(2)

RADC-TR-89-259, Vol X (of twelve)
Interim Report
October 1989

NORTHEAST ARTIFICIAL INTELLIGENCE CONSORTIUM ANNUAL REPORT - 1988 Time Oriented Problem Solving

Syracuse University

James F. Allen

This effort was funded partially by the Laboratory Director's fund.

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700

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

RADC-TR-89-259, Vol X (of twelve) has been reviewed and is approved for publication.

APPROVED: Northrup Fowler III

NORTHRUP FOWLER III
Project Engineer

APPROVED: Raymond P. Urtz Jr.

RAYMOND P. URTZ, JR.
Technical Director
Directorate of Command & Control

FOR THE COMMANDER: James W. Hyde III

JAMES W. HYDE III
Directorate of Plans & Programs

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (COES) Griffiss AFB NY 13441-5700. This will assist us in maintaining a current mailing list.

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

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS N/A
2a. SECURITY CLASSIFICATION AUTHORITY N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) N/A	5. MONITORING ORGANIZATION REPORT NUMBER(S) RADC-TR-89-259, Vol X (of twelve)
6a. NAME OF PERFORMING ORGANIZATION Northeast Artificial Intelligence Consortium (NAIC) 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Rome Air Development Center (COES)
6c. ADDRESS (City, State, and ZIP Code) Science & Technology Center, Rm 2-296 111 College Place, Syracuse University Syracuse NY 13244-4100	7b. ADDRESS (City, State, and ZIP Code) Griffiss AFB NY 13441-5700
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Rome Air Development Center 8b. OFFICE SYMBOL (If applicable) COES	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F30602-85-C-0008
8c. ADDRESS (City, State, and ZIP Code) Griffiss AFB NY 13441-5700	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. 62702F; PROJECT NO. 5581; TASK NO 27; WORK UNIT ACCESSION NO 13

11. TITLE (Include Security Classification)
NORTHEAST ARTIFICIAL INTELLIGENCE CONSORTIUM ANNUAL REPORT - 1988
Time Oriented Problem Solving

12. PERSONAL AUTHOR(S)
James F. Allen

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Interim	FROM Jan 88 TO Dec 88	October 1989	24

16. SUPPLEMENTARY NOTATION This effort was funded partially by the Laboratory Directors' Fund. This effort was performed as a subcontract by the University of Rochester to Syracuse University, Office of Sponsored Programs.

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	
12	05		Artificial Intelligence, Temporal Logic, Planning, Plan Recognition

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The Northeast Artificial Intelligence Consortium (NAIC) was created by the Air Force Systems Command, Rome Air Development Center, and the Office of Scientific Research. Its purpose is to conduct pertinent research in artificial intelligence and to perform activities ancillary to this research. This report describes progress that has been made in the fourth year of the existence of the NAIC on the technical research tasks undertaken at the member universities. The topics covered in general are: versatile expert system for equipment maintenance, distributed AI for communications system control, automatic photointerpretation, time-oriented problem solving, speech understanding systems, knowledge base maintenance, hardware architectures for very large systems, knowledge-based reasoning and planning, and a knowledge acquisition, assistance, and explanation system.

The specific topic for this volume is a model theory and aciomatization of a logic for reasoning about planning in domains of concurrent actions.

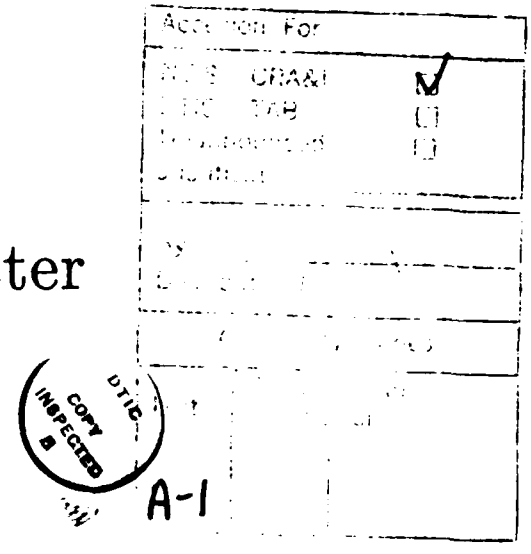
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL Northrum Fowler III	22b. TELEPHONE (Include Area Code) (315) 330-7794	22c. OFFICE SYMBOL RADC (COES)

DD Form 1473, JUN 86 — Previous editions are obsolete. — SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

Item 10. SOURCE OF FUNDING NUMBERS (Continued)

Program Element Number	Project Number	Task Number	Work Unit Number
62702F	5581	27	23
61102F	2304	J5	01
61102F	2304	J5	15
33126F	2155	02	10
61101F	LDFP	27	01

Northeast Artificial Intelligence Consortium
1988 Annual Report



Volume 10
University of Rochester

Time Oriented Problem Solving
Principal Investigator: James F. Allen

This work was supported by the Air Force Systems Command, Rome Air Development Center, Griffiss Air Force Base, New York 13441-5700, under Contract Number F30602-85-C-0008 which supports the Northeast Artificial Intelligence Consortium (NAIC).

Contents

10.1 Executive Summary

We have made excellent progress this year on several different aspects of our overall task of exploring problem solving in enriched temporal domains. The main contributions have been the following:

- the development of a domain for motivating our research in real-time execution of plans involving temporally extended actions.
- a new approach to causal reasoning that rejects domain-dependent approaches to solving the frame problem [Web88b];
- the formalization of two different types of abstraction in planning and an explication of their properties, one involving the relaxation of operator constraints, and the other an extension of inheritance hierarchies [Ten88];
- the public release of the knowledge representation language Rhet, and the additional of new functionality, such as extended type reasoning.
- the development of a non-reified first-order temporal logic that has a well defined syntax, semantics, and proof-theory and is easily implemented using a type-based theorem prover such as Rhet.

We describe each of these contributions in more detail below. In addition to the research *carried out*, *we organized* and hosted the Rochester Planning Workshop, which took place at the University of Rochester, October 27-29. This workshop received support from both the NAIC and AAAI. In attendance were many of the leading researchers in automated planning, with presentations made by Nils Nilsson, Stan Rosenschein, Drew McDermott, and Ray Perrault, to name just a few. By all reports, the workshop was quite successful, and has helped put the University of Rochester and the NAIC at the forefront of institutions engaged in planning research.

We have developed a planning domain based on model trains, ARMTRAK, which captures the conceptual simplicity of the blocks world while extending it sufficiently to a planner with more realistic capabilities. Some of the capabilities that this planner must account for are: actions that have non-uniform temporal extent, knowledge acquisition actions, concurrent actions and resource conflicts, a dynamic world that changes while the agent computes, and incomplete knowledge. A simulation based version of ARMTRAK is under development. In addition, the functionality of the simulation is being integrated with the image processing equipment in the vision lab.

For the past twenty years, solutions to the persistence (frame) problem have been based on domain-independent nonmonotonic assumptions, including such favorites as STRIPS, circumscriptive commonsense reasoning, and chronological minimization. There are, however, serious technical and philosophical problems with this paradigm. On close inspection, it becomes clear that the Yale shooting problem should not be solved by a domain-independent mechanism, as is commonly believed. As an alternative, Jay Weber [Web88b] has developed a framework for reified logical causal theories that solves the persistence problem using

appropriate information about the domain. For example, in the shooting problem it is important to reason that the gun could not have been unloaded since the reasoning agent did not unload it, and no other agent could have unloaded it while the agent was holding it.

An ongoing interest has been in defining a precise notion of abstraction in planning. This has led us to investigate multi-level representations of the same problem domain. Our motivation has been in specifying the formal semantics that we would like to hold between the different levels, and seeking those syntactic transformations that will preserve this semantics. We have investigated two methods of abstraction; the first involves extending inheritance hierarchies from object classes to relations between objects and actions over objects. A domain that includes Bottles and Cups at the low level, with relations InBottle and InCup and operations pourBottle and pourCup, could be abstracted to a simpler representation with objects of type Container, the relation InContainer, and the operation pourContainer. Thus, the abstract level does not distinguish between the different but analogous object types. The second method of abstraction involves relaxing the constraints under which operators can be applied, similar to the approach taken in ABSTRIPS. We have shown how previous methods of doing this would result in inconsistent systems, and demonstrate a set of constraints on the abstraction mapping that ensure both that the abstract levels are consistent, and also that solutions to low level problems have solutions in the abstract space as well.

Rhet is a multi-strategy KR system that does contextual based reasoning (forward and backward chaining) using enhanced horn clauses. Its features include: 2 major modes for representing knowledge – as Horn Clauses or as frames, which are interchangable; a type subsystem for typed and type restricted objects (including variables); E-unification; negation; forward and backward chaining; complete proofs (prove, disprove, find the KB inconsistent, or claim a goal is neither provable nor disprovable); full LISP compatibility; upward compatibility with HORNE; contextual reasoning; truth maintenance; frames having KL-1 type features, plus arbitrary predicate restrictions on slots within a frame as well as default values for slots; and a separate subsystem providing advanced user-interface facilities, graphics, and ZMACS interface on the lisp machines. The following features will soon be added: incremental compilation; default reasoning; user-declarable reasoning subsystems; and Allen and Koomen's [AK88] TEMPOS time interval reasoning subsystem.

Many problems in artificial intelligence require reasoning about events or states of the world that have temporal extent. Standard first-order logics have proven useful for reasoning about static propositions and their consequences, but have not been readily adaptable to the greater demands of temporal reasoning. For instance, "block A is on block B" can be represented as $On(a, b)$, but "block A is on block B from 7 to 12" is less obviously represented. One approach is to add to the predicates additional arguments denoting the temporal interval to which the assertion should be associated: $On(7, 12, a, b)$. This approach has received little past research attention, being typically abandoned in favor of *reified* logics [All84, Lif86, Sho88] containing truth predicates relating atemporal propositions. Although there have been many benefits to adopting this approach, and we continue to use reified logics in much of our research, there have been drawbacks as well. These include a complex unification and type mechanism, since arbitrary structure is often encoded in the terms. In addition, there has been criticism by Shoham [Sho88] leveled at the representation of Allen [All84] to the effect that it does not have a well-defined semantics, especially with regards to the embedding of quantification and logical connectives within the terms. This has led us to

develop a simple first-order logic obtained by including the additional temporal arguments, showing that this preserves the first-order structure of the propositions, has a clear semantics, and a standard proof-theory which, with few augmentations, is easily implemented on the current generation of automated theorem provers. In addition, we demonstrate [BTK89] that little is lost in using this logic, since it subsumes Shoham's logic.

10.2 Introduction

The overall aim of our research has been to develop planning representations that overcome the constraints and problems of earlier planning systems having impoverished temporal models. Much of our previous work has involved the development of representations that explicitly encode temporal knowledge, enabling one to reason about concurrent events [All84] and events beyond the agent's control [PA88].

As a natural development of this research, we are exploring richer causal models and a novel approach to the *frame* problem, concurrent planning and activity, and a means for encoding similarities between problem instances in the domain. Our research has been driven by a desire to add these capabilities and to explore the concomitant problems that arise from their use. The results of this effort have been impressive. In the following sections, we will describe these research efforts in more detail.

10.3 Time-Oriented Problem Solving

10.3.1 Planning and Execution

We have developed a planning domain based on model trains, ARMTRAK, which captures the conceptual simplicity of the blocks world while extending it sufficiently to exercise planners with more realistic capabilities. Over the summer a simulation based version of ARMTRAK was developed. Currently, an effort to integrate the functionality of the simulation with the image processing equipment in the vision lab is being implemented.

The inputs to the ARMTRAK are simple: switch settings and engine power; but many different scenarios can be generated from these simple inputs. Because the train movement is independent of the planner, plans which succeed at one point may fail as the train progresses. The planner must therefore take into account temporal constraints on the generation and execution of its plans. Moreover, model trains are notorious for falling off the track. Therefore, plans may succeed or fail due to unexpected effects of the primitives. Finally, in certain cases there may be no plans which achieve the desired goal, but which may partially achieve that goal. In this case, all possible plans fail, and the planner must select the most useful plan from the set of failing plans.

Traditional planning has concentrated on issues of correctness and performance in controlled environments. Planning of this type has been well served by the blocks world. This planning domain does not, however, address several new problems raised by attempts to build robots which plan actions in the real world. While planning domains must be sufficiently simple for humans to reason about, they must also be sufficiently complex to exercise planning programs. Though the blocks world admirably captures the first requirement, it

fails on the second. The blocks world makes five simplifying assumptions that effect the nature of the planners which operate on it. These assumptions are:

1. only the planner has effects on the world,
2. the planners actions always cause expected effects,
3. only one action can occur at one time,
4. a goal must be completely achieved for a plan to succeed, and
5. complete information is always available.

Due to these simplifying assumptions, the blocks world fails to support sufficient complexity to exercise planning under three major categories of constraints:

1. real-time planning,
2. planning with the possibility of unexpected results from primitive actions, and
3. selecting between equally attractive plans.

ARMTRAK consists of a model train layout coupled with a sensorium. A set of commands manipulates the layout; the sensorium satisfies a set of queries. The layout consists of track and trains. The track comes in curved or straight sections. The curved sections are categorized by the radius of the circle which the track describes. Switches are sections of track at which a choice between two alternate sections must be made. The state of the switch determines which of the two alternatives is taken. Decouplers are sections of track which can uncouple cars. Trains are composed of a locomotive and cars. Signals sent to the locomotive specify the power supplied to the engine. The velocity of the train is determined by its power, its weight, and the grade of the track it is on. When a car backs into another car the cars connect. Commands can be sent to the switches, the trains and the decouplers. Commands to the switches change the state of the switch; commands to the engines change the power going to the engines; commands to the decouplers uncouple the cars.

ARMTRAK will have advantages over the blocks world due to increased flexibility along four axes. First, ARMTRAK will follow the actual passage of time more closely. Flexibility along this axis will allow exercises involving planners which maintain complex temporal models. Next, complex goals can be specified allowing the use of planners which attempt to achieve optimal results rather than blindly achieving a goal set for it. Also, the domain will have a sufficient number of elements so that realistic conversations can be generated about the domain allowing the use of a discourse system involving plan recognition. Finally, unexpected results can occur allowing the use of planners designed to take advantage of serendipity.

10.3.2 Causal Reasoning

Historically, the *frame problem* has been discussed within the context of the situation calculus [Hay81]. Recently, many researchers [MS88, San88, Web88b, Sho88] have examined

the incarnation of the frame problem within representations incorporating an explicit discrete time line, usually calling it the *persistence problem.* Despite the often overlooked fact that this representation is no more expressively powerful than the situation calculus, these researchers have found that the direct application of techniques from situation-based solutions [Lif87, Hau87] are not sufficient to solve the persistence problem. We believe this predicament is caused by the following:

- In most examples, transformations on situations are expressed using specific actions, e.g. $result(load, S)$ [Lif87]. When this is done, it is trivial to deduce that any another distinct action did not occur, thereby reducing the frame problem to reasoning with complete knowledge about what actions occurred. Note that Lifschitz' approach [Lif87] would not solve the shooting problem [HM86] if he expressed the wait step as $result(a, S)$ where $type(a, wait)$.

- For explicit time, action occurrences are represented by a relation between actions and time points, e.g. $\texttt{occurs}(\texttt{load}, 0)$ [Web88b]. In general, the assertion of a specific action occurrence says nothing about the occurrences of other actions.

- The persistence problem is concerned with ascertaining which actions did *not* occur, because to reason that a property did not change over time, it is necessary to reason that no action occurred that could cause the change.

- Therefore, the typical explicit time translation of examples like the shooting problem scenario [HM86] actually contains less domain information about action non-occurrences. Something must be added to provide this lost information.

Several researchers have independently suggested circumscribing the `occurs` predicate [MS88, Web88b, San88] to entail the needed non-occurrence information. Thus if an agent does not know that a particular action occurred, it may assume that the action did not occur. We have found that this approach has the following problems:

- The results of the circumscription depend on the particular definitions of the actions. For example, if there are two mutually exclusive action terms for "the door stays open" and "the door closes", circumscription produces minimal models for both non-occurrence assumptions.

- If "causal chains of action" are allowed [MS88], i.e. actions can *generate* other actions depending on the context, then unintuitive assumptions are made about those contexts. For example, suppose an agent performs a `trigger` action at time 0 but does not know whether the gun is `loaded` at 0, while believing the following domain axiom:

$$\texttt{occurs}(\texttt{trigger}, t) \wedge \texttt{holds}(\texttt{loaded}, t) \rightarrow \texttt{occurs}(\texttt{shoot}, t)$$

  Circumscribing `occurs` entails $\neg\texttt{occurs}(\texttt{shoot}, 0)$, which combined with the above generation rule entails $\neg\texttt{holds}(\texttt{loaded}, 0)$. Thus the agent assumes the gun is not loaded when in doubt, simply as an artifact of the domain independent solution to the persistence problem. We call this the *generation/minimization problem.*

- We have a philosophical objection to a domain-independent rule that claims domain facts may be inferred purely from ignorance.

Fueled by these objections, we have developed a formal theory of action reasoning based on an explicit time line that has the following features:

- The occurs predicate is not circumscribed. Instead, non-occurrences are entailed by domain-dependent *executability axioms* which specify necessary conditions for action occurrences, as in the template:

$$\forall m[\texttt{occurs}(\epsilon, m) \rightarrow \bigwedge_{i=1}^{\nu} \texttt{holds}(\phi_i, m)]$$

  Particular domain axioms have constants instead of greek letters. An action is believed to not occur when at least one of its necessary conditions is believed to not hold.

- Effects of actions are entailed by axioms of the form:

$$\forall m[\bigvee_{i=1}^{\nu} \texttt{occurs}(\epsilon_i, m) \rightarrow \texttt{holds}(\phi, m)]$$

  When adding the definition of negation, this form extends to negative effects:

$$\forall p, t[\texttt{holds}(p, t) \equiv \neg\texttt{holds}(\overline{p}, t)].$$

- Non-effects of actions are entailed by *cause closure axioms* [Sch89], which specify the possible conditions for property change, as in:

$$\forall m[(\texttt{holds}(\phi, m) \wedge \neg\texttt{holds}(\phi, m')) \rightarrow \bigvee_{i=1}^{\nu} \texttt{occurs}(\epsilon_i, m)]$$

  By the contrapositive, a property is believed to remain holding when it is believed that no possible cause occurs. Incidently, these axioms are similar to the theorems added when the *causes* predicate is circumscribed in Lifschitz' situation calculus solution [Lif87].

- External events (not inspired by the planning agent) are supported through *event enabling axioms*, which specify sufficient conditions for action occurrences, as in:

$$\forall m[\bigwedge_{i=1}^{\nu} \texttt{holds}(\phi_i, m) \rightarrow \texttt{occurs}(\epsilon, m)]$$

  This ability to infer the occurrence of events based solely on properties is actually quite novel, and because of this, past formal frameworks have not supported external events.

The above features constitute a solution to the persistence problem which does not suffer from the generation/minimization problem, nor does it create an ambiguity between action terms. For example, the theory demonstrates how an explicit time version of the shooting problem scenario does not contain enough information to produce the popular interpretation

of the outcome in a domain-independent manner. Lastly, although the above theories are monotonic, we can add nonmonotonic rules in one of the many ways, e.g. using Reiter's default rules [Rei80]:

$$\frac{\texttt{holds(Loaded},m) \wedge \texttt{holds(HasGun(Me)},m) : \mathbf{M}\ \texttt{holds(Loaded},m')}{\texttt{holds(Loaded},m')}$$

Literally, this rule says "if you are holding a loaded gun and it is consistent that it is still loaded at the next time point, then assume it will still be loaded". Another way to phrase this is "you would (probably) know if a gun became unloaded while you were holding it", in the sense that provability means knowledge. Condoning the use of nonmonotonicity in this manner may appear to contradict our previous arguments, but this is not actually the case. We have criticized the *domain-independent* use of nonmonotonicity for persistence. Nonmonotonic rules such as the above make claims about particular properties, i.e. that these properties will usually persist given certain information. How often the default rule must be right depends on how costly it is to be wrong, and the cost of ascertaining a more accurate evaluation of that property. Ultimately, this sort of judgement should be founded on statistical information about degrees of belief in properties given belief in other properties. This direction is the subject of our current research [Web88a].

10.3.3 Abstraction in Planning

Planning involves reasoning about the effect of actions on the world. Josh Tenenberg's dissertation [Ten88] explores the use of abstraction in planning in order to simplify the reasoning task for an automated agent. An abstract representation can be constructed from a concrete one by ignoring details and including only those aspects of primary importance. The abstract representation can be used to solve problems that do not crucially depend upon the ignored details; the concrete representation can be used otherwise, and to specialize the abstract solutions.

Two significant issues emerge which form the basis of this research. First, the abstract views must sanction plan construction for frequently occurring problems, yet never sanction the deduction of contradictory assertions. Attempting to extend the applicability of an abstraction often occurs at the cost of consistency, while preserving consistency can weaken a system and limits its utility. Second, a correspondence between the abstract and concrete views must be maintained so that abstract solutions bear a precise relationship to the concrete level solutions derived from them. Without this correspondence, there is little justification for the resource expenditure necessary to construct and use the abstract level.

Abstraction has been explored in two settings. In the first, relaxing the operator preconditions of a STRIPS system [FN71] was used in a manner similar to that of Sacerdoti in his ABSTRIPS system [Sac74]. We attacked the problem that plagues Sacerdoti's work, of the potential for inconsistencies to arise at the abstract levels due to the presence of sentences in the world description that contain eliminated preconditions. For instance, if *HandEmpty* is ignored at the abstract level, the agent might be *Holding* two blocks at the same time, violating one of its world axioms.

Our approach is to partition the predicates based upon their inferential relationships to one another within the world description. Each partition is assigned a *criticality* value

reflecting its measure of importance, and an abstraction hierarchy is induced by eliminating partitions of decreasing importance. So, for example, *HandEmpty* and *Holding* would belong to the same partition, since they are related in the axiom

$$\forall x.Holding(x) \supset \neg HandEmpty$$

and hence would have the same criticality. Whenever *HandEmpty* preconditions are eliminated, so are the preconditions, add's, delete's, and sentences of the world description that involve any of the predicates in *HandEmpty*'s partition. This approach is shown to preserve consistency. In addition, systems abstracted in this fashion have the *upward-solution property*, where the existence of a concrete level solution implies the existence of abstractions to this solution at each higher level.

In the second setting, analogy is used as the basis for determining which aspects of a theory are details. Objects belong to object classes arranged in inheritance hierarchies. Hierarchies of object types organized in subclass/superclass relations have been used extensively in artificial intelligence for a variety of reasoning tasks [Hen79, Was85]. These hierarchies have an intuitive semantics: each element of a subclass is an element of the superclass to which it is related. Therefore, each property known to be true of each element of the superclass will necessarily be true of each element of the subclass. So, for example, every object that is a *Bottle* is a *Container*, and all properties of *Containers* are also properties of *Bottles*, hence the frequent use of ISA in the literature to describe the relationship between inheritance classes. Object classes having a common superclass can be considered *analogous*.

Inheritance is extended in a novel fashion to *all* of the predicates used in the world descriptions of a STRIPS system (not just the object *type* predicates) while still preserving the desired semantics through the use of a *predicate mapping function*, a many–1 function that maps concrete level predicates to abstract level predicates. So, for example, predicate mappings enable *Bottle* and *Cup* to map to *Container*, as one would expect, and likewise *InBottle* and *InCup* map to *InContainer*. An abstract representation is constructed by extending the mapping function to world descriptions, and to operators. Mapping concrete level sentences prohibits the inclusion of sentences that distinguish between analogous predicates. For instance, given the concrete level axiom

$$\forall x.Bottle(x) \supset NarrowNeck(x),$$

the abstract axiom

$$\forall x.Container(x) \supset NarrowNeck(x),$$

is *not* included in the abstract world description, since

$$\forall x.Cup(x) \supset NarrowNeck(x),$$

is not also true at the concrete level. That is, the *NarrowNeck* axiom distinguishes *Bottles* from *Cups*, and hence is not mapped. However, the axiom

$$\forall x.Container(x) \supset HoldsStuff(x),$$

would occur at the abstract level, since *HoldsStuff* is common to both *Bottles* and *Cups*.

Operators are mapped according to similar constraints, in that operators cannot be included which distinguish between analogous predicates. Thus, *pourContainer* is an abstract level action, assuming that both *pourBottle* and *pourCup* are at the concrete level. However, the action *openContainer* is not an abstract action, since (presumably), *Bottles* but not *Cups* are opened. By observing these constraints, the abstract levels are guaranteed to be consistent. In addition, they exhibit the *downward-solution property*, where the existence of an abstract solution implies the existence of specializations of this solution at each lower level.

10.3.4 Rhetorical

The Project

As we introduced in last year's report, Rhetorical (Rhet) is a programming and knowledge representation system offering formally well defined tools for building an automated reasoning system. It's emphasis is on flexibility of representation, allowing the user to decide if the system will basically operate as a theorem prover (e.g. like PROLOG), a frame-like system (e.g. like KL-1), or an associative network.

Rhet offers two main modes of inference: a horn clause theorem prover, and a forward chaining mechanism. To supplement these modes, Rhet includes specialized reasoners for dealing with equality of ground terms, types of terms (any of ground terms, variables, or functions), contextual reasoning, and constraint proof mechanisms. Rhet allows a reasonably functional interface with lisp allowing one to escape from Rhet to Lisp, or from Lisp to Rhet as needed[1].

Accomplished in 1988:

By the end of the year, we have virtually the full Rhet language supported and running in an interpreted fashion. In particular, from our expectations list last year and other additions we have achieved:

- Publication of two TRs, numbers 238 and 239 in February 1988, and revised in October of 88. These are the User Manual and Programmer's Guide respectively.
- Structured type reasoning is fully implemented with the exception of certain advanced functionality, e.g. classification[2].
- An inequality reasoner has been added, which keeps the prover from assuming or allowing two objects that have been declared unequal to be made equal.
- Belief modal operators are fully supported.
- Structured type definitions have been extended to include uninterpreted relations, which allow the user to associate arbitrary Rhet objects with a type declaration. For

[1] Normally, one would either code one's project in Lisp and escape to Rhet to handle the KR tasks, or code in Rhet and possibly escape into Lisp for efficiency or special i/o issues.

[2] Exceptions to the full implementation of the Rhet language are carefully noted in TR238.

example, one might define a *Precondition* relation on a plan type whose value would be a list of preconditions for executing a plan instance of this type.

- Considerable work stabilizing and documenting the system
- Some effort improving performance, which we expect to continue into 1989.

Expectations for 1989

Our current reasonable expectations for 1989 include:

- Two new releases of the system (currently we have just released version 15.0), with associated documentation updates.
- Version 16.0 will include more performance enhancements which we expect to at least double the rated LIPS of v15.
- V16.0 will also include a demonstration application (plan recognition) developed two different ways, one written in Lisp, and utilizing Rhet as a KR system only, and the other written entirely within Rhet.
- We expect the above effort to have some effect on Rhet's user interface, which we hope to improve to continue making Rhet easier to use.
- V17.0 will include at least a prototype axiom compiler. This will at a minimum compile the general unification algorithm down to a specialized unifier for the head of the clause[3].
- A move toward a more generic platform for using Rhet. While we expect that Rhet development will continue to require the specialized tools only available on advanced lisp platforms, we would like to see Rhet usable on a wider variety of workstations. To that end we will be moving toward a more generic brand of common-lisp for its implementation, and recoding the user interface for use under the X window system standard. This should make rehosting Rhet easier in the future, should the need arise.

Our more ambitious dreams for 1989 include:

- Additional performance improvement for Rhet including such facilities as goal caching, axiom reordering, and incremental hashing.
- A more advanced editor interface, integrating the Rhet environment more solidly into the Lisp machine paradigm. Since this paradigm has been picked up in recent years by more traditional platforms (e.g. Sun's using GNU Emacs) this should not hinder our standardization effort.
- Set Reasoning

[3]This is where PROLOG compilation tends to gain most of its speed so we will try it first.

- More advanced compiler capabilities, including taking better advantage of intelligent backtracking[4].

10.3.5 A Non-Reified Temporal Logic

In the logic that we have developed,[5] (which we will call "BTK") propositions are associated with time objects by including temporal arguments to the functions and predicates. For example, one can represent the assertion "the President of the USA in 1962 died in 1963" as $Died\big(1963, president(1962, USA)\big)$. Temporal objects are distinguished from non-temporal objects by partitioning both the universe of discourse and the set of symbols in the language used to denote the universe. One can thus specify, for each function and predicate symbol, some number n of temporal arguments and some number m of non-temporal arguments, and for each function symbol, whether it evaluates to a temporal or non-temporal object. Although our logic is sorted, we could equally well have used an unsorted, classical first-order logic, there being standard equivalences between theories in these different logics (see, e.g., [Wa52] or [Wal87, p. 7]).

This approach contrasts with the work of Shoham [Sho88], who introduces a non-standard first-order logic (which we will refer to as "STL") that has a complex semantics, and a more restrictive syntax. One of the main drawbacks of his logic is that, since he has chosen to abandon standard first order syntax and semantics, first order proof theory no longer applies. Hence, Shoham's logic requires a new proof theory, a proof theory which has not yet been provided. This means that *currently there is no formal basis for any use of Shoham's logic to reason about temporal propositions.*

It may not be very difficult to provide a proof theory for Shoham's logic, but this in itself would not suffice in providing a useful tool for reasoning. One would also have to develop expertise in automating such a proof theory. The fact that our temporal logic has a standard syntax and proof theory means that we can take advantage of over 20 years of research in automated reasoning. Since our logic is first-order, we can avail ourselves of the current automated reasoning technology.

We show that Shoham's logic is subsumed by the logic we have developed by defining two transformations, a syntactic transformation, π_{syn}, and a semantic transformation, π_{sem}. π_{syn} maps sentences of STL to sentences of BTK, while π_{sem} maps models of STL to models of BTK. Using these two transformations we can show that any STL model is transformable into a BTK model in such a way that the set of sentences satisfied by the BTK model[6] includes the transformed set of STL sentences satisfied by the STL model. In other words, any set of STL sentences can be rewritten as a set of BTK sentences without eliminating any models which satisfy those sentences.

[4]LIPS benchmarks do not backtrack, therefore any performance penalty paid for the overhead of setting up for intelligent backtracking cannot be compensated for by better performance on the benchmark. Part of this task then may be to develop benchmarks that more accurately expose performance issues in real problems.

[5]This work was carried out jointly with Fahiem Bacchus, at the University of Waterloo, and Hans Koomen, at the University of Rochester.

[6]A model, $\mathcal{M}$, satisfies a sentence, α, written $\mathcal{M} \models \alpha$, if $\alpha^{\sigma} = \top$, i.e., if α is true under σ, the interpretation of the model.

10.3.6 Rochester Planning Workshop

The Rochester Planning Workshop: "From Formal Systems to Practical Systems" was held October 27-29, at the University of Rochester, with support provided by AAAI, the department of Computer Science at the University of Rochester, and the Northeast Artificial Intelligence Consortium (NAIC). The program committee consisted of James Allen, Jay Weber, and Josh Tenenberg.

The thrust of the workshop was to explore the scientific and engineering issues that currently impede progress in the development of systems that solve real planning problems in real environments. Our interest in these issues was not limited to robotic applications, but included a broad range of others, including natural language understanding, scheduling, and distributed problem solving. Approximately sixty people attended the workshop, all of whom are actively engaged in planning research.

The workshop was organized into five parts: an overview of planning motivations (consisting of four separate introductory talks), and four panels. Stan Rosenschein began the workshop with his talk "Robotics and Planning," which discussed fruitful approaches to bridging the gap between the different languages and methodologies of research in robotics and in planning. James Allen discussed "Natural Language and Plan Reasoning," where he proposed that virtually all planning problems can be cast as natural language problems. In the talk "Knowledge Based Scheduling: A Review," Mark Fox outlined the basic approaches that have been used in large-scale scheduling problems. And Victor Lesser presented a talk "Controlling Problem Solving Through Planning," which proposed the explicit encoding of control knowledge as operations for a planning agent to improve its problem solving performance.

For each of the four technical panels we chose a panel organizer to whom we provided a general topic. The organizer was responsible for choosing additional panel members, and for giving specific instructions to these panelists as to the actual content of their presentations. We recommended that the panels consist of a position presented by each panelist, followed by a commentary showing how the positions relate to each other and to the panel theme. The results were quite different for each panel, largely due to the diverse ways in which the panel concept evolved for each organizer.

Austin Tate organized the panel "Planning and Execution," with additional panelists Mark Drummond, Paul Morris, Nils Nilsson, and Stanley Rosenschein. Mark Fox's panel "Micro vs. Macro Planning" included Amy Lansky and N.S. Sridharan. Henry Kautz organized the panel "Planning and Plan Recognition," with panelists Eugene Charniak, C. Raymond Perrault, and Brad Goodman. The Workshop concluded with Drew McDermott's panel "Reactive vs. Strategic Planning," having as panelists David Chapman, Michael Georgeff, and Thomas Dean.

A number of the presentations offered are quite novel, and deserving of wider dissemination within the planning research community. We believe that workshops of the kind that we hosted are essential for the vitality of artificial intelligence as a field. Our institution is committed to continuing this kind of interchange with the larger research community.

10.4 Publication List

Allen, J.F. and Hayes, P.J., "Moments and Points in an Interval-Based Temporal Logic," (submitted December 1988), Computational Intelligence.

Allen, J.F. and Miller, B.W., "The Rhetorical Knowledge Representation System: A User's Manual," TR 238, Computer Science Department, University of Rochester, February 1988; revised September, 1988.

Guez, S., "Even Argumentation is Linguistic," (submitted December 1988), *International Joint Conference on Artificial Intelligence.*

Miller, B.W., "The Rhet Programmer's Guide (for Version 14.4)," TR 239, Computer Science Department, University of Rochester, March 1988; revised September, 1988.

Tenenberg, J.D., "Abstraction in Planning," Ph.D. Dissertation and TR 250, Computer Science Department, University of Rochester, May, 1988.

Tenenberg, J.D., "Extending Inheritance Abstraction to Symbolic Planning Systems," 1988-89 *Computer Science and Engineering Research Review*, University of Rochester, October, 1988.

Tenenberg, J.D., "Inheritance in Automated Planning," (submitted November 1988), *First International Conference on Principles of Knowledge Representation and Reasoning.*

Tenenberg, J.D., (with Fahiem Bacchus and Hans Koomen), "A Non-Reified Temporal Logic," (Submitted November 1988), *First International Conference on Principles of Knowledge Representation and Reasoning.*

Weber, J.C., "Plan-Based Goal-Oriented Classification," TR 243, Computer Science Department, University of Rochester, February, 1988.

Weber, J.C., "A Versatile Approach to Action Reasoning," TR 237, Computer Science Department, University of Rochester, March, 1988.

Weber, J.C., Tenenberg, J.D., and Allen, J.F. (editors), "Advance Proceedings of the Rochester Planning Workshop," Computer Science Department, University of Rochester, October, 1988.

Bibliography

[All84] James Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM* 26:11, 1983.

[AK88] James Allen and Hans Koomen. **Must get this reference**

[Alt87] Richard Alterman. Issues in Adaptive Planning. Technical Report 304, Division of Computer Science, University of California at Berkeley, 1987.

[AF88] John Anderson and A. Farley. Plan Abstraction Based on Operator Generalization. Proceedings of the Seventh National Conference on Artificial Intelligence, 1988.

[BTK89] Fahiem Bacchus and Josh Tenenberg and Hans Koomen. A Non-Reified Temporal Logic. First Internation Conference on Principle of Knowledge Representation and Reasoning, Toronto, Ontario, Canada, 1989.

[FN71] Richard Fikes and Nils Nilsson. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence* 2:189-208, 1971.

[Hau87] Brian Haugh. Simple causal minimizations for temporal persistence and projection. In *Proceedings of IJCAI-87*, pages 218–223, June 1987.

[Hay81] Patrick J. Hayes. The frame problem and related problems in artificial intelligence. In Bonnie Lynn Webber and Nils J. Nilsson, editors, *Readings in Artificial Intelligence*, pages 223–230. Tioga, 1981.

[Hen79] Gary Hendrix. Encoding Knowledge in Partitioned Networks. In *Associative Networks*, Findler (ed.), Academic Press, 1979.

[HM86] Steve Hanks and Drew McDermott. Default reasoning, nonmonotonic logics, and the frame problem. In *Proceedings of AAAI-86*, August 1986.

[Lif86] Vladimir Lifschitz. On the Semantics of STRIPS. Proceedings of the Workshop on Reasoning about Actions and Plans, Timberline, Oregon, 1986.

[Lif87] Vladimir Lifschitz. Formal theories of action: Preliminary report. In Frank M. Brown, editor, *The Frame Problem in Artificial Intelligence: Proceedings of the 1987 Workshop*. Morgan Kaufman, April 1987.

[MS88] Leora Morgenstern and Lynn Andrea Stein. Why things go wrong: A formal theory of causal reasoning. In *Proceedings of AAAI-88*, pages 518–523, August 1988.

[Nau87] Dana Nau. Hierarchical Abstraction for Process Planning. Proceedings of Second International Conference in Applications of Artificial Intelligence in Engineering, 1987.

[PA88] Richard Pelavin. A Model of Concurrent Actions Having Temporal Extent. National Conference on Artificial Intelligence, Seattle, 1987.

[Rei80] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1,2):81–132, April 1980.

[Rob65] J. Robinson. A Machine-Oriented Logic Based on the Resolution Principle. Journal of the ACM 12:23-41, 1965.

[Sac74] Earl Sacerdoti. Planning in a Hierarchy of Abstraction Spaces. *Artificial Intelligence* 5, 1974.

[San88] Erik Sandewall. Non-monotonic entailment for reasoning about time and action – part i: sequential actions. working paper, August 1988.

[Sch89] Lenhart Schubert. Solving the original frame problem without frame axioms or nonmonotonicity. In Henry Kyburg and Ron Loui, editors, *Selected papers from the 1988 Society for Exact Philosophy Conference*, 1989.

[Sho88] Yoav Shoham. *Reasoning about Change: Time and Causation from the Standpoint of Artificial Intelligence.* The MIT Press, Cambridge, MA, 1988.

[Ten88] Josh Tenenberg. Abstraction in Planning. Ph.D. Dissertation and TR 250, University of Rochester, 1988.

[Was85] K. Wasserman Understanding Hierarchically Structured Objects: Unifying Representation and Generalization. Ph.D. Thesis, Columbia University, 1985.

[Wa52] Hao Wang. Logic of Many-Sorted Theories. *JSL* 17, 1952.

[Wal87] Christoph Walther. A Many-Sorted Calculus Based on Resolution and Paramodulation. Pitman Research Notes in Artificial Intelligence 1987.

[Web88a] Jay C. Weber. Statistical inference and causal reasoning. submitted to IJCAI-89, December 1988.

[Web88b] Jay C. Weber. A versatile approach to action reasoning. Technical Report 237, Computer Science Department, University of Rochester, March 1988.