to promote joint ventures between university bodies, research institutes and innovative industry, to act as an incubator for high-tech initiatives and to accelerate the development cycle of innovative products.

At the present time, part of the Institute is being reorganized into several high-tech units supported by and connected within the Technology park at the "Jožef Stefan" Institute, established as the beginning of a regional Technology park "Ljubljana". The project is being developed at a particularly historical moment, characterized by the process of state reorganisation, privatisation and private initiative. The national Technology Park will take the form of a shareholding company and will host an independent venture-capital institution.

The promoters and operational entities of the project are the Republic of Slovenia, Ministry of Science and Technology and the Jožef Stefan Institute. The framework of the operation also includes the University of Ljubljana, the National Institute of Chemistry, the Institute for Electronics and Vacuum Technology and the Institute for Materials and Construction Research among others. In addition, the project is supported by the Ministry of Economic Relations and Development, the National Chamber of Economy and the City of Ljubljana.

Jožef Stefan Institute
Jamova 39, 61000 Ljubljana, Slovenia
Tel.:+386 61 1259 199, Fax.:+386 61 219 385
Tlx.:31 296 JOSTIN SI
E-mail: matjaz.gams@ijs.si
Contact person for the Park: Iztok Lesjak, M.Sc.
Public relations: Ines Černe

CONTENTS OF INFORMATICA, Volume 17 (1993) pp. 1-420

Articles

Dear Subscribers and Collaborators of Informatica

Within the new international issuing and editing, journal *Informatica* terminates this volume by the fourth number. The year 1993 was aimed to the promotion of the journal in the broader European and global regions. With this intention we have distributed *Informatica* to different places and, simultaneously, we have appealed for an active collaboration in the form of. submitting of papers, editing and personal promoting of the journal in the respective professional communities.

Since we intend to settle the payment of subscription and the list of subscribers in the coming year, we ask the readers to inform us upon the subscription status or just fill out the attached order-form for the journal *Informatica* and fill out the questionary.

Please send your order-form and questionary to the address:

Dr. Rudi Murn
Jožef Stefan Institute
Jamova c. 29
61111 Ljubljana, Slovenia

INFORMATION FOR CONTRIBUTORS

1 Submissions and Refereeing

Please submit three copies of the manuscript with good copies of the figures and photographs to one of the editors from the Editorial Board or to the Contact Person. At least two referees outside the author's country will examine it, and they are invited to make as many remarks as possible directly on the manuscript, from typing errors to global philosophical disagreements. The chosen editor will send the author copies with remarks. If the paper is accepted, the editor will also send copies to the Contact Person. The Executive Board will inform the author that the paper has been accepted, in which case it will be published within one year of receipt of the original figures on separate sheets and the text on an IBM PC DOS floppy disk or by e-mail - both in ASCII and the Informatica IMgX format. Style (attached) and examples of papers can be obtained by e-mail from the Contact Person.

2 News, letters, opinions

Opinions, news, calls for conferences, calls for papers, etc. should be sent directly to the Contact Person.

```
\documentstyle[twoside,informat]{article}

\begin{document}
  \title{TITLE OF THE ARTICLE}
   \author{First Author \\ Address \\ E-mail \\
      AHD \\
      Second Author \\ Address \\ E-mail}
  \titleodd{TITLE II HEADER}
  \authoreven{Author's name in header}
  \keywords{article keywords}
  \edited{editor in charge}
  \received{date}
  \revised{date}
  \accepted{date}
  \abstract{abstract — around 200 words}
  \maketitle

  \section{Introduction}
    Text of Introduction.

  \section{Subject}
    Text of the Subject.

  \begin{figure}
    An example of a figure over 1 column.
    \caption{Caption of the figure 1}
  \end{figure*}

  \begin{figure*}
    An example of a figure over 2 columns.
    \caption{Caption of the figure 2}
  \end{figure*}

  \begin{thebibliography}{99}
    \bibitem{} First Author: {\sl Title},
      Magazine, Vol. 1, No. 1.
  \end{thebibliography}

\end{document}
```

```
\def\journal{Informatica {\bf 17} page xxx—yyy}
\immediate\writel6{'Informatica' VI.2  byB."Z}
\newif\iftitle \titlefalse
\hoffset=llmm \voffset=8mm
\oddsidemargin=-21mm \evensidemargin=-14mm
\t opmargin=-33mm
\headheight=17mm \headsep=10mm
\footheight=8.4mm \footskip=52.5mm
\textheight=242mm \textwidth=170mm
\columnsep=5mm \columnseprule=Opt
\tHocolumn \sloppy \flushbottom
\parindent lem
Meftmargini 2em \leftmargin\leftmargini
Meftmarginv . 5em \leftmarginvi . 5em
\def\labelitemi{\bf —} \def\labelitemii-[~}
\def\labelitemiii{—}
\setcounter{secnumdepth}{3}
\def\maketitle{\twocolumn [/.
  \vbox{\hsize=\textwidth\Large\bf\raggedright
  \uppercase{\8title}}\vss\bigskip\bigskip \vbox{
  \hsize=\textwidth \8author}\bigskip\smallskip
  \vbox{\hsize=\textwidth {\bf Keywords:}
    \8keywords}
  \bigskip \hbox{{\bf Edited by:} \8edited}
  \smallskip
  \hbox{{\bf Received:}
    \hbox to 10em{ \Qreceived\hss}
    {\bf Revised:}\hbox to 10em{ \9revised\hss}
    {\bf Accepted:}\hbox to 10em{ \8accepted\hss}}
  \bigskip \vbox{\hsize=\textwidth
    \leftskip=3em \rightskip=3em \sl \8abstract}
  \bigskip\bigskip]\t itletrue}
\def\maketitleauthor{\twocolumn[y,
  \vbox{\hsize=\textwidth \Large\bf\raggedright
  \«title}\vss \bigskip\bigskip
  \vbox{\hsize=\textwidth \8author}
  \bigskip\bigskip] \gdef\8title{}\titletrue}
\def\makeonlytitle{\twocolumn['/,
  \vbox{\hsize=\textwidth \Large\bf\raggedright
  \8title}\vss\bigskip\bigskip]
  \gdef\8title{}\titletrue}
\def\8title{} \def\8author{}
\def\8titleHO \def\8authorH{}
\def\8keywords{} \def\fiedited{} \def\8abstract{}
\def\8received{} \def\Srevised{} \def\8accepted{}
\def\authoreventl{\gdef\8authorH{fl}}
\def\titleodd#l{\gdef\8titleH{\uppercase{#l}}}
\def\keysords#l{\gdef\8keywords{#l}}
\def\editedtl{\gdef\aedited{#l}}
\def\received#l{\gdef\8received{#l}}
\def\revised#l{\gdef\8revised{*l}}
\def\acceptedtl{\gdef\8accepted{»l}}
\long\def\abstracttl{\gdef\8abstract{»l}}
\def\section{\8startsection {section}
  {I}{\z8}{-3.5ex plus -lex minus -.2ex}
  {2.3ex plus .2ex}{\Large\bf\raggedright}}
\def\subsection{\8startsection{subsection}
  {2}{\z8}{-3.25ex plus -lex minus -.2ex}
  {1.5ex plus .2ex}{\large\bf\raggedright}}
\def\subsubsection{\8startsection{subsubsection}
  {3}{\z8}{-3.25ex plus -lex minus -.2ex}
  {l.Sex plus .2ex}{\normalsize\bf\raggedright}}
\def\8evenhead{\hbox to 3em{\bf\thepage\hss}
  {\small\journal\hfil \iftitle\else
  \8authorH \fi}\global\titlefalse}
\def\8oddhead{{\small\iftitle\else \8titleH \fi
  \hfil\journal}\hbox to 3em{\hss\bf\thepage}
  \global\titlefalse}
\def\«evenfoot{\hfil} \def\9oddfoot{\hfil}
\endinput
```

REVIEW REPORT

Basic Instructions

Informatica publishes scientific papers accepted by at least two referees outside the author's country. Each author should submit three copies of the manuscript with good copies of the figures and photographs to one of the editors from the Editorial Board or to the Contact Person. Editing and refereeing are distributed. Each editor can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. The names of the referees should not be revealed to the authors under any circumstances. The names of referees will appear in the Refereeing Board. Each paper bears the name of the editor who appointed the referees.

It is highly recommended that each referee writes as **many remarks as possible directly on the manuscript,** ranging from typing errors to global philosophical disagreements. The chosen editor will send the author copies with remarks, and if accepted also to the Contact Person with the accompanying completed Review Reports. The Executive Board will inform the author that the paper is accepted, meaning that it will be published in less than one year after receiving original figures on separate sheets and the text on an IBM PC DOS floppy disk or through e-mail - both in ASCII and the Informatica LaTeX format. Style and examples of papers can be obtained through e-mail from the Contact Person.

Opinions, news, calls for conferences, calls for papers, etc. should be sent directly to the Contact Person.

Date Sent:

Date to be Returned:

Name and Country of Referee:

Name of Editor:

Title:

Authors:

Additional Remarks:

All boxes should be filled with numbers 1-10 with 10 as the highest rated.

The final mark (recommendation) consists of two orthogonal assessments: scientific quality and readability. The readability mark is based on the estimated perception of average reader with faculty education in computer science and informatics. It consists of four subfields, representing if the article is interesting for large audience (interesting), if its scope and approach is enough general (generality), and presentation and language. Therefore, very specific articles with high scientific quality should have approximately similar recommendation as general articles about scientific and educational viewpoints related to computer science and informatics.

☐ SCIENTIFIC QUALITY

☐ Originality

☐ Significance

☐ Relevance

☐ Soundness

☐ Presentation

☐ READABILITY

☐ Interesting

☐ Generality

☐ Presentation

☐ Language

☐ FINAL RECOMMENDATION

☐ Highly recommended

☐ Accept without changes

☐ Accept with minor changes

☐ Accept with major changes

☐ Author should prepare a major revision

☐ Reject

INFORMATICA
AN INTERNATIONAL JOURNAL OF COMPUTING AND INFORMATICS
INVITATION, COOPERATION

Since 1977, Informatica has been a major Slovenian scientific journal of computing and informatics, including telecommunications, automation and other related areas. In its 17th year, it is becoming truly international, although it still remains connected to Central Europe. The basic aim of Informatica is to impose intellectual values (science, engineering) in a distributed organisation.

Informatica is a journal primarily covering the European computer science and informatics community - scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the Refereeing Board.

Informatica is free of charge for major scientific, educational and governmental institutions. Others should subscribe (institutional rate 50 DM, individual rate 25 DM, and for students 10 DM). Send a check for the appropriate amount to Slovene Society Informatica, Vožarski pot 12, Ljubljana, account no. 50100-620-133-27620-5159/4.

Please, return this questionnaire.

QUESTIONNAIRE

☐ Send Informatica free of charge

☐ Yes, we subscribe

☐ We intend to cooperate (describe):

☐ Proposals for improvements (describe):

Informatica is published in one volume (4 issues) per year.

If possible, send one copy of Informatica to the local library.

Please, complete the order form and send it to Dr. Rudi Murn, Informatica, Institut Jožef Stefan, Jamova 39, 61111 Ljubljana, Slovenia.

ORDER FORM - INFORMATICA

Name: ....

Title and Profession (optional): ....

....

Home Address and Telephone (optional): ....

....

Office Address and Telephone (optional): ....

....

E-mail Address (optional): ....

Signature and Date: ....

EDITORIAL BOARDS, PUBLISHING COUNCIL

Informatica is a journal primarily covering the European computer science and informatics community; scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor from the Editorial Board can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the Refereeing Board. Each paper bears the name of the editor who appointed the referees. Each editor can propose new members for the Editorial Board or Board of Referees. Editors and referees inactive for a longer period can be automatically replaced. Changes in the Editorial Board and Board of Referees are confirmed by the Executive Editors.

The coordination necessary is made through the Executive Editors who examine the reviews, sort the accepted articles and maintain appropriate international distribution. The Executive Board is appointed by the Society Informatika. Informatica is partially supported by the Slovenian Ministry of Science and Technology.

Each author is guaranteed to receive the reviews of his article. When accepted, publication in Informatica is guaranteed in less than one year after the Executive Editors receive the corrected version of the article.

Volume 17 Number 4 December 1993 ISSN 0350-5596

Informatica

An International Journal of Computing and Informatics

Contents: