Reasoning About Action I:
A Possible Worlds Approach

Matthew L. Ginsberg
David E. Smith

Computer Science Department
Stanford University
Stanford, California 94305

Reasoning About Action I: A Possible Worlds Approach

Abstract

Reasoning about change is an important aspect of commonsense reasoning and planning. In this paper we describe an approach to reasoning about change for rich domains where it is not possible to anticipate all situations that might occur. The approach provides a solution to the frame problem, and to the related problem that it is not always reasonable to explicitly specify all of the consequences of actions. The approach involves keeping a single model of the world that is updated when actions are performed. The update procedure involves constructing the nearest world to the current one in which the consequences of the actions under consideration hold. The way we find the nearest world is to construct proofs of the negation of the explicit consequences of the expected action, and to remove a premise in each proof from the current world. Computationally, this construction procedure appears to be tractable for worlds like our own where few things tend to change with each action, or where change is regular.

1 Introduction

1.1 The problem

An important aspect of commonsense reasoning is the ability to reason about change. For example, imagine a robot responsible for various household chores. It must be able to infer that if it moves the lamp, the shade and cord move, too. But when it tries to move the bookcase, with the lamp on top, the cord will come tight, and the lamp will fall off and break. On the other hand, a napkin, or piece of paper on top of the bookcase might flutter off, but probably wouldn't cause any harm. Then too, the robot must know where to find things, like the vacuum, the carpet, the closet, or the bookcase. This requires the ability to infer that furniture and appliances stay in the same place unless specifically moved, and that these objects don't change their shape or color unless someone deliberately modifies them. There are effectively an infinite number of such reasoning situations that a household robot might come up against. It is therefore essential to its function that the robot be able to reason about actions and their effects on the world. Such abilities are also essential in somewhat more specialized domains, such as reasoning about mechanical devices and processes, and reasoning about analog circuits. For these applications a reasoner must be able to infer what will happen when various adjustments or stimuli are applied, or when parts malfunction.

Reasoning about action gives rise to three classical problems. The first is the *frame problem*, recognized and so named by McCarthy and Hayes [23]. The difficulty is that of indicating and inferring all those things that do not change when actions are performed and time passes. For example, when the lamp is moved from the bookcase to the endtable, we need to determine that the vacuum cleaner and the carpet remain stationary, and do not change shape or color.

The second, and closely related problem, is the *ramification problem* (so named by Finger [6]). The difficulty is that it is unreasonable to explicitly record all of the consequences of actions. For example, if we move the bookcase, all of the books inside move along with it. The papers and doilies on top move also. The arrangement of furniture is changed, and certain parts of the carpet become covered or uncovered. Some of the heating ducts and pictures may also be blocked. For any given action there are essentially an infinite number of possible consequences that might occur, depending upon the details of the situation in which the action occurs.

The third problem is called the *qualification problem*, also named by McCarthy [20]. The problem is that the number of preconditions for each action is immense. Imagine all of the things that could prevent the robot from moving the bookcase: it could be too heavy with all of the books in it, it could be too fragile to survive the move, it could be fastened to the floor, the floor where we might want to put it might be too weak to hold it, the door might be too small, the house might catch on fire, or there might be a nuclear war. Computationally, we cannot afford to check all of these unlikely possibilities explicitly.

In this paper we focus on the frame and ramification problems—specifically, our objective is to provide a formal mechanism for reasoning about action that provides 1) an epistemologically convenient way of expressing information about the domain, and 2) a computationally effective means of reasoning about the consequences of actions. In doing so, we will assume that not all of the consequences of actions are explicitly described. We will, however, ignore the qualification problem, assuming that all of the qualifications for the actions being considered are explicitly provided. In [11, 12] we show how the same techniques can be used to advantage in solving the qualification problem.

1.2 Classical approaches

1.2.1 The monotonic approach

The earliest proposal for formalizing reasoning about change is due to McCarthy [19, 23]. The approach involves providing explicit axioms for inferring what propositions hold following each possible action. These axioms are of two sorts: 1) *action axioms* indicating the things that change when each action takes place, and 2) *frame axioms* indicating the things that remain unaffected by each action.

As an example, consider a simple move operation for our household robot. If an object, x, and destination, l, are both clear, moving x to l will result in x being at l following the action. Formally, we can express this as:

$$\mathtt{clear}(x)_s \wedge \mathtt{clear}(l)_s \rightarrow \mathtt{on}(x, l)_{\mathtt{do}(\mathtt{move}(x,l),s)},$$

where the notation p_s means that the proposition p holds in the situation (or at the time) s, and the expression $\mathtt{do}(a, s)$ refers to the situation (or time) when the action a is completed.[1]

In addition to this action axiom, we need frame axioms indicating the facts that are unaffected when a move action takes place. For example, we need a frame axiom stating that all other objects don't move. This can be expressed as:

$$\mathtt{on}(y, l')_s \wedge y \neq x \rightarrow \mathtt{on}(y, l')_{\mathtt{do}(\mathtt{move}(x,l),s)}.$$

Similarly, we need frame axioms stating that moving does not change the color or shape of objects:

$$\begin{array}{l} \mathtt{color}(x, c)_s \rightarrow \mathtt{color}(x, c)_{\mathtt{do}(\mathtt{move}(u,v),s)}, \\ \mathtt{shape}(x, c)_s \rightarrow \mathtt{shape}(x, c)_{\mathtt{do}(\mathtt{move}(u,v),s)}, \end{array}$$

and so forth.

There are two basic problems with the monotonic approach — an epistemological one, and a computational one. The epistemological difficulty is that we must provide explicit frame axioms for every action and every relation of interest, stating under what circumstances facts involving the relation don't change when the action is performed. In general, if there are a different possible actions, and r different relations, on the order of ar frame axioms are required in order to determine which facts persist from one time to the next.[2] To make matters worse, these axioms may be very complex. Witness the case of the on relation for a move operation: If the object being considered is a piece of paper on a bookcase, the paper will stay when the bookcase is moved, but if the object is a lamp, it will stay if the bookcase is not moved far, but will fall off if the lamp is plugged in and the bookcase is moved beyond the reach of the cord. In short, the monotonic approach is cumbersome at best because the set of frame axioms required is large and complex, and must be constantly augmented as additional relations or actions are added.

The second problem with the monotonic approach is that it is computationally intractable if there are many facts in the database. To determine what is true after some action or event has taken place, the reasoner must examine every fact in its world model, and prove that it still holds. If the database contains n facts and each action only changes one fact, $n - 1$ deductions will have to be performed for each action or event that takes place.

1.2.2 The default approach

The basic problem with the monotonic approach is that we need to state all of the things that don't change explicitly. What we want to do is provide axioms only for the things that change and write a single *default* axiom stating that everything else persists. There are many different nonmonotonic formalisms that could be used for this purpose [25, 21, 22, 24]. The

[1]We are being deliberately vague about whether we are using situation calculus, or a temporal logic such as that of Allen [1]. Although the monotonic approach was originally conceived in the situation calculus, the approach is equally applicable to temporal logics.

[2]Yoav Shoham has pointed out that frame axioms are also needed for compound sentences. We are assuming here that the database consists of only atomic facts and their negations.

basic idea in all of these formalizations is to write an axiom stating that facts persist in the absence of information to the contrary. Using Reiter's default rules [25] we could express this as:

$$\frac{p_s : p_{\mathtt{do}(a,s)}}{p_{\mathtt{do}(a,s)}} \tag{1}$$

which states that if p is true in situation (or time) s and p is still consistent after the action a then we can infer p after the action.[3]

The default approach does not suffer from the epistemological difficulties inherent in the monotonic approach because only a single default rule is required, independent of the number of relations and actions appearing in the domain. However, the default approach still suffers from serious computational problems; to determine what is true about the world after an action has been performed, the default frame axiom must be examined once for every fact of interest.

1.2.3 The STRIPS approach

An entirely different approach to reasoning about change is the STRIPS approach [5]. The basic insight in STRIPS is that the world does not change much from one instant to the next. Thus STRIPS keeps a single model of the world that is updated to reflect the result of any action that takes place. The STRIPS approach describes actions in terms of preconditions, add lists and delete lists. The intention is that a given action can be executed if all of its preconditions are satisfied. The result of the execution is to add the facts in the add list to the world description while removing the facts in the delete list.

While this approach provides an effective solution to the frame problem, it cannot handle inferred consequences, and thus fails to solve the ramification problem. All consequences of actions must be explicitly listed in the add and delete lists, and the system designer is responsible for maintaining satisfactory descriptions of the actions. As the domain becomes more complex, this becomes an increasingly difficult task. Consider our robot/furniture example; if the bookcase is moved from one location to another, where it obscures vents or pictures, all of these consequences must be explicitly provided in the add and delete lists. In addition, the absolute locations of the television, bird and bird cage change. If there were a lamp on top, moving the bookcase would cause the cord to tighten, and the lamp would fall off and break. All of these consequences would have to be specified explicitly in the add and delete lists for the action. This makes the descriptions of actions cumbersome and difficult for complex domains. Furthermore, if new facts are ever added to the database, such as the weight on various locations of the floor, all actions that can potentially affect those facts must be updated. This is also a cumbersome task.

[3]Hanks and McDermott [14] have pointed out a serious technical difficulty with representing default frame axioms using nonmonotonic formalisms. The basic problem is that these formalisms allow unintended extensions in addition to those one would expect. Much recent work in nonmonotonic logic has been aimed at solving this problem [16, 18, 17, 28]. Although the attempts differ considerably in terms of specifics, the basic default approach remains unchanged. Since this issue is not the primary concern of this paper, we will continue to characterize these approaches in terms of the straightforward default rule given above.

These problems with STRIPS are actually a reflection of a more general problem. It is remarked in [7] that the problem of maintaining consistency of a database in the presence of new information is at best semi-decidable. It is therefore not theoretically possible for STRIPS to achieve this consistency maintenance via the simple mechanism of delete lists.

The STRIPS designers were aware of this. They got around the problem by requiring that the knowledge base contain only facts of a very specific nature, essentially guaranteeing through this that their description retains consistency. It is not clear how to maintain the validity of the restrictions on the knowledge base (essentially that it contain only atomic propositions) in more complex domains. Lifschitz [15] has also discussed this issue, and concludes that the approach used in STRIPS is viable when the facts used to describe the world can be guaranteed to be chosen from some predetermined set. Lifschitz calls the sentences in this predetermined set *essential.*

As our example illustrates, this is an increasingly restrictive assumption as the domain becomes more complex. The reason is that our domain information may increasingly be non-atomic (e.g., "all of the books are in the bookcase"), and it will be difficult to specify the form of this information in advance.

1.3 The Possible Worlds Approach

Suppose that we are given an action and a situation. Our approach to reasoning about action is to take the result of the action to be the nearest world to the given one in which the consequences of the action hold. For example, if our robot were considering moving the bookcase from one location to another, the expected result would be the nearest world to the current one in which the bookcase was at its new location. In this world, the bookcase would no longer be at its old location, and everything on or in it would also be at the new location. Furthermore, heating ducts and pictures might be covered in this new world as a result of the new position of the bookcase.

The way we find the nearest world is to construct proofs of the negation of the explicit consequences of the expected action, and remove a premise in each proof from the current world. In the case of the robot, one proof that the bookcase cannot be at its new location is based on the fact that it is at its old location, and that an object cannot be in two places at once. One of these two statements must therefore be removed, and the latter has the status of a *domain constraint* or *protected* law. Another proof that the bookcase cannot be in its new location stems from the fact that the heating duct is not obscured, and the bookcase would obscure it. Therefore the former fact must also go, and so forth. Computationally, this construction procedure appears to be tractable for worlds like our own where very few things tend to change with each action, or where change is regular.

We will refer to the nearest world for an action in a situation as a *possible world*, and will refer to our approach as the possible worlds approach. Formally, this approach is very similar to the default approach; we will show in Section 4 that, given the same set of axioms and domain constraints, the two approaches normally yield the same result. Computationally, however, the approaches differ, in that a clear and effective implementation exists for the possible worlds approach, but not for the default approach.

The computational mechanism used by the possible worlds approach is, in many respects, similar to the STRIPS approach. Like STRIPS, the result of an action is characterized as an update to a single model of the world corresponding to the sentences that are true at the time the action is performed.[4]

However, unlike the STRIPS approach, the possible worlds approach does not require a complete description of the consequences of each action. Instead, the world model is updated inferentially from the explicitly stated consequences of the action, and *domain constraints*. The domain constraints specify the relationships of different things in the world, such as the relationship between one object being on another object, and the locations of the two objects.

1.4 Conventions

Standard first-order predicate calculus will be used throughout this paper to represent sentences about the world. Variables are written in italics; free variables are assumed to be universally quantified. We will also use standard mathematical notation for dealing with sets; the symbol $\subset$ will be used to mean *strict* set inclusion.

For simplicity, we will use the situation calculus throughout the remainder of the paper. As mentioned earlier, the notation p_s will mean that the proposition p holds in the situation s, and the expression $\mathtt{do}(a, s)$ refers to the situation when the action a is completed.

The restriction to situation calculus does not appear to be a fundamental limitation. Historical information or information about the future could still be maintained using a temporal logic such as that of Allen [1]. In such a case, the situation in which an action or event takes place is just considered to be the collection of sentences that hold at the time at which the action or event begins.

1.5 Organization

The following section (Section 2) describes the basic possible world construction. This formalization is also discussed in somewhat greater detail in [7]. Section 3 presents a formalization of actions and of reasoning about actions that naturally incorporates an implicit description of the necessary frame axioms.

The bulk of the paper (Section 4) consists of comparisons of the computational characteristics of our approach with other approaches to reasoning about action. First, in Section 4.1, we examine the space requirements of the various approaches. In particular we consider the relationship between the number of frame axioms required for the monotonic approach, and the number of domain constraints required for the default and possible worlds approaches.

[4] Both the STRIPS approach and the possible worlds approach avoid the Hanks-McDermott problem [14], since actions are described by incremental changes to a database as opposed to being described by formulae that are universally quantified over time or situations.

We also consider the space requirements of an equivalent STRIPS add and delete list representation for those cases where it is possible. In Section 4.2 we go on to consider the time requirements of the different approaches, both for investigating a single proposition and for deriving the entire world state resulting from a sequence of n actions. For the monotonic approach, this involves considering the number of frame axioms that must be examined; for the default and possible worlds approaches, this involves considering the number of domain constraints that must be examined.

2 Possible worlds

2.1 Theoretical construction

We first define a *partial world* to be any consistent collection of sentences describing the domain in question. There will in general be many specific sentences that are logically independent of the domain description; these will be left undetermined in the partial world being considered.

Now, suppose that we have some partial world S describing the current situation, and wish to add the facts in some new set C. The problem is that we cannot simply take the union of the two sets, $S \cup C$, because C may be inconsistent with S.

Instead, we will consider consistent subsets of $S \cup C$ that contain C. In considering these subsets, there will also be other facts in S that we want to preserve. Domain constraints often have this property; consider a blocks world fact such as

$$\mathtt{on}(x, y) \wedge y \neq z \rightarrow \neg \mathtt{on}(x, z), \tag{2}$$

indicating that a block can be in only one place at a time. This constraint can be expected to hold independent of the modifications we might make to our world description. We cater to this formally by supposing that we have identified some set P containing all of those sentences in our language having this "protected" or preferred status.

Definition 2.1 *Given a set S of logical formulae, a set P of protected sentences, and an additional set of formulae C, a* potential world for C in S *is defined to be any subset $T \subseteq S \cup C$ such that:*

1. *$C \subseteq T$. T contains the formulae in C that are explicitly to be added in constructing the new partial world.*
2. *$P \cap S \subseteq T$. Any protected sentence in S is preserved when the partial world is constructed.*[5]
3. *T is consistent.*

[5]It might seem more natural to assume that $P \subseteq S$, so that S already contains all protected formulae, and this is in fact done in [7]. The formulation used here is needed for the treatment of the qualification problem in [11, 12].

Note that the "nearness" of a potential world for C in S to the original world S is reflected by how large the subset is. Thus, if T_1 and T_2 are both potential worlds for C in S, and $T_1 \subseteq T_2$, T_2 is at least as close to S as T_1 is. This leads us to define a *nearest potential world* or *possible world* for C in S as any potential world for C in S that is not a subset of any other potential world for C in S.[6]

Definition 2.2 *Given a set S of logical formulae, a set P of protected sentences, and an additional set of formulae C, a* possible world for C in S *is defined to be any subset $T \subseteq S \cup C$ such that:*

1. $C \subseteq T$,
2. $P \cap S \subseteq T$,
3. *T is consistent,*
4. *T is maximal subject to these constraints.*

This definition closely follows that of [7] and, as mentioned there, is similar to notions developed by Fagin et. al. [4] in considering the semantics of database updates.

We will denote the collection of all possible worlds for C in S by $W(C, S)$.

2.2 An example

In order to illustrate this construction, let us introduce an example similar to that described in Section 1. The example is shown in Figure 1.

The domain in question contains a table, a chest, a plant, a piece of fine art and a bookcase (which itself contains a bird and a television). Ventilation for the room is provided by a pair of ducts under the floor; if both of these are blocked, the room becomes stuffy. Putting an object on the table will obscure the picture.

Formally, the initial situation shown in the figure can be described as follows (the * and # symbols should be ignored for the moment):

	`on(bird,top-shelf)`	`rounded(bird)`	
*#	`on(tv,bottom-shelf)`	`rounded(plant)`	
	`on(chest,floor)`		
*	`on(plant,duct2)`	`duct(duct1)`	
	`on(bookcase,floor)`	`duct(duct2)`	(3)
*	`blocked(duct2)`	`in(bottom-shelf,bookcase)`	
	`¬obscured(picture)`	`in(top-shelf,bookcase)`	
#	`¬stuffy(room)`		

[6] Note that both potential worlds and possible worlds are only partial. The phrase "possible partial world" seems rather unwieldy, however.

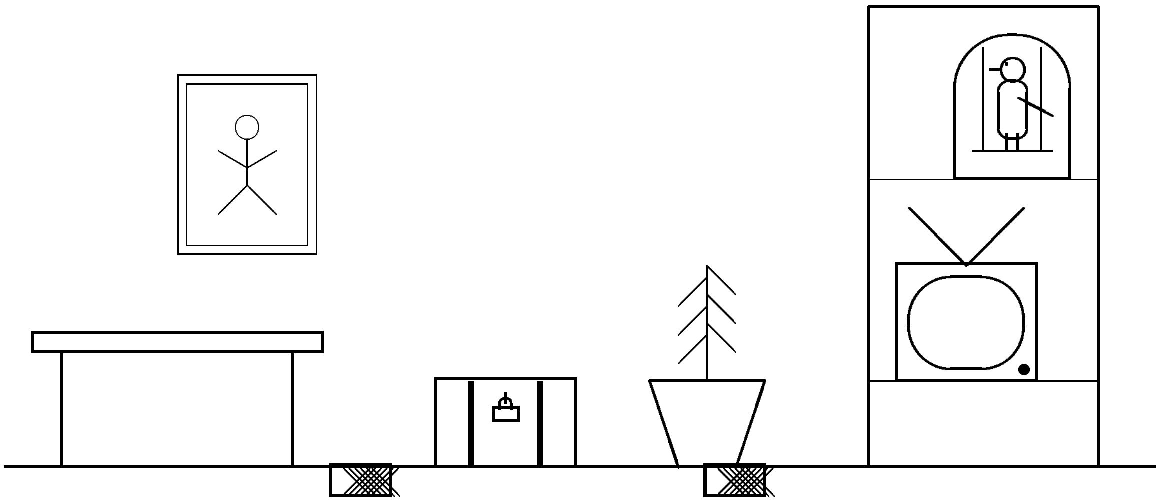

Figure 1: A household domain

There are also the associated domain constraints:

$$\mathtt{on}(x,y) \wedge y \neq z \rightarrow \neg\mathtt{on}(x,z) \tag{4}$$
$$\mathtt{on}(x,y) \wedge z \neq x \wedge y \neq \mathtt{floor} \rightarrow \neg\mathtt{on}(z,y) \tag{5}$$
$$\mathtt{rounded}(x) \rightarrow \neg\mathtt{on}(y,x) \tag{6}$$
$$\mathtt{duct}(d) \wedge \exists x.\mathtt{on}(x,d) \rightarrow \mathtt{blocked}(d) \tag{7}$$
$$\exists x.\mathtt{on}(x,\mathtt{table}) \leftrightarrow \mathtt{obscured(picture)} \tag{8}$$
$$\mathtt{blocked}(\mathtt{duct1}) \wedge \mathtt{blocked}(\mathtt{duct2}) \leftrightarrow \mathtt{stuffy(room)} \tag{9}$$

The first domain constraint indicates that an object can be in only one place at any given time; the second that two different objects cannot be in the same place. The third indicates that no object can be on top of a rounded object; the bird (more accurately, the bird cage) and the plant both fit this description. Domain constraint (7) indicates that anything on a duct blocks it. The final two domain constraints define the conditions under which the portrait will be obscured or the room will be stuffy.

Assuming that we are not prepared to drop the domain constraints appearing at the end of the above description or any of the "structural" facts appearing on the right hand side of (3), there is a unique possible world corresponding to the new fact $\mathtt{on}(\mathtt{tv},\mathtt{chest})$. This possible world corresponds to the removal of the fact indicating that the television is on the bottom shelf of the bookcase. It is necessary to remove this fact since, in light of the domain constraint indicating that an object can be in only one place at any given time, it is inconsistent with the new location of the television.

There is also a unique world corresponding to the television moving to the table. As before, we must remove the fact stating that the television is in the bookcase, but we must also remove the fact that the picture is not obscured. Each of these facts is inconsistent with the new location.

2.3 Automatic generation of possible worlds

One possible implementation of the ideas presented in Section 2.1 involves constructing the possible worlds for a set C by examining proofs of $\neg q$ for the various sentences $q \in C$. Suppose that our theorem prover generates n such proofs $\pi_1, \ldots, \pi_n$, and that these proofs depend upon the (possibly overlapping) sets of unprotected database facts $S_1, \ldots, S_n$ respectively.[7] The intention here is that *all* of the facts in S_i must hold in order for the proof given by π_i to be valid.

To construct a world where the facts in C are consistent, we must remove from our original scenario enough to ensure that each of these proofs of some $\neg q$ now fails. Removing any fact in S_i is enough to invalidate the corresponding proof π_i, so that a subset T of $S \cup C$ with $(P \cap S) \cup C \subseteq T$ will be consistent (in the sense that our theorem prover will not derive inconsistencies from it) if and only if $(S - T) \cap S_i \neq \emptyset$ for each $i \in 1, \ldots, n$. Since non-maximal subsets of $S \cup C$ do not correspond to possible worlds for C in S, we need only select one member from each of the S_i's for deletion from the state description of the possible world, and we have:

Algorithm 2.3 *Let C be as above. Then every possible world for C in S is of the form*

$$S \cup C - \{s_1, \ldots, s_n\} \tag{10}$$

for some choice of $s_1, \ldots, s_n$ with $s_i \in S_i$.

It follows that we can generate all of the possible worlds for C in S by considering each expression of the form (10) and testing it for maximality. Assuming that $|S_1| \cdots |S_n|$ (the product of the cardinalities of the various S_i's) is manageably small, this will provide us with an effective means for enumerating these possible worlds.

For most actions in domains of interest to AI, this seems to be a reasonable assumption.[8] Returning to our usual example, suppose that our set C is

$$\{\texttt{on}(\texttt{tv}, \texttt{table})\}.$$

There are two proofs of $\neg$`on(tv, table)`. One uses the facts that `on(tv, bottom-shelf)` and that the television can only be in one place at a time. The other uses the domain fact $\neg$`obscured(picture)` together with the domain constraint (8). This gives us:

$$\begin{aligned} S_1 &= \{\texttt{on}(\texttt{tv}, \texttt{bottom-shelf})\} \\ S_2 &= \{\neg\texttt{obscured(picture)}\} \end{aligned}$$

[7] In theory, there will be an infinite number of proofs of the various $\neg q$ if there are any, since one can always add irrelevant steps to any proof. The following algorithms can take the S_i to be the minimal subsets of S from which some $\neg q$ follows.

[8] This need not always be the case, however. Consider the action of starting a nuclear war.

Each of these sets is of cardinality 1, and we therefore conclude immediately that the unique possible world for $\texttt{on(tv, table)}$ is as described in Section 2.2.

We will assume throughout this paper that the computational effort involved in constructing possible worlds for C in S from proofs of negations of elements of C is small compared to the effort involved in enumerating the proofs themselves. Essentially, this amounts to the assumption that any particular domain fact appears in only a small number of the proofs being considered.

The possibility of modifying the approach we have presented to deal with a situation where this assumption fails is discussed in [7]. The basic idea presented there is to work through the various proofs of $\neg q$ for $q \in C$ in a "breadth-first" fashion, attempting to construct a single possible world for C in S before beginning the construction of another. This has the advantage that the single world so constructed can then be used to eliminate *potential* worlds constructed later that are included in it; further work on these partial worlds is thus avoided. For problems with a great deal of internal correlation (such as the diagnostic problems discussed in [7] and [26]), this appears to be a more satisfactory approach than a straightforward implementation of the ideas presented in this section, with the result that, once again, the time needed to generate the possible worlds is on a par with that needed to enumerate the proofs of the negations of facts in C.

The approach we have outlined instead works "depth-first" through the proofs of the various $\neg q$, constructing the possible worlds for C in parallel [7]. The effect of this will be to generate many *potential* worlds early in the construction process, and this will allow us to obtain lower bounds on the distance to the nearest world for C in S even before all of the proofs of the $\neg q$ have been examined. It is argued in [9] that this has substantial impact on the control of planning search.

2.4 Multiple worlds

Suppose we return to the example in Figure 1, but instead of moving the television onto the chest or the table, we move it to duct 1. Here, there are *two* possible worlds that result. As usual, we must remove the fact giving the original location of the television from both.

Where the two worlds differ is in their handling of the stated domain fact indicating that the room is not stuffy, which is inconsistent with the plant being located on duct 2. One possibility is to simply remove the fact indicating that the room is not stuffy; the two facts that must be removed are those marked with a # in (3).

The other possibility is to remove the fact that duct 2 is blocked by the plant. In order to do this, we must remove the fact that the plant is located on duct 2, resulting in the possible world where the facts marked with an asterisk have been removed.

More precisely we note that there are essentially three proofs of $\neg\texttt{on(tv, duct1)}$. One uses the fact that the television is in the bookcase. Another uses the facts that the room is not stuffy and duct 2 is blocked. The third uses the facts that the room is not stuffy and the plant is on duct 2 (from which it *follows* that duct 2 is blocked):

$$S_1 = \{\texttt{on(tv, bottom-shelf)}\}$$

$$
\begin{aligned}
S_2 &= \{\neg \texttt{stuffy(room)}, \texttt{blocked(duct2)}\} \\
S_3 &= \{\neg \texttt{stuffy(room)}, \texttt{on(plant, duct2)}\}
\end{aligned}
$$

This leads us to consider the four potential worlds:

$$
\begin{aligned}
W_1 &= S \cup C - \{\texttt{on(tv, bottom-shelf)}, \neg \texttt{stuffy(room)}\} \\
W_2 &= S \cup C - \{\texttt{on(tv, bottom-shelf)}, \neg \texttt{stuffy(room)}, \texttt{on(plant, duct2)}\} \\
W_3 &= S \cup C - \{\texttt{on(tv, bottom-shelf)}, \texttt{blocked(duct2)}, \neg \texttt{stuffy(room)}\} \\
W_4 &= S \cup C - \{\texttt{on(tv, bottom-shelf)}, \texttt{blocked(duct2)}, \texttt{on(plant, duct2)}\}
\end{aligned}
$$

The worlds W_2 and W_3 are non-maximal (W_1 is a superset of each) and need not be considered further.

The two remaining worlds, W_1 and W_4, correspond to legitimate but distinct possibilities in light of our limited knowledge of the world. In W_1, the television blocks the duct and the room becomes stuffy; in W_4, the plant is dislodged from duct 2 (presumably because of the power of the ventilation system), and air continues to circulate.

In this particular example, it seems reasonable to assume that moving the television to a position where it blocks duct 1 will cause the room to become stuffy (W_1) rather than one where the plant has moved (W_4). But this is a consequence of facts not recorded in the database: that the plant is heavy, for example. If it were only a piece of paper blocking the second duct, we might well prefer a result where the paper had moved to one where the room became stuffy.

Our formal construction is unable to select between these two possible worlds because neither of them is a subset of the other. This is to be expected: the information in our database is insufficient to determine, in and of itself, which of the two possible worlds will be the consequence of moving the television to duct 1. Resolving the ambiguity will require the introduction of additional information about the television, plant and ventilation system.

There are several ways we might capture this; unfortunately, none of the approaches we are about to suggest is completely satisfactory at this point, and the remainder of this section should be viewed as preliminary.

One approach is to restrict the possible worlds considered as results of the action by incorporating domain-dependent information in the possible worlds construction. We do this by thinking of the subsets of $S \cup C$ as ordered not by set inclusion, but by some other partial order $\leq$ that extends it[9] (so that $T_1 \subseteq T_2 \Rightarrow T_1 \leq T_2$). We now repeat the definition exactly, defining possible worlds to correspond to subsets that are maximal not merely in a set-theoretic sense, but that are maximal with respect to $\leq$ as well.

One possible partial order involves prioritizing the facts in the database in some way, perhaps preferring the removal of facts involving certain predicates over the removal of facts involving others. For example, we might prefer removing facts involving the relation `stuffy` over removing facts involving `on`.

[9] In [7], only partial orders on subsets of S were considered. The advantage of the present scheme is that it allows the partial order to depend upon the set C.

The simplest formalization of this requires that we identify some subset T of our initial world description that contains the preferred facts in S. We now define a partial order on S by saying that $U < V$ if either $U \subset V$ or $U \cap T \subset V \cap T$, so that we attempt not to remove preferred facts from our database. (This partial order is clearly an extension of set inclusion.)

In our living room example, we might take T to be the collection of facts of the form $\mathtt{on}(x, y)$. When considering moving the television to duct 1, this leads us to select W_1 in our example, since this world reverses fewer `on` facts than W_4 does. (W_1 is the world in which the room becomes stuffy.) Unfortunately, it is not clear how to establish a partial order on the facts in a database, or even that a single partial order will suffice for a wide range of actions.

A second approach to the original ambiguity involves the incorporation of reason maintenance information [2, 3] into our database, and to then remove any sentence that becomes unsupported when a new possible world is constructed.

As an example, we could record the fact that the room's being not stuffy is a consequence of duct 1's being unblocked. Since the result of moving the television to duct 1 is to block it, $\neg$`stuffy(room)` will become unsupported and will therefore be removed from the database independent of the status of the fact `on(plant,duct2)`. The new database (where the stuffiness of the room is undetermined, but the plant stays where it was) is consistent, and we once again conclude that the room becomes stuffy (since both ducts are blocked).

The advantage of this approach is that it does not depend on the use of domain dependent information (such as an identification of the set of preferred sentences). The disadvantages are that it depends upon more powerful theoretical mechanisms, including some formal understanding of the reason maintenance process. Although progress is being made in this area [8, 27], difficult problems remain. It also appears that the simple approach proposed here may remove more facts than necessary in more complex examples.

A final approach resolves the ambiguity between W_1 and W_4 by appealing to a more detailed description of the domain under consideration. In the example we are considering, we might include precise information about the workings of the ventilation system, the weight of the plant, and basic physical knowledge about air pressure, forces, and so on.

Unfortunately, this sort of a redescription may not always be available, and the computational impact of such an ontological shift is likely to be substantial. This idea is also discussed in [7].

3 Formalizing actions

Suppose that we are given some initial world description S, together with some collection A of actions. Each action $a \in A$ will be defined using a precondition $p(a)$ and a consequence set $C(a)$. Informally, the precondition must hold if the action is to be possible, and the result of the action is to add the facts in $C(a)$ to the world description. We will assume throughout this paper that the precondition $p(a)$ is complete, in the sense that the action can definitely be performed if the precondition is satisfied.[10] The consequence of the action, however, will *not* be assumed complete, since there may be many ramifications of an action in any particular situation. Not all of these need be explicitly included in the consequence set.

Given a world description and an action, we would like to define the result of the action to be the world obtained by adding to the existing description the consequences of the action. The difficulty, of course, is that this world may be inconsistent for a variety of reasons, and we therefore take the result of the action to be the possible world corresponding to its consequences.

Specifically, suppose we are given some world description S and some action a having consequences $C(a)$, and $W(C(a), S)$ is nonempty. We define the *result* of a in S, to be denoted $r(a, S)$, as the intersection of all of the possible worlds for $C(a)$ in S. We adopt this conservative definition in order to ensure that any sentence in $r(a, S)$ is valid in *all* of the possible worlds for $C(a)$ in S.[11] If $W(C(a), S) = \emptyset$, we take $r(a, S) = S$. (We include this for completeness only; an action whose consequences have *no* possible world is effectively impossible.)

It is straightforward to extend this definition to an action *sequence*; if $a_1, \ldots, a_n$ is a sequence of actions, we can define a corresponding sequence of situations recursively by:

$$s_i = \begin{cases} S, & \text{if } i = 0, \\ r(a_i, s_{i-1}), & \text{if } 0 < i \leq n. \end{cases} \tag{11}$$

We will say that a sequence of actions is *viable* if $s_{i-1} \models p(a_i)$ for all $i \in 1, \ldots, n$.

As an example, consider once again the scenario of Figure 1. In this domain, there is a single action $\texttt{move}(x, y)$, which moves the object x to the location y. The preconditions for the action are that x and y both be clear (or that x is clear and y is the floor), and that y not be rounded, and the consequence is that x is on y:

$$\begin{aligned} p(\texttt{move}(x,y)) &= \texttt{clear}(x) \wedge [\texttt{clear}(y) \vee y = \texttt{floor}] \wedge \neg\texttt{rounded}(y) \\ C(\texttt{move}(x,y)) &= \{\texttt{on}(x,y)\}. \end{aligned}$$

Consider now the action sequence $\langle \texttt{move}(\texttt{tv}, \texttt{duct1}), \texttt{move}(\texttt{chest}, \texttt{table}) \rangle$ where we move the television to duct 1 and then move the chest to the table.

We have already discussed the first of these actions in some detail; certainly this action is possible in the initial situation. In the absence of domain-specific information as discussed in Section 2.4, the result of this action will be to remove from the domain description all of the facts that are marked *either* with a * or with a # in the domain description.

Even in the face of our inability to select between the possible worlds describing the consequences of the first action, the precondition to the second remains provable. We therefore conclude that the entire action sequence is viable.

---

[10] Generally, of course, actions have a variety of preconditions. All of these can be conjoined to obtain the single formula $p(a)$.

[11] The converse need not hold, since a fact may follow from a disjunction of the facts characterizing the individual worlds. This can be handled by saying that some sentence p is valid after performing an action a whenever p is counterfactually implied by the consequences of the action [7].

This would not have been the case had the first action been to move the television to the chest. Now, `clear(chest)` fails in the (unique) possible world obtained by adjoining `on(tv, chest)` to our initial world description, so that the precondition to the second action fails.

4 Comparison with other approaches

In Section 1.2 we gave brief descriptions of the monotonic approach, the default approach, and the STRIPS approach to reasoning about action. Our objective in this section is to provide a more rigorous comparison of the computational properties of these approaches with those of the possible worlds approach.

We will assume we are given some initial state and some sequence of actions, and are interested in the complexity of 1) determining whether or not some specific formula holds after the actions are executed (for example, the precondition to a hypothetical next action), and 2) in constructing the entire final state reflecting the execution of the action sequence. The symbol α will be used to refer to the number of action types in the domain, and r will be used to refer to the number of relation symbols appearing in the database. For simplicity, we will assume that the database contains only atomic facts and their negations. (Note that neither the default approach nor the possible worlds approach requires this assumption.)

4.1 Space requirements

In order to properly compare the efficiencies of the different approaches, we need to know the relationship between the sets of axioms required for each. We therefore start by considering the relationship between the number of frame axioms required for the monotonic approach and the number of domain constraints required for the default and possible worlds approaches.

4.1.1 Frame axioms

In general, for each relation symbol in our domain and each possible action, we will need one frame axiom, indicating the conditions under which that relation is preserved when the action takes place. We will also generally need a second rule indicating the conditions under which the negation of the relation is preserved when the action takes place. Thus for the `on` relation and `move` action we would need the two frame axioms:

$$\begin{aligned} \mathtt{on}(y,l')_s \wedge y \neq x &\rightarrow \mathtt{on}(y,l')_{\mathtt{do}(\mathtt{move}(x,l),s)} \\ \neg\mathtt{on}(y,l')_s \wedge l' \neq l &\rightarrow \neg\mathtt{on}(y,l')_{\mathtt{do}(\mathtt{move}(x,l),s)} \end{aligned}$$

In general, if there are α different possible actions, and r different relations, $2\alpha r$ frame axioms are required in order to determine which facts persist from one time to the next.

It might seem that some of these $2\alpha r$ frame axioms may be unnecessary in certain domains. For example, if we know that an object's being blue persists through a move action, why should we need to know that its being non-red also persists? Of course, the fact that an object can only be one color at a time is a domain constraint; the possibility of replacing a set of frame axioms with a smaller set of domain constraints is the subject of the following section.

4.1.2 Domain Constraints

In order to investigate the relationship between frame axioms and domain constraints, we will take a collection of frame axioms and derive domain constraints from them. The form of the derived domain constraints will enable us to make the comparison.

Suppose, then, that we have some frame axiom stating that if some condition π_+ holds, then q will persist through an action a. In other words, if q holds in s and the preconditions to a hold in s, then $\pi_+ \rightarrow q_{\mathbf{do}(a,s)}$:

$$q_s \wedge p(a)_s \rightarrow [\pi_+ \rightarrow q_{\mathbf{do}(a,s)}].$$

We assume in addition that π_+ is a minimal persistence requirement, so that q persists *only* if π_+ holds. This allows us to strengthen the above expression to:

$$q_s \wedge p(a)_s \rightarrow [\pi_+ \leftrightarrow q_{\mathbf{do}(a,s)}]. \tag{12}$$

Related to the persistence of q is the persistence of $\neg q$, and we assume that π_- is a persistence condition for it (we only need to examine persistence conditions for q and $\neg q$ because of our assumption that the database consists entirely of atomic facts and negations of atomic facts):

$$\neg q_s \wedge p(a)_s \wedge \pi_- \rightarrow \neg q_{\mathbf{do}(a,s)}. \tag{13}$$

Note that we do not need to require π_- to be minimal.

We will call any π_+ satisfying (12) a *positive persistence condition* for q and a, and any π_- satisfying (13) a *negative persistence condition* for q and a.[12]

If we denote by $c(a)$ an arbitrary element of the consequence set $C(a)$, we now have the following:

Theorem 4.1 *Let π_+ and π_- be positive and negative persistence conditions for q and a, and set $\lambda = \neg\pi_+ \wedge \pi_-$. Then if $p(a)$ is a negative persistence condition for λ,*

$$p(a)_s \rightarrow [\lambda \wedge c(a) \rightarrow \neg q]_{\mathbf{do}(a,s)}. \tag{14}$$

Proof. We first show that

$$p(a)_s \wedge \lambda_s \rightarrow [c(a) \wedge \neg q]_{\mathbf{do}(a,s)}. \tag{15}$$

[12]Because of the bidirectional implication appearing in (12), it follows that a positive persistence condition for q is a negative persistence condition for $\neg q$, but not necessarily vice versa. In actuality, it is only the reverse arrow in (12) that we need, and Theorem 4.1 could be strengthened to reflect this.

It is clear that

$$p(a)_s \rightarrow c(a)_{\mathtt{do}(a,s)}.$$

With regard to q, we note that if $\neg q$ holds before the action a, it will hold afterwards by virtue of λ's implying the negative persistence condition π_-. Alternatively, if q holds before the action, $\neg q$ will once again hold afterwards due to the failure of the positive persistence condition π_+, together with (12).

We rewrite (15) as

$$p(a)_s \rightarrow [\lambda_s \rightarrow (c(a) \wedge \neg q)_{\mathtt{do}(a,s)}]. \tag{16}$$

If $p(a)$ is a negative persistence condition for λ, we have from (13) that

$$\neg\lambda_s \wedge p(a)_s \rightarrow \neg\lambda_{\mathtt{do}(a,s)},$$

or

$$p(a)_s \rightarrow [\lambda_{\mathtt{do}(a,s)} \rightarrow \lambda_s].$$

Combining this with (16) gives us

$$p(a)_s \rightarrow \{[\lambda_{\mathtt{do}(a,s)} \rightarrow \lambda_s] \wedge [\lambda_s \rightarrow (c(a) \wedge \neg q)_{\mathtt{do}(a,s)}]\},$$

so that

$$p(a)_s \rightarrow [\lambda \rightarrow (c(a) \wedge \neg q)]_{\mathtt{do}(a,s)}, \tag{17}$$

or

$$p(a)_s \rightarrow [\lambda \wedge c(a) \rightarrow \neg q]_{\mathtt{do}(a,s)}. \qquad \square \tag{18}$$

Given this result, it seems appropriate to treat

$$\lambda \wedge c(a) \rightarrow \neg q$$

as a domain constraint for the domain in question, since this constraint holds in any state that can be reached by performing the action a.[13]

As an example, consider a simple blocks world situation where $q = \mathtt{on}(x, y)$ is the domain fact, and $a = \mathtt{move}(u, v)$ is the action. The frame axiom is given by:

$$\mathtt{on}(x,y)_s \wedge u \neq x \rightarrow \mathtt{on}(x,y)_{\mathtt{do}(\mathtt{move}(u,v),s)},$$

so the positive persistence condition π_+ is $u \neq x$.

A negative persistence condition must satisfy

$$\neg\mathtt{on}(x,y)_s \wedge \pi_- \rightarrow \neg\mathtt{on}(x,y)_{\mathtt{do}(\mathtt{move}(u,v),s)}.$$

Since

$$\neg\mathtt{on}(x,y)_s \wedge v \neq y \rightarrow \neg\mathtt{on}(x,y)_{\mathtt{do}(\mathtt{move}(u,v),s)},$$

[13]It was in order to treat the result in this fashion that we moved $c(a)$ to the premise of the implication, taking the result of the theorem to be (18) rather than the stronger (17).

we take $\pi_- = v \neq y$. Setting $\lambda = (v \neq y \wedge u = x)$, the calculated domain constraint is:

$$v \neq y \wedge u = x \wedge \mathtt{on}(u,v) \rightarrow \neg\mathtt{on}(x,y),$$

which we rewrite as:

$$v \neq y \wedge \mathtt{on}(x,v) \rightarrow \neg\mathtt{on}(x,y).$$

This is precisely the usual constraint indicating that a block can be in only one place at a time.

If our persistence conditions are such that some quality *always* persists through a certain action type (as with `color` and `shape` for the `move` operation), an examination of (12) makes it clear that we can take π_+ to be $\pi_+ = t$ (truth). Then we obtain $\lambda = f$, and the associated domain constraint, $\lambda \wedge c(a) \rightarrow \neg q$, will be vacuous.

Here is another example, where λ is in fact state-dependent: Again in a blocks world domain, suppose that if we move a blue block onto another blue block, the block being moved changes color. Thus, blueness persists if and only if either the block being considered is not the one being moved, or if the target block is itself non-blue:

$$\mathtt{blue}(x)_s \rightarrow [\mathtt{blue}(x)_{\mathtt{do}(\mathtt{move}(u,v),s)} \leftrightarrow x \neq u \vee \neg\mathtt{blue}(v)].$$

We see from this that we can take the positive persistence condition π_+ to be $x \neq u \vee \neg\mathtt{blue}(v)$. Since non-blueness persists through any motion, we take π_- to be t, so that $\lambda = [x = u \wedge \mathtt{blue}(v)]$.

To see that $\neg\lambda$ persists, we must check that

$$x \neq u \vee \neg\mathtt{blue}(v)_s \rightarrow x \neq u \vee \neg\mathtt{blue}(v)_{\mathtt{do}(\mathtt{move}(u,v),s)},$$

or that

$$\neg\mathtt{blue}(v)_s \rightarrow \neg\mathtt{blue}(v)_{\mathtt{do}(\mathtt{move}(u,v),s)}.$$

Since non-blueness always persists, we conclude that

$$x = u \wedge \mathtt{blue}(v) \wedge \mathtt{on}(u,v) \rightarrow \neg\mathtt{blue}(x)$$

holds after moving u to v. The associated domain constraint is:

$$\mathtt{blue}(v) \wedge \mathtt{on}(u,v) \rightarrow \neg\mathtt{blue}(u), \tag{19}$$

indicating that one blue block cannot be on top of another.

This example is one in which the domain constraint posited as a result of an expression such as that in (14) may *not* be valid for the domain in question. We might, for example, be working in a domain where one blue block could be (initially) on top of another, but we are incapable of *moving* a blue block onto another.

This can be captured by adding to the consequences of the move operator a marker of the form $\mathtt{moved}(u,v)$, indicating that u has been moved to v. Having done this, (19) becomes:

$$\mathtt{blue}(v) \wedge \mathtt{moved}(u,v) \wedge \mathtt{on}(u,v) \rightarrow \neg\mathtt{blue}(u),$$

capturing exactly the fact that blue blocks cannot be moved to other blue blocks. In general, similar dummy consequences can be introduced to enable domain constraints to adequately describe any set of positive and negative persistence conditions. This guarantees the epistemological adequacy of using domain constraints.

We will call a domain *persistence-constrained* if all of the domain constraints can be obtained from the frame axioms without having to insert an additional clause like $\texttt{moved}(u, v)$. In general, if there are $2\alpha r$ frame axioms these will lead to at most αr domain constraints. However, this is far too conservative for a persistence-constrained domain with many relation and action types. In this case we can expect many relations to persist through many of the actions, and the result will be that the number of domain constraints will frequently be much smaller than the number of frame axioms. It also seems likely that many of the frame axioms in large domains will generate identical domain constraints, further decreasing the number of domain constraints relative to the number of frame axioms.

Theorem 4.2 *In a persistence-constrained domain having $2\alpha r$ frame axioms the number of domain constraints is $\kappa\alpha r$, where κ is generally much less than one for large domains.* □

4.1.3 Explicit consequences

In Section 1.2 we remarked on the fact that the STRIPS, or *explicit consequences* approach is not epistemologically adequate for general reasoning about action where non-atomic facts are allowed. However, this is not a concern for many applications. Assuming that we can represent all of the effects of every action explicitly using STRIPS add and delete lists, what is the complexity of the resulting descriptions for a domain having $\kappa\alpha r$ domain constraints?

To derive this we consider a single action that, in the possible worlds formulation, adds a single consequence c to the database. On the average, there will be $\kappa\alpha$ domain constraints involving the relation symbol appearing in c, and the validity of any subset of these may require the deletion of some domain fact when the action is executed.

In order to handle the fact that the delete list for some action may vary depending upon the circumstances in which the action is executed, we will need to split the original action into a variety of special cases, specifying preconditions, add lists and delete lists for each. Since there are potentially 2^κ distinct action subtypes, we get

Theorem 4.3 *The space required to describe a domain containing $\kappa\alpha r$ domain constraints using explicit consequences may grow as rapidly as $\alpha 2^{\kappa\alpha}$.* □

As an example of this, consider the `move` operator in the example we have been considering. STRIPS will certainly need to distinguish among situations where the target of the motion is the table (blocking the picture), where it is a duct (potentially making the room stuffy) and where it is a location of another type.

Given that the target of the motion is one of the two ducts, STRIPS must also distinguish between situations where the other duct is blocked (so that the room does indeed become stuffy) and where the other duct is not blocked. But even this is insufficient, since moving an object from one duct to the other will not make the room stuffy. Additionally, we need to consider the possibility that the location we are moving *from* is either the table (making the picture visible) or a duct (clearing the room).

The following table summarizes the possibilities. We have described the possible motions in terms of their starting and ending locations, with "other" indicating a location which is not the table or a duct. Motions to a duct also indicate whether the other duct is clear or blocked. We also indicate which of the four potential database facts needs to be deleted (in addition, of course, to deleting the fact giving the object's initial location).

	Deleted fact(s)			
Motion type	obscured	¬obscured	stuffy	¬stuffy
other–other				
other–table		x		
other–duct (clear)				
other–duct (blocked)				x
table–other	x			
table–duct (clear)	x			
table–duct (blocked)	x			x
duct–other			x	
duct–table		x	x	
duct–duct				

Instead of a single move operator, we have eleven. Since there are only seven distinct possibilities for the delete lists, it is possible to reduce this to a set of seven move operators; the preconditions to each will be fairly complex, as they will need to determine which of the eleven cases above applies for any particular motion.

The example with which we have been working is fairly simple, but seems typical of the exponential difficulties that will be suffered by any approach that must explicitly list all consequences of all actions.

4.2 Time

Using the information about space requirements derived in the previous sections, we can now consider the time complexity of the various approaches.

4.2.1 The monotonic approach

Let us suppose that in order to investigate some proposition q after performing an action, we need to examine an average of l domain facts to determine whether or not they held before the action was executed. Determining the color of a block after moving it, for example, involves simply discovering its color before the motion, so that $l = 1$ in this case.

To investigate q after a sequence of n actions, we will therefore need to examine l database facts after $n-1$ actions, l^2 after $n-2$ actions, and so on.

There will, of course, be some overlap among the many facts investigated at early times, and it will be effective to cache these facts as we consider them. In general, if we suppose that we are working in a domain containing d facts in the state description, a random collection of x of these facts (possibly with repetition) will involve, on average,

$$d\left[1-\left(1-\frac{1}{d}\right)^x\right]$$

distinct domain facts. (A proof of this can be found in the appendix.) The time spent investigating q will be spent investigating each of these domain facts at various times; if t is the time required to investigate each of them, the total time required will be:

$$\sum_{i=1}^{n} t\left\{d\left[1-\left(1-\frac{1}{d}\right)^{(l^i)}\right]\right\}.$$

A Taylor expansion of this result leads to:

Lemma 4.4 *The time taken to investigate the query q after a sequence of n actions in the monotonic approach is approximately given by:*

$$t(q)=\begin{cases} td(n-1), & \text{if } l^n \gg d,\\ \frac{tl(l^n-1)}{l-1}, & \text{if } l^n \ll d.\end{cases} \tag{20}$$

If we are constructing the entire final state, we should clearly cache all of the intermediate results. This gives us the following:

Theorem 4.5 *In a domain described by a large number d of domain facts, the time needed to investigate the validity of a single proposition q after a sequence of n actions is*

$$\frac{tl(l^n-1)}{l-1},$$

where t and l are as described above. The time needed to construct the entire state resulting from the action sequence is

$$ntd. \quad \square$$

4.2.2 The possible worlds approach

The possible worlds approach is somewhat simpler to analyze. Suppose that the result of an action is to add x new facts $q_1, \ldots, q_x$ to the database, that r_i is the relation symbol appearing in q_i, and that there are $c(r_i)$ domain constraints involving r_i. The possible worlds construction requires that we investigate the $c(r_i)$ domain constraints for each consequence q_i to determine whether or not they lead to proofs of $\neg q_i$. If it takes time t' to examine each domain constraint, the total time spent will therefore be approximately $\sum_1^x t'c(r_i)$.[14] Assuming that the constraints are distributed uniformly among the relation symbols, the time spent constructing the state corresponding to the consequences of the action will be $xt'c/r$, where c is the number of constraints on the domain in its entirety, and r is the number of relation symbols.

Lemma 4.6 *The time spent analyzing a sequence of n actions using the possible worlds approach is given by*

$$\frac{nxt'c}{r}. \qquad \square$$

Note that this result is independent of whether we are interested in only a single proposition q, or are interested in constructing the entire state resulting from the action sequence. This is because there is a basic computational difference between the monotonic approach and the possible worlds approach. The possible worlds approach proceeds by constructing the final state *in its entirety*, while the monotonic approach work by considering the validity of specific formulae in the final state. (We will see in Section 4.2.3 that similar remarks hold for the default approach.)

We will assume that the time t' needed to investigate each domain constraint is comparable to the sum of the time t needed to examine a persistence condition such as π_+ and the time t_c needed to examine a consequence $c(a)$ of the (now hypothetical) action a. Using this, together with the result from Theorem 4.2 that $c = \kappa\alpha r$ we get:

Theorem 4.7 *In a persistence-constrained domain, the time spent analyzing a sequence of n actions using the possible worlds approach is given by*

$$\kappa n x(t + t_c)\alpha. \qquad \square \tag{21}$$

The relative magnitudes of t and t_c may vary depending upon the fashion in which the domain is being described. In most monotonic approaches, for example, t will be fairly small, since little inference needs to be done in order to determine if some specific domain fact or persistence condition holds. In the absence of substantial interaction or "chaining" between the derived domain constraints, we can expect t_c to be comparably small, but this need not always be the case.

4.2.3 The default approach

In [25], Reiter gives a proof procedure for default logics. Defining `PREREQUISITES` and `CONSEQUENCES` in the obvious fashion for a collection of default rules and denoting by W the facts known to hold in the default theory, he notes that a formula β will be a member of *some* extension of the default theory if we can find

[14]Note that if several domain constraints appear in some proof, t' will become larger. This has an analog in the monotonic approach, where it may be necessary to derive information about some domain fact instead of simply looking for the fact in the database. This would be reflected in an increase of the t appearing in Theorem 4.5.

... a subset D_0 of the defaults such that $W \cup \texttt{CONSEQUENCES}(D_0) \vdash \beta$. For $i \geq 1$, if D_{i-1} has been determined, then determine a subset D_i of the defaults such that $W \cup \texttt{CONSEQUENCES}(D_i) \vdash \texttt{PREREQUISITES}(D_{i-1})$. If, for some k, $W \vdash \texttt{PREREQUISITES}(D_{k-1})$ and if $W \cup_{i=0}^{k-1} \texttt{CONSEQUENCES}(D_i)$ is satisfiable, then a proof of β has been found.

Even in the best case where $k = 1$, we will have to investigate the satisfiability of $W \cup \texttt{CONSEQUENCES}(D_0)$. Since β is entailed by D_0, this can be no easier than investigating the satisfiability of $W \cup \{\beta\}$.

Any practical implementation will assume W itself to be consistent, so that the satisfiability of $W \cup \{\beta\}$ can be determined simply by attempting to prove $\neg\beta$ from W. This will involve investigating each of the c/r domain constraints; if the time needed to investigate each is comparable to the time t' needed to investigate a single domain constraint in the possible worlds approach, we conclude that it will take time $\frac{t'c}{r}$ to examine all of them. Substituting $t' = t + t_c$ and $c = \kappa\alpha r$, we obtain:

Theorem 4.8 *The time spent analyzing a sequence of n actions using the default approach is as given by Theorem 4.5, with t replaced with*

$$(t + t_c)\kappa\alpha. \qquad \square$$

We should also remark at this point that there are instances where the default and possible worlds approaches will generate different situations corresponding to the results of some specific action. In order to make this point precise, we need the following result:

Theorem 4.9 *Let S be some collection of sentences, and P the protected sentences in our language. Now consider a default theory in which the default rules are given by*

$$\frac{: w}{w}$$

for each $w \in S - P$, so that w holds in the absence of information proving it false. The facts in the default theory consist of q, together with the sentences in P.

The possible worlds for q in S are in natural correspondence with the extensions of the above default theory.

A proof of this result can be found in [7].

This correspondence is also discussed in [26]; Reiter goes on to show how it is possible to capture some sorts of domain-dependent information in this framework, although it does not appear that possible worlds generated using arbitrary partial orders will always correspond naturally to results obtained using some default theory.

It follows from Theorem 4.9 that if there is a *unique* possible world corresponding to the consequences of an action, a statement will be true in that possible world if and only if it follows from the default rule given by (1). But if there are multiple extensions or possible worlds (corresponding perhaps to an action such as `move(tv,duct1)` whose consequences are only partially known), the proof procedure given by Reiter will accept any statement that is true in *any* of these possible worlds, while the construction we have given, based as it is upon the intersection of the possible worlds, will require a potential plan to achieve its goal for *any* of the possible outcomes of the action or actions in question. In realistic planning situations, this seems likely to be the more effective approach, since it requires the planning agent to consider all possible outcomes of his actions, as opposed to hoping optimistically that they will result in his achieving his goal. Unfortunately, Reiter gives no method for determining whether or not a formula holds in every extension of a default theory.

4.2.4 STRIPS

Because STRIPS modifies its database directly using add and delete lists, the principal computational cost involved in this approach is simply that of finding which of the $\alpha 2^{\kappa\alpha}$ possible add and delete lists should be used when a particular action is executed. Locating any particular add and delete list will take time at best logarithmic in the number of possibilities, so we can expect the time taken by the STRIPS approach to be on the order of $t\kappa\alpha$ (the t appears as an approximation to the time needed to determine if some particular add-delete pair is appropriate in any given situation). Analyzing a sequence of n actions can therefore be expected to take time $nt\kappa\alpha$.

4.3 Comparisons

We can summarize the results of the previous sections in the following:

Theorem 4.10 *In a persistence-constrained domain with α action types, r relations, and a large number d of domain facts, the space and time required by the various approaches is:*

	possible worlds	monotonic	default	STRIPS
space	$\kappa\alpha r$	$2\alpha r$	$\kappa\alpha r$	$\alpha 2^{\kappa\alpha}$
time				
single query	$nx(t+t_c)\kappa\alpha$	$\frac{tl(l^n-1)}{l-1}$	$\frac{(t+t_c)\kappa\alpha l(l^n-1)}{l-1}$	$nt\kappa\alpha$
entire state	$nx(t+t_c)\kappa\alpha$	ntd	$n(t+t_c)\kappa\alpha d$	$nt\kappa\alpha$

One can expect x, t, t_c, l and α to all be approximately independent of the size of the domain being investigated. If we assume additionally that $d \gg x$ (few facts change with each action), and that κ is sufficiently small that $d \gg \kappa\alpha x$ as well, we can conclude that the possible worlds approach will be more efficient than the monotonic approach when either:

1. $l > 1$ and n is large, or
2. The entire final state needs to be constructed.

The first condition corresponds to a case where a great many facts need to be investigated at early times in order to determine the validity of even a single one at a later time.

If $x = 1$, so that each action has a single consequence, some simple algebraic manipulation shows that the possible worlds approach is uniformly more efficient than the default one. Informally, the reason for this is that both the default approach and the possible worlds approach involve a complete search through the proof tree for some proposition. In the default case, the proposition being considered is the negation of the consequence of the default rule; a complete search is necessary because the default rule is generally valid. In the possible worlds approach, we must explicitly examine all proofs of the negations of the consequences of the action being considered. These tasks are of comparable computational complexity; the difference between the approaches is that the possible worlds approach requires only x such proof attempts, while the default approach will in general require many.

Finally, we see that the STRIPS approach is exponentially expensive in terms of the space used to describe a particular domain; although this approach needs less time than the others to evaluate the results of any particular action, the times needed by the STRIPS and possible worlds approaches are roughly comparable. (This is true even though the activities performed by the two approaches are completely different: The possible worlds approach is inferentially investigating domain constraints, while the STRIPS approach is searching through an extensive collection of add and delete lists.)

5 Conclusion

In this paper we have described an approach to reasoning about action using possible worlds. The approach involves keeping a single model of the world that is updated when actions are performed. The update procedure involves constructing the nearest world to the current one in which the consequences of the actions under consideration hold. The way we find the nearest world is to construct proofs of the negation of the explicit consequences of the expected action, and remove a premise in each proof from the current world. Computationally, this construction procedure appears to be tractable for worlds like our own where very few things tend to change with each action, or where change is regular.

The approach developed here is in some sense a formalized extension of the STRIPS approach; in both cases, a single model of the world is maintained, which is updated to reflect the result of any particular action. The difference is that in STRIPS, consistency is maintained after the execution of an action via the delete lists; in our approach, it is maintained inferentially using the constraints describing the domain in question. In some sense, the STRIPS descriptions of the actions "compile" the interactions between these actions and the domain constraints into the delete list.

In this paper we have focussed specifically on the frame and ramification problems, but have ignored the qualification problem. As mentioned earlier, we have applied the possible worlds construction described here to the qualification problem as well. In these cases one is forced to decide among different possible worlds: worlds in which the action is prevented by hidden qualifications, and worlds in which the consequences of the action overcome the qualifications. This work is described in detail in [11, 12].

We have also applied the possible worlds approach described here to the construction of a working planning system. For planning problems we construct the closest possible world to the initial state for which the desired goal holds. The procedure for constructing this state is the same as that described in Section 3. The planning process is guided by seeking those actions that will lead to the nearest possible world from the initial one where a goal or subgoal holds. This planning system and the control procedure based on nearness of possible worlds are described in detail in [9, 10].

Appendix

Proposition A.1. *Suppose we have an urn containing d distinct items. Then if we make x selections from the urn, replacing each item selected, the number of different items selected will be, on average,*

$$d\left[1-\left(1-\frac{1}{d}\right)^x\right].$$

Proof. Suppose that we fix d, and denote by $p(n,x)$ the probability of drawing n different items in x drawings, and by $f(x)$ the expected number of items selected. We are therefore trying to show that

$$f(x)=d\left[1-\left(1-\frac{1}{d}\right)^x\right].$$

If we are to select n items in $x+1$ drawings, then we must either select $n-1$ items in the first x drawings, followed by selecting a new item in the last drawing, or select n items in the first x drawings, followed by selecting an old item in the last one. Thus

$$p(n,x+1)=p(n,x)\frac{n}{d}+p(n-1,x)\left[1-\frac{n-1}{d}\right].$$

We also have that

$$f(x)=\sum_1^d ip(i,x),$$

so that

$$\begin{aligned}
f(x+1) &= \sum_1^d ip(i,x+1)\\
&= \sum_1^d ip(i,x)\frac{i}{d}+\sum_1^d ip(i-1,x)\left(1-\frac{i-1}{d}\right)\\
&= \sum_1^d p(i,x)\frac{i^2}{d}+\sum_0^{d-1} jp(j,x)\left(1-\frac{j}{d}\right)+\sum_0^{d-1} p(j,x)\left(1-\frac{j}{d}\right)\\
&= dp(d,x)+\sum_0^d jp(j,x)-dp(d,x)+\sum_0^d p(j,x)-\frac{1}{d}\sum_0^d jp(j,x)
\end{aligned}$$

$$= \left(1-\frac{1}{d}\right) f(x)+1.$$

We also know that $f(0)=0$.

For $x=0$, the expression in the statement of the theorem does indeed reduce to 0; we must therefore show that it satisfies the recursion relation appearing above. But we have:

$$\begin{aligned}
f(x+1) &= d\left[1-\left(1-\frac{1}{d}\right)^{x+1}\right] \\
&= d\left[1-\left(1-\frac{1}{d}\right)^{x}\left(1-\frac{1}{d}\right)\right] \\
&= d-d\left(1-\frac{1}{d}\right)^{x}+\left(1-\frac{1}{d}\right)^{x} \\
&= f(x)+\left[1-\frac{f(x)}{d}\right] \\
&= \left(1-\frac{1}{d}\right) f(x)+1. \qquad \square
\end{aligned}$$

Acknowledgement

This work has been supported by DARPA under grant number N00039-86-C-0033 and by ONR under grant number N00014-81-K-0004. We would like to thank the Logic Group for providing, as ever, a cooperative and stimulating — and demanding — environment in which to work. We would also like to thank Mike Genesereth, Vladimir Lifschitz, Drew McDermott, Yoav Shoham, Narinder Singh and Marianne Winslett for their comments on various drafts of this paper.

References

[1] J. F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23:123–154, 1984.

[2] J. de Kleer. An assumption-based truth maintenance system. *Artificial Intelligence*, 28:127–162, 1986.

[3] J. Doyle. A truth maintenance system. *Artificial Intelligence*, 12:231–272, 1979.

[4] R. Fagin, J. Ullman, and M. Vardi. On the semantics of updates in databases. In *Proceedings Second ACM Symposium on Principles of Database Systems*, pages 352–365, Atlanta, Georgia, 1983.

[5] R. Fikes and N. J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[6] J. J. Finger. *Exploiting Constraints in Design Synthesis.* PhD thesis, Stanford University, Stanford, CA, 1987.

[7] M. L. Ginsberg. Counterfactuals. *Artificial Intelligence*, 30:35–79, 1986.

[8] M. L. Ginsberg. Multi-valued logics. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 243–247, 1986.

[9] M. L. Ginsberg. Possible worlds planning. In *Proceedings of the 1986 Workshop on Planning and Reasoning about Action*, pages 213–243, Timberline, Oregon, 1986. Morgan Kaufmann.

[10] M. L. Ginsberg and D. E. Smith. Possible worlds planning. In preparation.

[11] M. L. Ginsberg and D. E. Smith. Possible worlds and the qualification problem. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 212–217, 1987.

[12] M. L. Ginsberg and D. E. Smith. Reasoning about action II: The qualification problem. *Artificial Intelligence*, 35:311–342, 1988.

[13] C. C. Green. Theorem proving by resolution as a basis for question-answering systems. In B. Meltzer and D. Mitchie, editors, *Machine Intelligence 4*, pages 183–205. American Elsevier, New York, 1969.

[14] S. Hanks and D. McDermott. Nonmonotonic logics and temporal projection. *Artificial Intelligence*, 33:379–412, 1987.

[15] V. Lifschitz. On the semantics of STRIPS. In *Proceedings of the 1986 Workshop on Planning and Reasoning about Action*, Timberline, Oregon, 1986.

[16] V. Lifschitz. Pointwise circumscription: Preliminary report. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 406–410, 1986.

[17] V. Lifschitz. Formal theories of action. In *Proceedings of the 1987 Workshop on the Frame Problem in Artificial Intelligence*, Lawrence, Kansas, 1987.

[18] V. Lifschitz. Pointwise circumscription. In M. L. Ginsberg, editor, *Readings in Nonmonotonic Reasoning.* Morgan Kaufmann, Los Altos, CA, 1987.

[19] J. McCarthy. Programs with commmon sense. In M. Minsky, editor, *Semantic Information Processing*, pages 403–418. MIT, Cambridge, 1960.

[20] J. McCarthy. Epistemological problems of artificial intelligence. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 1038–1044, Cambridge, MA, 1977.

[21] J. McCarthy. Circumscription – a form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980.

[22] J. McCarthy. Applications of circumscription to formalizing common sense knowledge. *Artificial Intelligence*, 28:89–116, 1986.

[23] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Mitchie, editors, *Machine Intelligence 4*, pages 463–502. American Elsevier, New York, 1969.

[24] R. Moore. Semantical considerations on nonmonotonic logic. *Artificial Intelligence*, 25:75–94, 1985.

[25] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[26] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.

[27] R. Reiter and J. de Kleer. Foundations of assumption-based truth maintenance systems: Preliminary report. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 183–188, 1987.

[28] Y. Shoham. Chronological ignorance. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 389–393, 1986.