Representing Frame Axioms as Defaults

Scott Douglas Goodwin
Department of Computer Science

Research Report CS-87-48
July 1987

Representing Frame Axioms as Defaults

by

Scott Douglas Goodwin

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, 1987

Abstract

The advent of non-monotonic reasoning systems provides the opportunity to explore solutions to the frame problem using consistency-based methods. In particular, the Theorist theory formation framework, in view of its clear semantics, is a good framework within which to pursue investigations of the frame problem. Here, we examine the issues involved in using Theorist's defaults to represent frame axioms as a means of solving the frame problem.

This dissertation defends the thesis that *theory formation, together with a particular theory preference heuristic, is a simple, intuitively appealing way to solve the frame problem.* We call this preference heuristic chronological persistence. We give evidence to support the claim that the only reasonable approach to formalising rational belief is to use "scientific-like" theory formation. Since viewing reasoning as theory formation, rather than as deduction from our knowledge, has a solid basis in the philosophy of science, *we believe that the theory formation/theory preference framework will turn out to be a fundamental paradigm for knowledge representation and rational reasoning in AI.*

We present a theory formation representation scheme together with a semantically well-defined theory preference heuristic for dealing with the frame problem, and we specify a search procedure for selecting a preferred theory. As well, an implementation of this search procedure is described. Our approach to the frame problem is compared and contrasted with other recent proposals. We also propose a *continuum hypothesis* — it seems that axiomatisations based on theory formation/theory preference, circumscription, and negation-as-failure can be viewed as different points on a continuum where the degree of *theory preference explicitness varies*. For example, in negation-as-failure, preference information derived from syntactic structure is "compiled into" the proof procedure; while in circumscription, preference is expressed as priorities. This dissertation is offered as evidence to support the above claims.

Acknowledgements

I would like to thank my supervisor, Randy Goebel, above all for his constant encouragement. I deeply appreciate his valuable leadership and guidance, and thank him for the (intellectual, emotional, etc.) suffering he endured in order to help me reach my potential. His unceasing enthusiasm for research provided me with an endless source of energy. Through him, I came to appreciate the 'science' in computer science and the 're' in research. He convinced me that an unimplementable theory has little value. He taught me the importance of semantics and the usefulness of writing for clarifying thought. He has a knack for finding the good in things; he showed me how to strengthen ideas by building on their strong points. For all he has done for me, I owe him a lifetime supply of beer.

I would also like to thank David Poole. His insistence on precision and on having clear semantics helped me both understand and express my ideas. His technical skill aided in debugging my ideas. Above all, I would like to thank David for provoking me. He always seemed to know the right question to ask. As Randy once said, "David is like a child who proves his buddy's block tower is flawed by kicking it over. He then says, 'Build it stronger next time.'"

Special thanks to Romas Aleliunas for reading my dissertation. His suggestions enabled me to provide a clearer, more mathematically rigorous treatment of the material in chapter 3, and his suggestions led to the use of improved notation.

I would also like to thank the other faculty members, especially: Robin Cohen, Maarten Van Emden, and Marlene Jones. Through much effort behind the scenes, they have provided an intellectual environment conducive to research. Special thanks to Robin for organising many workshops and other events.

Also deserving much thanks are the many members of the Logic Programming and Artificial Intelligence Group, especially: Eric Neufeld, Denis Gagné, André Trudel, Ken Jackson, Paul Van Arragon, Bruce Kirby, Marco Ariano, Peter Van Beek, Kelly Arrey, Ross Bryant, Amar Shan, and many others.

Thanks also to the boys in Portable #1: Dave Gillies, Trevor Smedley, Steve Lawrence, Steve Furino, Greg Schaffer, Marco Ariano, Alan Williams, and Bruce Kirby. Together, we braved the cold winters, the muddy springs, the humid summers, and the muddy autumns out in our *temporary* offices.

I would also like to thank my high school math teacher, Kasmir Watroba, for instilling in me a love for mathematics and for starting me on the road to higher learning.

Deep thanks to my parents: Frank and Colleen, and to my sisters: Cathy, Vicki, Janice, Kelly, Lynn, and Tracy; they have provided much support, both financial and emotional.

Finally, I would like to thank my friend Charley Kasmierski because he likes to see his name in print, but more importantly, for preventing me from taking things too seriously.

Table of Contents

List of Figures

Chapter 1
Introduction

1.1. Introduction

Change is a fundamental aspect of the world and dealing with it is intellectually challenging. To succeed in a demanding environment requires planning. Failure to plan can result in missed opportunities, unfulfilled needs, and unsatisfied wants and goals. Planning involves choosing a course of action which leads to a desired situation. To plan, it is necessary to envision the changes that ensue in following alternative paths. Performing actions can set off long chains of cause and effect. The underlying difficulty of envisioning future situations is, therefore, that of keeping track of all the consequences of actions. Since all features of the world are subject to change, describing the effects of actions requires specifying, for each feature, if and how it is affected.

The problem of specifying the effects of actions in a practical way has been called the frame problem. In view of the infinite number of possible features, describing actions can be perplexing. Fortunately, there seems to be a common sense approach to handling this difficulty. Because the world is a fairly stable place (actions have limited impact), actions can be described by

stating the features of the world that are changed by the actions and by making the assumption that all other features are unaffected. When contradictions arise as a result of this assumption, measures can be taken to re-establish consistency. Thus, contradiction and consistency-restoration are fundamental aspects of planning in dynamic worlds.

The ability to recognise and recover from inconsistencies is important not only to humans, but to the evolution of machine intelligence as well. Without this ability, handling change is difficult, if not impossible. The advent of non-monotonic reasoning systems provides the opportunity to explore solutions to the frame problem using consistency-based methods. In particular, the Theorist system [Poole87b], in view of its clear semantics, seems to be a good framework within which to pursue investigations of the frame problem. Thus, we propose to examine the issues involved in using Theorist's defaults to represent frame axioms as a means of solving the frame problem.

1.2. Artificial Intelligence

Artificial Intelligence (AI) is the branch of computer science that studies techniques for constructing programs that perform intellectually demanding tasks. Since the goal of AI is to build programs that behave intelligently, some notion of intelligence is needed to judge the results. Because intelligence appears to involve many information-processing and information-

representing components, it has not been precisely defined; nor does it seem possible to provide a definition in the usual sense. It is possible to give sufficient conditions for *general intelligence* (cf. [Turing63]); however, a purely behavioural description provides a clearer understanding of what intelligence is. Intelligence can be treated as a black box whose properties can be ascertained through observation of intelligent behaviour. Humans are said to possess *natural intelligence*; they exhibit certain behaviours associated with intelligence (e.g., problem solving, learning, language comprehension). Animals, like dolphins and chimpanzees, also possess a form of natural intelligence. Machine intelligence and natural intelligence are both instances of general intelligence.

Studies in artificial and natural intelligence share a symbiotic relationship. What is learned about machine intelligence often provides insight into natural intelligence and vice versa. Studying intelligence with machines has the advantage of easy experimentation. It is usually a simple matter to add or remove some piece of knowledge from a computer program. Animal brains can not be 'carved up' with the same degree of precision. The ability to add and delete pieces of knowledge allow their importance to be tested. Machine studies of intelligence have another important advantage. Because computer models are precise, implementing a theory can uncover conceptual errors. Viewing computation as a metaphor for intellectual activity adds pre-

cision to theories of intelligence. Computer science concepts provide useful analogies and a new language to describe thinking methods and epistemological structures.

There are advantages to studying natural intelligence as well, since it has ready-made subjects possessing intelligence (e.g., humans). Natural intelligence also provides a metric by which to judge machine models of intelligence. Thus, researchers in artificial and natural intelligence can benefit from each other's work.

This duality between artificial and natural intelligence suggests a methodology for studying general intelligence. The behaviour of human subjects is observed for some intellectual task. Certain intellectual mechanisms are proposed to explain the observed data. These mechanisms are then implemented on a computer. Experimentation with the machine models yield predictions about behaviour that are then tested for in the human subjects. The proposed theory is then revised as required to explain the differences between the predictions and the observations and the cycle is repeated. It is no coincidence that this process is similar to that used by other scientists. A physical phenomenon is observed. A mathematical model is proposed to explain it. The model makes certain predictions that are tested and the theory is revised as needed. Both processes are instances of the scientific method.

One difference, however, is that the pencil and paper models of the physicist are replaced by computer models of intelligent mechanisms. Like intelligence, these models are objects of study in AI. This particular form of the scientific method is an approach to studying general intelligence that is suggested by the duality between artificial and natural intelligence.

Since AI is a branch of computer science, the ultimate concern must be with machine implementations of algorithms and data structures. Many abstract theories of intelligence are possible, but unimplementable theories are of little worth. The only acceptable explanations of behaviour in AI are those expressed in terms of computation. Embedded in the notion of machine intelligence is the tacit assumption that there is nothing magical about intelligence, that it is reducible to computation (symbol manipulation), and that it can be embodied by computer programs. To the extent that this is the case, AI research can provide insight into the nature of intelligence.

Together with the scientific goal of devising an information-processing theory of intelligence, there is also an engineering goal of building programs that solve problems. The goal of AI is not to create an artifact with human intelligence; AI is not obsessed with mimicking humans. This does not, however, rule out using whatever human methods seem appropriate to the goal of solving problems. In pursuing the engineering goal of AI, the scientific goal should not be forgotten (and vice versa).

AI is a science — a computational study of intelligence. The test of an AI theory's mettle is how close the behaviour of its implementation is to the behaviour that the theory is trying to explain. Though AI is not as task-oriented as engineering, it is concerned with solving problems. Therefore, AI is an attempt to make computers more useful in solving problems by understanding and employing the principles underlying intelligence.

1.3. Knowledge Representation

Knowledge representation is recognised as one of the central problems in AI. Knowledge representation is an area of AI research that is concerned with knowledge and its representation in machines. This definition leads to two questions.

The first question is, 'What is a representation?' A representation is a set of conventions for describing things. According to Brachman and Levesque [Brachman85], it involves recording, in some language, descriptions that correspond to the thing being represented. The representation could be *symbolic* or *iconic*. Symbolic representations describe things using inherently meaningless symbols. A symbol becomes meaningful when a user assigns a meaning to it and thereafter interprets it according to its assigned meaning. Iconic representations describe things more directly (e.g., diagrams).

A *knowledge representation scheme* consists of a representation language (with its associated semantics) and an inference mechanism. The representation language may be a formal language, a data structure, a diagram, etc. In knowledge representation, the representation language is used to explicitly express facts about the world. An inference mechanism is used to derive implicit facts from expressions in the language. The nature of this mechanism depends on the representation language. For logical languages, it is usually based on deduction. Generally, it is some set of procedures for manipulating symbols or icons.

The second question the definition of knowledge representation leads to is, 'What is knowledge?' This question is at least as difficult to answer as the question, 'What is intelligence?' Philosophers have been struggling with it since the time of Aristotle. Ability to behave intelligently is often linked to knowledge. Knowledge is often associated with facts, but this begs the question, 'What is a fact?' Is it a fact that unsupported objects fall or is it a belief based on the theory of gravity?

The question 'What is knowledge?', like that of 'What is intelligence?', can best be dealt with by using the more pragmatic behavioural approach. The computational metaphor used to deal with intelligence applies equally well to knowledge. Thus, knowledge is represented as data structures and procedures. The behaviour of such machine representations of knowledge

can then be compared with knowledgeable behaviour and judged accordingly. A general hypothesis adopted by many researchers in AI is that a computer program capable of acting intelligently must have a scheme for representing its knowledge and beliefs, as well as a means of using that representation scheme to determine its behaviour. This assumption is what Brian Smith calls the *knowledge representation hypothesis*.

> "Any mechanically embodied intelligent process will be comprised of structural ingredients that a) we as external observers naturally take to represent a propositional account of the knowledge that the overall process exhibits, and b) independent of such external semantic attribution, play a formal but causal and essential role in engendering the behaviour that manifests that knowledge." [Smith85, p. 33]

Most AI researchers accept the knowledge representation hypothesis, at least as a working hypothesis. The origins of this hypothesis can be traced back to McCarthy who believed that for a system to behave intelligently it must be able to represent in language facts about the world, must be able to draw conclusions based on the facts, and act based on the conclusions [McCarthy68]. For McCarthy, the starting point in knowledge representation is to view fact manipulation as the basis of intelligent behaviour. For a machine to display intelligent behaviour, McCarthy argued, it would need an adequate representation of the world and the ability to answer a wide variety of questions based on that representation.

McCarthy and Hayes [McCarthy69, McCarthy77] showed how knowledge representation schemes could be analyzed in two parts, which they call the *epistemological* part and the *heuristic* part. What does the representation have to be like to allow all the kinds of knowledge needed for intelligence to be expressed?

> "What kinds of facts about the world are available to an observer with given opportunities to observe, how can these facts be represented in the memory of a computer, and what rules permit legitimate conclusions to be drawn from these facts?" [McCarthy77]

The epistemological part of a representation scheme is concerned mainly with the issue of *expressiveness*. A representation is called *epistemologically adequate* if it is suitably expressive. The heuristic part of a knowledge representation scheme involves the reasoning process. It is concerned with the representation's use. Can the representation be used to answer questions or solve problems in a reasonable amount of time? Efficiency in using the representation determines its *heuristic adequacy*.

Although many knowledge representation schemes are possible, there is widespread disagreement over what form of representation is most appropriate. McCarthy advocates explicit representations based on logical languages [McCarthy77]. Others have argued that knowledge should be represented procedurally [Minsky72, Hewitt73, Winograd72]. The logic/non-logic, declarative/procedural controversy has been discussed in many places (cf. [Hayes77, Winograd85]).

According to Winograd, the proceduralist's position is based on the belief that knowledge is primarily *knowing how*, and that knowledge is intimately intertwined with procedures for its use. Some things we know are best explained as procedures. It is easier to represent what we know about processes (e.g., baking) with procedures. Another advantage to procedural representations is the ease of use of second-order knowledge. The final advantage is that it is simpler to incorporate domain-specific heuristic knowledge in procedures.

The declarativist position is based on the belief that knowledge is mainly *knowing that*. The basis of their approach to intelligence is a general set of procedures for manipulating facts together with facts about the domain of interest. According to Winograd, declaratives have the advantages of flexibility and economy; that is, a single fact can be used in many ways. Procedural representations need separate procedures for each different use. Declaratives also have the advantages of understandability, modifiability, and learnability, because facts can be viewed independently. Interaction between them is determined solely by the inference mechanism. Thus, they are more modular. Finally, declaratives offer the advantages of accessibility and communicability, since it is easier to explain to others what declaratives mean.

Attempts have been made to develop representation schemes with the advantage of both declaratives and procedures. One approach is to compile declarative facts into procedures (cf. [Sandewall73, Sussman73]). Logic programming can also be seen as a way of combining declarative and procedural representation. Here logic provides both declarative semantics and procedural semantics [Lloyd84]. Such work has lead to the blurring of declarative/procedural distinctions. Though the controversy has more or less dissolved, the role of logic in knowledge representation is still in dispute.

Although the use of logic in AI has been the subject of heated debate, many AI researchers have come to recognise the need for representation languages with well-defined *semantics*. In the absence of an associated semantic theory, just what an expression in a representation language says about the world is unclear. Symbols alone are meaningless. The function of a semantic theory is to assign meanings to the symbols of the language, and to assure that all users of the language interpret the symbols in the same way. Without such a precise specification of the meaning of expressions, comparisons between different knowledge representation languages are impossible. The advantage logical languages have over other representation languages is their ready-made formal semantics (e.g., Kripke possible-world semantics, or Tarskian semantics).

McCarthy [McCarthy77] separates the use of logic into three areas (epistemological research, representation structures, and programming languages). He considers languages based on first-order logic to be suitable for use in epistemological research. Viewing the world as consisting only of individuals and relations, seems powerful enough to express most facts about the world. Thus, logic is an adequate tool for analyzing knowledge. McCarthy views the use of programs with data structures consisting of first-order sentences to be analogous to the use of an interpreter. Such representations are easy to understand and are flexible; but, they are slow to use. Since logic can be used as a programming language, logic can serve as the basis for both specification and implementation.

Given a suitable representation for the facts about the world, by what process can valid conclusions be drawn? A representation scheme must be more than just a representation language; it must also provide an *inference mechanism*, by which, implicit conclusions may be derived from explicit facts. The need for inference mechanisms does not imply the need for *logical deduction*. What is needed, however, is a non-procedural specification of what conclusions may be inferred. Here too, logic has an advantage over other representation schemes. It has a ready-made specification for the set of allowable conclusions — logical consequence. The *proof theory* of logical languages derives all the logically valid conclusions providing it is *correct*

(sound) and *complete*. The correctness and completeness of a proof procedure for a logical language is determined by appealing to the semantic theory of the language.

The intractability of all correct and complete proof procedures for first-order logic is often cited as a criticism of logic. But as noted by Brachman and Levesque [Levesque85], this is not a property of proof procedures, but of the problem itself. Any representation with the expressive power of first-order logic suffers from the same intractability problems. It is important to note that intractability is the worst case behaviour and that the average case behaviour may be quite reasonable. Brachman and Levesque suggest two ways to minimise the effects of intractability: limit the expressiveness of the language or relax the correctness and completeness requirement. Thus, the criticism of logic, which is really a criticism of knowledge representation languages in general, can be addressed.

1.4. Planning and Problem Solving

Another area of AI research is problem solving. This area investigates methods of describing and solving problems on computers. What is a problem? The concept of problem has not been completely formalised. One important paradigm in AI for the concept of problem is the *situation-space model*. Though not all problems are easily described in this formalism, the situation-space model has been found to be useful for investigating problem

solving. For example, General Problem Solver (GPS) [Newell63] used a simple version of this model.

The situation-space model consists of two kinds of entities: *situations* and *actions*. A situation is the complete state of the world at an instant of time. In real world problems, it is usually impossible to completely describe a situation. Hence, the situation-space model allows for partially described situations. An action in the situation-space model is an entity that corresponds to our intuitive notion of an action; that is, an action links a situation with the situation that results from performing the action. A *problem* in the situation-space model consists of an *initial situation*, a set of actions and their specification, and a *desired situation* or *goal*. The specification of an action consists of the rules governing its applicability and a specification of its effects (i.e., how it transforms one situation into another). A problem in the situation-space model can be further described by indicating situations to be avoided.

The situation-space model does not require complete specification of situations or actions. The result of applying an action is not necessarily a unique situation or unique partially described situation. There may be many situations in the space of possible situations that correspond to the described effects of an action. Consequently, it is not always possible to answer all conceivable questions about a situation. Such ambiguities make problem solving more difficult but not necessarily impossible.

The *solution* to a problem in the situation-space model is any sequence of actions that when applied to the initial situation leads to the desired situation while avoiding the undesirable situations. The solution may be partially specified. It could depend on conditions that cannot be determined in advance. Thus, a solution may be a simple sequence of actions or a complex network of conditionally selected paths.

Planning problems constitute an important class of problems. A *planning problem* is one that involves deciding on a course of action to achieve some purpose. The representation of a course of action is called a *plan*. It is a *solution* in the situation-space model. Similarly, the purpose to be achieved corresponds to the desired situation, and the world in which planning takes place corresponds to the initial situation. The simplicity of this transformation seems to equate planning with problem solving. The situation-space model was motivated by the planning problem, but planning is only an instance of problem solving.

In planning problems, as in all problems, knowledge representation is a major concern. How is the information about the world and its laws of change to be represented in a machine? The situation-space model gives insight into the nature of planning problems and their solutions, but it is not a knowledge representation scheme. Therefore, a representation language, an inferencing mechanism, and the corresponding semantic theory are needed to mechanise planning problems in the situation-space model.

The *situation calculus* of McCarthy and Hayes [McCarthy69] is one knowledge representation scheme (based on classical first-order logic) for the situation-space model. A problem in situation calculus is a theory in first-order logic together with a conjecture (theorem) to be proved. The logical theory describes the initial situation, the possible actions, and the rules governing their applicability. The desired situation is specified in the conjecture. The individuals of the world described by the theory are the situations and objects. Actions may be treated as either functions (see Formulation I below) or as individuals (see Formulation II below). Facts about the world which are situation independent, are expressed as ordinary propositions. In the theory ordinary propositions are typically represented by predicates.

McCarthy and Hayes introduce the notion of *fluent* for propositions and functions whose denotation depends on the situation. Assertions which vary with the situation are called *propositional fluents*. A propositional fluent is a function whose domain is the space of situations and whose range is {**true,false**}. Propositional fluents are typically represented by predicates with a situation argument. Certain functions which vary with the situation and whose values are situations, are called *situational fluents*. A situational fluent is a function whose domain and range is the space of situations.

Propositional and situational fluents are used in the *laws of motion*. These dynamic laws describe the effects of passing from one situation to

another. In the theory, laws of motion are represented by axioms involving predicates and functions, each having situation arguments (i.e., representations of propositional fluents and situational fluents). An action can be represented, in the theory, as either a function or as an individual.

In Green's formulation I [Green81], action functions are used to map one situation into another. A typical law of motion, in this formulation, is represented by an axiom of the form:

$$\forall s\ [\ P(s) \supset Q(f(s))\ ]$$

where
- **s** is a situation variable,
- **P** is a proposition describing a situation,
- **f** is an action function,
- **Q** is a proposition describing the new situation,
- **f(s)** denotes the new situation.

In contrast, Green's formulation II [Green81] treats actions as individuals. Treating actions as individuals makes it possible to talk about actions independently of situations. This method corresponds to the notion of state transformations. A state transition function is introduced that maps situations to new situations depending on the action parameter. In this formulation, typical laws of motion have the form:

$$\forall s\ [\ P(s) \supset Q(f(a_1,s))\ ]$$

where
- **s** is a situation variable,
- **P** is a proposition describing a situation,
- **f** is the state transition function,
- $\mathbf{a_1}$ is an action constant,
- **Q** is a proposition describing the new situation,
- $\mathbf{f(a_1,s)}$ denotes the new situation.

To make the use of the state transition function more convenient, Green adds the axioms (which define **nil** and **g**):

$\forall s\ [\ f(nil,s) = s\ ]$
$\forall s,a_i,a_j\ [\ f(a_j,f(a_i,s)) = f(g(a_i,a_j),s)\ ].$

The first axiom indicates that the null action has no effect. A composite action $g(a_i,a_j)$ that has the same effect as the sequential execution of a_i and a_j is defined by the second axiom. These axioms make it possible to talk about sequences of actions via an equivalent composite action. In Green's formulation II, the initial situation is represented as an individual constant and is described by a set of axioms. Given this set of axioms and the axioms for the laws of motion, certain conclusions can be drawn on the basis of the inference rules of first-order logic.

As Green [Green69] has shown, the solution to a problem represented in situation calculus can be found by *question answering* using resolution-based theorem-proving. A statement corresponding to the question: 'Does there exist a sequence of actions which lead to a situation satisfying the conditions of the desired situation?', is posed as a conjecture to be proven. The form of the conjecture is:

$\exists p\ G(f(p,s_0))$

where	p	is a sequence of actions,
	G	is a proposition describing the desired situation,
	f	is the state transition function,
	s_0	is a constant denoting the initial situation,
	$f(p,s_0)$	denotes the desired situation.

A value for the existentially quantified variable, which is a solution to the problem, is found by the theorem-proving process. Green has also shown how conditionals can be incorporated into the solution. As Green noted, formulation II is more complex than formulation I, but this representation allows quantification over, or specification of initial situation and action sequence. Thus, situation calculus (formulation II, in particular) is an epistemologically adequate representation scheme (for simple planning, at least).

Kowalski [Kowalski79] has made a distinction between problem solving by *finding* and problem solving by *showing*. Question answering without free variables corresponds to *showing* there is a solution, while question answering with free variables corresponds to *finding* a solution. For instance, the query **P(a)** is a showing problem (i.e., try to show that **P** is true of individual **a**). The query $\exists x\ \mathbf{P(x)}$ is a finding problem (i.e., try to find an individual for which **P** is true). In planning, it is important to know not only that a solution exists, but also what that solution is. Hence, planning problems are solved by *finding* a solution (i.e., a plan). In general, the search space for 'finding' problems is larger than for 'showing' problems.

One of the major difficulties in planning is that of limiting search. If the number of possible orderings of actions is considerable, how can an ordering to achieve the goal be efficiently chosen? Many approaches to planning have

been developed to address this issue. Nonhierarchical planning (e.g., STRIPS [Fikes71]) makes no distinction between actions on the basis of importance. Consequently, nonhierarchical planners are easily bogged down by unimportant details. Hierarchical planners (e.g., NOAH [Sacerdoti77]) address this problem by planning at various levels of detail. Script-based planners (e.g., MOLGEN [Friedland79]) operate by retrieving and refining stored skeletal plans. Finally, opportunistic planners have a more flexible control strategy. The Hayes-Roths' model of opportunistic planning [Hayes-Roth80] uses a blackboard control mechanism. Planning decisions are made only when there is a reason to do so. The problem of limiting search was a motivating factor in the development of these approaches.

Another major problem in planning is that of interacting subgoals. Whenever a problem has more than one condition to satisfy (*conjunctive goal*), there is the risk of having conflicting goals. Sometimes solving subgoals in the wrong order makes a solution impossible (e.g., painting the ladder before painting the ceiling). Different strategies are used by the various approaches for dealing with the interdependent subgoal problem. One strategy is based on the *linear assumption*, that is, 'subgoals are independent and thus can be achieved in an arbitrary order' [Sussman73]. Under this strategy, actions are selected without regard to ordering and when interactions arise, they are dealt with. *Least-commitment* is an alternative assump-

tion used by NOAH. Action orderings are postponed as long as possible under this strategy. *The problem of interacting subgoals and the problem of limiting search are both crucial issues in planning.*

1.5. Frame Problem

In planning, it is necessary to envision the future situations that stem from alternate courses of action; but performing actions can set off long chains of cause and effect, making it difficult to keep track of all the consequences. To determine the outcome of a sequence of actions, the effects of the actions must be specified. By describing the changes, if any, of each feature of the world, the effects of an action can be enumerated.

In the framework of logic, the effects of an action can be described by indicating, for each relation in the world, whether its truth-value changes when an action is performed. The trouble is that if there are M actions and N relations then $M \times N$ statements are required to describe all effects of the actions. If most actions resulted in drastic changes in the world (e.g., earthquakes, wars), then this would not be unreasonable; however, since most actions have only a small local effect on the world, one expects not to have to explicitly state every unaffected aspect of a situation. This intuitive knowledge that the world is relatively stable and not highly interconnected is what Simon [Simon67] calls the *empty world hypothesis*. This problem of specifying the effects of an action in a practical way (taking advantage of the empty world hypothesis) is what is known as the *frame problem*.

Hayes [Hayes73] describes the frame problem as the problem of finding adequate collections of *laws of motion*. In the situation calculus, these dynamic laws specify the effects of passing from one situation to another in terms of propositional fluents. The situation resulting from performing an action **a** in situation **s** is designated by **do(a,s)** (we will now use **do** instead of **f** or **result**). As Hayes [Hayes71] points out, laws of motion of the form **A(s)⊃B(do(a,s))** are inadequate. The situations **s** and **do(a,s)** are different (usually); therefore, inferring the properties of **do(a,s)** from the properties of **s** cannot be justified, a priori. If a propositional fluent **P** holds in situation **s**, there is no reason to deduce **P** holds in situation **do(a,s)**, even if **P** and **a** seem unrelated. So, in addition to describing properties which are changed, those which are not changed must be described as well. Thus, dynamic laws of the form **C(s)⊃C(do(a,s))** are also necessary, according to Hayes.

Laws of this form that describe invariant properties are called *frame axioms*. In general, many frame axioms are required to completely describe actions. The number of frame axioms is determined both by the number of actions and by the number of properties under consideration. For non-trivial problems, the number of frame axioms becomes unwieldy. The frame problem, cast in situation calculus, is the problem of describing actions with laws of motion and efficiently managing these laws.

The frame problem is not a result of the choice of representation scheme, nor is it simply an implementation difficulty, nor a problem resulting from the use of logical languages; it merely reflects the difficulty in expressing all the consequences of an action. The difficulty of maintaining internal representations of a changing external environment is at the heart of the frame problem. As will be seen later, *this difficulty varies with the choice of ontology*. The frame problem is a problem of dynamic representation. It is the problem of reconciling the structure of the internal representation with the changes in the world brought about by performing actions. Thus, the frame problem is a bona fide knowledge representation problem.

The example in [Raphael71] provides a illustration of the frame problem:

> "Suppose the initial world description contains the following facts (expressed in some suitable representation, whose precise form is beyond our immediate concern):
>
> (F1) A robot is at position **A**.
> (F2) A box called **B1** is at position **B**.
> (F3) A box called **B2** is on top of **B1**.
> (F4) **A,B,C,D** are all positions in the same room.
>
> Suppose, further, that two kinds of actions are possible:
>
> (A1) The robot goes from **x** to **y**, and
> (A2) The robot pushes **B1** from **x** to **y**,
>
> where **x** and **y** are in {**A,B,C,D**}. Now consider the following possible tasks:
>
> Task (1): The robot should be at **C**.
>
> This can be accomplished by the action of type **A1**, 'Go from **A** to **C**'. After performing the action, the system should 'know' that facts

F2 through F4 are still true, that is, they describe the world after the action, but F1 must be replaced by:

(F5, i.e., F1′) The robot is at position **C**.

Task (2): **B1** should be at **C**.

Now a 'push' action must be used, and both F1 and F2 must be changed.

One can think of simple procedures for making appropriate changes in the model, but they all seem to break down in more complicated cases. For example, suppose the procedure is:
Procedure (a): 'Determine which facts change by matching the task specification against the initial model.'
This would fail in task (1) if the problem solver decided to get the robot to **C** by pushing **B1** there (which is not unreasonable if the box were between the robot and **C** and pushing were easier than going around), thus changing **F2**.
Procedure (b): 'Specify which facts are changed by each action operator.'
This procedure is also not sufficient, for the initial world description may also contain derived information such as

(F6) **B2** is at position **B**,

which happens to be made false in task (2).
More complicated problems arise when sequences of actions are required.
Consider:

Task (3): The robot should be at **D** and, simultaneously, **B2** should be at **C**.

The solution requires two actions, 'Push **B1** from **B** to **C**' and 'Go from **C** to **D**', in that order. Any effective problem solver must have access to the full sets of facts, including derived consequences that will be true as a result of each possible action, in order to produce the correct sequence." [Raphael71, p. 160-161]

The frame problem is fundamentally the problem of dealing with change. Any knowledge representation scheme for worlds that are subject to change must somehow address the frame problem. Solving the frame problem means

finding a way of specifying actions, while taking into account, the heuristic knowledge that in applying actions most relations in the world are invariant.

1.6. Our Idea

This dissertation defends the thesis that *theory formation, together with a particular theory preference heuristic, is a simple, intuitively appealing way to solve the frame problem*. We call this preference heuristic chronological persistence. We claim that the only reasonable approach to formalising rational belief is to use "scientific-like" theory formation. Since viewing reasoning as theory formation, rather than as deduction from our knowledge, has a solid basis in the philosophy of science; *we believe that the theory formation/theory preference framework will turn out to be a fundamental paradigm for knowledge representation and rational reasoning in AI.*

We also propose a *continuum hypothesis* — it seems that axiomatisations based on theory formation/theory preference, circumscription, and negation-as-failure can be viewed as different points on a continuum where the degree of theory preference explicitness varies. For example, in negation-as-failure, preference information derived from syntactic structure is "compiled into" the proof procedure; while in circumscription, preference is expressed as priorities.

To support our thesis, we first trace the history of proposed solutions to the frame problem in the first half of chapter 2. The proposed solutions are divided into three groups: the axiomatisation approaches, the procedural approaches, and the consistency approaches. The second half of chapter 2 discusses in greater detail a particular consistency approach, namely, theory formation. The last section of chapter 2 discusses the need for theory preference which arises because there can be multiple theories that explain our observations.

Chapter 3 presents a theory formation representation scheme together with a semantically well-defined theory preference heuristic for dealing with the frame problem.

Chapter 4 provides a specification of a search procedure for selecting a preferred theory. As well, an implementation of this search procedure is described. The final section of chapter 4 discusses some additional implementation issues.

In the first section of chapter 5, our approach to the frame problem is compared and contrasted with other recent proposals. The next section speculates on future directions for research. In the final section, we give our conclusions.

Chapter 2
Approaches to the Frame Problem

2.1. Introduction

The literature contains many suggestions for dealing with the frame problem. These suggestions can be classified (roughly) into three groups. The first group, which we call the *axiomatisation approaches*, is characterised by descriptions, axioms, or general rules for when properties do or do not change. The second group, which we call the *procedural approaches*, addresses the frame problem by providing specialised procedures for determining the effects of actions. The final group, which we call the *consistency approaches*, makes the assumption that properties remain unchanged unless a contradiction arises.

The first part of the chapter briefly surveys the proposed solutions to the frame problem within each of these three classifications. It is important to note the *varying ontological commitments* made in each proposed solution and how those choices affect the success achieved by each approach. The last part of the chapter contains a section which describes in detail the consistency approach we have chosen to pursue, namely, theory formation. As well, a section describing theory preference is provided.

2.2. Axiomatisation Approaches

An elementary approach to the frame problem is the *complete world description approach* in which each situation is explicitly represented. Each complete world description is treated as an object and actions are treated as operators that transform one object into another. A GPS-like system can then be used for problem solving. While this method addresses the problem of specifying the effects of actions, it is not practical for worlds described by many facts, or for worlds with many possible actions, or for plans consisting of long sequences of actions. Because it becomes impractical to generate and store all the complete world descriptions in non-trivial problems, and because this approach does not take advantage of the empty world hypothesis, this approach is inadequate for solving the frame problem.

In the *assignment approach* [Manna70, Fikes70], changes in the world due to actions are modelled by the device of assignment. The current world is explicitly represented and the effects of actions are described by assignment statements. Properties that are unaffected by an action are automatically carried along when the current world is updated. The use of assignment to depict the effects of actions seems to presuppose a world with trivial physics, according to Hayes [Hayes71]. The lack of side-effects in assignments precludes the representation of complex interactions. Thus, this approach is also inadequate for non-trivial problems.

The *situation variable approach* [McCarthy69, Green81], improves on the complete world description approach. Facts are associated with named situations and actions are viewed as functions that map situation names to situation names (i.e., the name of the situation, in which an action is applied, is mapped to the name of the resulting situation). The problem with this approach is that an inordinate number of frame axioms are needed to carry unaffected facts along through situation changes (some relief is obtained by a slight change of ontology — cf. Kowalski's meta-language approach).

Another approach to the frame problem suggested by McCarthy and Hayes [McCarthy69] is based on the programming language notion of block structures (or modules). Facts are classified into groups that are action-independent (i.e., when an action is performed, it affects elements of one group while leaving the other groups undisturbed). In blocks world problems, there is a common sense notion that moving a block leaves its colour unchanged. This suggests that facts about colour and facts about location should be grouped separately; however, this grouping is inappropriate, when, for instance, the location is a bucket of paint. This approach does contribute somewhat toward solving the frame problem, but it is insufficient in view of the coarseness of classifications.

The *causal connection approach* of Hayes [Hayes71] axiomatises the dependencies between the objects in the world. The scheme provides a finer

degree of classification than the blocks approach. The basic idea is that properties of objects will not be affected by a given action unless the object and the action are causally related. The causal connection approach, in effect, axiomatises the common sense laws of cause and effect (cf. minimising uncaused changes [Lifschitz87]). This method goes a long way toward solving the frame problem and it, unlike some of the others, has a clear semantic theory. Nevertheless, it too is an inadequate solution, since it suffers from a local version of the frame problem. The local frame problem, according to Hayes, is:

> "When some property, however trivial, of an object is altered by an action, then the object must have been connected to the action, ... , and therefore any property of the object is liable to have changed when the action is performed." [Hayes71, p. 514]

Hayes observed this problem in his axiomatisation of the monkey and bananas problem. In that formulation, there is a causal connection between the thing held by the monkey and the monkey itself. There is also a connection between the monkey and the action 'move'. Thus, there is an indirect causal link between the thing held by the monkey and the action 'move'. Because of this link, extra frame axioms are needed to infer that the thing held by the monkey before it moves is still held afterwards. Thus, the causal connection approach fails to solve the frame problem.

The *world predicate approach*, due to Waldinger [Raphael71], views the entire collection of facts about a particular world as a single entity (a model M). A world predicate **P** with domain **models** $\times$ **situation-names**, is used in defining situations and laws of motion. **P(M,s)** means that **s** is the name of a world that **M** is a model of. The initial model $\mathbf{M_i}$ can be specified as a set of ordered n-tuples representing relations (e.g., **<At,Robot,A>**). The axiom $\mathbf{P(M_i,S_0)}$ defines the initial world (where $\mathbf{M_i}$ is given explicitly and $\mathbf{S_0}$ is a constant naming the initial situation). This method makes it possible to describe, in a single axiom, the relations that change when performing an action and those that do not. Raphael [Raphael71] illustrates this with:

$$\forall \mathbf{x,y,\bar{w},s}\ [\ \mathbf{P(\{<At,Robot,x>,\bar{w}\},s)} \supset \mathbf{P(\{<At,Robot,y>,\bar{w}\},go(x,y,s))}\ ]$$

where	**x,y**	are location variable
	$\mathbf{\bar{w}}$	is a variable representing all the unaffected relations
	s	is a situation variable
	P	is the world predicate
	go	is an action function
	At	is a constant representing a relation
	Robot	is a constant representing a robot.

According to Raphael, this approach has the advantages of the situation variable approach (i.e., problem solving and answer construction are handled by the theorem-prover). Furthermore, the unaffected properties of the world are automatically carried along through situation changes. In spite of these advantages, the need for an extended logic to handle sets and n-tuples, the requirement for complex pattern-matching, and the restrictions placed on

inferencing (because properties are stored as data), render this approach suspect.

Kowalski's [Kowalski79] *meta-language approach* to the frame problem treats both situations and statements as individuals. The binary relation, **holds(R,S)**, represents that statement **R** is true of situation **S**. Situations are represented by constants or by a term of the form **result(A,S)** that names the situation resulting from performing the action **A** in situation **S**. There is a straightforward translation between statements in Green's representation and statements in Kowalski's representation. The situation parameter of statements in Green's formulation is removed to form Kowalski-type statements. In Kowalski's representation, the situation parameter is an argument of the **holds** relation. For instance, **on(X,Y,S)** in Green's representation becomes **holds(on(X,Y),S)** in Kowalski's representation. This has the advantage that a single frame axiom, namely,

holds(R,S) & preserves(A,R) ⊃ holds(R,result(A,S))

where	**R**	is a statement variable
	S	is a situation variable
	A	is an action variable
	result(A,S)	is the situation resulting from doing action **A** in situation **S**
	holds(R,S)	means that statement **R** is true in situation **S**
	preserves(A,R)	means that action **A** does not change statement **R**

can be used to express all the statements that remain unchanged when an

action is performed. The problem with this method lies in the definition of preserves. Whenever the axiomatisation of the world is supplemented by additional statements it is necessary to amend the **preserves** axioms for each action. This is clearly unsatisfactory. Another problem is that Kowalski defines **preserves** in terms of a *syntactically* defined relation **diff**. This jeopardises the semantics of the frame axiom.

2.3. Procedural Approaches

Raphael's *context graph approach* [Raphael71] deals with the frame problem by augmenting the standard theorem-proving approach with a data structure (a context graph) and procedures for manipulating it. A *context* is the set of possible situations for which a given predicate is true. Predicates with parameters determine families of contexts. The initial context is the set of all possible situations satisfying the conjunction of all predicates defining the initial world. In this framework, solving a problem means finding any member of the context corresponding to the goal predicate. Actions are specified by an operator name, a parameter list, preconditions, and results. Any relation in the preconditions may be designated as *transient*. For example, Raphael defines the action **go** as:

go(x,y):	{ **Path(x,y)** ∧ **At(x)** | **At(y)** }	
where	**go**	is an operator name
	x,y	are parameters corresponding to locations
	Path(x,y)	means locations **x** and **y** are connected
	At(x)	is a transient precondition describing the robot's current location
	At(y)	describes the robot's resulting location

When an operator is applied, the defining predicate of the current context is changed by deleting transients and conjoining results, thus, defining a new (resulting) context.

Achievable contexts are those that can be reached by a finite sequence of operators. A *sufficient* context is one from which a subset of the goal context can be reached. The problem solving process, in this formulation, involves finding an achievable context that is a subset of a sufficient context.

This can be done by working forward through achievable contexts, or working backward through sufficient contexts or both. Ordinary theorem-proving is used to test operators and results and to instantiate parameters, but separate procedures are used to maintain the context graph (trees of achievable and sufficient contexts together with the operators that connect them).

Difficulties arise when dealing with dependencies. All expressions previously deduced from a transient expression may no longer hold after it is deleted. Further, it is not clear, how this approach can be adapted to worlds with incomplete information.

The *STRIPS approach* [Fikes71] is related to the context graph approach. It separates theorem-proving (used within a given world) from searching the space of world models (for which it uses a GPS-like means-end analysis strategy). This separation allows STRIPS to have more complex and general world models than GPS and more powerful search heuristics than Green's theorem-prover.

STRIPS eliminates the need for frame axioms by making the assumption that relations are not affected by a given action unless explicitly specified otherwise. This is what Waldinger [Waldinger77] calls the *STRIPS assumption*. Relations, in the STRIPS approach, are classified into *primitive* and *non-primitive*. Non-primitive relations are defined in terms of primitive ones. Actions are defined by three entities: an add list, a delete list, and a precondition expression. If a world model satisfies the precondition expression of an action then the action is applicable to the world model. The add list of an action consists of the primitive relations that are added to a world model when the action is applied. The delete list of an action consists of the primitive relations that should not be automatically carried forward by the STRIPS assumption. The initial world is given by an explicit list of relations and subsequent worlds are described by cumulative add and delete lists.

This approach works well with simple problems, but the add-delete list representation is too restrictive (difficult to describe complex actions). The STRIPS assumption breaks down in complex worlds with many interactions between objects. As well, adding a new primitive predicate requires changing the specification of all actions which affect it (i.e., specifications aren't modular). Thus, this approach is not suitable for complicated problems.

The *AI programming language approach* (e.g., PLANNER [Hewitt71], QA4 [Rulifson72], CONNIVER [Sussman72], QLISP [Wilber76]) represents

actions procedurally, but, as Waldinger [Waldinger77] noted, problem solvers implemented in this way have the STRIPS assumption structurally embedded within the language's context mechanism. Typically, a context mechanism operates in a similar way to variable scoping in conventional programming languages. Assertions made in one context will be visible in descendant contexts (unless subsequently changed or retracted). There are many examples of problem solving systems using this approach (cf. [Winograd72, Buchanan74, Fahlman74, Fikes75]). Even though complex effects of actions are easily represented in AI programming languages, the intimate structural tie to the STRIPS assumption makes this approach inadequate. It suffers from the same problems as the STRIPS approach.

2.4. Consistency Approaches

Rescher's [Rescher64] approach to the analysis of *counterfactual reasoning* can be applied to the frame problem. The method employed in this approach is to transfer properties from a situation to its successor as long as it is consistent to do so. Thus, the effect of performing an action **a** in situation **s** can be described as follows. If Γ_s is the set of properties describing the situation **s**, and $\Gamma_{do(a,s)}$ is the set of properties describing situation **do(a,s)**, and $\Phi_{do(a,s)}$ is the set of properties of situation **do(a,s)** directly inferred from the laws of motion ($\Phi_{do(a,s)} \subset \Gamma_{do(a,s)}$), then $\Gamma_{do(a,s)}$ is a maximal consistent subset (MCS) of the possibly inconsistent set $\Gamma_s \cup \Phi_{do(a,s)}$, that is:

$$\Gamma_{do(a,s)} \in \mathbf{MCS}\ [\ \Gamma_s \cup \Phi_{do(a,s)}\ ]\ \wedge\ \Gamma_{do(a,s)} \supset \Phi_{do(a,s)}.$$

According to Hayes [Hayes73], there are two major problems with this approach:

1) Consistency is not decidable;
2) The MCS of an inconsistent set is not necessarily unique.

Rescher deals with the second problem through the use of *modal categories*. These are hierarchical classifications of assertions. Thus, in addition to being an MCS, a candidate for $\Gamma_{do(a,s)}$ must also satisfy a hierarchical restriction, namely, preference is given to including properties that are in lower modal categories. Formally,

$$[\ \phi_j \notin \Gamma_{do(a,s)}\ \&\ \phi_j < \phi_i\ ] \supset [\ (\ \Gamma_{do(a,s)} - \{\phi_i\}\ ) \cup \{\phi_j\}\ \textbf{is inconsistent}\ ]$$

where $\phi_j < \phi_i$ if property ϕ_j is in a lower modal category than property ϕ_i.

Thus, if there is an MCS that includes ϕ_j then that MCS is preferred over another MCS which differs from the first by including ϕ_i instead of ϕ_j. One way to define the modal categories is to relate them to causal connection [Simon66] (i.e., $\phi_j < \phi_i$ if ϕ_j causes ϕ_i or $\neg\phi_i$). The problems of this approach are common to all consistency-based methods. Thus, consistency-based solutions of the frame problem all involve finding practical ways of dealing with the problems of decidability and multiple maximal consistent subsets.

The *formal literature approach* [McCarthy69 also cf. Brown86] is another consistency-based approach to the frame problem. A formal literature may be thought of as a formal language with a history. The possible sentences of a literature depend on previously asserted sentences (e.g., new terms can not be used until they are defined and their usage is restricted by their definition). An example of a formal literature is first-order logic extended to include the modal operators: **consistent**(ϕ), **normally**(ϕ), and **probably**(ϕ), together with the following rules of inference:

1. If ϕ is consistent with σ (the set of sentences in the literature so far) then **consistent**(ϕ) may be added (this rule is non-computable, in general)
2. **normally**(ϕ), **consistent**(ϕ) $\vdash$ **probably**(ϕ)
3. $\phi \vdash$ **probably**(ϕ)
4. If $\phi_1, \phi_2, \cdots, \phi_n \vdash \phi$ then
   probably(ϕ_1), $\cdots$,**probably**(ϕ_n) $\vdash$ **probably**(ϕ).

Whenever **probably**(π) and **probably**($\neg\pi$) (and therefore, **probably(false)**) occur, a search for a contradiction is called for. This approach is unnecessarily complicated since it attempts to solve both the frame problem and the problems associated with formalising the concepts of consistency, normality, and likelihood in one fell swoop rather then dealing with the problems separately. As well, this approach does little to solve the decidability and MCS problems. In fact, things are made even worse because consistency isn't finitely axiomatisable.

Sandewall's [Sandewall72] approach to the frame problem is to introduce the modal operator **UNLESS**, together with a frame inference rule:

$$\frac{\mathbf{IS(o,p,s)} \quad \mathbf{UNLESS(\ ENDS(o,p,succ(s,a))\ )}}{\mathbf{IS(o,p,succ(s,a))}}$$

where	**IS(o,p,s)**	means object **o** has property **p** in situation **s**
	succ(s,a)	means the situation resulting from doing action **a** in situation **s**
	ENDS(o,p,succ(s,a))	means object **o** ceases to have property **p** in situation **succ(s,a)**
	IS(o,p,succ(s,a))	means object **o** has property **p** in situation **succ(s,a)**.

The frame inference rule is intended to mean: if **IS(o,p,s)** is provable and **ENDS(o,p,succ(s,a))** is not provable then **IS(o,p,succ(s,a))** is proven.

This formalism suffers from the same problem as the other consistency-based methods, since the provability of an expression is undecidable and since multiple MCSs are still a problem. An example illustrating the MCS problem is:

$$\mathbf{A}$$
$$\mathbf{A} \wedge \mathbf{UNLESS(B)} \supset \mathbf{C}$$
$$\mathbf{A} \wedge \mathbf{UNLESS(C)} \supset \mathbf{B}.$$

In this example, there are two MCSs: one of these contains B and the other contains C. Sandewall suggested three ways of dealing with this problem:

1. use a precedence ordering on the axioms;
2. restrict the expressiveness of the language to prevent such situations from occurring; or,
3. allow a sentence to be: a theorem, not a theorem, or undetermined.

None of these suggestions is completely satisfactory. The first suggestion gives no advice on how to arrive at the precedence ordering in advance. As well, it is not clear that there always exists a precedence ordering which uniquely determines the desired consistent subset. The second suggestion is also unsatisfactory because it may be necessary to severely restrict the expressiveness of the language to eliminate multiple MCSs. Therefore, interesting problems may no longer be expressible. The last suggestion is the least desirable of the three because it may be impossible to infer anything useful. Unless a statement is true in every MCS or false in every MCS then its truth value is undetermined. Overall, Sandewell's approach, like the formal literature approach, fails to separate the representation and reasoning aspects of the frame problem (since **UNLESS**, which essentially means not provable, is a reasoning component).

The advent of nonmonotonic reasoning systems [McCarthy80, Reiter80, McDermott80] has brought on a resurgence of effort into solving the frame problem. Initial attempts using these systems met with difficulty [Hanks85]. Hanks and McDermott found that using default (or other nonmonotonic)

reasoning to deal with the frame problem inevitably results in the need to choose between extensions (or between models). They essentially rediscovered the multiple MCS problem [Hayes73]. Several researchers have turned their attention to this problem. The discussion of their work will be postponed until chapter 5 where we can compare their results with our own. The rest of the chapter will discuss the particular consistency approach that we have chosen to pursue, namely, theory formation.

2.5. Theory Formation

The development of nonmonotonic reasoning systems was, in part, motivated by the inability of logical deduction to capture certain forms of rational inference. In order to draw useful conclusions about a domain for which our knowledge is incomplete or inaccurate, we must make assumptions. Given additional information, we may retract some of our assumptions, and with them, the conclusions they entail.

Attempts to equate logical deduction with rational inference quickly run into trouble. Logical deduction is monotonic, that is, the conclusions sanctioned by a set of axioms are still sanctioned when the set of axioms is augmented with new axioms. Thus, McDermott and Doyle [McDermott80, p. 44] note that: "Monotonic logics lack the phenomenon of new information leading to a revision of old conclusions". Because logical deduction is monotonic and because rational inference is not, attempts were made to extend

classical logic to allow nonmonotonic reasoning [McCarthy80, Reiter80, McDermott80].

Israel has critisised these formalisations of nonmonotonic reasoning because, as he points out, they are based on a misconception.

> "The researchers in question seem to believe that logic — deductive logic, for there is no other kind — is centrally and crucially involved in the fixation and revision of belief. Or to put it more poignantly, they mistake so-called deductive rules of inference for real, honest-to-goodness rules of inference. Real rules of inference are precisely rules of belief fixation and revision; deductive rules of transformation are precisely not." [Israel80, p. 100]

While logical deduction is useful in determining what conclusions follow from a set of axioms, it says nothing about how to rationally determine and maintain a set of beliefs. Thus, Israel goes on to say:

> "The crucial point here, though, is that adherence to a set of deductive rules of transformation is not a sufficient condition for rational belief; it is sufficient (and necessary) only for producing derivations in some formal system or other. Real rules of inference are rules (better: policies) guiding belief fixation and revision. Indeed, if one is sufficiently simple-minded, one can even substitute for the phrase 'good rules of inference', the phrase '(rules of) scientific procedure' or even 'scientific method.'" [Israel80, p. 100]

It seems that Israel believes that what's necessary for the formalisation of rational belief is not an extension of logic, but rather the specification of a set of rational epistemic policies of belief fixation and revision. Furthermore, these policies must necessarily be heuristic in nature and are akin to scientific procedures.

> "... reflect on how odd it is to think that there could be a purely proof-theoretic treatment of scientific reasoning. A heuristic treatment, that is a treatment in terms of rational epistemic policies, is not just the best we could hope for. It is the only thing that makes sense." [Israel80, p. 101]

Thus, the only reasonable approach to formalising rational belief seems to be to use "scientific-like" theory formation. Such an approach has been taken by Poole and his colleagues [Poole87b] in the Theorist project at the University of Waterloo. Based on a philosophy inspired by Popper [Popper58], Theorist views reasoning as scientific theory formation (rather than as deduction). Science is concerned, not merely with collecting facts, but with explaining them. Consequently, reasoning in the Theorist framework involves building "scientific" theories that explain a set of observations.

The scientific method has been very fruitful. Consider, for example, the Russian chemist Dmitri Mendeleev. By 1869, sixty-three elements had been discovered. Mendeleev arranged these elements in a table according to their atomic weights and chemical properties. In doing this, he was able to show a relationship between atomic weight and chemical properties. His periodic table expressed the theory that chemical properties are the consequence of atomic weight.

This theory led to the discovery of new elements and the prediction of their properties. From the positions left blank in the table, he described the chemical properties of three undiscovered elements. These were discovered

during the next 15 years. By using his table, he found that the properties predicted by the atomic weight of gold disagreed with the observed properties. This inconsistency between theory and observation led to the discovery that the previously supposed atomic weight of gold was incorrect. The theory expressed by the periodic table withstood testing, and was, therefore, corroborated by the evidence. The methodology of observation, hypothesis, prediction, testing, and evaluation, which has proved so fruitful in science, should also be useful when applied to prescientific, commonsense reasoning and rational belief.

What are scientific theories? This question has been studied in great detail by philosophers of science [Popper58, Hempel65, Rescher70]. To them, theories are sets of sentences — usually general sentences or laws. These laws express relations that hold between various properties. Nomic necessity and hypothetical force distinguish laws from ordinary generalisations. To say that a generalisation has nomic necessity is to say that the relationship it expresses is somehow necessary. This element of necessity extends to the unobserved, the unrealised, and the hypothetical counterfactual. For example, from the law that objects denser than water sink, we can infer that if an ice cube were denser than water (which it isn't) then it would sink.

Thus Rescher describes lawfulness in the following:

> "It is built into our very concept of a law of nature that such a law must, if it be of the universal type, correspond to a universal generalization that is claimed to posses nomic necessity and is denied to be of a possible merely accidental status. If the generalizations were claimed to hold *in fact* for all places and times, even this would not of itself suffice for lawfulness: it would still not be a law if its operative effectiveness were not also extended into the hypothetical sphere. The conception of a universal law operative in our concept of causal explanation is thus very complex and demanding. A lawful generalization goes beyond claims of a merely factual generalization as such; it involves claims not only about the realm of observed fact, but about that of hypothetical counterfact as well. And just these far-reaching claims are indispensable to the acceptance of a generalization as lawful and is a formative constituent of our standard concept of a universal law of nature." [Rescher70, p. 102]

A question that has troubled philosophers since early times is: 'Where do laws come from?' Given that universal laws govern the unobserved, it is clear that they cannot be arrived at deductively from our experience; similarly, given that universal laws govern the unobservable, the hypothetical, and the counterfactual, it is clear that they cannot be arrived at inductively either. The question remains, therefore, how do we come to know laws — necessary and universal truths — given that our experience is limited to the actual and the particular? To this question, we might add: 'How do laws acquire nomic necessity and hypothetical force?' Rescher answers these questions as follows:

> "The basic fact of the matter — and it is a fact whose importance cannot be overemphasized — is that the elements of nomic necessity and hypothetical force are not to be extracted from the evidence. They are not *discovered* on some basis of observed fact at all; they are *supplied*. The realm of hypothetical counterfact is inaccessible to observational or experimental explanation. Lawfulness is not found

> in or extracted from the evidence, it is superadded to it. *Lawfulness is a matter of imputation.* When an empirical generalization is designated as a law, this epistemological status is *imputed* to it. Lawfulness is something which a generalization could not *in principle* earn entirely on the basis of warrant by the empirical facts. Men impute lawfulness to certain generalizations by according them a particular role in the epistemological scheme of things, being prepared to use them in special ways in inferential contexts (particularly hypothetical contexts), and the like.
>
> When one looks at the explicit formulation of the overt *content* of a law all one finds is a certain generalization. Its lawfulness is not a part of what the law asserts at all; it is nowhere to be seen in its overtly expressed content as a generalization. Lawfulness is not a matter of what the generalization *says*, but a matter of *how it is to be used.* By being prepared to put it to certain kinds of uses in modal and hypothetical contexts, it is *we*, the users, who accord to a generalization its lawful status thus endowing it with nomological necessity and hypothetical force. Lawfulness is thus not a matter of the assertive content of a generalization, but of its epistemic status, as determined by the ways in which it is deployed in its applications." [Rescher70, p. 107]

The key point to draw from this is eloquently summarised in the last sentence of the above quote: "Lawfulness is thus not a matter of the assertive content of a generalization, but of its epistemic status, as determined by the ways in which it is deployed in its applications." As we shall see, the power of the Theorist system derives from the special epistemic status it gives to possible hypotheses. The nonmonotonic nature of Theorist results from the way in which possible hypotheses are used. Before considering this further, let us examine the issue of scientific explanation.

What is it to give an explanation? The deductive-nomological model of explanation [Hempel48] defines an explanation as a valid deductive argument. The premisses of the argument include nomological general statements (laws) and other singular statements (called initial conditions). These statements deductively entail the conclusion. In this sense, the conclusion is explained by the premisses, that is, to explain a state of affairs is to deduce its description from a law. Thus, scientific explanation involves a subsumption argument under laws.

A distinction can be made between *potential* explanations and *actual* explanations. A *potential* explanation is a valid deductive argument whose premisses entail the conclusion. For an explanation to be an *actual* explanation, its singular premisses must be true and its general premisses must be well-confirmed, lawful generalisations.

When the element of time is crucially involved in an explanation, we call it a *temporal* explanation. Two important kinds of temporal explanation are: *prediction* and *retrodiction*. Note that the term prediction (and retrodiction), as used here, refers to the explanation itself rather than the conclusion. Just as an explanation can be actual or potential, predictions and retrodictions can also be actual or potential.

A *potential* prediction has the structure of a potential explanation where a time dependent conclusion follows from premisses consisting of general state-

ments and time dependent initial conditions. As well, the time parameters of the premisses must predate the time parameter of the conclusion. For example, the following is a potential prediction of **Q(t)**:

$$\frac{C_1(t_1), \ldots, C_m(t_m) \quad \text{(Initial Conditions)}}{L_1, \ldots, L_n \quad \text{(universal generalisations)}} \qquad Q(t) \quad \text{(conclusion)}$$

provided each $t_i < t$. In addition to the requirements of a potential prediction, an *actual* prediction must have fact-asserting singular premisses, and well-confirmed, law-asserting general premisses. Furthermore, the time parameters in the predictive argument are constrained as follows: if the present time is t_{now} then each $t_i \leq t_{now} < t$.[1]

A second form of temporal explanation is *retrodiction*. A *potential* retrodiction has the structure of a potential explanation where a time dependent conclusion follows from premisses consisting of general statements and time dependent initial conditions. As well, the time parameters of the premisses must postdate the time parameter of the conclusion. For example, the following is a potential retrodiction of **Q(t)**:

---

[1] See [Rescher70, pp. 33ff.] for a discussion of the need to constrain the chronological direction of predictive arguments.

$$\frac{C_1(t_1),\ \ldots,\ C_m(t_m) \qquad \text{(Initial Conditions)}}{L_1,\ \ldots,\ L_n \qquad \text{(universal generalisations)}}$$
$$Q(t) \qquad \text{(conclusion)}$$

provided each $t_i > t$. In addition to the requirements of a potential retrodiction, an *actual* retrodiction must have fact-asserting singular premisses, and well-confirmed, law-asserting general premisses. Furthermore, the time parameters in the retrodictive argument are constrained as follows: if the present time is t_{now} then $t_{now} > t$.

The definitions above correspond to the deductive nomological model of explanation. In this model, explanation takes the form of a deductive argument. If the explanatory premisses are true, a deductive argument provides conclusive evidence for the conclusion. When our premisses are not so certain as to allow such conclusive explanations, we have at least two alternatives: we can turn to probabilistic explanations (cf. [Rescher70, Hempel65, Carnap50]), or we can tentatively assume the premisses are true and treat the explanation as potentially refutable.

The latter mode of explanation above is described by Popper [Popper58]. In this approach, the premisses of a deductive argument can be of two types: those that we accept as true, and those we treat as assumptions. These two types of premisses taken together form a potential explanation of the deductively entailed conclusions. In the spirit of the scientific method, a potential

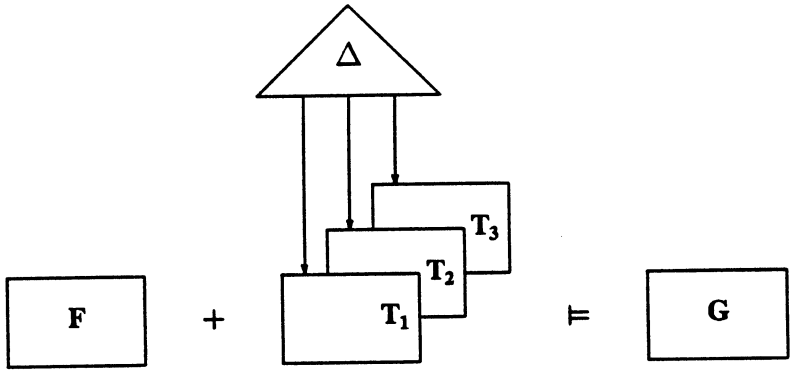
explanation can be subjected to crucial experiments, and having survived the tests, the potential explanation becomes well-confirmed. Thus Popper writes:

> "A scientist, whether theorist or experimenter, puts forward statements, and tests them step by step. In the field of the empirical sciences, more particularly, he constructs hypotheses, or systems of theories, and tests them against experience by observation and experiment." [Popper58, p. 27]

> "From a new idea, put up tentatively, and not yet justified in any way — an anticipation, a hypothesis, a theoretical system, or what you will — conclusions are drawn by means of logical deduction. These conclusions are the compared with one another and with other relevant statements, so as to find what logical relations (equivalence, derivability, compatibility, incompatibility) exist between them." [Popper58, p. 32]

This is the philosophy underlying the Theorist framework [Poole87b]. Instead of viewing reasoning as deduction from our knowledge, reasoning may be better modelled by scientific theory formation. Consequently, reasoning in the Theorist framework involves building "scientific" theories that explain a set of observations. While the intuition underlying Theorist stems from the deductive nomological model of explanation described above, there is a notable difference. The Theorist framework does not require explanations to contain lawful general statements. Thus, explanations in Theorist are usually only potential explanations. Because of this, it may be more appropriate to think of Theorist's explanations as "prescientific" (or commonsense) theories.

Figure 2.1 illustrates the Theorist framework.



Find T_i, a set of ground instances of elements of Δ, such that:
$F \cup T_i \models G$, and
$F \cup T_i$ is consistent
where Δ is a set of possible hypotheses (or defaults)
T_i is a set of ground instances of possible hypotheses
F is a set of facts (statements to which we are committed)
G is a set of observations (or desired conclusions).

Figure 2.1. The Theorist Framework

The current implementation of Theorist[2] uses full first-order clausal logic as its representation language. Statements are divided into two types: *facts* and *possible hypotheses* (or defaults). Facts are statements that we accept as true and thus constitute ordinary assertions in a logical theory. Possible

[2] All examples in this dissertation were run on Theorist version 0.21 which compiles statements to Waterloo Unix Prolog (WUP) [Cheng84].

hypotheses, however, are given a different epistemological status than facts. They are used as a kind of schema for axioms — possible hypotheses can be viewed as the specification for the generation of logical theories which are extensions of the facts. Instances drawn from the set of possible hypotheses can be used to construct consistent explanations for a set of observations (or desired conclusions).

Let **F** be the set of facts, Δ the set of possible hypotheses, $\mathbf{A}=\{\mathbf{d}_1,\mathbf{d}_2,\ldots\}$ the set of instances of elements of Δ, and **G** the set of observations. Theorist reasons by first trying to deduce **G** from **F**. Failing that, it next tries to find elements of **A** which conjoined to **F** allow **G** to be deduced. Thus, Theorist constructs explanations of the form $\mathbf{F} \cup \mathbf{T}_i$ where $\mathbf{T}_i$ is a subset of the elements of **A** (i.e., a set of instances of possible hypotheses).

It is important to note here that $\mathbf{F} \cup \mathbf{T}_i$ is an ordinary logical theory and therefore has well-defined semantics. In contrast, $\mathbf{F} \cup \Delta$ is *not* a logical theory since it contains defaults. In essence, defaults can be viewed as a form of axiom schemas together with the meta-inference rule: "use an instance of the axiom schema if it is consistent to do so."

Returning to the process of explanation construction, we note that in addition to the requirement that $\mathbf{F} \cup \mathbf{T}_i$ constitute an explanation of **G**, that is, $\mathbf{F} \cup \mathbf{T}_i \models \mathbf{G}$, there is also the requirement that the explanation be consistent. In other words, we require that $\mathbf{F} \cup \mathbf{T}_i$ be consistent. This con-

sistency check can be viewed as submitting a potential explanation to "scientific" testing.

That determining the consistency of a set of first order axioms is, in general, undecidable should not be seen as blemish on the methodology underlying Theorist. Rather, we should be surprised if theory formation were decidable. In practice, we could submit our explanations to a partial consistency check. Passing the partial consistency test can be viewed as confirming evidence for the potentially refutable explanation. This view of reasoning has a basis in the philosophy of science (cf. [Popper58, Rescher70, Hempel65]).

2.6. Theory Preference

It is well known in deductive logic that there can be many arguments for the same conclusion. The same is true in theory formation — there can be many potential explanations of the same observation. Faced with a multiplicity of explanations and a desire to draw useful conclusions, we are often forced to chose between the competing theories. For example, if upon returning home you find your door open, it could be explained at least two ways. One possibility is that you merely forgot to close it when you left. Another possibility is that a robber is lurking inside. Each explanation indicates a different course of action. Therefore, a choice between these potential explanations is necessary. In this case, acting on the basis of the second possibility seems to be the most prudent course of action.

In general, there can be many factors that influence the choice between competing explanations. In science, there are several criteria for theory preference. The relative simplicity of a theory, its explanatory strength and power, its degree of confirmation, its coherence with the overall system of scientific explanations, its predictive power — all these play a role in determining which theory to prefer from among competing alternatives. For example, the Copernican view of the solar system was preferred over the Ptolemaic view because its relative simplicity. Of the criteria above, two need to be defined: explanatory strength and explanatory power. The degree to which one explanation renders a conclusion more likely than an alternative explanation is its explanatory strength. The explanatory power of an explanation is the degree to which an explanation renders a conclusion more likely than any alternative conclusion. The criteria above are not the only criteria for theory preference. Indeed, it seems that the criteria for theory preference are, at least in part, domain dependent.

The issue of theory preference has been considered for several domains [Poole87b, Poole85a, Jones85, Poole86, Jackson87]. In domains with inheritance hierarchies, the explanation that uses the most specific knowledge is preferred; while in learning domains, the most general explanation is desired; in diagnosis, the most probable, the most serious, or the most specific explanation may be sought; and in analogical reasoning, the simplest and most

relevant explanation is preferred. Preference criteria for vision [Gagné86] and for user modelling [vanArragon86] are also being investigated.

The scientific method suggests a general methodology for knowledge representation and reasoning in AI. We can represent our domain knowledge as facts, and then provide possible hypotheses to supplement our incomplete domain knowledge. From these possible hypotheses, theories can be constructed to explain our observations. When there are multiple conflicting potential explanations, then theory preference can be used to choose between them. Thus, in addition to the facts and possible hypotheses, theory preference heuristics (heuristics because theories are potentially refutable) must be defined.

Chapter 3
Planning in a Theory Formation Framework

3.1. Introduction

In reasoning about actions, one generally assumes that most things are unaffected by a given action. McCarthy calls this the "common sense law of inertia." From this assumption, we expect that relations not explicitly changed by an action are unchanged after performing the action; that is, we expect the truth-value of most relations to *persist* when performing actions. In this chapter, we will develop a representation that incorporates this assumption about persistence. This notion of persistence will be fundamental to our planning representation.

3.2. Representing Planning Problems in Theorist

The first step in the process of representing planning problems is to formalise the notion of a planning problem. As we have seen, the situation-space model is one way to do this. Armed with this formalisation, the next step in the representation process is to develop or choose a suitable knowledge representation scheme in which to express the problem.

For a knowledge representation scheme to be considered suitable for planning, it must address the frame problem. Representation schemes which support consistency-based reasoning seem the most suitable, as consistency-based methods offer promise in solving the frame problem. One such knowledge representation scheme, Theorist [Poole87b], in addition to supporting the use of consistency-based methods, has a clean well-defined semantics, making it an ideal candidate in which to cast planning problems. How then, can planning problems be represented within this framework?

Having chosen the situation-space model as a paradigm for the concept of problem and having chosen Theorist as a knowledge representation scheme for planning problems, it must next be determined how to integrate these choices. The question that must be answered is: 'How are the entities of the situation-space model (situations and actions) to be expressed in the representation language of Theorist?' This, of course, depends on what the representation language is.

The Theorist knowledge representation scheme does not restrict the choice of representation language — almost any language that allows contradictions will do [Goebel85] — we make the assumption here that the representation language is full first-order (clausal) logic. Given this assumption, the question posed above becomes: 'How are the entities of the situation-space model to be expressed in first-order logic?' This has already been accomplished in situation calculus.

Recall, three variants of situation calculus are: Green's formulation I, Green's formulation II, and Kowalski's meta-language approach. The first of these, as was seen previously, is not as robust as the second or third, so it will not be considered any further. Both the second and third variants of situation calculus form suitable foundations on which to build representations for planning problems in Theorist.

How can the above representational building blocks be used by Theorist to represent the situation-space model? To begin to answer this, it will be helpful to consider how first-order logic is used by each of these two variants of situation calculus.

3.2.1. Review of Situation Calculus

The first-order logic characterisation of the situation-space model begins by classifying its entities (situations, actions, objects, properties, etc.). Entities are classified as either individuals or relations. *One possible classification*[3] admits situations, actions, and objects as individuals in the ontology of the representation language (we shall call this *ontology-G* — which corresponds to that of Green's formulation II). Another possible ontological commitment (choice of individuals) is to treat statements, situations, actions, and objects as individuals (we shall call this *ontology-K* — which corresponds to Kowalski's representation).

---

[3] Note that the frame problem must be solved with respect to a particular choice of ontology.

In both ontologies, situations are named by functions that map situations and actions to situations. Terms consisting of *constant symbols* and *function symbols* are used to represent individuals. In ontology-G, the initial situation and the effects of actions are described by axioms involving relations on situations, whereas in ontology-K the initial situation and the effects of actions are described by axioms involving a single relation (represented by the predicate **holds**) that relates statements to situations. As was mentioned previously, special axioms called frame axioms are used in situation calculus to describe properties that persist from one situation to another. All of these axioms are represented by expressions made up of terms, predicates, and connectives.

In *using*[4] this representation, the properties of the goal or desired situation are treated as conjectures (or theorems) to be proven. The solution to the planning problem is a sequence of actions that leads to a situation satisfying the desired properties. This sequence of actions can be extracted from the situation parameter in the proven conjectures.

An important property of these characterisations of the situation-space model is that they have well-defined semantics. Since planning problems in the situation calculus are formalised in first-order logic, standard Tarskian semantics can be used to specify their meaning.

---

[4] Note that goal-directedness is not part of the representation but part of its use.

To illustrate situation calculus representations of planning problems, consider the axiomatisation of the Yale shooting scenario (cf. [Hanks86]).

A = {	The set of axioms (A):
	Initial Situation:
¬ loaded(0),	The gun is not loaded
alive(0),	John is alive
	Action: load
loaded(do(load,S)),	The gun is loaded after the action load
	Action: wait (no known changes)
	Action: shoot
¬ alive(do(shoot,S)) ← loaded(S),	John dies when shot with a loaded gun
¬ loaded(do(shoot,S)),	After shooting, the gun is not loaded
	Frame Axioms:
loaded(do(wait,S)) ↔ loaded(S),	
alive(do(load,S)) ↔ alive(S),	
alive(do(wait,S)) ↔ alive(S),	
alive(do(shoot,S)) ↔ alive(S)	
∧ ¬ loaded(S)}	

Figure 3.1. Yale Shooting Scenario - Axiomatisation for Ontology-G

In this example, there are two relations: **loaded** and **alive**. The relation **loaded** is true of a situation when there is a loaded gun aimed at John. Similarly, the relation **alive** is true of a situation whenever John is alive in that situation. Initially, the gun is loaded and aimed at John, who is alive. There are three actions defined in this example: **load**, **wait**, and **shoot**. For simplicity, these actions have no preconditions. The first action, **load**, has the effect that, whenever it is performed, the gun is loaded in the resulting situation. The second action, **wait**, has no known effects. The final action,

A = {	The set of axioms (A):
	Initial Situation:
¬ holds(loaded,0),	The gun is not loaded
holds(alive,0),	John is alive
	Action: load
holds(loaded,do(load,S)),	The gun is loaded after the action load
	Action: wait (no known changes)
	Action: shoot
¬ holds(alive,do(shoot,S)) ← holds(loaded,S),	John dies when shot with a loaded gun
¬ holds(loaded,do(shoot,S)),	After shooting, the gun is not loaded
	Frame Axioms:
holds(loaded,do(wait,S)) ↔ holds(loaded,S),	
holds(alive,do(load,S)) ↔ holds(alive,S),	
holds(alive,do(wait,S)) ↔ holds(alive,S),	
holds(alive,do(shoot,S)) ↔ holds(alive,S)	
∧ ¬ holds(loaded,S)}	

Figure 3.2. Yale Shooting Scenario - Axiomatisation for Ontology-K

shoot, has the effect that, whenever it is performed in a situation in which a loaded gun is aimed at John, he is not alive in the resulting situation. As well, the gun is no longer loaded.

Figures 3.1 and 3.2 are axiomatisations for ontology-G and ontology-K respectively. Note that the frame axioms of ontology-K can be replaced by the single frame axiom

holds(R,do(A,S)) ↔ holds(R,S) ∧ preserves(A,R),

together with a suitable definition for **preserves(A,R)**. Thus, *the choice of ontology can aid in solving the frame problem.*

3.2.2. Casting the Situation-Space Model in Theorist

Both ontology-G and ontology-K form suitable foundations on which to build representations for planning problems in Theorist; but, the treatment of axioms is different in Theorist than in situation calculus. Axioms in Theorist are represented as either *facts* or *defaults* (possible hypotheses). To specify a planning problem, we must describe the initial situation and the laws of motion. For now, we will restrict our attention to worlds where the initial situation can be described solely by facts.

The laws of motion are treated differently. The effects of an action partition relations describing a world into two groups: relations that are known to be changed by the performance of the action form one group, and all other relations — those *presumed* to be unaffected by the performance of the action — form the other group. Laws of motion describing the relations that are known to change are expressed as facts, while the laws of motion for the relations presumed invariant are expressed as defaults (as suggested by Reiter [Reiter80, p. 85]). These defaults correspond to frame axioms [Green81]. Collections of ground instances of these *frame defaults* form theories from which predictions can be made.

For some problems, it is desirable to treat some action effects and some initial conditions as defaults. For example, the normal effect of an action can be represented as a default. As well, the closed world assumption can be

expressed with defaults. In this chapter, however, we restrict the set of possible hypotheses to frame defaults. This is done in order to concentrate on the problem of defining a preference measure for planning without the need to consider possible interactions between various types of defaults. While we expect that our preference heuristic will have to be modified to allow general hypotheses, it is hoped that the modifications will not be too severe. We will leave this question open for further study.

Once a planning problem has been represented as described above, it is solved by finding a situation that satisfies the goal description. This situation is named by the sequence of actions from which it results, and is described by the facts together with a supporting *persistence theory* (a collection of ground instances of frame defaults). In this framework, a solution to a planning problem has two components: a *plan* (extracted from the term that names the goal situation) which achieves the goal, and a persistence theory which predicts the goal description.

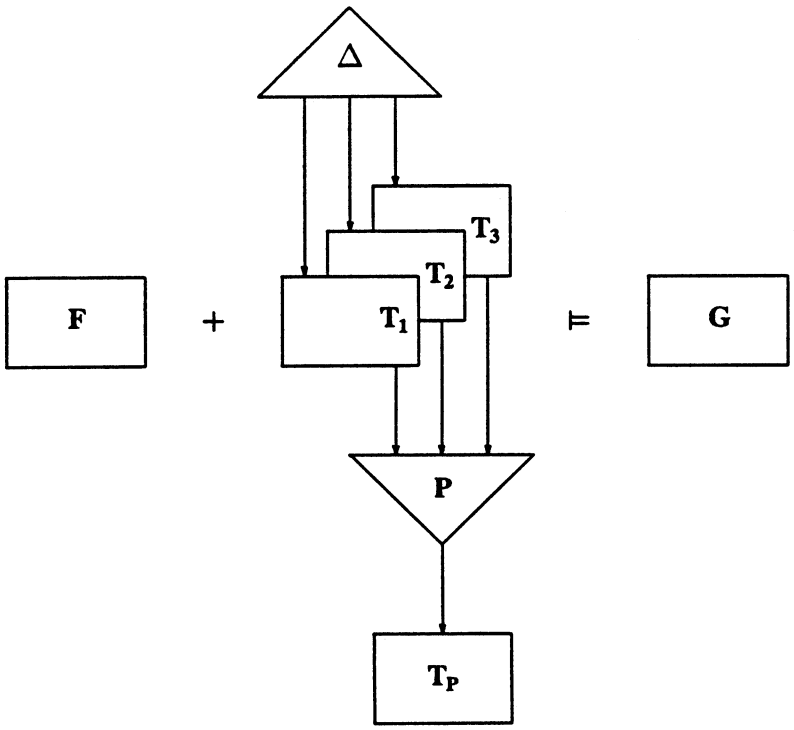
Since it is not known in advance whether the goal description or its negation will hold, they are treated as competing *predictions*. From the theories predicting the goal or its negation, one seeks a preferred persistence theory from which the expected truth of the goal description is determined. The intuition is to select a preferred theory that is most likely, in some sense. For instance, it would be reckless to prefer a theory *merely* because it predicts the

goal will hold! Therefore, the derived plan (and hence its quality) depends on the preferred persistence theory. In turn, the actions that are presumed applicable and the presumed effects of those actions depend on the predictions of the preferred theory.

This is quite different from the treatment of symptoms in diagnosis, where symptoms are *observations* to be explained. There is, however, a close relationship between observations and predictions; both are consequences of a theory of the world. The difference between them is that observations (if accurate) are true in the world whereas predictions may or may not be true in the world. From this we see that the quality of a heuristic measure of preference depends on its ability to select a theory that makes predictions which are usually true in the world. From the above, we see that their is an important difference between *observation and prediction.* The issue of theory preference is considered more closely in section 3.4. Figure 3.3 depicts planning in the Theorist framework (cf. [Poole87b]).

3.2.3. The Impact of True Negation

The choice of representation language — full first-order logic in this case — affects how and what facts are expressed. In typical planning systems based on Horn clause logic, only statements in positive form are asserted (e.g., **dead** instead of ¬**alive**). Negative information is deduced by employing the *negation as failure* rule [Clark78]. In contrast, the Theorist



Find T_i, a set of ground instances of elements of Δ, such that:
$F \cup T_i \models G$, and
$F \cup T_i$ is consistent
where Δ is a set of frame defaults
T_i is a set of ground instances of frame defaults
F is a set of facts describing the initial situation and laws of motion
G is a goal description (or its negation).
Then choose $T_P \in \{T_1, T_2, \ldots\}$
such that $\forall i\ T_P >^{cp} T_i$
where $>^{cp}$ is a theory preference heuristic
T_P is a preferred theory
P is a procedure for choosing a preferred theory.

Figure 3.3. Planning in a Theory Formation/Theory Preference Framework

framework, with full first-order logic, uses true negation to express negative information.

The use of true negation affects the way in which the domain of interest is axiomatised. Kowalski [Kowalski79] has noted that in answering queries involving universal quantification and negation, the full *iff* form of definitions is needed (whereas Horn clause planners provide only the *if* halves of definitions). Thus, when describing relations in the domain, care must be taken to provide both the *if* halves and the *only-if* halves of definitions. In Theorist, this is especially important as failure to provide the *only-if* half of a definition allows models that are inconsistent with the intended definition. Some theories that are inconsistent with the intended model are not inconsistent with the models of the underspecified logical theory.

In addition to its effect on the representation of facts, the explicit representation of negative information also affects the form of the representation of defaults. In particular, it will soon become apparent (cf. section 3.2.4.1) that the direct representation of the usual frame axioms as defaults is not sufficient to express the notion of persistence (negative information must be taken into account). Therefore, because of the explicit representation of negative information, extra care must be taken to ensure that the specification of problems correspond to their intended meaning. Fortunately, this is not difficult in view of the well-defined semantics of first-order logic.

3.2.4. Representing Frame Axioms as Defaults

Any representation for planning problems must address the frame problem. It must somehow encode the assumption that most relations are unaffected by a given action, that is, generally their truth-value persists after performing the action. How can this notion of persistence be represented in Theorist? Different answers to this question result from each of the two ontologies being considered. Each will be examined separately.

3.2.4.1. Representation in Ontology-G

Recall that ontology-G corresponds to Green's formulation-II where situations, actions, and objects (but not statements) are admitted as individuals into the ontology of the representation language. In Green's formulation-II, persistence is represented by frame axioms. These frame axioms capture the notion of persistence but, as has been shown, suffer from serious drawbacks. How can the expressive power of frame axioms be exploited without incurring their shortcomings?

Reiter [Reiter80] suggests that defaults can be used to represent frame axioms. The default rule schema for frame axioms in Rieter's logic is:

$$\text{(3.1)} \qquad \frac{R(\bar{x},s) : M\ R(\bar{x},f(\bar{x},s))}{R(\bar{x},f(\bar{x},s))}$$

This inference rule schema allows the conclusion that every action (or state

transition) leaves the truth-value of every relation unchanged unless it can be proven otherwise. In Theorist, however, we would use the related default axiom schema:

(3.2) $[\bar{x},s]\ R(\bar{x},f(\bar{x},s)) \leftarrow R(\bar{x},s)$

(The parameters in square brackets are to be instantiated). This default schema means that instances of the statement

$$R(\bar{x},f(\bar{x},s)) \leftarrow R(\bar{x},s)$$

can be used in an explanation of the goal as long as the explanation is consistent with the facts. The translation from 3.1 to 3.2 is only approximate for two reasons. First, Theorist is able to make deductions based on the contrapositive of an implication, that is, implication in Theorist is true logical implication. For example, in addition to being able to derive $R(\bar{x},f(\bar{x},s))$ from $R(\bar{x},s)$ when it is consistent, it is also possible to derive $\neg R(\bar{x},s)$ from $\neg R(\bar{x},f(\bar{x},s))$ when it is consistent.

The second reason the translation is only approximate is that theories in Theorist differ from extensions in Reiter's logic. Extensions are the theorems following from maximal (and consistent) sets of ground instances of defaults, whereas theories are sets of instances of defaults (consistent with the facts) that explain the observations; there is no implied maximality criterion.

The default schema (3.2) presented above is not sufficient to express the notion of persistence. The inadequacy of the default schema arises in connection with the choice of full first-order logic in Theorist. The explicit representation of negative information, which the use of first-order logic entails, makes it necessary to augment the usual frame axioms for positive information with frame axioms for negative information. Thus, in addition to the default schema (3.2) that expresses the persistence of positive information, the default schema:

(3.3) $[\bar{x},s]\ \neg R(\bar{x},f(\bar{x},s)) \leftarrow \neg R(\bar{x},s)$

is needed to capture the persistence of negative information. This default schema means that instances of the statement

$$\neg R(\bar{x},f(\bar{x},s)) \leftarrow \neg R(\bar{x},s)$$

can be used in an explanation of the goal as long as the explanation is consistent with the facts. Default schema (3.2) and default schema (3.3) can be combined into a single equivalence default schema:

(3.4) $[\bar{x},s]\ R(\bar{x},f(\bar{x},s)) \leftrightarrow R(\bar{x},s)$

This default schema (3.4) is used to represent persistence in ontology-G. It means that instances of the statement

$$R(\bar{x},f(\bar{x},s)) \leftrightarrow R(\bar{x},s)$$

can be used in an explanation of the goal as long as the explanation is

consistent with the facts. Intuitively, this means that we can assume a given actions doesn't change the truth-value of a given relation unless this assumption is inconsistent with the facts (and other assumptions already made).

Figure 3.4 illustrates the Theorist axiomatisation for the Yale shooting scenario using ontology-G. Two sets of statements are given. One set, **F**, describes the relations that are accepted as true in the world. Within this set, there are two kinds of axioms: those that describe the initial situation, and those that describe the changes caused by the performance of actions.

A second set of statements, Δ, contains a frame default for each *primitive* [Fikes71] relation occurring in **F**. Collections of instances (the variables in the square brackets are to be instantiated) of these defaults form theories. Whenever an instance of a frame default is included in a theory, the truth-value of the corresponding relation is preserved when performing the particular action in the particular situation (e.g., **alive(do(load0))** $\leftrightarrow$ **alive(0)** means the truth-value of **alive** is unaffected when performing the action **load** in situation **0**).

3.2.4.2. Representation in Ontology-K

How can the notion of persistence be represented in Theorist if ontology-K is chosen? Recall that ontology-K corresponds to Kowalski's representation where statements, situations, actions, and objects are admitted as individuals into the ontology of the representation language. In

F = {	The set of facts (**F**):
	Initial Situation:
¬ loaded(0),	The gun is not loaded
alive(0),	John is alive
	Action: load
loaded(do(load,S)),	The gun is loaded after the action load
	Action: wait (no known changes)
	Action: shoot
¬ alive(do(shoot,S)) ← loaded(S),	John dies when shot with a loaded gun
¬ loaded(do(shoot,S))}	After shooting, the gun is not loaded
Δ = {	The set of Frame Defaults (Δ):
[A,S] loaded(do(A,S)) ↔ loaded(S),	
[A,S] alive(do(A,S)) ↔ alive(S)}	

Figure 3.4. Yale Shooting Scenario - Theorist Axiomatisation for Ontology-G

Kowalski's representation [Kowalski79], a single frame axiom is used to express persistence.

In Theorist, this frame axiom can be represented as a default. The default corresponding to the frame axiom of Kowalski's representation is:

(3.5) **[R,A,S] holds(R,do(A,S)) ↔ holds(R,S)**

This default (3.5) is used to represent persistence in ontology-K. (Note that this is a default, not a default schema. Compare the difference between how default schema 3.4 and default 3.5 are used in Fig. 3.4 and Fig. 3.5 respectively.) In ontology-K, the issues arising as a result of the explicit representation of negation appear both at the meta-level and at the object-level since we

can introduce negation at the object-level, i.e., **holds(not(R),S)**. Thus, the above default is expressed as an equivalence similar to those of ontology-G.

Figure 3.5 gives the Theorist axiomatisation of the Yale shooting scenario using ontology-K. Note that under ontology-K, a single frame default replaces the frame defaults for each relation in ontology-G. Again we must emphasize that the choice of ontology impacts the solution to the frame problem.

F = {	The set of facts (**F**):
	Initial Situation:
¬ holds(loaded,0),	The gun is not loaded
holds(alive,0),	John is alive
	Action: load
holds(loaded,do(load,S)),	The gun is loaded after the action load
	Action: wait (no known changes)
	Action: shoot
¬ holds(alive,do(shoot,S)) ← holds(loaded,S),	John dies when shot with a loaded gun
¬ holds(loaded,do(shoot,S))}	After shooting, the gun is not loaded
Δ = {	The set of Frame Defaults (Δ):
[R,A,S] holds(R,do(A,S)) ↔ holds(R,S)}	

Figure 3.5. Yale Shooting Scenario - Theorist Axiomatisation for Ontology-K

3.3. Using the Representation in Planning

Planning can be characterised as a process of deductive question answering [Green81] and search. For example, in a simple forward planner, a tree of possible situations is searched to find one that satisfies the goal description. Question answering is used to determine whether the goal description is satisfied in the current situation. When it is not satisfied, neighbour nodes of the current situation in the search tree can be generated by using question answering to determine which actions are possible, i.e., which actions have their preconditions satisfied in the current situation.

This separation of planning into question answering and search allows the representation of planning domain knowledge to be considered independently of the planning search strategy. The Theorist-based representation described earlier is intended to form the basis of the question answering component of a planning system.

Once a planning problem has been expressed, how is the representation used to find a plan? Recall that in a typical theorem proving planner (e.g., [Green81]), the initial situation, the effects of actions, and the goal situation are described by axioms in first-order logic, then an attempt is made to prove the conjecture that there exits a situation satisfying the goal description. If a proof is found then a plan is produced as a side-effect. A planning system of this type reasons by drawing conclusions from its knowledge.

This view of reasoning has serious drawbacks arising from the monotonic nature of logical deduction. Our proposed treatment of the goal stems from the position espoused by Poole et al. [Poole87b]. Instead of viewing reasoning as deduction from knowledge, it is viewed as the formation of "scientific" theories [Popper58]. Thus, we view the planning process as the building of a theory of the way the world is. If the goal is achievable then its properties will be a consequence of the facts and the theory of the world, that is, the goal description is a prediction of the theory.

Given a theory of the world, verifying that the goal description is a prediction of the theory amounts to the same thing as proving the conjecture that a situation exists which satisfies the goal description. So one approach to planning would be to first construct a theory of the world and then construct a plan using Green's approach, but the world is a complicated place. Constructing a complete theory of the world would be an arduous task. Instead, we require a theory of the world that only includes information relevant to determining if and how the goal is achievable, that is, we require goal-directedness.

How can such a theory be constructed? To begin to answer this question, let us first examine a very naive approach to planning which we call *reckless planning*, since it sanctions any theory that predicts the goal description without regard to the theory's reasonableness or likelihood. By consider-

ing the shortcomings of this approach, a more reasonable and more cautious approach, which we call *conservative planning*, can be developed.

3.3.1. Reckless Planning or Planning by Wishful Thinking

Recall that in the theory formation framework, a solution to a planning problem is a sequence of actions that achieves the goal together with a persistence theory that predicts the goal description. In general, there will be multiple consistent persistence theories making different predictions. One approach to planning would be to ask, for each sub-goal, whether there exists a persistence theory which predicts the sub-goal. This approach is reckless for several reasons.

First, no consideration is given to the reasonableness or likelihood of the theories. For example, a persistence theory which predicts dice will continue to roll seven supports a plan to get rich playing "craps," but the theory, while possible, is unlikely.

Second, no consideration is given to the mutual consistency of the theories. Each sub-goal may be predicted by a different theory and these theories may be mutually incompatible, that is, there may be no consistent theory which unifies the individual theories corresponding to each sub-goal.

For example,[5] in Lesotho all northern roads are long and smooth or short

[5] Steve Furino suggested this example.

and rough. A driver in Mokhotlong wishes to go to Thaba Tseka. He is at a fork in the road where one branch leads to the long smooth route and the other leads to the short rough route. Two sub-goals are: not to run out of gas and not to get a flat tire. There is a theory for each sub-goal, namely, if the route selected is short then he has enough gas and if the route selected is smooth then the tires will be fine.

Obviously, the two theories are incompatible. Forming a plan to achieve the goal by achieving each sub-goal with the above actions and theories would be reckless because it treats sub-goals independently. It does not take interactions between the sub-goals into account — it does even consider that different incompatible actions are required to achieve the sub-goals.

Third, this approach is reckless because alternatives are not considered. Even if a single theory predicts all the sub-goals and is more likely than all other theories predicting the sub-goals, planning based on this theory is reckless unless theories predicting the negation of the sub-goals are considered.

Fourth, the reckless approach to planning does not prohibit making needless assumptions. It looks for *any* theory which predicts the sub-goals, not necessarily the least presumptive theory (i.e., the theory making the fewest assumptions while still predicting the goal).

Finally, when there is incomplete knowledge, a situation may arise when there are several alternatives with no basis on which to choose. The reckless approach would be to arbitrarily choose the most favourable alternative. The reckless approach to planning can be characterised as a form of 'wishful thinking.' It asks: 'What would the world have to be like in order to make my objectives achievable?' and then it assumes the world is like that!

3.3.2. Conservative Planning

These deficiencies of reckless planning suggest a more reasonable approach — conservative planning. By considering these deficiencies, we can outline some properties that a conservative planning approach should have. Conservative planning should attempt to view the world the way it is — or at least, the way it is expected to be — and then make decisions accordingly, without making unreasonable assumptions.

The issue of theory preference arises in connection with the first shortcoming of reckless planning. When is one persistence theory more reasonable or more likely than another? This question will be examined in detail in section 3.4. For now, it suffices to note that some form of theory preference is necessary in order to plan conservatively.

The second deficiency of reckless planning indicates the need for a single unifying persistence theory for the set of sub-goals. This theory, if it exists, might possibly be constructed by finding the most likely theory for the first

sub-goal. Next each of the remaining sub-goals is checked to see whether it is predicted by that theory. If there is some sub-goal not predicted by the theory then an embellishment of the theory is sought (subject to the likelihood constraint) which predicts the sub-goal. Whether or not this method of constructing a unifying theory works in general, it is clear that a conservative approach to planning should have a means of finding a unifying theory.

The third shortcoming of the reckless approach can be addressed by ensuring that there is no theory for a negated sub-goal which is more likely than the theory predicting all the sub-goals described above.

The fourth shortcoming can be addressed by choosing the least presumptive theory. Sometimes this form of conservatism is too cautious, for example, in order to plan how to get to your car you can assume that your car is where you left it and that the weather will remain the same during the walk to your car. It would be less presumptive to exclude the assumption about the weather, but this is overly cautious.

Finally, whenever there is a choice to be made and there is no information on which to base the choice then the conservative approach to planning should incorporate a conditional into the plan. Thus, conservative planning should attempt to view the world the way it is and to make decisions accordingly, without making unreasonable assumptions.

3.4. Theory Preference Criteria for Planning

In order to plan conservatively, some means is necessary to distinguish among the multiple competing persistence theories. When a scientist is faced with a choice among competing theories, several factors influence the choice — corroboration with the evidence, predictive power, simplicity, unifiability with other knowledge, etc. In planning, when presented with competing theories (each making different predictions), we desire a means to choose among theories — a theory preference heuristic. This heuristic should prefer the theory that corresponds to our intuition about persistence. The heuristic should simultaneously satisfy three criteria:

1) *accuracy* — it should select only theories that makes predictions corresponding to our common sense expectations;

2) *sufficiency* — if the goal description (or its negation) is expected, then it should be predicted by the selected theory;

3) *resource conservatism* — it should select a theory with maximal obtainable *accuracy* for minimal computational effort, that is, we want to make as many accurate predictions as possible while doing only enough computation as is necessary to determine whether the goal is expected.

To make the discussion of this heuristic more concrete, consider whether, in the shooting scenario, John will be alive after the actions **load**, **wait**, and **shoot**. For simplicity, ontology-G will be used throughout the rest of the

chapter, although similar conclusions can be drawn using ontology-K. As well, we will often abbreviate situation names as follows:

1 ≡ **do(load,0)**
2 ≡ **do(wait,do(load,0))**
3 ≡ **do(shoot,do(wait,do(load,0)))**.

There are 29 consistent theories that describe the invariance of relations over the path from situation **0** to situation **3**. These are formed from ground instances of elements of Δ. Of these, two are particularly interesting:

$\mathbf{T_1}$ = {**loaded(do(wait,do(load,0)))** ↔ **loaded(do(load,0))**,
alive(do(load,0)) ↔ **alive(0)**,
alive(do(wait,do(load,0))) ↔ **alive(do(load,0))**,} and

$\mathbf{T_2}$ = {**alive(do(load,0))** ↔ **alive(0)**,
alive(do(wait,do(load,0))) ↔ **alive(do(load,0))**,
alive(do(shoot,do(wait,do(load,0)))) ↔ **alive(do(wait,do(load,0)))**,
loaded(do(shoot,do(wait,do(load,0)))) ↔ **loaded(do(wait,do(load,0)))**}.

The statements in the theories are instances of frame defaults that record two different sets of assumptions about the action sequence **load** then **wait** then **shoot**. Note that these two theories conflict since

$\mathbf{F} \cup \mathbf{T_1} \models$ ¬**alive(do(shoot,do(wait,do(load,0))))** while
$\mathbf{F} \cup \mathbf{T_2} \models$ **alive(do(shoot,do(wait,do(load,0))))**.

Of the remaining 27 theories,

1 predicts **alive(do(shoot,do(wait,do(load,0))))**,
12 predict **¬alive(do(shoot,do(wait,do(load,0))))**, and
14 make no prediction regarding **alive** or **¬alive**.

In our intended model (i.e., the one that corresponds to our intuitions), however, **¬alive(do(shoot,do(wait,do(load,0))))** is true (Figure 3.6).

Situations:
1 ≡ do(load,0)
2 ≡ do(wait,do(load,0))
3 ≡ do(shoot,do(wait,do(load,0)))

Intended Model Features				
	0	1	2	3
loaded	F	T	T	F
alive	T	T	T	F

T_1 Model Features				
	0	1	2	3
loaded	F	T	T	F
alive	T	T	T	F

T_2 Model Features				
	0	1	2	3
loaded	F	T	F	F
alive	T	T	T	T

Figure 3.6. Essential Features of Models for the Yale Shooting Scenario

---

At this point, one might wonder if the existence of multiple consistent yet conflicting theories indicates that our axiomatisation of the problem is incorrect. In their work, Hanks and McDermott [Hanks85] show that such multiple consistent conflicting solutions are inevitable and they conclude that nonmonotonic reasoning systems are inherently incapable of adequately representing even simple default reasoning problems. However, we claim, the problem is not due to any deficiency of the reasoning system; it is merely the result of a weak set of axioms.

One might ask whether it is possible to restrict the set of consistent theories by strengthening the set of facts so that the new set of consistent theories all predict **¬alive(do(shoot,do(wait,do(load,0))))** (e.g., adding as a fact that loading persist while waiting). The answer, of course, is yes — frame axioms could be added to the set of facts — but we are trying to avoid the inherent problems of frame axioms. *Therefore, we initially allow theories which disagree with our intended model and then use theory preference to discriminate between them.*

We wish to provide a theory preference heuristic based on our intuitions about the persistence of relations. To do this, we must first formalise the concept of persistence. The notion of persistence is intended to reflect what McCarthy calls the "common sense law of inertia;" that is, when an action is performed, most things remain unchanged. In order to formalise this intuition, a semantic account of persistence is necessary.

We will use standard Tarskian semantics; that is, the world is described in terms of individuals and relations on individuals. The domain of discourse contains three types of individuals: situations, actions, and ordinary objects (i.e, ontology-G). For the purpose of discussion, the language in which logical theories are expressed will be full first-order clausal logic; however, any language capable of expressing a contradiction is sufficient (cf. [Goebel85]).

Definition 3.1.

Let **R** be a relation in the domain with a vector of arguments $\bar{x}$ as well as a situation argument **s**. We say that $\mathbf{R}(\bar{x})$ is a *propositional fluent* [McCarthy69] because the truth-value of the corresponding relation, $\mathbf{R}(\bar{x},s)$, varies with the situation. When we say $\mathbf{R}(\bar{x})$ is true in situation **s**, we mean $\mathbf{R}(\bar{x},s)$ is true.

Definition 3.2.

Let **do** be a function that maps actions and situations to situations. Thus, **do(a,s)** is the situation that results from doing action **a** in situation **s**.

Definition 3.3.

Let $s_n = do(a_n,do(a_{n-1},\ldots,do(a_1,s_0))\ldots)$ be a situation in the domain. The situations s_0 to s_n determine a *path* which we write as $\langle s_0,s_n\rangle$. The *length of a path* $\langle s_0,s_n\rangle$ is defined to be **n**, the number of actions on the path from s_0 to s_n. A *unit path* is a path of length one.

Definition 3.4.

Given a consistent theory **T**:

a) A *primitive* propositional fluent $\mathbf{R}(\bar{x})$ is said to *persist* in **T** over the unit path $\langle s,do(a,s)\rangle$ if

$$\mathbf{F} \cup \mathbf{T} \models \mathbf{R}(\bar{x},s) \equiv \mathbf{R}(\bar{x},do(a,s));$$

b) A *persistence set*, consisting of elements of the form $\mathbf{persist}(\mathbf{R}(\bar{x}),do(a,s))$, describes the propositional fluents that persist over a given path. For a given theory **T** and a path $\langle s_0,s_n\rangle$, we define the following persistence sets:

$$P^{i}_{T,R(\bar{x})} = \begin{cases} \{\text{persist}(R(\bar{x}),\text{do}(a_i,s_i))\} & \text{if } F \cup T \models R(\bar{x},s_i) \leftrightarrow R(\bar{x},\text{do}(a_i,s_i)) \\ \varnothing & \text{otherwise} \end{cases}$$

$$P^{i}_{T} = \bigcup_{R(\bar{x})} P^{i}_{T,R(\bar{x})}$$

$$P_T = \bigcup_{i} P^{i}_{T}$$

Here $P^{i}_{T,R(\bar{x})}$ indicates whether the propositional fluent $R(\bar{x})$ persists over unit path $\langle s_i,s_{i+1}\rangle$. P^{i}_{T} indicates all the propositional fluents that persist over unit path $\langle s_i,s_{i+1}\rangle$, and P_T describes all the propositional fluents that persist over the entire path (i.e., $\langle s_0,s_n\rangle$).

In definition 3.4a persistence is defined in relation to primitive [Fikes71] propositional fluents. This ensures that the truth-value of a defined (non-primitive) propositional fluent is preserved only when its associated primitives persist.

A domain-dependent *partial ordering* of persistence sets (denoted by $\geq^p$) can be defined to reflect the supposed relative likelihoods of each persistence set. Thus if two persistence sets differ on a highly persistent propositional fluent $R(\bar{x})$, the persistence set which includes it would be higher ranked. We leave open the question about how $\geq^p$ should be defined, but we will now consider some possibilities.

As indicated, the partial ordering $\geq^p$ is a domain dependent ranking of the likelihood of persistence sets. If all we know about the likelihood of a propositional fluent persisting is that it is more likely to persist than not (i.e., the commonsense law of inertia), then we can define $\geq^p$ to be $\supseteq$.

When we know more about the persistence characteristics of various propositional fluents, then this information can be included by providing an arbitrarily complex definition of $\geq^{P}$. For example, if we know that all propositional fluents in the domain of interest are equally likely to persist, then we can define $\geq^{P}$ to be a comparison of the cardinality of persistence sets (cf. the example in chapter 5 figure 5.1). In this case, the partial ordering $\geq^{P}$ is defined as the total ordering $\geq$ on the cardinality of persistence sets.

When the domain knowledge about persistence probabilities is complex (e.g., metal fatigue, device failure rates, etc.), Neufeld's work on commonsense probabilistic theory comparators may be useful [Neufeld87].

This semantic definition of persistence can be used to distinguish theories. For instance, in the Yale shooting scenario, we can arbitrarily assign weights to the propositional fluents **loaded** and **alive**. These weights can be used to represent the relative probability of the propositional fluents persisting. By taking $\geq^{P}$ to be a comparison of weighted measures of the sets of persisting propositional fluents in competing theories, we can prefer the theory with the greatest weight. Figure 3.7 illustrates how theory $\mathbf{T_2}$ can be preferred over $\mathbf{T_1}$ in the Yale shooting scenario when arbitrarily assigning **alive** a weight of 2 and **loaded** a weight of 1. This table is computed by counting the unit paths over which each propositional fluent persists in the indicated theory (cf. Fig. 3.6), e.g., in theory $\mathbf{T_2}$ the truth-value of **loaded** is

Path: <0,do(shoot,do(wait,do(load,0)))>

R	W	$\|P_{T_1,R}\|$	$\|P_{T_2,R}\|$	$W \times \|P_{T_1,R}\|$	$W \times \|P_{T_2,R}\|$
loaded	1	1	1	1	1
alive	2	2	3	4	6
		3	4	5	7

Figure 3.7. Weighted Persistence Comparison for Yale Shooting Scenario

invariant only over the unit path <2,3>, hence the entry '1' at the intersection of row **loaded** and column $|P_{T_2,R}|$. The table shows that the weight of T_2 is greater than the weight of T_1. While it may seem reasonable to weight **alive** more than **loaded**, this naive approach is inadequate since T_2 is mistakenly preferred. A more sophisticated definition of $\geq^p$ is needed.

Before turning to the definition of our proposed theory preference heuristic, we will first define the another possible preference heuristic based on the global maximisation of persistence. After defining this, we will show be means of a counterexample that global maximisation of persistence is an inadequate heuristic for reasoning in temporal domains. The following two definitions define the global maximisation heuristic.

Definition 3.5.

A consistent theory T_1 is said to be *more persistent* than consistent theory T_2 over the path $\langle s_1,s_n\rangle$ if $P_{T_1} \geq^p P_{T_2}$ where P_T is the persistence set for T over $\langle s_1,s_n\rangle$. The partial ordering $\geq^p$ on persistence sets defines a partial ordering $\geq^{gp}$ on consistent theories. Thus, $T_1 \geq^{gp} T_2$ means T_1 is more persistent than T_2.

Definition 3.6.

A consistent theory T_{GMP} is said to be a *globally maximally persistent* theory over a path if $\neg\exists T_k$, $T_k >^{gp} T_{GMP}$ over the path where each T_k is a consistent theory.

We can use these definitions to examine Reiter's [Reiter80, p. 85] formalisation of the STRIPS assumption. In Reiter's terms, a *default theory* $\Delta = (D,F)$ consists of defaults D and facts F. The extensions of a default theory with

$$D = \frac{R(S):\ M\ R(do(A,S))}{R(do(A,S))}$$

are the maximal consistent sets of consequences of $F \cup D$, that is, extensions are generated from the facts together with maximally consistent subsets of the elements of D. Therefore, when $\geq^p$ is taken to be $\supseteq$, extensions correspond to the consequences of globally maximally persistent theories (as these are formed from maximally consistent subsets of the elements of Δ). Thus, both T_1 and T_2 (globally maximally persistent theories) of the Yale shooting scenario have corresponding extensions in Reiter's logic. Since only T_1 corresponds to the intended model for the Yale shooting scenario, this shows that the global maximisation of persistence is inadequate. Therefore, global

maximal persistence is not an appropriate heuristic measure of preference for planning.

Since actions are performed in sequence, it does not make sense to *globally* maximise persistence over a path. Instead, persistence should be maximised step by step (in chronological order[4] — cf. [Hanks85, Hanks86, Shoham86a, Shoham86b]). To see the intuition behind this, consider the following example.

Suppose we have two switches connected to a nuclear device. Switch A arms the device and switch B detonates the device. If we assume both switches continue to function after throwing switch A, we expect great changes in the world after throwing switch B. Alternatively, if we do not make the above assumption, then we expect that nothing much will change after throwing switch B. The two alternatives are globally maximally persistent, but the first alternative is the expected one. From this we can see that we should maximise persistence in first step of a path before we consider the second step. We do this to reflect the idea that causality flows forward in time. We do not want to assume the switches fail in order to avoid drastic changes in the future.

---

[4] This concept is due to Yoav Shoham.

An important property the preference criterion should exhibit is *compositionality* over sub-paths, that is, the predictions of the preferred theory for the entire path to agree with the predictions of the preferred theory for a sub-path. In the above example, if the global maximal persistence criterion was applied to the first unit path, the (unique) preferred theory would predict that the switches continued to function; while if applied to the whole path, two conflicting theories arise. Therefore, the global maximal persistence criterion lacks the compositionality property.

The notion of step by step maximisation is formalised in the following definitions.

Definition 3.7.

A consistent theory T_1 is said to be *chronologically more persistent* than consistent theory T_2 over the path $\langle s_1, s_n \rangle$ if either

a) $\forall i < n,\ P^i_{T_1} \geq^p P^i_{T_2}$, or

b) $\exists j < n,\ P^j_{T_1} >^p P^j_{T_2}$ and
$\forall i < j,\ P^i_{T_1} \geq^p P^i_{T_2}$

where P^i_T is the persistence set for T over $\langle s_i, s_{i+1} \rangle$. The partial ordering $\geq^p$ on persistence sets defines a partial ordering $\geq^{cp}$ on consistent theories. Thus, $T_1 \geq^{cp} T_2$ means T_1 is chronologically more persistent than T_2.

Definition 3.8.

A consistent theory T_{CMP} is said to be *chronologically maximally persistent* (CMP) over a path if $\neg \exists T_k,\ T_k >^{cp} T_{CMP}$ over the path where each T_k is a consistent theory.

It should be mentioned that, in general, there is no guarantee of a unique CMP theory. When there are multiple CMP theories, our intuition

about persistence offers no further assistance — the problem is simply underspecified. In the absence of any additional domain knowledge, there is no basis by which to determine a "best" theory. We can, however, form a conditional plan based on the disjunction of the CMP theories $\mathbf{DT_{CMP}} \equiv \bigvee_k \mathbf{T_{CMP_k}}$. Note that $\mathbf{DT_{CMP}}$ is not a "scientific" theory since it is not a set of instances of defaults; it is, however, a logical theory. Because at least one of the CMP theories is expected to describe the world, the predictions of $\mathbf{F} \cup \mathbf{DT_{CMP}}$ are expected to be true in the world. Therefore, even in this case, we can still describe what we expect to be true in the world.

How can we use CMP to answer the query about whether John is expected to be alive in the Yale shooting scenario (Fig. 3.4)? In this example, all we know about the relative likelihood of a relation persisting is that, by the common sense law of inertia, it is more likely to persist than not. Therefore, it is appropriate to take $\geq^{\mathbf{P}}$ to be $\supseteq$. In this case, $\mathbf{T_1}$ is the unique chronologically maximally persistent theory over $\langle 0,3 \rangle$. As we have seen earlier, $\mathbf{T_1}$ corresponds to the intended model; its predictions correspond to our expectations about the domain. Therefore, the query about whether whether John is expected to be alive is answered by determining what $\mathbf{T_1}$ predicts. Here $\mathbf{T_1}$ predicts $\neg$**alive(3)**.

We conjecture that CMP theories correspond to our expectations in planning. One possible argument against this claim is given by Kautz [Kautz86, pp. 404-405]. He describes a scenario where a car is left in a parking lot and at some later time it is *observed* to be missing.[5] Under chronological maximisation of persistence, the car is expected to be in the parking lot until the shortest possible time before it is observed missing. This is unreasonable since the car could have been stolen or towed away any time between when it was parked and when it was observed missing.

This argument fails to distinguish the *task of predicting* the future outcome of a series of events, and the *task of explaining* how a particular state of affairs could have come about. In the above example, it seems reasonable to expect that your car will remain where it is parked until you return unless you have reason to believe an event will occur which affects your car's location. When trying to explain why your car is not where it was expected to be, it seems reasonable to hypothesize that some unknown event must have occurred which changed the car's location. Since the task of predicting is relevant to planning, but the task of explaining is not, Kautz's argument is irrelevant to planning. So we are still justified in making the claim that chronological maximisation of persistence is an appropriate heuristic for planning.

---

[5] This example is better represented in an ontology of time intervals and events. For discussion, we assume our heuristic can be recast in such an ontology.

Additionally, the CMP heuristic satisfies two of the three criteria we gave earlier: accuracy and sufficiency. The third criterion we set out (resource conservatism) does not involve the specification of what is expected to hold in the domain, but rather, it involves computational issues — what subset of what is expected can be reasonably computed. While CMP seems to capture our expectations about persistence, CMP theories are usually not the most economical choice. In a CMP theory, many of the propositional fluents that persist are simply irrelevant to the problem at hand. Including frame defaults for these irrelevant fluents is computationally costly (because of the required consistency checking). Therefore, a better theory would exclude irrelevant instances of frame defaults. The need to simplify theories by eliminating irrelevant details motivates the following definitions (cf. [Poole86]).

Definition 3.9.

A partial ordering on simplicity of consistent theories predicting **G** is defined as follows. Let $\mathbf{T_1}$ and $\mathbf{T_2}$ be distinct consistent theories predicting **G**. $\mathbf{T_1}$ is *simpler* ($\succeq^s$) than $\mathbf{T_2}$ (w.r.t. **G**) iff $\mathbf{F} \cup \mathbf{T_2} \models \mathbf{T_1}$.
Syntactically, **Theorems**$(\mathbf{F} \cup \mathbf{T_1}) \subseteq$ **Theorems**$(\mathbf{F} \cup \mathbf{T_2})$.

Definition 3.10.

A consistent theory is *simplest* w.r.t. **G** if it is a maximal element of the partial ordering $\succeq^s$ on the set of consistent theories predicting **G**.

Simplest theories contain the minimal amount of information necessary to make a given prediction. Because simplest theories can be generated in a goal directed manner (cf. [Poole87b]), and because they involve less consistency checking, they are computationally less costly than CMP theories.

Let us see why simplicity cannot be used to determine the expected outcome of a sequence of actions. In the example above, there is a single simplest theory predicting $\neg$**alive(do(shoot,do(wait,do(load,0))))**, namely:

$$\mathbf{T_3} = \{\textbf{loaded(do(wait,do(load,0)))} \leftrightarrow \textbf{loaded(do(load,0))}\}.$$

There is also a simplest theory predicting **alive(do(shoot,do(wait,do(load,0))))**. It is:

$$\begin{aligned}\mathbf{T_4} = \{&\textbf{alive(do(load,0))} \leftrightarrow \textbf{alive(0)},\\ &\textbf{alive(do(wait,do(load,0)))} \leftrightarrow \textbf{alive(do(load,0))}\}.\\ &\textbf{alive(do(shoot,do(wait,do(load,0))))} \leftrightarrow \textbf{alive(do(wait,do(load,0)))}\}.\end{aligned}$$

It should be clear that simplicity alone is not an appropriate heuristic for theory preference in planning, since it does not satisfy the accuracy criterion (e.g., $\mathbf{T_4}$ is a simplest theory but its prediction does not agree with the intended model). In other words, simplicity does give us the computationally least expensive way to predict a goal, but it does not tell us whether to expect the goal to be true in the world.

Our proposed preference heuristic combines simplicity and chronological persistence. The trade-off between resource conservatism and accuracy is reflected in a trade-off between simplicity and chronological persistence — *increasing simplicity and decreasing chronological persistence decreases computational cost and accuracy*, while *decreasing simplicity and increasing chronological persistence increases computational cost and accuracy*. To arrive at our

proposed preference heuristic, we must choose a point on the trade-off that satisfies the three criteria: accuracy, sufficiency, and resource conservatism. We want our preference heuristic to involve only as much computation as is necessary to determine whether our goal is expected, and then, with little further computation, it should select a theory (or disjunction of theories) that maximises accuracy (i.e., maximises the subset of what is expected that gets predicted). Thus, we are willing to approximate CMP theories to decrease computational cost.

We do this as follows. From the partial ordering of theories defined by the chronological persistence, we take the disjunction of a particular set of theories $\mathbf{DT_A} \equiv \bigvee_{\mathbf{k}} \mathbf{T_k}$. Each $\mathbf{T_k}$ is simpler than a (or the) CMP theory. Therefore, $\mathbf{DT_A}$ (the disjunction of theories that approximates CMP theories) has the property that for all **P**, $\mathbf{F} \cup \mathbf{DT_A} \models \mathbf{P}$ implies $\mathbf{F} \cup \mathbf{DT_{CMP}} \models \mathbf{P}$, that is, every prediction of $\mathbf{DT_A}$ is a prediction of $\mathbf{DT_{CMP}}$, but not every prediction of $\mathbf{DT_{CMP}}$ is necessarily a prediction of $\mathbf{DT_A}$. Consequently, $\mathbf{DT_A}$ may be *uninformed* about some features of the world (i.e., the intended model corresponding to $\mathbf{DT_{CMP}}$), but it is not *misinformed.*

$\mathbf{DT_A}$ can be thought of as representing a point on the trade-off between resource conservatism and accuracy. This point is determined by the $\mathbf{T_k}$ above. The sufficiency criterion adds another constraint for determining $\mathbf{DT_A}$. We must ensure that $\mathbf{DT_A}$ predicts the goal if it is expected (or predicts its

negation if the negation is expected). To do this we make use of the following theorem and definitions:

Definition 3.11.

A *maximal* theory that predicts **G** is a maximal element of the partial ordering determined by the dual of $\geq^{s}$. (Intuitively, it is a least simple (most complex) theory predicting **G**. Note that the partial ordering is over consistent theories.)

Definition 3.12.

The partial ordering $\geq^{p}$ is *sensibly* defined if $P^{i}_{T_1} \supseteq P^{i}_{T_2}$ implies $P^{i}_{T_1} \geq^{p} P^{i}_{T_2}$. (Intuitively, the partial ordering is sensibly defined if it is consistent with the common sense law of inertia, i.e, relations are more likely to persist than not).

Note that $\geq^{p}$, when sensibly defined, is a refinement of $\supseteq$ and that $\geq^{cp}$ is a refinement of $\geq^{s}$. Thus, $T_1 \geq^{s} T_2$ implies $T_2 \geq^{cp} T_1$. This important observation is fundamental to the definition of DT_A and it allows us to give the following theorem.

Theorem 3.1

If, for each maximal theory M_k that doesn't predict **G**, there exists a consistent theory T_k that predicts **G**, such that T_k is chronologically more persistent than M_k, then all CMP theories predict **G** provided $\geq^{p}$ is sensibly defined.

Proof

Suppose there is a CMP theory T_{CMP} that doesn't predict **G**. There is a maximal theory **M** that doesn't predict **G** for which T_{CMP} is simpler than **M** (w.r.t. $\neg G$ or w.r.t. $G \vee \neg G$ depending on what T_{CMP} predicts). Since T_{CMP} is simpler than **M** and since $\geq^{p}$ is sensibly defined, the persistence set for T_{CMP} is contained in the persistence set for **M** for each unit path. Therefore, **M** is chronologically more persistent than T_{CMP}. The only theory chronologically more persistent than a CMP theory is itself. Therefore, $T_{CMP} \equiv M$. Hence, T_{CMP} is a maximal theory that doesn't predict **G**. By hypothesis, there exists a consistent theory T_k that

is chronologically more persistent than T_{CMP}. Again, the only theory chronologically more persistent than a CMP theory is itself; therefore, $T_{CMP} \equiv T_k$. But T_k predicts G. So T_{CMP} predicts G. This contradicts the supposition. Therefore, no such CMP theory exists.

Recall that we wish to ensure that DT_A predicts the goal if it is expected (or predicts its negation if the negation is expected). We do this by ensuring that each maximal theory not predicting the goal (assuming the goal is expected) is below at least one of the T_k in the partial ordering of chronological persistence. Now by theorem 3.1, all CMP theories predict the goal. Consequently, the goal is expected and DT_A predicts it. Thus, DT_A is defined as the disjunction of least chronologically persistent theories which cover the maximal theories not predicting the goal.

This specifies our preference heuristic. DT_A satisfies the sufficiency criterion by construction. It satisfies the resource conservatism criterion since we need to compute the T_i to determine if the goal is expected. It also satisfies the accuracy criterion since each T_i is simpler than a CMP theory and therefore their predictions are a subset of the expected properties.

As a simplification, from now on, we will only consider problems where either the goal or its negation is expected (as is often the case). With this assumption, we can restrict our attention to theories for the goal and theories for the negation of the goal.

Let us illustrate the above definition with a few examples. In the Yale Shooting Scenario, the unique maximal theory predicting **alive(3)** is T_2, and T_1 is the only theory predicting **¬alive(3)** that is chronologically more persistent that T_2. So in this case, $DT_A \equiv T_1 \equiv T_{CMP}$. Thus, in this example, we were unable to realise any savings. Examples can be constructed where this is not the case.

For instance, consider the sky-diving scenario (Fig. 3.8).

F = {	The set of facts:
	Initial Situation:
defective(0),	The parachute is defective
alive(0),	John is alive
¬ happy(0),	He is not happy
	Action: guzzle
happy(do(guzzle,S)),	Guzzling beer makes John happy
	Action: jump
¬ alive(do(jump,S)) ← defective(S)}	Sky diving with the defective parachute is fatal
Δ = {	The set of Frame Defaults:
[A,S] defective(do(A,S)) ↔ defective(S),	
[A,S] alive(do(A,S)) ↔ alive(S),	
[A,S] happy(do(A,S)) ↔ happy(S)}	

Figure 3.8. Sky-diving Scenario

Here, John is initially alive and happy and he has a defective parachute. He guzzles some beer and then goes sky-diving. The action **guzzle** results in John being happy and the action **jump** results in John being dead if the parachute is defective. We want to know what is expected after the action sequence **guzzle** and then **jump**. The CMP theory is

$$\begin{aligned} T_{CMP} = \{&\textbf{defective(do(guzzle,0))} \leftrightarrow \textbf{defective(0)}, \\ &\textbf{defective(do(jump,do(guzzle,0)))} \leftrightarrow \textbf{defective(do(guzzle,0))}, \\ &\textbf{alive(do(guzzle,0))} \leftrightarrow \textbf{alive(0)}, \\ &\textbf{happy(do(jump,do(guzzle,0)))} \leftrightarrow \textbf{happy(do(guzzle,0))}\}. \end{aligned}$$

In this example,

$$\begin{aligned} T_1 = \{&\textbf{defective(do(jump,do(guzzle,0)))} \leftrightarrow \textbf{defective(do(guzzle,0))}, \\ &\textbf{alive(do(guzzle,0))} \leftrightarrow \textbf{alive(0)}, \\ &\textbf{alive(do(jump,do(guzzle,0)))} \leftrightarrow \textbf{alive(do(guzzle,0))}, \\ &\textbf{happy(do(jump,do(guzzle,0)))} \leftrightarrow \textbf{happy(do(guzzle,0))}\} \end{aligned}$$

is the unique maximal theory predicting **alive(do(jump,do(guzzle,0)))**. Here, the theory

$$\begin{aligned} T_2 = \{&\textbf{defective(do(guzzle,0))} \leftrightarrow \textbf{defective(0)}, \\ &\textbf{alive(do(guzzle,0))} \leftrightarrow \textbf{alive(0)}\} \end{aligned}$$

is the simplest theory predicting $\neg$**alive(do(jump,do(guzzle,0)))** that is chronologically more persistent than T_1. Therefore, in this example, $DT_A \equiv T_2$. Thus, we are able to eliminate the frame default instances

defective(do(jump,do(guzzle,0))) ↔ defective(do(guzzle,0))
happy(do(jump,do(guzzle,0))) ↔ happy(do(guzzle,0))

from T_{CMP}. While we expect some computational savings for this, we do lose some accuracy. Here, we expect

defective(do(jump,do(guzzle,0))) and
happy(do(jump,do(guzzle,0))).

T_{CMP} makes these predictions, but DT_A does not. Thus, we see a case where DT_A approximates T_{CMP}. Note that if we were interested in whether defective and happy are expected in the final situation then they would be part of the goal. Making them part of the goal would result in selecting a different DT_A which would make the above predictions. Consequently, in cases where DT_A only approximates DT_{CMP}, the information loss is unimportant while the computational savings are beneficial.

Now let us turn to an example where there are multiple CMP theories and where DT_A is a disjunction of theories. Figure 3.9 depicts a partial ordering of theories according to their chronological persistence. Theories that predict an unspecified goal are represented by circles; ellipses represent theories that predict the negation of the goal; theories that are independent of the goal (and its negation) are represented by boxes. In this example, T_{11} and T_{12} are CMP theories. Thus, $DT_{CMP} \equiv T_{11} \vee T_{12}$. This theory predicts the goal since both T_{11} and T_{12} predict the goal. Note that T_8, which predicts the goal, is chronologically more persistent than T_6 and T_2, each of which

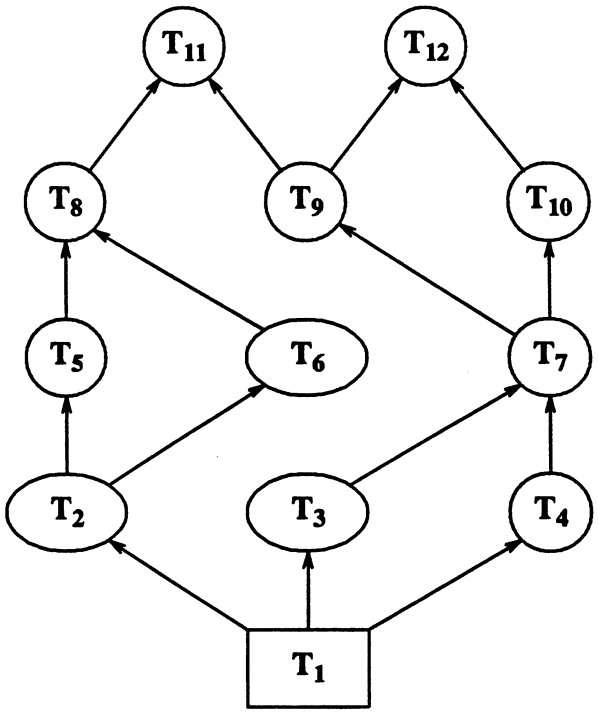


Figure 3.9. Partial Ordering $\geq^{cp}$

predict the goal's negation. As well, T_7, which predicts the goal, is chronologically more persistent than T_3, which predicts the negation of the goal. Both T_7 and T_8 are chronologically more persistent than T_1, the only theory in this example which is independent of the goal. Thus, all theories that don't predict the goal lie below T_8 or T_7 in the partial ordering. So $DT_A \equiv T_7 \vee T_8$.

In the next chapter, we will describe a procedure for computing DT_A.

Chapter 4
Implementation Issues

4.1. Introduction

The goal of this chapter is to specify a search procedure[6] for constructing $\mathbf{DT_A}$ in the previous chapter. We begin the task of specifying the procedure by first specifying the search space. The search space is the poset determined by the set of consistent theories and the partial ordering ($\geq^{cp}$). Given the search space, we must next choose a search strategy. Many search strategies are possible; the one we use is based on Berliner's B* algorithm [Berliner79]. We then describe one implementation of this procedure; we provide an example of its operation and finally, we briefly discuss some other implementation issues.

4.2. The Search Space

The problem of constructing the $\mathbf{DT_A}$ can be viewed as the search of a graph whose nodes represent consistent theories. More specifically, the search space is the poset determined by the set of consistent theories $\overline{\Delta}$, (i.e.,

[6] We do mean procedure, not algorithm, because of the theorem proving and consistency checking involved (these are semi-decidable, in general).

the set of sets of ground instances of elements of Δ that are consistent with **F**) and the partial ordering ($\geq^{cp}$). Recall that $\geq^{cp}$ is determined by $\geq^{p}$. If $\geq^{p}$ is taken to be $\supseteq$ then the nodes adjoining a given node in the poset (for $\geq^{cp}$) are such that their corresponding theories differ from the given node's theory by a single instance of a frame default. Finally, the nodes of the search space can be partitioned into 3 classes according to the relation between the theory represented by the node and the goal, that is, whether the theory predicts the goal, predicts the goal's negation, predicts neither (note that inconsistent theories are pruned from the search space).

As an example, consider the search space for the Yale shooting scenario with $\geq^{p}$ defined to be $\supseteq$ and with the goal **¬alive(do(shoot,do(wait,do(load,0))))**. The minimal element in the search space corresponds to the empty theory, while the maximal element corresponds to the theory:

$$\mathbf{T_{max}} = \{\mathbf{loaded(do(wait,do(load,0)))\leftrightarrow loaded(do(load,0)),}$$
$$\mathbf{alive(do(load,0))\leftrightarrow alive(0),}$$
$$\mathbf{alive(do(wait,do(load,0)))\leftrightarrow alive(do(load,0))}\}.$$

The persistence set corresponding to the above theory is:

$$\mathbf{P_{T_{max}}} = \{\mathbf{persist(loaded,do(wait,do(load,0))),}$$
$$\mathbf{persist(alive,do(load,0)),}$$
$$\mathbf{persist(alive,do(wait,do(load,0)))}\}.$$

If two theories, say T_1 and T_2, are such that $T_1 >^{cp} T_2$ then the node representing T_1 is a descendant of the node representing T_2. The adjoining nodes in the search space are determined by the definition of $>^{cp}$ which in turn depends on the definition of $\geq^{p}$. Consequently, the components of the specification of the search space are the set of consistent theories (determined by the set of primitive propositional fluents, the path to the goal, and the facts) and the partial order $\geq^{p}$.

4.3. The Search Procedure

There are many possible strategies that could be used to search the search space described in the previous section. Of these, one that seems well suited for this particular search space is Berliner's B* search algorithm [Berliner79]. In this algorithm, each node is given two evaluations: a pessimistic one and an optimistic one.

The intuition behind the two evaluations is that the true evaluation of the node likely lies somewhere between the bounds. Thus, the two evaluations are intended to delimit the range of uncertainty in the true value of the node, i.e., they provide a range of values between which the values found in the node's sub-tree likely lie. Hopefully, the bounds are accurate, in which case, the values in a given sub-tree will be between the range given at the root of the sub-tree.

During the search, as new nodes are expanded and their corresponding evaluations are backed-up, the range at the root is narrowed. In this way, the search dynamically refines the initial (static) evaluation of the root node. The search is terminated when the pessimistic value of an arc[2] at the root is no worse than the optimistic value of any other arc at the root. Intuitively, this means that a node has been identified that is at least as good as the best descendant of the node's siblings.

When at the root, there are two strategies that can be employed: PROVEBEST and DISPROVEREST. The PROVEBEST strategy attempts to raise the pessimistic bound of the most optimistic node so that it is no worse than the optimistic bound of any of its sibling nodes. In contrast, the DISPROVEREST strategy tries to lower, below the pessimistic bound of the most optimistic node, the optimistic bounds of all its sibling nodes. When the B* search terminates it will have identified the best first step toward the goal.

To adapt the B* algorithm for searching our theory poset, we must specify an static evaluation function that returns optimistic and pessimistic bounds on a leaf node; as well, we must specify the root node and how to generate successor nodes. Additionally, we must specify how to compare evaluations (i.e., we must specify $\geq^{P}$). As we will soon see, we must also

---

[2] By value of an arc, we mean value of the arc's corresponding node.

specify the termination condition, since the usual one for the B* search is too strong for our purposes (we will explain what we mean by this later).

We define the pessimistic value of a leaf node to be the persistence set corresponding to theory represented by the node; we define the optimistic value of a leaf node to be the persistence set corresponding to the most presumptive extension of the node's theory which is not currently known to be inconsistent. The root node of the search tree is the empty theory. Its static evaluation would return a pessimistic value equal to the empty (persistence) set and an optimistic value equal to the persistence set indicating all propositional fluents persist on each unit path. So for the example in the previous section, the theory corresponding to the root node is $\mathbf{T_{root}} = \mathbf{T_{min}} = \{\}$, the pessimistic evaluation is $\mathbf{PESS} = \mathbf{P_{min}} = \{\}$, and the optimistic evaluation is

OPT = {**persist(loaded,do(load,0))**,
persist(alive,do(load,0)),
persist(loaded,do(wait,do(load,0))),
persist(alive,do(wait,do(load,0))),
persist(loaded,do(shoot,do(wait,do(load,0)))),
persist(alive,do(shoot,do(wait,do(load,0))))}.

The children of a node are formed by extending the node's theory with an additional instance of a frame default. This implies that a node is always simpler than its children. As new nodes are generated, inconsistent nodes are pruned — instances of frame defaults found to be inconsistent with the

current node's theory (and the facts) are removed from the possible extensions to the theory. *Finding inconsistent extensions results in lowering the optimistic bound on the current theory, while finding consistent extensions has the effect of raising the pessimistic bound of the current theory.*

An optimisation is possible in the generation process. Since the chronological persistence criterion prefers theories extended by chronologically earlier frame defaults, there is no need to generate children formed from later defaults (as they cannot affect the backed-up scores). As a further optimisation, the children of the root can be treated as a special case. Because the goal (and its negation) may be logically independent of the theories represented by certain nodes in the search space and because we are only interested in deciding between the goal and its negation, we can skip these nodes by defining the children of the root node to be the nodes representing the simplest theories predicting the goal and the simplest theories predicting its negation. *This optimisation relies on the simplifying assumption we made in chapter 3, namely, that either the goal or its negation is expected.* Under this assumption, we do not need to consider theories that are independent of the goal. The procedure could be made more general by removing this optimisation. Finally, it often happens that the descendant of one node is a duplicate of the descendant of another node. The procedure given here does not attempt to merge these nodes, but clearly this would be desirable.

The next part of the search procedure to specifying is how to compare evaluations. This involves specifying the partial ordering $\geq^{P}$. Recall that this partial ordering is domain dependent. When the domain is chosen, $\geq^{P}$ is defined by indicating under what conditions $P^{i}_{T_1} \geq^{P} P^{i}_{T_2}$ (i.e., specify the ordering on persistence sets).

Finally, we must define the termination condition. This condition follows from Theorem 3.1 in the previous chapter. Essentially, if we ensure that *there is at least one theory predicting the goal that is chronologically more persistent than each maximal theory not predicting the goal, then all CMP theories predict the goal.* Consequently, we can terminate the search when the optimistic value of each theory not predicting the goal is "less than" the pessimistic value of at least one of the theories predicting the goal. Of course, the meaning of "less than" depends on $\geq^{P}$. Note that the termination check is done for the nodes attached to the root every time new scores are backed up to that level.

Upon termination, the procedure identifies the subset of nodes attached to the root which predict the expected goal. DT_A can be extracted by taking the disjunction of the theories corresponding to the leaf nodes that are descendants of the identified subset of nodes.

4.4. An Implementation of the Search Procedure

One implementation of the search procedure outlined in the previous section is provided in appendix A. The implementation was written in Waterloo Unix Prolog (WUP) [Cheng84]. It only identifies the subset of nodes attached to the root whose theories predict the expected goal, but we have described in the previous section how $\mathbf{DT_A}$ can be extracted.

In order to find the maximal element of $\overline{\Delta}$, and to find the simplest theories predicting the goal and those predicting its negation, and to check the consistency of the newly generated theories, the search procedure is interfaced to Theorist version 0.21 (appendix B). This version of Theorist modifies David Poole's version 0.2 to include checks for looping caused by the equivalences used in frame defaults (dealing with equivalence with be further discussed in section 4.6).

The major data structures of the program are the node and the search tree. A node consists of a name (implemented as a sequentially generated positive integer), a theory (implemented as a list of instances of frame defaults), the candidates (implemented as a list of instances of frame defaults that can be used to extend the theory), an arc (implemented as a list of instances of frame defaults that were used to extend the parent theory), and a score (implemented as a list representing the persistence sets corresponding to the optimistic and pessimistic evaluations). The search tree is represented as a list of terms; each term describes an arc in the tree.

Before the search begins, the initial list of candidates (i.e., all the instances of frame defaults in the maximal element of $\overline{\Delta}$) must be constructed. This is done by scanning the axiomatisation for defaults. Ground instances of these defaults are generated for each action and situation in the goal path. Note that the set of ground instances formed for these actions and situations is finite. Since in planning the goal path is not known in advance, the current implementation can only check plans. This approach only works when all defaults are frame defaults and when the only parameters of the frame defaults are the action and situation parameter.

When the initial list of candidates has been constructed, the search begins from the root node (whose theory is the empty theory). New nodes are generated by augmenting the current node's theory with an instance of a frame default drawn from the list of candidates (as an optimisation the selection is limited to the earliest instances, i.e., those corresponding to the earliest unit path which has candidates). This process could be improved by making it goal-directed. As described earlier, the children of the root are treated as a special case — all simplest theories predicting the goal or its negation are children of the root. When inconsistent nodes are discovered, they are pruned and the list of candidates for the is updated accordingly. The static evaluation assigns the persistence set corresponding to the node's theory as the pessimistic bound and it assigns the pessimistic bound augmented by the current list of candidates as the optimistic bound. As was mentioned earlier,

there are two strategies that could be followed at the root of the search tree: PROVEBEST and DISPROVEREST. Of these, only PROVEBEST is currently implemented.

Recall that the termination condition (given in the previous section) depends on the definition of $\geq^p$. In order to define $\geq^p$ the user must provide the definition of the predicate: **p_ge(P1,P2)**. This predicate is true when the persistence set represented by list **P1** is $\geq^p$ the persistence set represented by list **P2**. For example, if $\geq^p$ is taken to be $\supseteq$ then **p_ge(P1,P2)** is defined as:

p_ge(P1,P2) ← subset(P2,P1);

Or if $\geq^p$ is a comparison of the cardinality of persistence sets then **p_ge(P1,P2)** is defined as:

p_ge(P1,P2) ← length(P1,LP1) length(P2,LP2) ge(LP1,LP2);

Note that the predicate **length** determines the cardinality of a set. Any arbitrarily complex definition of $\geq^p$ is possible, but it must be *sensibly* defined (cf. definition 3.12) for Theorem 3.1 to be valid. With some modification, the comparison could take into account the persistences of propositional fluents over earlier unit paths or even the past sequence of actions. This would be useful in domains where current persistence depends on past (e.g., metal fatigue). Many other improvements to the program in appendix A are possible.

4.5. Test Results

The axiomatisations for two test problems are included in appendix C. These axiomatisations are expressed in the syntax of Theorist version 0.21. The major types of statements are:

fact p(X);	which means $\{\mathbf{p(X)}\} \subseteq \mathbf{F}$,
fact n(p(X));	which means $\{\neg \mathbf{p(X)}\} \subseteq \mathbf{F}$,
fact p(X) <- q(X);	which means $\{\mathbf{p(X)} \leftarrow \mathbf{q(X)}\} \subseteq \mathbf{F}$,
default p(X);	which means $\{\mathbf{[X]\ p(X)}\} \subseteq \Delta$.

Note that asserting an implication as a fact means that both the implication and its contrapositive can be used for inferencing. Because the representation language of Theorist version 0.21 does not include equivalence, frame defaults were represented as follows:

```
fact r(do(A,S)) <- r(S) frame_r(A,S);
fact n(r(do(A,S))) <- n(r(S)) frame_r(A,S);
default frame_r(A,S);
```

Still awaiting implementation is a method for generating ground instances of frame defaults which have parameters other than the action and situation (i.e., of the form **frame_r(X,A,S)**). For now we use a more verbose axiomatisation, e.g., in the second test problem we use **frame_ok_s(A,S)**, **frame_ok_a(A,S)**, and **frame_ok_b(A,S)** instead of **frame_ok(X,A,S)**

The first test problem is the Yale shooting scenario. For this test, $\geq^{p}$ was defined as $\supseteq$, since all we know about persistence in this domain is the common sense law of inertia. The goal for the test was

alive(do(shoot,do(wait,do(load,0)))).

The results of this test are in appendix D. The program was able to correctly identify the subset of nodes (attached to the root) whose corresponding theory predicts the expected

n(alive(do(shoot,do(wait,do(load,0))))).

Here the only node attached to the root that predicts the expected result of the actions is node 2. To extract $\mathbf{DT_A}$, we must take the disjunction of the leaf node descendant of node 2. A formated listing of the final search tree is included in appendix D. From this we see that the only leaf node descendant of node 2 is node 4. Thus,

$\mathbf{DT_A}$ = {**frame_alive(load,0),**
frame_alive(wait,do(load,0)),
frame_loaded(wait,do(load,0))}.

As was noted in chapter 3, $\mathbf{DT_A}$ for the Yale Shooting Scenario is equivalent to $\mathbf{DT_{CMP}}$, that is, no savings could be realised for this problem.

For the second test problem (the light circuit problem of chapter 5), $\geq^{\mathbf{p}}$ was defined as a comparison of the cardinality of persistence sets. This was done because of the given failure rate information. Since the failure rates of the devices are the same, we give the same weight to the persistences of the propositional fluents that indicate whether the devices are operational.

The goal for this problem was **ok_s(do(flip,0))**. The results of this test are in appendix D. Again the program correctly identified the subset of nodes that predict the expected result. For this problem, there are two nodes attached to the root that predict **n(ok_s(do(flip,0)))**, namely, node 2 and node 3 (see the formatted search tree listing in appendix D). Taking the disjunction of the leaf node descendant of nodes 2 and 3 (i.e., the disjunction of nodes 4 and 5) gives

$$\mathbf{DT_A} = \{\mathbf{frame\_ok\_a(flip,0)},\ \mathbf{frame\_ok\_b(flip,0)}\}.$$

Notice that nodes 4 and 5 represent the same theory — ideally, the procedure should merge these nodes. In this problem, as in the previous one, $\mathbf{DT_A} \equiv \mathbf{DT_{CMP}}$. Therefore, no savings are possible for this problem.

The third test (the sky-diving scenario — figure 3.8) illustrates a problem for which $\mathbf{DT_A}$ is different from $\mathbf{DT_{CMP}}$. Here we are interested in whether John will be alive after the actions **guzzle** and **jump**. The program correctly selects node 2 (see appendix D) as the subset of nodes predicting the expected result (i.e., **n(alive(do(jump,do(guzzle,0))))**). For this problem,

$$\mathbf{DT_A} = \{\mathbf{frame\_alive(guzzle,0)},\ \mathbf{frame\_defective(guzzle,0)}\}.$$

As was noted in chapter 3, the unique CMP theory is

T_{CMP} = {**frame_alive(guzzle,0),**
frame_defective(guzzle,0),
frame_defective(jump,do(guzzle,0)),
frame_happy(jump,do(guzzle,0))}.

By comparing, we note that DT_A eliminates two of the frame default instances. Thus, a savings is realised.

4.6. Other Implementation Issues

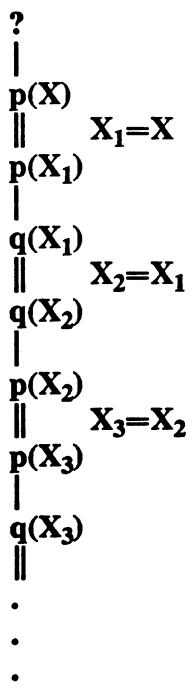
Here we briefly discuss three implementation issues: dealing with equivalence, undecidability and the reverse skolemisation problem. The problem with equivalence arises because frame defaults are equivalence assumptions. When deriving the consequences of a logical theory that includes equivalences, resolution proofs may contain infinite branches. For example, given the axioms:

p(X) ← q(X),
q(X) ← p(X),
q(a)

and the query:

?p(X)

then the left most branch of the proof tree has the form



The presence of this infinite branch in the proof tree will prevent finding the solution **X=a** when a depth first left to right search is used.

The above example could be solved by reordering the the axioms. Another way to solve this problem is to include a loop check in the theorem prover. Covington [Covington85] suggests that infinite branches can be avoided by 'blocking' the current branch of the proof tree whenever the new subgoal 'matches' a subgoal higher in the tree. Poole and Goebel [Poole85] show "that this works only in limited cases," and they argue that "these cases can be better avoided by slight modifications of the program, rather than by increasing the complexity of all programs with a rule that has very limited applicability." They also note that "it does work for biconditionals,

but in this case one would probably prefer to choose a canonical element of the equivalence class, and convert all other equivalent cases to that element."

While it may be preferable to eliminate infinite branches through the use of canonical elements, we have adopted the simpler loop checking approach. Whenever a new subgoal is "syntactically identical" (cf. **is_same** predicate in appendix B) to a subgoal that occurred earlier in the current branch of the search tree then the branch can be blocked. Since this loop checking approach uses a linear search of the ancestor subgoals, the theorem prover's efficiency diminishes as the length of the proof branch increases. Perhaps the canonical element approach will not suffer from this performance degradation.

Another implementation issue is that of dealing with undecidability. In the Theorist framework, a theory is said to explain a set of observations if the theory together with the facts is consistent and entail the observations, i.e.,

$\mathbf{F} \cup \mathbf{T} \models \mathbf{G}$, and
$\mathbf{F} \cup \mathbf{T}$ is consistent.

This can be implemented by using derivability in place of validity. As Poole et al. explain [Poole87c], the construction of consistent explanations can be done in two steps. First, try to prove **G** from the facts and instances of elements of Δ, i.e.,

$$\mathbf{F} \cup \mathbf{T} \vdash \mathbf{G}$$

Second, verify that each hypothesis used in the explanation is consistent, i.e.,

$$\mathbf{F} \cup \mathbf{T} \nvdash \neg \mathbf{t}, \forall \mathbf{t} \in \mathbf{T}$$

Note that each step uses a first-order theorem prover, and since first-order satisfiability is only semidecidable, the two steps taken together are completely undecidable. Thus, when Theorist is asked to explain an arbitrary set of observations, it might never return an answer, even if a solution exists. In critical applications, this would be unacceptable.

One way to deal with this problem is to limit the expressiveness of the underlying language so that satisfiability is decidable. As Levesque and Brachman have pointed out, there is a trade-off between expressive power and computational tractability [Levesque85]. An example of a restricted language which is decidable yet more expressive than propositional logic is that proposed by Patel-Schneider [Patel-Schneider86]. Instead of limiting the expressiveness of the underlying language, another approach is to tentatively accept explanations whose consistency is only partially tested. As was mentioned in chapter 2, such potentially refutable explanations are not as well-confirmed as explanations which are known to be consistent with the (known) facts. This approach seems to have a solid basis in the philosophy of science (cf. [Popper58, Rescher70, Hempel65]).

Finally, there is the problem of reverse skolemisation. Poole [Poole87b] has explained how this problem arises when hypotheses have variables. Essentially, the problem is that variables in hypotheses need to have their quantification reversed for consistency checking. In the examples given in this dissertation, all hypotheses were ground so that the reverse skolemisation problem is avoided; however, it is possible to envision situations where it is desirable to have variables in hypotheses. For instance, in that Yale shooting scenario, the relation **alive** could have an argument representing the person who is alive (e.g., **alive(john,0)**). Rather than have an instance of the frame default

[X,A,S] alive(X,do(A,S)) ↔ alive(X,S)

for each person, e.g.,

T = {alive(john,do(load,0)) ↔ alive(john,0),
alive(fred,do(load,0)) ↔ alive(fred,0),
etc.},

it may be desirable to assume the equivalence is true for all persons:

T = {alive(X,do(load,0)) ↔ alive(X,0)}.

If this is done, then consistency checking is subject to the reverse skolemisation problem. A solution to this problem has been proposed by Poole [Poole87b].

Chapter 5
Conclusions

5.1. Related Work

Hanks and McDermott [Hanks85, Hanks86] show that using default (or other nonmonotonic) reasoning to deal with the frame problem inevitably results in the need to choose between multiple models. Because the popular forms of nonmonotonic reasoning don't seem to provide a mechanism for choosing between models, they come to the discouraging conclusion that logic is inadequate as an AI representation language. They turn to a direct procedural characterisation to describe default reasoning processes and give an algorithm that generates their intended model for a set of axioms. In our terms, this model is a model of $\mathbf{F} \cup \mathbf{T}_{\mathbf{CMP}}$. We have demonstrated how this theory can be arrived at in a simple, intuitive theory formation framework through a semantically well-defined theory preference heuristic. Hopefully, this disspells the perception of inadequacy of logic stemming from Hanks' and McDermott's work.

Another related idea is that of Lifschitz [Lifschitz85, Lifschitz86a, Lifschitz86b]. He makes the observation that the usual forms of circumscription [McCarthy80] are inadequate for dealing with axiomatisations of

planning problems (cf. [McCarthy86]). To overcome this inadequacy, he introduces *pointwise circumscription*. Our notion of chronological maximisation of persistence is analogous to a form of prioritised pointwise circumscription which prefers "minimisation at earlier moments of time". More recently, Lifschitz has devised a formalism which uses ordinary circumscription to minimises "causes" and "preconditions." [Lifschitz87] The claim is that minimising causes allows minimisation of uncaused changes which captures the intuition behind the commonsense law of inertia. Minimising preconditions minimises the qualifications for the performance of action. Thus, Lifschitz' recent work addresses both the frame and qualification problem. Lifschitz is able to avoid the multiple model problem because the *causes* relation has no situation parameter. This is similar to Kowalski's *preserves* relation [Kowalski79, pp. 136-137] which is minimised through negation-as-failure. While not having a situation parameter in causes (or preserves) does avoid the multiple model problem, it seems to do this by sacrificing expressiveness. For instance, the changes induced by flipping a toggle switch depend on the previous situation; so it seems in this case, the causes relation for the action **flip** needs a situation parameter. We see once again that the choice of ontology an important consideration in dealing with the frame problem.

The work of Shoham [Shoham86a, Shoham86b] is also related to the work presented here. Work on the *initiation problem*[1] led him to the idea of *chronological maximisation of ignorance*. While the initiation problem is different from the frame problem, solutions to each problem reflect the need to maximise (or minimise) step by step (i.e., chronologically).

Recent work by Kowalski and Sergot [Kowalski86a, Kowalski86b] also addresses the frame problem. More specifically, Kowalski [Kowalski86b] proposes a first order *persistence* axiom that specifies how one can deduce whether a relation holds at a given time in a particular temporal database. A database is formalised as the ground terms of a logical theory specified in *event calculus*; the first order ground atomic formula **holds(r,t)** is true when **r** is an instance of a relation, and **t** is a time interval over which the relation is true. Database update constraints are specified in terms of **terminate** and **initiate** conditions on relations, events and actions. The epistemological aspect of the frame problem is claimed to be solved by axiomatising the database in terms of a single relation **holds** (cf. [Kowalski79]), and relying on the persistence axiom's use of negation-as-failure to assume that nothing affects the truth of a relation unless explicitly declared. The heuristic aspect of the frame problem is viewed as efficiently using the persistence axiom to answer

---

[1] This is the qualification problem in a temporal setting.

the general question of whether an instance of a relation **r** holds at time **t**. This problem is solved by describing an efficient method for implementing the use of the persistence axiom.

This notion of persistence, rendered as a first order axiom that specifies what "holds" in terms of how explicit initiate and terminate conditions affect the truth of relations, is only weakly related to the notion of persistence described here. Kowalski's formulation has no explicit concept of the update constraints being contingent or assumable if consistent, consequently there is no possibility of multiple conflicting answers to the question "does **r** hold at time **t**?" The persistence axiom relies on negation-as-failure while isolating those constraints that affect the truth of a relation so that different answers to the same question can be had due to intervening updates. However, there is no possibility of uncertain or multiple possible responses to questions of a relation's future persistence.

It seems that Kowalski assumes that frame axioms as defaults are unnecessary, as the concept of non-monotonicity can be handled more generally by using negation-as-failure within the persistence axiom. However, the burden to explicitly assert the affect of actions on relations remains; a default-like statement of the form "normally relation R persists" is not possible. Kowalski's alternative definition might be rephrased as "relation R persists, until I tell you otherwise."

Finally, Kautz [Kautz86] proposes a solution to the frame problem using a generalisation of circumscription. He defines the following partial ordering of models:

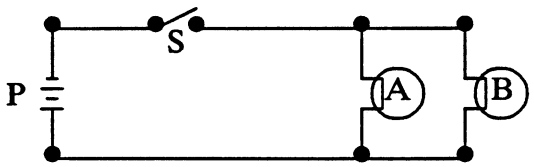
$$\mathbf{M1} \leq \mathbf{M2} \text{ if and only if}$$
$$\forall \mathbf{t,f} \,.\, \mathbf{(t,f)} \in \mathbf{M1[Clip]} \supset (\mathbf{(t,f)} \in \mathbf{M2[Clip]}) \vee$$
$$\exists \mathbf{t2,f2} \,.\, \mathbf{t2<t} \;\&\; (\mathbf{(t2,f2)} \in \mathbf{M2[Clip]}) \;\&\; (\mathbf{(t2,f2)} \notin \mathbf{M1[Clip]})$$
and
$$\mathbf{M1} < \mathbf{M2} \text{ if } \mathbf{M1} \leq \mathbf{M2} \text{ and not } \mathbf{M2} \leq \mathbf{M1}$$

The predicate **Clip(t,f)** is true when the persistence of a fact **f** ceases at time **t**. From this definition, a model **M1** is strictly better than **M2** (i.e., $\mathbf{M1} < \mathbf{M2}$) when they are identical (in terms of Clip) up to some time **t** at which some fact **f** changes in **M2** but not in **M1** (i.e., $\mathbf{(t,f)} \in \mathbf{M2[Clip]}$ but $\mathbf{(t,f)} \notin \mathbf{M1[Clip]}$).

Kautz's model ordering corresponds to the chronological maximisation of persistence when $\geq^{\mathbf{P}}$ is taken to be $\supseteq$. To illustrate this, consider the example in Figure 5.1 and two of the possible models corresponding to it (Fig. 5.2).

Here we have two lights (A and B) connected in parallel through a switch (S) to a power source (P). For the period of interest, the switch and the two lights have an equivalent failure rate. The power source and the wiring are completely reliable.

Under Kautz's model ordering, $\mathbf{M_1}$ and $\mathbf{M_2}$ are incomparable (both are minimal). When $\geq^{\mathbf{P}}$ is taken to be $\supseteq$ (i.e., the common sense law of inertia) then a theory $\mathbf{T_1}$ corresponding to $\mathbf{M_1}$ has the same chronological persistence



F = {	
¬ on(s,0),	**¬ hold(0,on(s))**
¬ on(a,0),	**¬ hold(0,on(a))**
¬ on(b,0),	**¬ hold(0,on(b))**
ok(s,0),	**hold(0,ok(s))**
ok(a,0),	**hold(0,ok(a))**
ok(b,0),	**hold(0,ok(b))**
on(s,1),	**hold(1,on(s))**
¬ on(a,1),	**¬ hold(1,on(a))**
¬ on(b,1),	**¬ hold(1,on(b))**
on(a,T) ←	**hold(T,on(a)) ←**
on(s,T) ∧ ok(s,T) ∧ ok(a,T),	**hold(T,on(s)) ∧ hold(T,ok(s)) ∧ hold(T,ok(a))**
n(on(a,T)) ← n(on(s,T)),	**¬ hold(T,on(a)) ← ¬ hold(T,on(s))**
n(on(a,T)) ← n(ok(s,T)),	**¬ hold(T,on(a)) ← ¬ hold(T,ok(s))**
n(on(a,T)) ← n(ok(a,T)),	**¬ hold(T,on(a)) ← ¬ hold(T,ok(a))**
on(b,T) ←	**hold(T,on(b)) ←**
on(s,T) ∧ ok(s,T) ∧ ok(b,T),	**hold(T,on(s)) ∧ hold(T,ok(s)) ∧ hold(T,ok(b))**
n(on(b,T)) ← n(on(s,T)),	**¬ hold(T,on(b)) ← ¬ hold(T,on(s))**
n(on(b,T)) ← n(ok(s,T)),	**¬ hold(T,on(b)) ← ¬ hold(T,ok(s))**
n(on(b,T)) ← n(ok(b,T))}	**¬ hold(T,on(b)) ← ¬ hold(T,ok(b))**
Δ = {	
[X,T] ok(X,T+1) ↔ ok(X,T)}	**hold(T+1,F) ⊕ clip(T+1,F) ← hold(T,F)**

Figure 5.1. Comparison with Kautz´s Model Ordering

Model M_1		
	0	1
ok(s)	T	T
ok(a)	T	F
ok(b)	T	F
on(s)	F	T
on(a)	F	F
on(b)	F	F

Model M_2		
	0	1
ok(s)	T	F
ok(a)	T	T
ok(b)	T	T
on(s)	F	T
on(a)	F	F
on(b)	F	F

Figure 5.2. Two of the Models for Figure 5.1

as T_1 (corresponding to M_2) over a path corresponding to the action between time **0** and time **1**. Instead, if $\geq^p$ is taken to be a comparison of the cardinality of the persistence sets (i.e., common sense law of inertia plus failure rate information) corresponding to the above theories then T_2 has more chronological persistence than T_1. Thus, incorporating the assumption that all persistences are equally likely by defining $\geq^p$ to be a comparison of cardinalities enables us to distinguish the two models. The preference heuristic reflects our expectation that the switch failing alone is a better theory than both lights failing.

5.2. Future Work

There are numerous ways that our simple representation scheme can be extended. In the examples given earlier, to axiomatise the initial situation, it was necessary to provide assertions for both the positive and the negative information. Often, the negative assertions greatly outnumber the positive assertions. If this is generally the case, we would like to invoke the *closed world assumption* [Reiter78, Reiter85], that is, we would like to have to specify only positive information about the initial situation and have all negative information inferred by default. The closed world assumption can be applied to all relations in the domain, or just to some, but its use implies perfect knowledge of the relations it is applied to. If we have incomplete information about some relation then the closed world assumption should not be applied to that relation (unless we want the system to behave as if it had perfect information). The closed world assumption can be represented in Theorist as a possible hypothesis. In ontology-G, for each primitive relation **r** in the domain of interest we could include a *closed world default* of the form:

[X] ¬r(X,0);

or in ontology-K, we could include a single closed world default:

[X] ¬holds(r(X),0).

As an example, consider the Yale shooting scenario.

F = {	The set of facts (F):
	Initial Situation:
alive(0),	John is alive
	Action: load
loaded(do(load,S)),	The gun is loaded after the action load
	Action: wait (no known changes)
	Action: shoot
¬ alive(do(shoot,S)) ← loaded(S),	John dies when shot with a loaded gun
¬ loaded(do(shoot,S))}	After shooting, the gun is not loaded
Δ = {	The set of defaults (Δ):
	The Closed World Defaults:
[] ¬ loaded(0),	
[] ¬ alive(0),	
	The Frame Defaults:
[A,S] loaded(do(A,S)) ↔ loaded(S),	
[A,S] alive(do(A,S)) ↔ alive(S)}	

Figure 5.3. Yale Shooting Scenario - Theorist Axiomatisation for Ontology-G

Compare the axiomatisation using closed world defaults (Fig. 5.3) with the axiomatisation which explicitly gives negative information about the initial situation (Fig. 3.4). One possible problem with the use of closed world defaults is their possible interaction with frame defaults. For instance, it is possible to explain **¬loaded(0)** using the closed world default **¬loaded(0)**; at the same time, it is possible to explain **loaded(0)** using the frame default **loaded(do(load,0)) ↔ loaded(0)** and the fact **loaded(do(load,0))**. It seems that this problem can be avoided by preferring theories with *maximal closure*. Maximal closure can probably be defined in a similar fashion to maximal

persistence, i.e., a theory which includes the closed world defaults of another theory is at least as good in terms of closure. It also seems that preferring maximal closure should be given priority over preferring (chronological) persistence.

Another area where the representation could be extended is in expressing incomplete knowledge. There are several areas where knowledge can be incomplete: initial conditions, action preconditions, action effects, and constraints. For example, if we have incomplete knowledge about a blocks world, then we might like to assume that, initially, the blocks are not too heavy and they are on the table [Lifschitz85, p. 2]. For the action **move(X,Y)** to succeed, we might assume that **clear(X)** is a precondition. This assumption is refuted when the only objects on **X** are trivial (e.g., a speck of dust [McCarthy86, p. 101]). Again, we might like to assume that the effect of moving **X** to **Y** is that **X** is on **Y**, but this may not be true if the resulting tower of blocks is unstable. Another assumption we might make is that a block can only be on one other block (or on the table). One can easily imagine a situation where this weak constraint is violated (e.g., the top block is very large and hence overlaps blocks adjacent to the bottom block). Figure 5.4 gives a partial axiomatisation of the above blocks world. The interesting question that remains unanswered about such axiomatisations is what kinds of interactions occur between the various defaults and what forms of theory

F = { The set of facts (F):
clear(a,0),
poss(do(move(X,Y),S)) ← on(Z,X,S) ∧ trivial(Z),
¬ on(X,Y,do(move(X,Y),S)) ← on(Y,Z,S) ∧ on(Z,V,S) ∧ on(V,W,S),
on(X,Y,S) ← on(X,Z,S) ∧ adjacent(Y,Z,S) ∧ very_large(X)}

Δ = { The set of defaults (Δ):
[X] ¬ too_heavy(X,0),
[X] ¬ on(X,table,0),
[X,Y,S] on(X,Y,do(move(X,Y),S)),
[X,Y,S] poss(do(move(X,Y),S)) ← clear(X,S) ∧ clear(Y,S),
[X,Y,S] ¬ on(X,Y,S) ← on(X,Z,S) ∧ diff(Y,Z)}

Figure 5.4. Blocks World - Theorist Axiomatisation for Ontology-G

preference are appropriate.

Other issues that require work in the future are:

1) conditionals vs. theory preference - instead of using theory preference to select a preferred theory, the disjunction of possible explanations could be used as a basis for conditional planning. The decision to use conditionals instead of using theory preference is tied to an effort versus certainty tradeoff, that is, using theory preference increases the knowledge about a situation but at the cost of increased computation. Sometimes the extra knowledge is not need. For instance, you can decide you don't need bus fare equally well knowing your car will start or knowing either your car will start or your neighbour will drive you to work. In this case, using extra computation (theory preference) to determine your car will start would be foolish.

2) error recovery - inevitably, even the best made plans may fail. When a plan is executed and the real world is found to differ from the predictions of our preferred theory, how can this theory (and, subsequently, our plan) be revised? When forming the plan initially, should competing theories be saved for use in error recovery? When an assumption has been violated, what conclusions must be retracted? Much work related to this question has been done (cf. [Fikes75, Hayes75, Stallman77, London78, Doyle79, Doyle80, deKleer79]).

3) relationship to other forms of nonmonotonic reasoning - comparison of axiomatisations based on theory formation/theory preference with those based on circumscription and with those based on negation as failure seem to suggest the existence of a continuum defined in terms of how assumptions are expressed and interpreted. The continuum ranges from axiomatisations where assumptions are selected by explicit preference heuristics, to axiomatisations where preference for assumptions is implicit (or 'compiled in'). For example, in negation-as-failure, preference information derived from syntactic structure is "compiled into" the proof procedure; while in circumscription, preference is expressed as priorities. Some evidence for this continuum has been presented [Goebel87]. Its is conjectured that program transformation techniques [Seki86] may prove useful for 'compiling' from one point on the continuum to another.

Another interesting direction future research might take is to see what insight can be gained from the philosophy of science. Earlier, we discussed the distinction between predictive and retrodictive explanations. One idea to pursue is suggested in [Goodwin87a]. There we make an important distinction between the task of predicting some future situation and the task of (retrodictively) explaining some past situation. It was suggested that preferred temporal theories for prediction tasks are ones that use predictive arguments. Likewise preferred temporal theories for explaining tasks are ones that use retrodictive arguments. Using the temporal structure of theories as a basis for theory preference is in accord with the philosophy of science.

Gagné and I are beginning to explore this area [Goodwin87b]. We have written a very simple Prolog program for prediction tasks. This program attempts to construct a purely predictive proof for the desired goal — potential proofs that involve retrodiction are failed. Like Theorist, the program incorporates the concept of fact and default. The nonmonotonic nature of defaults is simulated using negation as failure. Our preliminary results are promising. The program (Fig. 5.5) seems adequate for predicting tasks. It is interesting to note that the program seems to encode Gagné's and Poole's notion of theory formation *constraint* [Gagné87, Poole87]. Constraints are a form of theory preference. It will be interesting to see whether our proposed theory preference criterion can be expressed as constraints.

```
predict(G) <- predict(G [] []);

%     G is predicted by contradiction

predict(G Anc) <- neg(G NG) member(NG Anc) cut;

%     Loop deletion

predict(G Anc) <- member(A Anc) is_same(G A) cut fail;

%     Prediction from facts

predict(G Anc) <- fact(G []);
predict(G Anc) <- fact(G Body)
              all_prior(Body G)   % Temporal constraint
              predict_all(Body [G|Anc]);

%     Prediction from defaults
%     Negation as failure simulates defaults

predict(G Anc) <- default(G [])
              neg(G NG)
              not(predict(NG []));% Fail to predict NG

predict(G Anc) <- default(G Body)
              variable_free(Body)
              all_prior(Body G)   % Temporal constraint
              predict_all(Body [G|Anc])
              neg(G NG)
              not(predict(NG []));% Fail to predict NG

predict_all([]  Anc);
predict_all([H|R] Anc) <- predict(H Anc)
                    predict_all(R Anc);

%     Ensure each sub-goal's situation is no later than the goal's

all_prior([] G);
all_prior([H|R] G) <-
      prior(H G)
```

```
        all_prior(R G);

prior(H G) <-
        situation(H HS)
        situation(G GS)
        before(HS GS);

before(S S);
before(S1 do(A S2)) <- before(S1 S2);

situation(n(G) S) <- functor(G [H S]);
situation(G S) <- ne(G n(X)) functor(G [H S]);

neg(n(G) G);
neg(G n(G)) <- ne(G n(H));
```

Figure 5.5. Predictive Explanation Program

5.3. Conclusion

The frame problem is a fundamental aspect of planning — indeed, it is a fundamental problem in knowledge representation and reasoning. Many approaches to solving the frame problem have been proposed. Each has contributed something to our understanding of the problem. Many researchers have come to believe that the only reasonable approach to formalising rational reasoning and belief seems to be to use "scientific" theory formation. This position has a solid basis in the philosophy of science. In developing a representation scheme in a theory formation framework to deal with the frame problem, the problem of multiple theories arises. A method for selecting among competing theories based on a measure of persistence has been

presented. The validity of this idea is supported by an implementation. We have compared and contrasted our theory formation/theory preference approach with other recent proposals. Finally, we have speculated on possible directions for future research. The most interesting speculation is presented in the form of a hypothesis. We conjecture that the various forms of nonmonotonic reasoning can be usefully compared along a continuum represented the varying degrees of explicitness in the expression of theory preference. Because theory formation/theory preference is so useful, so simple, and so intuitively appealing, we suspect that it will turn out to be a fundamental paradigm in Artificial Intelligence. This dissertation is offered as evidence to support this conjecture.

References

[Berliner79]
H. J. Berliner (1979), The B* Tree Search Algorithm: A Best-First Proof Procedure, *Artificial Intelligence* **12**(1), 23-40.

[Brachman85]
R.J. Brachman and H.J. Levesque (1985, eds.), *Readings in Knowledge Representation*, Morgan Kaufmann Publishers, Los Altos.

[Brown86]
F.M Brown (1986), A comparison of the commonsense and fixed point theories of nonmonotonicity, *Proceedings of the National Conference on Artificial Intelligence (AAAI-86)*, August 11-15, University of Pennsylvania, Philadelphia, Pennsylvania, 394-400.

[Buchanan74]
J.R. Buchanan and D.C. Luckman (1974), On Automating the Construction of Programs, AIM-236, Artificial Intelligence Laboratory, Stanford University, Stanford, May [Informal Memo].

[Carnap50]
R. Carnap (1950), *Logical Foundations of Probability*, University of Chicago Press, Chicago, Illinois.

[Cheng84]
M.H.M. Cheng (1984), The design and implementation of the Waterloo Unix Prolog environment, M.Math thesis dissertation, Department of Computer Science, University of Waterloo, September, 114 [ICR Report 26; Department of Computer Science Technical Report CS-84-47].

[Clark78]
K.L. Clark (1978), Negation as failure, *Logic and Data Bases*, H. Gallaire and J. Minker (eds.), Plenum Press, New York, 293-322.

[Covington85]
M.A. Covington (1985), Eliminating unwanted loops in Prolog, *ACM SIGPLAN Notices* **20**(1), 20-26.

[deKleer79]
J. deKleer, J. Doyle, G.L. Steele, Jr., and G.J. Sussman (1979), Explicit Control of Reasoning, *Artificial Intelligence: An MIT perspective*, P.J. Winston and R.H. Brown (eds.), MIT Press, 93-116.

[Doyle79]
J. Doyle (1979), A Truth Maintenance System, *Artificial Intelligence* **12**, 231-272.

[Doyle80]
J. Doyle and P.E. London (1980), Selected Descriptor Index Bibliography to the Literature on Belief Revision, *SIGART Newsletter* **71**, 7-23.

[Fahlman74]
S.E. Fahlman (1974), A Planning System for Robot Construction Tasks, *Artificial Intelligence* **5**(1), Spring, North-Holland, Amsterdam, 1-49.

[Fikes70]
R.E. Fikes (1970), Ref-Arf: a system for solving problems stated as procedures, *Artificial Intelligence* **1**(1&2), Spring, North-Holland, Amsterdam, 27-120.

[Fikes71]
R.E. Fikes and N.J. Nilsson (1971), STRIPS: a new approach to the application of theorem proving in problem solving, *Artificial Intelligence* **2**(3&4), Winter, North-Holland, Amsterdam, 189-208.

[Fikes75]
R.E. Fikes (1975), Deductive Retrieval Mechanisms for State Description Models, *Proceedings of the Fourth International Joint Conference on Artificial Intelligence (IJCAI-75)*, September 3-8, Tblisi, USSR, 99-106.

[Friedland79]
P.E. Friedland (1979), Knowledge-based experiment design in molecular genetics, TR79-771, Computer Science Department, Stanford University, Stanford.

[Gagné86]
D. Gagné, R. Goebel, and D. Poole (1986), Theory Formation for High Level Scene Analysis, Department of Computer Science, University of Waterloo, Waterloo, Ontario, December, 14 pages.

[Gagné87]
D. Gagné (1987), The Multiple Extension Problem Revisited, Department of Computer Science, University of Waterloo, April.

[Goebel85]
R.G. Goebel, K. Furukawa, and D.L. Poole (1985), Using definite clauses and integrity constraints as the basis for a theory formation approach to diagnostic reasoning, Research report CS-85-50, Department of Computer Science, University of Waterloo, Waterloo, Ontario, December.

[Goebel87]
R.G. Goebel and S.D. Goodwin (1987), Applying theory formation to the planning problem, *The Frame Problem in Artificial Intelligence: Proceedings of the 1987 Workshop*, April 13-15, University of Kansas, Lawrence, Kansas, F.M. Brown (eds.), 207-232.

[Goodwin87a]
S.D. Goodwin (1987), Temporal Explanation: Prediction and Retrodiction, Department of Computer Science, University of Waterloo, Waterloo, Ontario, February, 14 pages [revised April].

[Goodwin87b]
S.D. Goodwin and D. Gagné (1987), Theory Formation: Prediction via constraints, Department of Computer Science, University of Waterloo, Waterloo, Ontario, May [in preparation].

[Green69]
C.C. Green (1969), Theorem proving by resolution as a basis for question-answering systems, *Machine Intelligence*, vol. 4, B. Meltzer and D. Michie (ed.), American Elsevier, New York, 183-205.

[Green81]
C.C. Green (1981), Application of Theorem Proving to Problem Solving, *Readings in Artificial Intelligence*, B.L. Webber and N.J. Nilsson (eds.), Morgan Kaufmann Publishers, Los Altos, California, 202-222.

[Hanks85]
S. Hanks and D. McDermott (1985), Temporal reasoning and Default Logics, TR YALEU/CSD/RR# 430, Computer Science Department, Yale University.

[Hanks86]
S. Hanks and D. McDermott (1986), Default Reasoning, Nonmonotonic Logic, and the Frame Problem, *Proceedings of the National Conference for Artificial Intelligence (AAAI-86)* **1**, August 11-15, University of Pennsylvania, Philadelphia, Pennsylvania, 328-333.

[Hayes71]
P.J. Hayes (1971), A Logic of Actions, *Machine Intelligence*, vol. 6, B. Meltzer and D. Michie (eds.), Edinburgh University Press, Edinburgh, Scotland, 495-520.

[Hayes73]
P.J. Hayes (1973), The frame problem and related problems in Artificial Intelligence, *Artificial and Human Thinking*, A. Elithorn and D. Jones (eds.), Jossey-Bass, London, England, 45-59.

[Hayes75]
P.J. Hayes (1975), A Representation for Robot Plans, *Proceedings of the Fourth International Joint Conference on Artificial Intelligence (IJCAI-75)*, September 3-8, Tblisi, USSR, 181-188.

[Hayes77]
P.J. Hayes (1977), In Defence of Logic, *Proceedings of the Fifth International Joint Conference on Artificial Intelligence (IJCAI-77)*, August 22-25, MIT, Cambridge, Massachusetts, 559-565.

[Hayes-Roth80]
B. Hayes-Roth (1980), Human Planning Processes, R-2670-ONR, Rand Corp., Santa Monica.

[Hempel48]
C. Hempel and P. Oppenheim (1948), Studies in the Logic of Explanation, *Philosophy of Science* **15**.

[Hempel65]
C.G. Hempel (1965), *Aspects of scientific explanation and other essays in the philosophy of science*, The Free Press, New York.

[Hewitt71]
C. Hewitt (1971), PLANNER: A Language for Proving Theorems in Robots, *Proceedings of the Second International Joint Conference on Artificial Intelligence (IJCAI-71)*, September 1-3, Imperial College, London, 295-301.

[Hewitt73]
C. Hewitt, P. Bishop, and R. Steiger (1973), A Universal Modular ACTOR Formalism for Artificial Intelligence, *Proceedings of the Third International Joint Conference on Artificial Intelligence (IJCAI-73)*, August, Stanford, 235-245.

[Israel80]
D.J. Israel (1980), What's wrong with non-monotonic logic, *Proceedings of the First Annual National Conference on Artificial Intelligence (AAAI-80)*, August 18-21, Stanford University, Stanford, California, 99-101.

[Jackson87]
W.K. Jackson (1987), A Theory Formation Framework for Learning by Analogy, Master's dissertation, Department of Computer Science, University of Waterloo, Waterloo, Ontario, January.

[Jones85]
M. Jones and D. Poole (1985), An expert system for educational diagnosis based on default logic, *Proceedings of the Fifth International Workshop on Expert Systems and their Applications*, May 13-15, Palais des Papes, Avignon, France, 573-583.

[Kautz86]
H. Kautz (1986), The logic of persistence, *Proceedings of the National Conference on Artificial Intelligence (AAAI-86)*, August 11-15, University of Pennsylvania, Philadelphia, Pennsylvania, 401-405.

[Kowalski79]
R.A. Kowalski (1979), *Logic for Problem Solving*, Artificial Intelligence Series 7, Elsevier North Holland, New York.

[Kowalski86a]
R.A. Kowalski and M.J. Sergot (1986), A Logic-based calculus of events, *New Generation Computing* **4**(1), Ohmsha Ltd. and Springer Verlag, 67-95.

[Kowalski86b]
R.A. Kowalski (1986), Database updates in the event calculus, DOC 86/12, Department of Computing, Imperial College, London, England, July, 29 pages.

[Levesque85]
H.J. Levesque and R.J. Brachman (1985), A fundamental tradeoff in knowledge representation and reasoning, *Readings in Knowledge Representation*, R.J. Brachman and H.J. Levesque (eds.), Morgan Kaufmann, Los Altos, California, 42-70.

[Lifschitz85]
V. Lifschitz (1985), Circumscription in the Blocks World, Stanford University, Stanford, December, 17 pages [unpublished].

[Lifschitz86a]
V. Lifschitz (1986), Pointwise Circumscription, Stanford University, Stanford, March, 15 pages [unpublished].

[Lifschitz86b]
V. Lifschitz (1986), Pointwise circumscription: preliminary report, *Proceedings of the National Conference on Artificial Intelligence (AAAI-86)*, August 11-15, University of Pennsylvania, Philadelphia, Pennsylvania, 406-410.

[Lifschitz87]
V. Lifschitz (1987), Formal Theories of Action, *The Frame Problem in Artificial Intelligence: Proceedings of the 1987 Workshop*, April 13-15, University of Kansas, Lawrence, Kansas [to appear].

[Lloyd84]
J.W. Lloyd (1984), *Foundations of logic programming*, Springer-Verlag, New York.

[London78]
P.E. London (1978), Dependency Networks as a Representation for Modelling in General Problem Solvers, TR-698, Computer Science Department, University of Maryland.

[Manna70]
Z. Manna (1970), The Correctness of Nondeterministic Programs, *Artificial Intelligence* **1**(1&2), Spring, North-Holland, New York, 1-26.

[McCarthy68]
J. McCarthy (1968), Programs with Common Sense, *Semantic Information Processing*, M. Minsky (ed.), MIT Press, Cambridge, Massachusetts, 403-418.

[McCarthy69]
J. McCarthy and P.J. Hayes (1969), Some Philosophical Problems from the Standpoint of Artificial Intelligence, *Machine Intelligence*, vol. 4, B. Meltzer and D. Michie (eds.), Edinburgh University Press, Edinburgh, 463-502.

[McCarthy77]
J. McCarthy (1977), Epistemological problems of Artificial Intelligence, *Proceedings of the Fifth International Joint Conference on Artificial Intelligence (IJCAI-77)*, August 22-25, MIT, Cambridge, Massachusetts, 1038-1044.

[McCarthy80]
J. McCarthy (1980), Circumscription—a form of non-monotonic reasoning, *Artificial Intelligence* **13**(1&2), April, North-Holland, Amsterdam, 27-39.

[McCarthy86]
J. McCarthy (1986), Applications of circumscription to formalising common-sense knowledge, *Artificial Intelligence* **28**(1), February, North-Holland, Amsterdam, 89-116.

[McDermott80]
D. McDermott and J. Doyle (1980), Non-Monotonic Logic I, *Artificial Intelligence* **13**(1&2), April, North-Holland, Amsterdam, 41-72.

[Minsky72]
M. Minsky and S. Papert (1972), Progress Report, AIM-252, Artificial Intelligence Laboratory, MIT, Cambridge, Massachusetts.

[Neufeld87]
E.M. Neufeld (1987), Thinking Logically about Probabilities, Department of Computer Science, University of Waterloo, Waterloo, Ontario, January.

[Newell63]
A. Newell and H.A. Simon (1963), GPS, a Program that Simulates Human Thought, *Computers and Thought*, E.A. Feigenbaum and J. Feldman (eds.), McGraw-Hill, New York, 279-293.

[Patel-Schneider86]
P.F. Patel-Schneider (1986), A Hybrid, Decidable, Logic-Based Knowledge Representation System, *Sixth Canadian Conference on Artificial Intelligence*, May 21-23, Ecole Polytechnique de Montreal, Montreal, Quebec, 210-214.

[Poole87a]
D. Poole (1987), A Logical Framework for Default Reasoning, Department of Computer Science, University of Waterloo, Waterloo, Ontario, April, 21 pages.

[Poole85a]
D.L. Poole (1985), On The Comparison of Theories: Preferring the Most Specific Explanation, *Proceeding of the Ninth International Joint Conference on Artificial Intelligence (IJCAI-85)*, August 16-18, UCLA, Los Angeles, California, 144-147.

[Poole85b]
D.L. Poole and R. Goebel (1985), On eliminating loops in Prolog, *ACM SIGPLAN Notices* **20**(8), 38-41.

[Poole86]
D.L. Poole (1986), Default Reasoning and Diagnosis as Theory Formation, CS-86-08, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, March, 19 pages.

[Poole87b]
D.L. Poole, R.G. Goebel, and R. Aleliunas (1987), Theorist: a logical reasoning system for defaults and diagnosis, *The Knowledge Frontier: Essays in the Representation of Knowledge*, N.J. Cercone and G. McCalla (eds.), Springer-Verlag, New York, 331-352 [in press].

[Poole87c]
D.L. Poole (1987), Variables in Hypotheses, *Proceedings of the Tenth International Joint Conference on Artificial Intelligence (IJCAI-87)* **1**, August 23-28, Milano, Italy [submitted].

[Popper58]
K.R. Popper (1958), *The Logic of Scientific Discovery*, Harper & Row, New York.

[Raphael71]
B. Raphael (1971), The frame problem in problem-solving systems, *Artificial Intelligence and Heuristic Programming*, N.V. Findler and B. Meltzer (eds.), American Elsevier, New York, 159-169.

[Reiter78]
R. Reiter (1978), On Closed World Data Bases, *Logic and Data Bases*, H. Gallaire and J. Minker (eds.), Plenum Press, New York, 55-76.

[Reiter80]
R. Reiter (1980), A logic for default reasoning, *Artificial Intelligence* **13**(1&2), April, North-Holland, Amsterdam, 81-132.

[Reiter85]
R. Reiter (1985), On Reasoning By Default, *Readings in Knowledge Representation*, R.J. Brachman and H.J. Levesque (eds.), Morgan Kaufman Publishers, Los Altos, 401-410.

[Rescher64]
N. Rescher (1964), *Hypothetical Reasoning*, North-Holland, Amsterdam.

[Rescher70]
N. Rescher (1970), *Scientific Explanation*, Collier-MacMillian Canada, Toronto.

[Rulifson72]
J.F. Rulifson, J.A. Derksen, and R.J. Waldinger (1972), QA4: a procedural calculus for intuitive reasoning, Technical Note 73, Stanford Research Institute, Menlo Park, California, November.

[Sacerdoti77]
E.D. Sacerdoti (1977), *A Structure for Plans and Behaviour*, Artificial Intelligence Series 3, Elsevier North-Holland, New York.

[Sandewall72]
E. Sandewall (1972), An Approach to the Frame Problem, and its Implementation, *Machine Intelligence*, vol. 7, B. Meltzer and D. Michie (eds.), Edinburgh University Press, Edinburgh, 195-204.

[Sandewall73]
E. Sandewall (1973), Conversion of Predicate Calculus Axioms, Viewed as Non-deterministic Programs, to Corresponding Deterministic Programs, *Proceedings of the Third International Joint Conference on Artificial Intelligence (IJCAI-73)*, August, Stanford, 230-234.

[Seki86]
H. Seki and K. Furukawa (1986), Compiling control by a program transformation approach, Institute for New Generation Computer Technology, Tokyo, Japan, November, 24 pages.

[Shoham86a]
Y. Shoham (1986), Chronological Ignorance: Time, Knowledge, Nonmonotonicity, and Causation, Computer Science Department, Yale University, April, 24 pages.

[Shoham86b]
Y. Shoham (1986), Chronological Ignorance: time, nonmonotonicity, necessity, and causal theories, *Proceedings of the National Conference for Artificial Intelligence (AAAI-86)* **1**, August 11-15, University of Pennsylvania, Philidelphia, Pennsylvania, 389-393.

[Simon66]
H.A. Simon and N. Rescher (1966), Cause and Counterfactual, *Philosophy of Science*, 323-340.

[Simon67]
H.A. Simon (1967), The Logic of Heuristic Decision Making, *The Logic of Decision and Action*, N. Rescher (ed.), University of Pittsburgh Press, Pittsburgh.

[Smith85]
B.C. Smith (1985), Prologue to "Reflection and Semantics in a Procedural Language", *Readings in Knowledge Representation*, R.J. Brachman and H.J. Levesque (eds.), Morgan Kaufmann Publishers, Los Altos, 31-39.

[Stallman77]
R.M. Stallman and G.J. Sussman (1977), Forward Reasoning and Dependency-directed Backtracking in a System for Computer-aided Circuit Analysis, *Artificial Intelligence* **9**, 135-196.

[Sussman72]
G.J. Sussman and D.V. McDermott (1972), Why conniving is better than planning, AI Memo 255A, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, April.

[Sussman73]
G.J. Sussman (1973), A Computational Model of Skill Acquisition, AI-TR-297, AI Laboratory, MIT, Cambridge, Massachusetts, August.

[Turing63]
A.M. Turing (1963), Computing machinery and intelligence, *Computers and Thought*, E.A. Feigenbaum and J. Feldman (eds.), McGraw-Hill, New York, 11-35 [reprinted from *Mind* **59**, 1950, 433-460, also in *Minds and Machines*, A.R. Anderson (ed.), Prentice-Hall, Englewood Cliffs, New Jersey, 4-30].

[vanArragon86]
P. vanArragon (1986), Using Scientific Theory Formation for User Modelling, Department of Computer Science, University of Waterloo, October, 30 pages.

[Waldinger77]
R. Waldinger (1977), Achieving Several Goals Simultaneously, *Machine Intelligence*, vol. 8, Edinburgh University Press, Edinburgh, 94-136.

[Wilber76]
M.A. Wilber (1976), A QLISP Reference Manual, TN-118, Artificial Intelligence Center, Stanford Research Institute, Menlo Park, March.

[Winograd72]
T. Winograd (1972), *Understanding Natural Language*, Academic Press, New York.

[Winograd85]
T. Winograd (1985), Frame Representations and the Declarative/Procedural Controversy, *Readings in Knowledge Representation*, R.J. Brachman and H.J. Levesque (eds.), Morgan Kaufman Publishers, Los Altos, 357-370.

Appendix A
An Implementation

```
%
%     The search procedure described in chapter 4 is used
%     to select the children of the root node whose descendant
%     leaf nodes form DT_A.
%     This search procedure is based on Berliner's B* search.
%     Best is the set of selected nodes.
%     Final_search_tree is  list representing the final state of
%     the search tree.
%

bstar(Best Final_search_tree) <-
      init_search_tree(Initial_node Initial_search_tree)
      bstar_top(Initial_node Best Initial_search_tree Final_search_tree);

%
%     Process the children of the root. An optimisation is employed
%     here. The children of the root are the simplest theories
%     predicting the goal or its negation.
%

bstar_top(Initial_node Best Initial_search_tree Final_search_tree) <-
      node.name(Initial_node Initial_name)
      first_expand(Initial_name Initial_search_tree New_search_tree)
      terminate(Initial_name New_search_tree Best Final_search_tree);

%
%     Generate and evaluate the children of the root.
%     Join them to the search tree.
%

first_expand(Cur_name Search_tree New_search_tree) <-
      get_node(Cur_name Search_tree Cur_node)
      node.object(Cur_node Object)
      generate(Object Child_objects)
      node.candidates(Cur_node Cand)
      init_candidates(Child_objects Cand Init_objects)
      evaluate(Init_objects Scored_nodes)
      join_nodes(Scored_nodes Cur_name Search_tree New_search_tree);

%
```

```
%     Generate and evaluate children of a (non-root) node.
%     Prune inconsistent nodes and update candidates list accordingly.
%     Join consistent nodes to search tree.
%

expand(Cur_name Search_tree Search_tree) <-
      child(Cur_name Search_tree Child)   % Test if already expanded
      cut;
expand(Cur_name Search_tree New_search_tree) <-
      get_node(Cur_name Search_tree Cur_node)
      node.object(Cur_node Object)
      generate(Object Child_objects)
      consistency_check(Child_objects Con_objects Incon_arcs)
      node.candidates(Cur_node Cand1)
      remove_arcs(Incon_arcs Cand1 Cand2)
      update_node.candidates(Cur_node Search_tree Cand2 Search_tree2)
      init_candidates(Con_objects Cand2 Init_objects)
      rank(Init_objects Cur_name Search_tree2 New_search_tree);

rank([] _ Tree Tree) <-
      nl;
rank([Object|Objects] Cur_name Search_tree2 New_search_tree) <-
      evaluate([Object|Objects] Scored_nodes)
      join_nodes(Scored_nodes Cur_name Search_tree2 New_search_tree);

consistency_check([] [] []);
consistency_check([Object|Rest] Con_objects Incon_arcs2) <-
      object.arc(Object Arc)
      object.theory(Object Theory)
      inconsistent(Theory Arc)
      cut
      consistency_check(Rest Con_objects Incon_arcs1)
      union(Arc Incon_arcs1 Incon_arcs2);
consistency_check([Object|Rest] [Object|Con_objects] Incon_arcs) <-
      consistency_check(Rest Con_objects Incon_arcs);

init_candidates([] _ []);
init_candidates([Object1|Con_objects] Cand1 [Object2|Unscored_objects]) <-
      object.arc(Object1 Arc)
      remove_arcs(Arc Cand1 Cand2)
      update_object.candidates(Object1 Cand2 Object2)
      init_candidates(Con_objects Cand1 Unscored_objects);

update_object.candidates(Object1 Cand2 Object2)
      object.theory(Object1 Theory)
      object.theory(Object2 Theory)
      object.candidates(Object2 Cand2)
      object.goal(Object1 Goal)
      object.goal(Object2 Goal)
```

```
        object.arc(Object1 Arc)
        object.arc(Object2 Arc);

update_node.candidates(Node1 Search_tree1 Cand2 Search_tree2) <-
        node.name(Node1 Name)
        node.name(Node2 Name)
        node.object(Node1 Object1)
        node.object(Node2 Object2)
        object.theory(Object1 Theory)
        maximum(Theory Cand2 Max)
        node.score(Node1 [_ Min])
        node.score(Node2 [Max Min])
        update_object.candidates(Object1 Cand2 Object2)
        parent(Name Search_tree1 Parent1)
        node.name(Parent1 Parent_name1)
        replace(Name Node2 Parent_name1 Search_tree1 Search_tree2);

remove_arcs(_ [] []);
remove_arcs(Arcs [CH1|CT1] [CH2|CT2]) <-
        del_all(Arcs CH1 CH2)
        remove_arcs(Arcs CT1 CT2);

covers_all(CG [] CG);
covers_all(CG [CNGH|CNGT] CG) <-
        member(N CG)
        covers(N CNGH)
        covers_all(CG CNGT CG);
covers_all([] CNG CNG);
covers_all([CGH|CGT] CNG CNG) <-
        member(N CNG)
        covers(N CGH)
        covers_all(CGT CNG CNG);

covers(N1 N2) <-
        node.score(N1 [_ Min])
        node.score(N2 [Max _])
        cp_gt(Min Max);

split([] [] []);
split([CH|CT] CG CNG) <-
        split_one(CH CG1 CNG1)
        split(CT CG2 CNG2)
        append(CG1 CG2 CG)
        append(CNG1 CNG2 CNG);

split_one(C [] [C])
        node.goal(C n(G))
        cut;
split_one(C [C] []);
```

```
%
%     Check for termination conditions.
%     If necessary, determine search strategy and search lower tree.
%

terminate(Cur_name Final_search_tree Best Final_search_tree) <-
        children(Cur_name Final_search_tree Children)
        split(Children CG CNG)
        covers_all(CG CNG Best)
        cut;

terminate(Cur_name Search_tree Best Final_search_tree) <-
        children(Cur_name Search_tree Children)
        select_strategy(Children Strategy)
        select_next_node(Strategy Children Next_node)
        node.name(Next_node Next_name)
        bstar_lower(Next_name Search_tree New_search_tree)
        terminate(Cur_name New_search_tree Best Final_search_tree);

%
%     Control search of lower tree. Generate and evaluate children
%     as necessary.
%

bstar_lower(Cur_name Search_tree New_search_tree) <-
        expand(Cur_name Search_tree Search_tree2)
        children(Cur_name Search_tree2 Children)
        search_lower(Cur_name Children Search_tree2 New_search_tree);

search_lower(_ [] Tree Tree);
search_lower(Cur_name Children Search_tree2 New_search_tree) <-
        calc(Children Score)
        explore(Cur_name Search_tree2 Score Search_tree3 New_score)
        update_node.score(Cur_name Search_tree3 New_score New_search_tree);

%
%     Search lower tree.
%

explore(Cur_name Search_tree Score Search_tree Score) <-
        get_node(Cur_name Search_tree Cur_node)
        node.score(Cur_node Cur_score)
        ne(Score Cur_score)
        cut;
explore(Cur_name Search_tree Score New_search_tree New_score) <-
        children(Cur_name Search_tree Children)
        select_next_node2(Children Next_node)
        node.name(Next_node Next_name)
```

```
        bstar_lower(Next_name Search_tree Search_tree2)
        children(Cur_name Search_tree2 New_children)
        search_deeper(Cur_name New_children Search_tree2
                        New_search_tree Score New_score);

search_deeper(_ [] Tree Tree Score Score);
search_deeper(Cur_name New_children Search_tree2
                New_search_tree Score New_score) <-
        calc(New_children Score2)
        explore(Cur_name Search_tree2 Score2 New_search_tree New_score);

%       Used to get node data give node name

get_node(Node_name Tree Node) <-
        node.name(Node Node_name)
        member(link(Parent_name Node) Tree);

%
%       Node data structure.
%
%       Node := [ Node_name Object Score ]
%       Node_name := nil | 0 | 1 | 2 | ...
%       Object := [ Theory Candidates Arc Goal]
%       Theory := [ P1 P2 ... Pn ]
%       Pi := list of persistence assumptions for unit path i
%       Candidates := [ C1 C2 ... Cn ]
%       Ci := list of potential persistence assumptions for unit path i
%       Arc := list of persistence assumptions added to parent to create child
%       Goal := indicates whether node's theory predicts goal or its negation
%       Score := [ Max Min ]
%       Max := optimistic score
%       Min := pessimistic score
%

node.name([Node_name | _] Node_name);

node.object([Name Object | _] Object);

node.theory(Node Theory) <-
        node.object(Node Object)
        object.theory(Object Theory);

node.candidates(Node Candidates) <-
        node.object(Node Object)
        object.candidates(Object Candidates);

node.arc(Node Arc) <-
        node.object(Node Object)
        object.arc(Object Arc);
```

```
node.goal(Node Goal) <-
        node.object(Node Object)
        object.goal(Object Goal);

node.score([Name Object Score] Score);

object.theory([Theory _ _ _] Theory);

object.candidates([_ Candidates _ _] Candidates);

object.arc([_ _ Arc _] Arc);

object.goal([_ _ _ Goal] Goal);

%
%       Used to retrieve parent node given child name.
%

parent(Child_name Tree Parent) <-
        node.name(Child Child_name)
        node.name(Parent Parent_name)
        child(Parent_name Tree Child);

%
%       Used to retrieve a child node given parent name.
%

child(Parent_name Tree Child) <-
        member(link(Parent_name Child) Tree);

%
%       Used to retrieve all children nodes given a parent name.
%

children(Parent_name Tree Children) <-
        all_of(Children Child child(Parent_name Tree Child));

%
%       Generate children nodes.
%

generate(Parent_object Child_objects) <-
        all_of(Child_objects Child_object arc(Parent_object Child_object));

%
%       Evaluate a list of node objects and return scored nodes.
%

evaluate(Objects Nodes) <-
```

```
        all_of(Nodes Node eval_one(Objects Node));

%
%       Select and evaluate one node object and return scored node.
%

eval_one(Objects Node) <-
        member(Object Objects)
        val(Object Node);

%
%       Initialise search tree.
%       Generate initial list of candidates.
%

init_search_tree(Initial_node Initial_search_tree) <-
        delta(Delta)
        node.name(Initial_node 0)
        node.object(Initial_node Object)
        object.theory(Object nil)
        object.candidates(Object Delta)
        object.arc(Object nil)
        object.goal(Object nil)
        node.score(Initial_node nil)
        record_next_name(nil 1)
        join(Initial_node nil nil Initial_search_tree);

%
%       Join a node to its parent in the tree.
%

join(Node Parent_name Tree New_tree) <-
        append([link(Parent_name Node)] Tree New_tree);

%
%       Join a list of children to their parent in the tree.
%

join_nodes([] _ Search_tree Search_tree);
join_nodes([Node|Node_tail] Cur_name Search_tree New_search_tree) <-
        join(Node Cur_name Search_tree Search_tree2)
        join_nodes(Node_tail Cur_name Search_tree2 New_search_tree);

%
%       Select a search strategy. Presently only PROVEBEST is
%       implemented.
%

select_strategy(Children Strategy) <-
```

```
        eq(Strategy prove_best);

%
%       Select next node to search using chosen search strategy.
%

select_next_node(Strategy Children Next_node) <-
        eq(Strategy prove_best)
        find_max_opt(Children Next_node _);

%
%       Find greatest optimistic and pessimistic score.
%

calc(Children [Max_opt Max_pess]) <-
        find_max_opt(Children Node1 [Max_opt _])
        find_max_pess(Children Node2 [_ Max_pess]);

%
%       Find the node with the highest optimistic score.
%

find_max_opt(Children Node [Max_opt Pess]) <-
        del(Node Children Rest)
        node.score(Node [Max_opt Pess])
        not(find_higher_opt(Rest Max_opt));

%
%       Try to find a node with a higher optimistic score than Max_opt.
%

find_higher_opt(Children Max_opt) <-
        member(Node Children)
        node.score(Node [Opt _])
        cp_gt(Opt Max_opt);

%
%       Try to find a node with a higher pessimistic score than Max_pess.
%

find_higher_pess(Children Max_pess) <-
        member(Node Children)
        node.score(Node [_ Pess])
        cp_gt(Pess Max_pess);

%
%       Find the node with the highest pessimistic score.
%
```

```
find_max_pess(Children Node [Opt Max_pess]) <-
        del(Node Children Rest)
        node.score(Node [Opt Max_pess])
        not(find_higher_pess(Rest Max_pess));

%
%       Update the score of a node in the search tree.
%

update_node.score(Cur_name Search_tree New_score New_search_tree) <-
        get_node(Cur_name Search_tree Cur_node)
        node.object(Cur_node Cur_object)
        parent(Cur_name Search_tree Parent_node)
        node.name(Parent_node Parent_name)
        node.name(New_node Cur_name)
        node.object(New_node Cur_object)
        node.score(New_node New_score)
        replace(Cur_name New_node Parent_name Search_tree New_search_tree);

%
%       Replace a node with an updated node.
%

replace(Node_name New_node Parent_name Search_tree New_search_tree) <-
        node.name(Node Node_name)
        del(link(Parent_name Node) Search_tree T)
        append([link(Parent_name New_node)] T New_search_tree);

%
%       Select next node to search (using PROVEBEST)
%

select_next_node2(Children Next_node) <-
        select_next_node(prove_best Children Next_node);

%
%       Used to keep track of last node name generated.
%

record_next_name(nil N) <-!
        assert(mod next_name(N) []);
record_next_name(Old New) <-
        retract(mod next_name(Old))
        assert(mod next_name(New) []);

%
%       Comparisons of chronological persistence
%       This corresponds to >^cp in chapter 3.
%
```

```
cp_gt([] []) <-
        p_gt([] []);
cp_gt([PH1|PT1] [PH2|PT2]) <-
        p_gt(PH1 PH2);
cp_gt([PH1|PT1] [PH2|PT2]) <-
        p_eq(PH1 PH2)
        cp_gt(PT1 PT2);

cp_ge([] []) <-
        p_ge([] []);
cp_ge([PH1|PT1] [PH2|PT2]) <-
        p_gt(PH1 PH2);
cp_ge([PH1|PT1] [PH2|PT2]) <-
        p_eq(PH1 PH2)
        cp_ge(PT1 PT2);

%
%       Comparisons of persistence
%       This corresponds to >^p in chapter 3.
%

p_gt(P1 P2) <-
        p_ge(P1 P2)
        not(p_ge(P2 P1));

p_eq(P1 P2) <-
        p_ge(P1 P2)
        p_ge(P2 P1);


%
%       User defined partial odering ≥^p.
%

%
%       This defines the partial ordering on persistence
%       sets in terms of subsets
%

p_ge(P1 P2) <-
        subset(P2 P1);

subset([] Y);
subset([H|T] Y) <-
        delete(H Y Y1)
        subset(T Y1);

%
%       This defines the partial ordering on persistence
```

```
%     sets in terms of cardinality
%

%p_ge(P1 P2) <-
%     cardinality(P1 C1)
%     cardinality(P2 C2)
%     ge(C1 C2);

cardinality(List Length) <-
      length(List Length);

%
%     Generate children of the root (simplest theories predicting
%     the goal).
%

arc(Parent Child) <-
      object.theory(Parent nil)
      prove(mod goal(G))
      explain([G] T)
      functor(G [_ S])
      group(T S Theory)
      object.theory(Child Theory)
      object.goal(Child G)
      object.arc(Child T);

%
%     Generate children of the root (simplest theories predicting
%     the goal's negation).
%

arc(Parent Child) <-
      object.theory(Parent nil)
      prove(mod goal(G))
      neg(G GN)
      explain([GN] T)
      functor(G [_ S])
      group(T S Theory)
      object.theory(Child Theory)
      object.goal(Child GN)
      object.arc(Child T);

%
%     Generate (non-root) children.
%

arc(Parent Child) <-
      object.theory(Parent Theory1)
      ne(Theory1 nil)
```

```
        object.candidates(Parent Candidates)
        first_nonempty(Candidates 1 CI I)
        member(C CI)
        join_to(Theory1 1 C I Theory2)
        object.theory(Child Theory2)
        object.arc(Child [C])
        object.goal(Parent Goal)
        object.goal(Child Goal);

%
%       Group persistence assumptions by unit path.
%

group(T do(A S) Theory) <-!
        all_of(DS1 D group_sit(T do(A S) D))
        group(T S DS2)
        append(DS2 [DS1] Theory);
group(T S []);

group_sit([TH|TT] do(A S) TH) <-
        functor(TH [R A S]);
group_sit([TH|TT] S D) <-
        group_sit(TT S D);

%
%       Find the first nonempty element of a list of lists.
%

first_nonempty([H|T] I H I) <-
        ne(H []);
first_nonempty([[]|T] J H I) <-
        sum(J 1 JP1)
        first_nonempty(T JP1 H I);

%
%       Join an element to the proper list-element of a list of lists.
%

join_to([TH1|TT] I CIH I [TH2|TT]) <-
        append([CIH] TH1 TH2);
join_to([TH|TT1] J CIH I [TH|TT2]) <-
        gt(I J)
        sum(J 1 JP1)
        join_to(TT1 JP1 CIH I TT2);

%
%       Delete a list of elements from a list.
%
```

```
del_all([] L L);
del_all([H|T] L1 L2) <-
        del(H L1 L3)
        del_all(T L3 L2);
del(X [X|Y] Y);
del(X [Y|Z1] [Y|Z2]) <-
        del(X Z1 Z2);
del(X [] []);

%
%       Evaluate a node object and return a node.
%

val(Object Node) <-
        object.theory(Object Theory)
        object.candidates(Object Candidates)
        object.arc(Object Arc)
        minimum(Theory Min)
        maximum(Theory Candidates Max)
        next_name(Next_name)
        sum(Next_name 1 New_next_name)
        record_next_name(Next_name New_next_name)
        node.name(Node Next_name)
        node.object(Node Object)
        node.score(Node [Max Min]);

%
%       Minimum score.
%

minimum(Theory Theory);

%
%       Maximum score.
%

maximum([] [] []);
maximum([TH|TT] [CH|CT] [MH|MT]) <-
        union(TH CH MH)
        maximum(TT CT MT);

%
%       Check for inconsistency.
%

inconsistent(T [IncDef]) <-     % T implies IncDef, check if T implies NegIncDef
        flatten(T Theory)
        cut                     % flatten produces garbage on backtracking
        neg(IncDef NegIncDef)
```

```
        makeBody(Theory Theory Theory [] [NegIncDef] EB PB [])
        prove(mod PB);

flatten([H|T] FL) <-
        flatten(H FH)
        flatten(T FT)
        append(FH FT FL);
flatten([] []);
flatten(X [X]);

%
%       Generate initial candidates list.
%       See chapter 4 for restriction on use with
%       defaults with non-situation, non-action parameters.
%

delta(Delta) <-
        prove(mod problem(Problem))
        reread(Problem)
        find(D)
        prove(mod goal(Goal))
        functor(Goal GL)
        append(Front [S] GL)
        ground(D S Delta)
        close(Problem);
find(Delta) <-
        read_things(L)
        cut
        match(L D)
        find(DR)
        append(D DR Delta);
find([]);
match([default D] [D]) <-
        cut;
match(L []);

ground(D do(A S) Delta) <-!
        all_of(GDS1 GD ground_sit(D do(A S) GD))
        ground(D S GDS2)
        append(GDS2 [GDS1] Delta);
ground(D S []);

ground_sit([DH|DT] do(A Sit) DH) <-
        functor(DH DHL)
        append(First [A Sit] DHL);
ground_sit([DH|DT] S GD) <-
        ground_sit(DT S GD);

%
```

```
% "union" will perform the union of the first two lists and put this
% union in the third list.
%
% i.e., union( [a] [b] X ) returns X = [a b]
%
%

union( [X|L1] L2 L3 ) <-
        member( X L2 )
        cut
        union( L1 L2 L3 );
union( [X|L1] L2 L3 ) <-
        union( L1 [X|L2] L3 );
union( [] L1 L1 );
```

Appendix B
Theorist version 0.21

```
%
%     Note: This is a modified version of Theorist 0.2 which was
%           written by David Poole.
%
%     INTERFACE
%

start <-
        more(/u/sdgoodwin/theo/.initmsg)
        restart;

restart <-
        loop;

restart <-
        write("restarting\n")
        restart;

loop <-
        mark(Stack)
        prompt
        read_things(L)
        incommline(L)
        release(Stack)
        loop;

incommline([exit]) <- abort;

incommline(L) <-
        interpretcommand(L);

%interpretcommand is how to add new commands

% EXTENDED COMMANDS (Could possibly redefine other commands)

interpretcommand(X) <-
        extendedcommand(X);

% SYSTEM COMMANDS
interpretcommand([help]) <-!
```

```
        more(/u/sdgoodwin/theo/.help);

interpretcommand([quit])!
        quit;

interpretcommand([clear])!
        clear_aux(mod)
        init_mod;

interpretcommand([consult File]) <-!
        reread(File);

interpretcommand([end]) <-
        cur_in(File)
        close(File);

% DECLARATIONS

interpretcommand([fact H "<-" | Body]) <-!
        neg(H HN)
        indexfact([HN|Body] []);

interpretcommand([fact H]) <-!
        neg(H HN)
        indexfact([HN] []);

interpretcommand([default D]) <-!
        default(D);

interpretcommand([default D H "<-" | B]) <-!
        default(D)
        append([D] B NB)
        neg(H HN)
        indexfact([HN|NB] []);

interpretcommand([askable A]) <-!
        askable(A);

interpretcommand([prolog A]) <-!
        prolog(A);

% QUERIES

interpretcommand([explain |G]) <-
        explain(G T)
        write("yes\nanswer:")
        write(G)
        write("\ntheory:")
        space_output([T])
```

```
        nl;

interpretcommand([explain | G]) <-!
        write("no\n");

interpretcommand([retry]) <-
        cut
        fail;

% UNKNOWN COMMAND

interpretcommand(X) <-
        write("Unknown command: ")
        space_output(X)
        nl;

prompt <-
        cur_in(stdin)
        cut
        write("\n: ");

prompt <-
        not(cur_in(stdin));

neg(n(X) X);
neg(X n(X)) <- ne(X n(Y));

indexfact([] _);
indexfact([G|S] []) <-!
        neg(G GN)
        fact(1 GN S)        % priority of positive clause (High=1 Low=2)
        indexfact(S [G]);
indexfact([G|S] E) <-
        neg(G GN)
        append(S E B)
        fact(1 GN B)        % priority of contra-positive clauses (High=1 Low=2)
        indexfact(S [G|E]);
```

```
%
%     COMPILER
%

% fact(Priority Head Body) means that "Head <- Body" is a fact;
%     Priority is used to give some control over the order prolog
%     searches for rules;
%     Head is an atomic symbol and Body is a list of atomic symbols
%     Ancestors keeps track of the subgoals in the proof tree
%     Loop detection feature added here

fact(P H B) <-
      makeBody(T1 T1 T2 H B Exbody Prbody Ancestors)
      assert(mod exl(P H Ancestors T1 T2) [noloop(B Ancestors) |Exbody])
      assert(mod prl(P H Ancestors T1) [noloop(B Ancestors) |Prbody]);

% makeBody(T0 T1 T2 L EB PB Ancestors), where T0 is the initial theory,
%     T1 the current theory, T2 the final theory, and L the initial
%     body means EB is the body for explaining, PB is the body for
%     proving, and A is the list of ancestor subgoals

makeBody(T0 T T H [] [] [] _);

makeBody(T0 T1 T2 H [G|R] [exr(G [H|A] T1 T3) | EB] [prr(G [H|A] T0)|PB] A) <-
      makeBody(T0 T3 T2 H R EB PB A);

% default(Name ) means Name is a default which can be used if consistent

default(D) <-
      neg(D ND)
      assert(mod prl(1 D _ T) [member(D T)])
      assert(mod exl(1 D _ T T) [member(D T)])
      assert(mod exl(1 D _ T1 [D|T1]) [not(member(D T1)) not(prr(ND [] T1))]);

% askable(P) means anything matching P can be asked of the user

askable(P) <-
      assert(mod exr(P [] T T) [ask_user(P)])
      assert(mod prr(P [] T) [ask_user(P)])
      assert(mod exr(n(P) [] T T) [ask_user(n(P))])
      assert(mod prr(n(P) [] T) [ask_user(n(P))]);

% explain(P T) means Pcan be explained with theory T

explain(G T) <-
      makeBody([] [] T [] G NL PB [])
      prove(mod NL)
      ;
```

```
% prolog(P) means anything matching P can be proven in prolog

prolog(P) <-
        assert(mod prl(1 P [] T) [P])
        assert(mod exl(1 P [] T T) [P]);
```

```
%
%     EXTENDED_COMMANDS
%

%
%     Clear command modified to do reinitialisation
%
extendedcommand([clear]) <-!
        clear_aux(mod)
        init_mod
        init_ext;

extendedcommand([problem Problem]) <-!
        prove(mod problem(Cur_Problem))
        update_problem(Problem Cur_Problem)
        ;
extendedcommand([problem]) <-!
        prove(mod problem(Cur_Problem))
        update_problem(Cur_Problem Cur_Problem)
        ;

extendedcommand([goal Goal]) <-!
        prove(mod goal(Cur_Goal))
        update_goal(Goal Cur_Goal)
        ;
extendedcommand([goal]) <-!
        prove(mod goal(Cur_Goal))
        update_goal(Cur_Goal Cur_Goal)
        ;

%
%     Find all simplest theories.
%
extendedcommand([theories Goal]) <-!
        interpretcommand([goal Goal])
        find_theories(Goal)
        ;
extendedcommand([theories]) <-!
        prove(mod goal(Goal))
        find_theories(Goal)
        ;
%
%     Find a preferred theory.
%
extendedcommand([bstar]) <-!
        init_lib
        bstar(Best_node Final_search_tree)
        write("\nBest node: ")
        write(Best_node)
```

```
        write("\n\nFinal Search Tree: ")
        write(Final_search_tree)
        ;

find_theories(Goal) <-
        all_theories(Goal Theories)
        output_all_theories(Goal Theories)
        neg(Goal NegGoal)
        all_theories(NegGoal NegTheories)
        output_all_theories(NegGoal NegTheories)
        ;

all_theories(Goal Theories) <-
        all_of(Theories Theory explain([Goal] Theory))
        ;

output_all_theories(Goal []) <-!
        write("\n\nThere are no theories explaining: ")
        write(Goal)
        nl
        ;
output_all_theories(Goal Theories) <-
        write("\n\nAll simplest theories explaining: ")
        write(Goal)
        nl
        output_all(Theories 0 N)
        nl
        ;

output_all([] N N);
output_all([H|T] I N) <-
        sum(I 1 IP1)
        write("\nTheory ")
        write(IP1)
        write(": ")
        output_theory(H)
        output_all(T IP1 N)
        ;

output_theory([]) <-!
        nl
        tab
        write([])
        ;
output_theory(T) <-
        output_names(T)
        ;

output_names([]);
```

```
output_names([N|Ns]) <-
        output_name(N)
        output_names(Ns)
        ;
output_name(N) <-
        nl
        tab
        write(N)
        ;

update_problem(Problem Cur_Problem) <-
        ne(Problem Cur_Problem)
        interpretcommand([clear])
        retract(mod problem(Cur_Problem))
        assert(mod problem(Problem) [])
        write("\nNew Problem: ")
        write(Problem)
        interpretcommand([consult Problem])
        nl
        ;
update_problem(Problem Cur_Problem) <-
        eq(Problem Cur_Problem)
        write("\nCurrent Problem: ")
        write(Cur_Problem)
        nl
        ;
update_goal(Goal Cur_Goal) <-
        ne(Goal Cur_Goal)
        retract(mod goal(Cur_Goal))
        assert(mod goal(Goal) [])
        write("\nNew Goal: ")
        write(Goal)
        nl
        ;
update_goal(Goal Cur_Goal) <-
        eq(Goal Cur_Goal)
        write("\nCurrent Goal: ")
        write(Cur_Goal)
        nl
        ;
init_lib <-
        module(test _)
        cut;
init_lib <-
        import(test .);
```

```
%
%     LOOP_CHECK
%

noloop([] A);
noloop([B|R] A) <-
        not(loop(B A))
        noloop(R A);

loop(G A) <-
        member(M A)
        is_same(G M);

is_same(G M) <-
        is_var(G)
        is_var(M)
        cut
        same_var(G M);
is_same(G M) <-
        atomic(G)
        atomic(M)
        cut
        eq(G M);
is_same([] []);
is_same(G M) <-
        is_list(G)
        is_list(M)
        cut
        eq(G [GH|GT])
        eq(M [MH|MT])
        is_same(GH MH)
        is_same(GT MT);
is_same(G M) <-
        is_functor(G)
        is_functor(M)
        cut
        functor(G GL)
        functor(M ML)
        is_same(GL ML);

%
%     GROUND
%

ground(Term) <-
        not(is_var(Term))
        eq(Term [H|T])
        cut
        ground(H)
```

```
        ground(T);

ground(T) <-
        is_functor(T)
        cut
        functor(T L)
        ground(L);

ground(T) <-
        not(is_var(T));
```

```
%
%     ASK_USER
%

% "ask_user" is a predicate that will ask interactively whether or not the
% given question is true.  If the question has variables, an instance is asked
% for, and backtracking is allowed until a reply of "none" is given.

% Q is assumed to be a functor
ask_user(Q) <-
        get_vars(Q [F|R])          % has at least one variable
        ask_user_instance(Q [F|R])
        ;
ask_user(n(Q)) <-
        cut
        not(asked(Q))
        get_vars(Q [])                    % no variables
        printf("is % true?\n" [Q])
        read_term(A)
        replied(Q A)
        negative_answer(A)
        cut
        ;
ask_user(Q) <-
        not(asked(Q))
        get_vars(Q [])                    % no variables
        printf("is % true?\n" [Q])
        read_term(A)
        replied(Q A)
        affirmative_answer(A)
        cut
        ;

ask_user_instance(n(Q) Vars) <-
        cut
        not(already_asked(Q Vars))
        printf("for which % is % false?\n" [Vars Q])
        read_terms(T)
        interpret_neg_ans(T Q Vars)
        ;

ask_user_instance(Q Vars) <-
        not(already_asked(Q Vars))
        printf("for which % is % true?\n" [Vars Q])
        read_terms(T)
        interpret_pos_ans(T Q Vars)
        ;

interpret_pos_ans([none] Q V) <-
```

```
        cut
        assert(mod asked(Q) [])
        default(n(Q))
        cut fail;

interpret_pos_ans([unknown] Q V) <-
        cut
        assert(mod asked(Q) [])
        cut fail;

interpret_pos_ans(T Q Vars) <-
        eq(T Vars)
        assert(mod asked(Q) [])
        fact(Q [])
        ;
interpret_pos_ans(T Q Vars) <-
        ask_user_instance(Q Vars)
        ;

interpret_neg_ans([none] Q V) <-
        cut
        assert(mod asked(Q) [])
        default(Q)
        cut fail;

interpret_neg_ans([unknown] Q V) <-
        cut
        assert(mod asked(Q) [])
        cut fail;

interpret_neg_ans(T Q Vars) <-
        eq(T Vars)
        assert(mod asked(Q) [])
        fact(n(Q) [])
        ;
interpret_neg_ans(T Q Vars) <-
        ask_user_instance(n(Q) Vars)
        ;

% this pred analyzes a term and returns a list of its variables
get_vars(Q []) <-
        atomic(Q);
get_vars(Q [Q]) <-
        is_var(Q);
get_vars(Q L) <-
        is_functor(Q)
        functor(Q [F|R])
        get_vars(R L);
get_vars(Q []) <-
```

```
        is_list(Q)
        eq(Q []);
get_vars(Q L) <-
        is_list(Q)
        eq(Q [F|R])
        get_vars(F L1)
        get_vars(R L2)
        app_no_dup(L1 L2 L);

% append two lists of variables without duplications
app_no_dup([] L L);
app_no_dup([F|R] L Z) <-
        in_var_list(F L)
        app_no_dup(R L Z)
        ;
app_no_dup([F|R] L [F|Z]) <-
        not(in_var_list(F L))
        app_no_dup(R L Z)
        ;

% this succeeds if the variable is in the list
in_var_list(V [F|R]) <-
        same_var(V F)
        eq(V' F')
        ;
in_var_list(V [F|R]) <-
        in_var_list(V R)
        ;

affirmative_answer(yes);
affirmative_answer(y);

negative_answer(no);
negative_answer(n);

replied(Q A) <-
        save_reply(Q A)
        assert(mod asked(Q)[]);

save_reply(Q A) <-
        affirmative_answer(A)
        fact(Q [])!;

save_reply(Q A) <-
        negative_answer(A)
        fact(n(Q)[])!;

save_reply(Q A) ;
```

```
already_asked(Q V) <-
        not(not(already_asked_and_bind(Q V)));

already_asked_and_bind(Q V) <-
        skolemize(V)
        asked(Q);

skolemize([]);

skolemize([H|T]) <-
        gensym(H)
        skolemize(T);

gensym(V) <-
        retract(mod seed(S))
        plus(1 S S1)
        assert(mod seed(S1) [])
        itoa(S1 A1)
        strcat("gen" A1 V);
```

```
%
%     .INIT (WUP initialisation file)
%

?import(iolib /u/prolog/library/iolib);
?init_mod;
?init_ext;
?start;

%
%     INIT (other initialisations)
%

init_mod <-
       assert(mod exr(G A T1 T2) [ground(G) cut ex(G A T1 T2) cut])
       assert(mod exr(G A T1 T2) [ex(G A T1 T2)])
       assert(mod ex(G A T T) [neg(G GN) member(GN A)])
       assert(mod ex(G A T1 T2) [exl(1 G A T1 T2)])
%      assert(mod ex(G A T1 T2) [exl(2 G A T1 T2)])
       assert(mod prr(G A T) [ground(G) cut pr(G A T) cut])
       assert(mod prr(G A T) [pr(G A T)])
       assert(mod pr(G A T) [neg(G GN) member(GN A)])
       assert(mod pr(G A T) [prl(1 G A T)])
%      assert(mod pr(G A T) [prl(2 G A T)])
       assert(mod asked(this_question_should_never_be_asked) [])
       assert(mod seed(0) [])
       ;

init_ext <-
       assert(mod problem([]) [])
       assert(mod goal([]) [])
       ;
```

```
%
%     START UP MESSAGE
%

Welcome to Theorist 0.21
For help type "help;"
All complaints to David Poole (dlpoole@watdragon)
          or Scott Goodwin (sdgoodwin@watdragon)
Have Fun

%
%     HELP FILE
%

Commands are:

help;               prints this message
quit;               exits from the system
exit;               exits to wup
clear;              clear the database

consult "File";          read input from File
end;                end of input from file

fact Clause;        makes the clause a fact
default Name;            makes the Name a default
default Name Clause;     makes the clause a default
prolog Atom;             Atom should be proven as in Prolog
askable Atom;            the user should be asked whether Atom

explain Goal;            find a theory to explain the goal
retry;              find another answer

To questions asked of the user reply
"yes;" "no;" or "unknown;" to "is" questions
"none;" "unknown;" "list of values...;" to "for which" questions

Extended Commands are:

problem "File";          reads input from File (unless already read)
problem;            displays the name of the current problem file
goal Goal;          sets the current goal to Goal;
goal;               displays the current goal;
theories Goal;           find all theories explaining Goal and its negation
theories;           find all theories for current Goal and its negation
bstar;              find the preferred theory
```

Appendix C
Axiomatisations

```
%
%     Yale Shooting Scenario
%

fact alive(0);
fact n(loaded(0));

fact loaded(do(load,S));

fact n(alive(do(shoot,S))) <- loaded(S);
fact n(loaded(do(shoot,S)));

fact alive(do(A S)) <- alive(S) frame_alive(A S);
fact n(alive(do(A S))) <- n(alive(S)) frame_alive(A S);

fact loaded(do(A S)) <- loaded(S) frame_loaded(A S);
fact n(loaded(do(A S))) <- n(loaded(S)) frame_loaded(A S);

default frame_alive(A S);
default frame_loaded(A S);

end;
```

```
%
%    Light Circuit Example
%

fact n(on(s,0));
fact n(on(a,0));
fact n(on(b,0));

fact ok(s,0);
fact ok(a,0);
fact ok(b,0);

fact on(s,do(flip,0));
fact n(on(a,do(flip,0)));
fact n(on(b,do(flip,0)));

fact on(a,S) <- on(s,S) ok(s,S) ok(a,S);
fact n(on(a,S)) <- n(on(s,S));
fact n(on(a,S)) <- n(ok(s,S));
fact n(on(a,S)) <- n(ok(a,S));

fact on(b,S) <- on(s,S) ok(s,S) ok(b,S);
fact n(on(b,S)) <- n(on(s,S));
fact n(on(b,S)) <- n(ok(s,S));
fact n(on(b,S)) <- n(ok(b,S));

fact ok_s(do(A,S)) <- ok_s(S) frame_ok_s(A,S);
fact n(ok_s(do(A,S))) <- n(ok_s(S)) frame_ok_s(A,S);

fact ok_a(do(A,S)) <- ok_a(S) frame_ok_a(A,S);
fact n(ok_a(do(A,S))) <- n(ok_a(S)) frame_ok_a(A,S);

fact ok_b(do(A,S)) <- ok_b(S) frame_ok_b(A,S);
fact n(ok_b(do(A,S))) <- n(ok_b(S)) frame_ok_b(A,S);

default frame_ok_s(A,S);
default frame_ok_a(A,S);
default frame_ok_b(A,S);

end;
```

```
%
%     Sky-diving Scenario
%

fact defective(0);
fact alive(0);
fact n(happy(0));

fact happy(do(guzzle,S));

fact n(alive(do(jump,S))) <- defective(S);

fact defective(do(A,S)) <- defective(S) frame_defective(A,S);
fact n(defective(do(A,S))) <- n(defective(S)) frame_defective(A,S);

fact alive(do(A,S)) <- alive(S) frame_alive(A,S);
fact n(alive(do(A,S))) <- n(alive(S)) frame_alive(A,S);

fact happy(do(A,S)) <- happy(S) frame_happy(A,S);
fact n(happy(do(A,S))) <- n(happy(S)) frame_happy(A,S);

default frame_alive(A,S);
default frame_happy(A,S);
default frame_defective(A,S);

end;
```

Appendix D
Test Results - Final Search Trees

```
%
%     Yale Shooting Scenario
%

Best = { 2 }

Root Node = 0
Children:
        Node = 2
        Theory =
                frame_loaded(wait,do(load,0))
        Max Score =
                frame_alive(load,0)
                frame_loaded(wait,do(load,0))
                frame_alive(wait,do(load,0))
                frame_alive(shoot,do(wait,do(load,0)))
                frame_loaded(shoot,do(wait,do(load,0)))
        Min Score =
                frame_alive(load,0)
                frame_alive(wait,do(load,0))
                frame_loaded(wait,do(load,0))

        Children:
                Node = 3
                Theory =
                        frame_alive(load,0)
                        frame_loaded(wait,do(load,0))
                Max Score =
                        frame_alive(load,0)
                        frame_loaded(wait,do(load,0))
                        frame_alive(wait,do(load,0))
                        frame_alive(shoot,do(wait,do(load,0)))
                        frame_loaded(shoot,do(wait,do(load,0)))
                Min Score =
```

```
            frame_alive(load,0)
            frame_alive(wait,do(load,0))
            frame_loaded(wait,do(load,0))

      Children:
            Node = 4
            Theory =
                  frame_alive(load,0)
                  frame_alive(wait,do(load,0))
                  frame_loaded(wait,do(load,0))
            Max Score =
                  frame_alive(load,0)
                  frame_loaded(wait,do(load,0))
                  frame_alive(wait,do(load,0))
                  frame_alive(shoot,do(wait,do(load,0)))
                  frame_loaded(shoot,do(wait,do(load,0)))
            Min Score =
                  frame_alive(load,0)
                  frame_alive(wait,do(load,0))
                  frame_loaded(wait,do(load,0))



Node = 1
Theory =
      frame_alive(load,0)
      frame_alive(wait,do(load,0))
      frame_alive(shoot,do(wait,do(load,0)))
Max Score =
      frame_alive(load,0)
      frame_alive(wait,do(load,0))
      frame_alive(shoot,do(wait,do(load,0)))
      frame_loaded(shoot,do(wait,do(load,0)))
Min Score =
      frame_alive(load,0)
      frame_alive(wait,do(load,0))
      frame_alive(shoot,do(wait,do(load,0)))
```

```
%
%    Light Circuit Example
%

Best = { 2, 3 }

Root Node = 0
Children:
      Node = 1
      Theory =
            frame_ok_s(flip,0)
      Max Score = [1]
      Min Score = [1]

      Node = 2
      Theory =
            frame_ok_a(flip,0)
      Max Score = [2]
      Min Score = [2]
      Children:
            Node = 5
            Theory =
                  frame_ok_b(flip,0)
                  frame_ok_a(flip,0)
            Max Score = [2]
            Min Score = [2]


      Node = 3
      Theory =
            frame_ok_b(flip,0)
      Max Score = [2]
      Min Score = [2]
      Children:
            Node = 4
            Theory =
                  frame_ok_a(flip,0)
                  frame_ok_b(flip,0)
            Max Score = [2]
            Min Score = [2]
```

```
%
%    Sky-diving Scenario
%

Best = { 2 }

Root Node = 0
Children:
      Node = 1
      Theory =
            frame_alive(guzzle,0)
            frame_alive(jump,do(guzzle,0))
      Max Score =
            frame_alive(guzzle,0)
            frame_alive(jump,do(guzzle,0))
            frame_defective(jump,do(guzzle,0))
            frame_happy(jump,do(guzzle,0))
      Min Score =
            frame_alive(guzzle,0)
            frame_alive(jump,do(guzzle,0))

      Node = 2
      Theory =
            frame_defective(guzzle,0)
      Max Score =
            frame_defective(guzzle,0)
            frame_alive(guzzle,0)
            frame_alive(jump,do(guzzle,0))
            frame_defective(jump,do(guzzle,0))
            frame_happy(jump,do(guzzle,0))
      Min Score =
            frame_alive(guzzle,0)
            frame_defective(guzzle,0)

      Children:
            Node = 3
            Theory =
                  frame_alive(guzzle,0)
                  frame_defective(guzzle,0)
            Max Score =
                  frame_defective(guzzle,0)
                  frame_alive(guzzle,0)
                  frame_alive(jump,do(guzzle,0))
```

```
        frame_defective(jump,do(guzzle,0))
        frame_happy(jump,do(guzzle,0))
Min Score =
        frame_alive(guzzle,0)
        frame_defective(guzzle,0)
```