# Reasoning About Indefinite Actions

**L. Thorne McCarty**
**Ron van der Meyden***
Computer Science Department
Rutgers University
New Brunswick, NJ 08903

## Abstract

In this paper, we view planning as a special case of reasoning about indefinite actions. We treat actions as predicates defined over a linear temporal order. This formalism permits the representation of concurrent activity. Suppose we have a set of abstract actions defined by Horn clauses from a set of basic actions. Let us assume that an abstract action $\psi$ has occurred, and ask whether a given condition $\phi$ is entailed by all the basic actions that constitute $\psi$. A countermodel to this hypothetical implication is then a plan for "doing $\psi$ and avoiding $\phi$." We propose a formalization of this problem using circumscription, and argue that this is the correct formalization if our action definitions include recursive rules. We then investigate two techniques for solving the problem: (1) a special type of inductive proof procedure, which is sound but not complete; and (2) a decision procedure that works for an interesting subclass of the general problem. Because of the use of recursive rules in our action definitions, we can use these techniques to reason about an abstract action that consists of "doing $\psi$ some finite number of times." We demonstrate the techniques on a simple example involving concurrent, repetitive actions.

## 1 Introduction

Planning problems often take the following form: There is an abstract action that we wish to perform, and a number of concrete ways to perform it. Some of these concrete actions have negative consequences. Is there some way to carry out the abstract action such that the negative consequences do not occur?

For example, consider the situation facing Evelyn F. Gregory in 1928. She was the owner of all the outstanding shares of the United Mortgage Corporation, which in turn owned 1000 shares of the Monitor Securities Corporation. The Monitor shares had appreciated substantially in value, and Mrs. Gregory wanted the United Mortgage Corporation to distribute them to her as a dividend. But this transaction, if carried out directly, would be taxable at ordinary income rates. Was there some way to structure the transaction to avoid this result? Mrs. Gregory's lawyer had an answer. A new company, named Averill, was created and incorporated in Delaware, and the Monitor shares were transferred to the Averill Corporation, in exchange for which Averill issued all of its stock directly to Mrs. Gregory. According to the Revenue Act of 1928, Section 112(i), these events were not taxable. Four days later, the Averill Corporation was liquidated, and the Monitor shares were distributed to Mrs. Gregory, its sole stockholder. This distribution, although taxable, would be treated as a capital exchange. Mrs. Gregory's lawyer had thus succeeded in converting an ordinary dividend into a capital gain, which was taxable at a much lower rate. The Board of Tax Appeals agreed with this characterization of the situation. *Evelyn F. Gregory*, 27 *B.T.A.* 223 (1932).

In this paper, we will investigate a general approach to planning problems of this sort. We assume that we have a number of *abstract actions* which are built up by *definitions* from a set of *basic actions*. These actions, both abstract and basic, will be interpreted over a *linear temporal order*. For example, 'DistributeAssets(umc, efg, $t_1$, $t_2$)', with $t_1 < t_2$, might be just such an abstract action. In addition, we will define various *queries* over this linear temporal order.

*The final version of the paper was prepared while the second author was visiting the School of Computer Science and Engineering, University of New South Wales, Kensington NSW, Australia.

For example, 'Tax(efg, $t_2$)' might be just such a defined query. Now consider the following implication:

$$\text{Tax}(\text{efg}, t_2) \Leftarrow \text{DistributeAssets}(\text{umc}, \text{efg}, t_1, t_2) \quad (1)$$

We interpret this as a *goal* to be *proven*. It means: "If we assume that 'DistributeAssets(umc, efg)' occurs over the interval from $t_1$ to $t_2$, does it follow that 'Tax(efg)' is true at $t_2$?" If the answer is 'Yes', then we know that every possible combination of basic actions that constitutes the defined action 'DistributeAssets(umc, efg, $t_1, t_2$)' results in a situation in which 'Tax(efg, $t_2$)' is true. Presumably, this is the conclusion that the Internal Revenue Service would like to establish. If the answer is 'No', then we know that there is some combination of basic actions that satisfies the definition of the abstract action 'DistributeAssets(umc, efg, $t_1, t_2$)' in which 'Tax(efg, $t_2$)' is not true. In other words, there exists a *countermodel* to the hypothetical implication in (1). This is the conclusion that the taxpayer would like to establish.

We thus view planning as a special case of reasoning about indefinite actions, a position with numerous antecedents in the AI literature. The first explicit statement of the close relationship between "Planning and Acting" was Drew McDermott's 1978 article with that title. "[P]roblem solving is part of the study of *action*," McDermott wrote [1978, p. 72]. "A problem is a difficult action." Within this tradition, various logics have been proposed for the representation of actions [McDermott, 1982; Allen, 1984; Georgeff and Lansky, 1985; Kowalski and Sergot, 1986; Shoham, 1987; Manna and Waldinger, 1987].

By formalizing actions as predicates over a linear temporal order and allowing abstract actions to be built up by means of definitions, we are able to give *hierarchical* definitions of *concurrent* actions, and can represent *non-linear plans* by giving incomplete information about the linear order. In this respect our work is closely related to an early paper by Allen and Koomen [1983] and a more recent paper by Eshghi [1988]. Allen and Koomen represent an action by a conjunction of predicates that hold over temporal intervals and can be decomposed, in turn, into more primitive predicates over temporal intervals. Eshghi's representation, using the Event Calculus of Kowalski and Sergot [1986], has a similar expressive power. In addition, Eshghi proposes a form of reasoning based on *abduction* [Peirce, 1931] rather than deduction. Whereas Allen and Koomen write the expansion of an abstract action as a *necessary* condition, Eshghi writes it initially as a *sufficient* condition and then turns the implication around by an abductive inference. This abductive approach to planning has subsequently been followed up by others [Shanahan, 1989; Missiaen, 1991; Denecker *et al.*, 1992].

Although our ultimate goal is to reason about actions with the complexity apparent in legal domains (see [McCarty, 1989; Schlobohm and McCarty, 1989]), our objectives in the present paper are more modest. We will study hypothetical implications in which the antecedent – e.g., 'DistributeAssets(umc, efg, $t_1, t_2$)' in (1) – is an action defined by a set of Horn clauses. The work of Allen and Koomen [1983] and Eshghi [1988] essentially allows the definition of actions by a set of *non-recursive* Horn clauses. We will extend this work by allowing *recursive* definitions as well. There are many situations in which recursive definitions are needed, one of the most important being the representation of repetitive actions. Given any action '$A(t_1, t_2)$', we often want to talk about an action '$R(t_1, t_n)$' which consists of "doing $A$ some finite number of times." For example: "George ran around the track some finite number of times." "The United Mortgage Corporation created some finite number of subsidiaries." "The Monitor shares were transferred to Mrs. Gregory through some finite chain of intermediaries." We can obviously represent these actions if we allow recursive definitions in our language. However, if we wish to use a recursively defined action in the antecedent of a hypothetical implication, as in (1), it is necessary to go beyond first-order logic, as we will see in Section 2 of this paper.

Section 2, following our work in [McCarty and Meyden, 1991], suggests that the proper way to formalize this problem is to *circumscribe* [McCarthy, 1980; Lifschitz, 1985] the action definitions. We give a precise characterization of the *circumscriptive query problem,* and illustrate it with a simple example of an inference involving concurrent, repetitive actions. However, circumscription is a second-order formalism. Can we ever hope to compute the correct response to a circumscriptive query? The answer is: Yes. The main technical contribution of the present paper is the development of two techniques for deciding whether a hypothetical implication is entailed by the circumscription of a set of action definitions. The first technique, presented in Section 3, is based on a special type of proof by induction, and is therefore sound but not complete. The second technique, presented in Section 4, is an actual decision procedure, which works for a restricted, but useful, class of queries. We illustrate the techniques in these two sections by solving the simple problem introduced in Section 2.

In Section 5, we present another example which shows how a planning problem can be solved by the construction of a countermodel to a hypothetical implication. This example also shows how the "closed world assumption" can be incorporated into our formalism. Section 6 then compares our approach to related work.

## 2 Circumscribed Action Definitions

In a recent paper [McCarty and Meyden, 1991], we suggested that indefinite information arises in common sense reasoning from the *circumscription* [McCarthy, 1980; Lifschitz, 1985] of definite rules. We will review our analysis briefly here, and then show how to extend it to the case of indefinite actions.

We begin with a Gedanken experiment. Suppose we have a set of *basic predicates*: 'Block($x$)', 'Red($x$)', 'Green($x$)', 'On($x, y$)', that express observable facts about the world. Suppose we also have sufficient conditions for a set of *defined predicates*, as follows:[1]

$$\mathbf{ChristmasBlock}(x) \Leftarrow \mathrm{Block}(x) \wedge \mathrm{Red}(x) \tag{2}$$

$$\mathbf{ChristmasBlock}(x) \Leftarrow \mathrm{Block}(x) \wedge \mathrm{Green}(x) \tag{3}$$

$$\mathbf{Above}(x, y) \Leftarrow \mathrm{On}(x, y) \tag{4}$$

$$\mathbf{Above}(x, y) \Leftarrow \mathrm{On}(x, z) \wedge \mathbf{Above}(z, y) \tag{5}$$

Now imagine a two-person communication situation in which the "speaker" applies these definitions to a world of observable facts, and reports some of his or her conclusions. For example, the speaker might report: '**ChristmasBlock**(a)' and '**Above**(a,b)'. It is our job (as the "hearer") to make inferences about the actual state of the world, even though we have not observed it directly. Note that this is essentially an *abductive* task [Peirce, 1931].

How should we formalize this problem? For the non-recursive rules (2)–(3), Clark's Predicate Completion [Clark, 1978] gives us the correct results. For example, the following implication is entailed by the completion of (2) and (3):

$$\mathrm{Red(a)} \vee \mathrm{Green(a)} \;\Leftarrow\; \mathbf{ChristmasBlock}(a),$$

and we can therefore conclude that 'a' is either 'Red' or 'Green'. However, for the recursive rules (4)–(5) the situation is more complex. Consider the following hypothetical implication:

$$(\exists w)\mathrm{On}(w, \mathrm{b}) \;\Leftarrow\; \mathbf{Above}(a, b). \tag{6}$$

[1] For clarity, we use boldface type for the defined predicates. We also use italic type for variables and roman type for individual constants.

Intuitively, if we have been told that '**Above**(a,b)' is true according to the definition in (4)–(5), then we ought to conclude that there is something on 'b'. But the completion of (4)–(5) would give us:

$$\begin{aligned}&\mathbf{Above}(x, y) \Rightarrow \\ &\quad \mathrm{On}(x, y) \\ &\quad \vee\, (\exists z)[\mathrm{On}(x, z) \wedge \mathbf{Above}(z, y)],\end{aligned} \tag{7}$$

and (4)–(5) and (7) together do not entail (6). The solution to this problem is to use the *circumscription* of '**Above**' in rules (4)–(5), instead of using predicate completion. We refer the reader to [McCarty and Meyden, 1991] for the details.

The extension of these ideas to action definitions is straightforward. Suppose we have a set of *basic actions*: 'A($x, t_1, t_2$)', 'B($x, y, t_1, t_2$)', 'C($y, t_2$)', in which $x$ and $y$ are *object variables* and $t_1$ and $t_2$ are *order variables*. Intuitively, 'A($x, t_1, t_2$)' asserts that some observable fact about $x$ is true over the time interval from $t_1$ to $t_2$. We now add *abstract actions*: '$\mathbf{P}(x, t_1, t_2)$', '$\mathbf{R}(x, y, t_1, t_2)$', etc. Abstract actions are defined by Horn clauses that are allowed to contain *order relations* in their antecedents, but may not contain function symbols. For example, we could define '$\mathbf{P}(x, t_1, t_2)$' as follows:

$$\mathbf{P}(x, t_1, t_2) \;\Leftarrow\; \mathrm{A}(x, t_1, t_2) \;\wedge\; t_1 < t_2 \tag{8}$$

$$\begin{aligned}\mathbf{P}(x, t_1, t_2) \;\Leftarrow\; &\mathrm{A}(x, t_1, t_3) \;\wedge\; \mathbf{P}(x, t_3, t_2) \;\wedge \\ &t_1 < t_3 < t_2\end{aligned} \tag{9}$$

The only difference between these action definitions and the definitions studied in [McCarty and Meyden, 1991] is the special treatment of order variables and order relations, but this difference is quite significant. The relation '$t_1 < t_2$' is interpreted over a *linear order*, and it means: "$t_1$ is strictly less than $t_2$." In other words, '<' is both transitive and irreflexive, and it satisfies the disjunctive constraint: $t_1 < t_2 \;\vee\; t_1 = t_2 \;\vee\; t_2 < t_1$. Because of this constraint, the order relations provide an additional source of indefinite information in our language.

As illustrated in (8)–(9), action definitions can be recursive. Thus the argument in [McCarty and Meyden, 1991] that Clark's Predicate Completion should not be used as a formalization of our Gedanken experiment applies here as well. Instead, we will *circumscribe* the defined actions. Let $\mathcal{R}$ be the set of action definitions, and let $\boldsymbol{P} = \,<P_1, P_2, \ldots, P_k>$ be a tuple consisting of all the predicates that appear on the left-hand side of the rules in $\mathcal{R}$. We denote the circumscription of $\boldsymbol{P}$ in $\mathcal{R}(\boldsymbol{P})$ by $Circ(\mathcal{R}(\boldsymbol{P});\boldsymbol{P})$. When we wish to talk about predicate completion, we write $Comp(\mathcal{R})$ instead. Now let $\phi$, the query, be a *positive existential formula* constructed from basic actions, abstract actions, and (possibly) order relations. If $\phi$ contains no abstract actions, it will be called a *basic* query. Let $\psi$ be a *conjunction* of basic actions, abstract actions, and (possibly) order relations. If there are free variables in $\phi$ or $\psi$, we assume that the goal $\phi \Leftarrow \psi$ is universally quantified at top level. The *circumscriptive query problem* is now: $Circ(\mathcal{R}(\boldsymbol{P});\boldsymbol{P}) \models \phi \Leftarrow \psi$.

Although the hypothetical implication (1) in Section 1 satisfies our constraints on $\phi$ and $\psi$, the action 'DistributeAssets(umc, efg, $t_1, t_2$)' is too complex to analyze here. We will therefore consider a simpler example, suggested by the following story:

**Mermaids Get the Sack**

Ethel and Daphne Mermaid are synchronized swimmers in the employ of Sam Silverscreen, movie mogul. On the set of "Swimming in the Rain," Sam is giving Ethel and Daphne instructions for the next scene. "What I want you to do today is very simple," Sam says. "Just keep swimming laps of the pool. As soon as you finish a lap, start the next. But, whenever you both start a lap at the same time you must also finish it at the same time. Don't stop till I tell you to."

"OK, boss," reply Ethel and Daphne (even their speech is synchronized.)

"Lights! Camera! Action!" says Sam, and the Mermaids start their synchronized swimming. Sam retires to his office for a nap on the casting couch.

When Sam returns some time later, he notices that Ethel has just completed a lap, whereas Daphne is still half way through hers. "Cut! Cut!" he yells when Daphne reaches the end of the pool. "Did either of you stop swimming at any stage?" he asks.

"No, boss," reply the Mermaids.

"Well, you're both fired anyway," says Sam.

Question: Why did the Mermaids get the sack?

Here is a formalization of the story: Let $\mathrm{A}(x, t_1, t_2)$ represent the action in which Mermaid $x$ swims one lap of the pool over the time interval from $t_1$ to $t_2$. The situation that would lead to the Mermaids' dismissal, assuming that neither stopped swimming, is then:

$$(\exists x, y, v_1, v_2, v_3) \quad [\mathrm{A}(x, v_1, v_2) \wedge \mathrm{A}(y, v_1, v_3) \wedge v_1 < v_2 < v_3] \tag{10}$$

Call this $\phi$. Sam Silverscreen knows that each Mermaid has completed some finite number of laps, and we can represent this situation recursively. Define the abstract action '$\mathbf{P}(x, t_1, t_2)$' by the Horn clauses in (8)–(9). Then '$\mathbf{P}(\mathrm{d}, u_1, u_2)$' is the action in which Daphne swims some finite number of laps from $u_1$ to $u_2$, and '$\mathbf{P}(\mathrm{e}, u_1, u_3)$' is the action in which Ethel swims some finite number of laps from $u_1$ to $u_3$. Figure 1 shows the situation that Sam observes (and infers) when he returns from his nap at time $u_3$. Thus we need to establish:

$$\phi \Leftarrow \mathbf{P}(\mathrm{d}, u_1, u_2) \wedge \mathrm{A}(\mathrm{d}, u_2, u_4) \wedge \mathbf{P}(\mathrm{e}, u_1, u_3) \wedge u_1 < u_2 < u_3 < u_4 \tag{11}$$

If we can show that (11) follows from the circumscription of '$\mathbf{P}$' in (8)–(9), then we know with certainty that the Mermaids disobeyed their instructions and were (legally!) fired.

How can we show this? First, let us recall some results from [McCarty and Meyden, 1991; Meyden, 1992 in press], which dealt with the simpler framework in which there are no order relations:

**Theorem 2.1** *For arbitrary $\mathcal{R}$, $\phi$, $\psi$ not containing order relations, the circumscriptive query problem $Circ(\mathcal{R}(\boldsymbol{P});\boldsymbol{P}) \models \phi \Leftarrow \psi$ is not decidable. However, the set $\{\langle \mathcal{R}, \phi, \psi \rangle \mid Circ(\mathcal{R}(\boldsymbol{P});\boldsymbol{P}) \not\models \phi \Leftarrow \psi\}$ is recursively enumerable.*

Thus, since adding order relations can only add to the complexity of the circumscriptive query problem, we cannot hope to have decidability in general, nor even a complete proof theory, since we are dealing with an undecidable set whose complement is recursively enumerable. However, we also identified in [McCarty and Meyden, 1991; Meyden, 1992 in press] a decidable subproblem:

**Theorem 2.2** *For arbitrary $\mathcal{R}$, $\psi$ and* basic *queries $\phi$, none of which contain order relations, the circumscriptive query problem $Circ(\mathcal{R}(\boldsymbol{P});\boldsymbol{P}) \models \phi \Leftarrow \psi$ is decidable.*

This gives us some hope that a similar result may hold when we add order relations. This would suffice for a solution of the Mermaid problem, since the query $\phi$ for this problem is basic. Unfortunately, the generalization does not go through.

**Theorem 2.3** *For arbitrary $\mathcal{R}$, $\psi$ and basic queries $\phi$ containing order relations, the circumscriptive query problem $Circ(\mathcal{R}(\boldsymbol{P});\boldsymbol{P}) \models \phi \Leftarrow \psi$ is not decidable.*

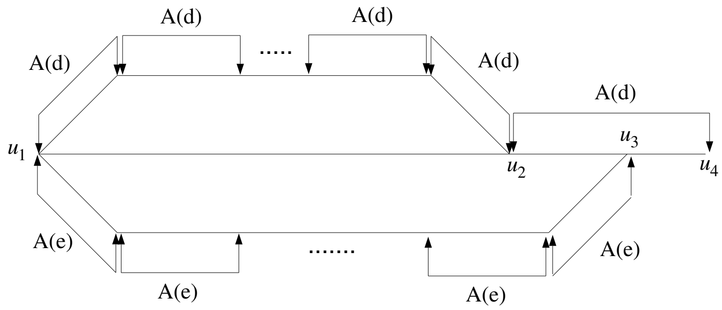


Figure 1: Why Did the Mermaids Get the Sack?

Thus, adding order relations makes the circumscriptive query problem strictly more complex. However, the complementary problem has not increased in complexity:

**Theorem 2.4** *For arbitrary $\mathcal{R}$, $\psi$, $\phi$ containing order relations, the set $\{\langle \mathcal{R}, \phi, \psi \rangle \mid Circ(\mathcal{R}(\boldsymbol{P});\boldsymbol{P}) \not\models \phi \Leftarrow \psi\}$ is recursively enumerable.*

Theorem 2.4 shows that the planning problem, as we defined it in Section 1, is semidecidable: If there exists a countermodel to the hypothetical implication $\phi{\Leftarrow}\psi$, then we will eventually be able to find it. But Theorem 2.3 and Theorem 2.4 together show that even for basic queries $\phi$, there is no possibility of a sound and complete proof procedure for the circumscriptive query problem itself, since the set $\{\langle \mathcal{R}, \phi, \psi \rangle \mid Circ(\mathcal{R}(\boldsymbol{P}); \boldsymbol{P}) \models \phi{\Leftarrow}\psi\}$ is not recursively enumerable.

However, all is not lost. In the following two sections of the paper, we will investigate two different techniques that are capable of solving the problem: "Why did the Mermaids get the sack?" In fact, as we will see at the end of Section 4, the mermaid example happens to belong to an interesting class of circumscriptive query problems that are decidable in polynomial time!

## 3 Prototypical Proofs

The definition in (8)–(9) has a relatively simple form: It consists of a nonrecursive Horn clause, (8), and a linear recursive Horn clause, (9). Let us call this a *linear recursive definition*. In this section we will show how to solve the circumscriptive query problem for linear recursive definitions, using a special type of inductive proof. Our solution is a novel application of *second-order intuitionistic logic programming* [McCarty, 1988a; McCarty, 1988b; McCarty and Meyden, 1991; Nadathur and Miller, 1988; Miller, 1989].

The first step is to simplify the problem. Suppose we apply predicate completion to (8) alone, to obtain the following:

$$\mathbf{P}(x, t_1, t_2) \Rightarrow \mathrm{A}(x, t_1, t_2) \ \wedge \ t_1 < t_2 \qquad (12)$$

We think of this as the *prototypical definition* of '$\mathbf{P}(x, t_1, t_2)$', and write it in general as '$P(\mathbf{x}; \mathbf{t}){\Rightarrow}P^0(\mathbf{x}; \mathbf{t})$'. Let $\mathcal{P}(\boldsymbol{P})$ be the collection of prototypical definitions for the recursively defined predicates in $\mathcal{R}$. We now note the following simple fact, which is proven in [McCarty, 1992]:

**Theorem 3.1** *If $Circ(\mathcal{R}(\boldsymbol{P}); \boldsymbol{P}) \models \phi{\Leftarrow}\psi$ then $Comp(\mathcal{R}) \cup \mathcal{P}(\boldsymbol{P}) \models \phi{\Leftarrow}\psi$.*

Intuitively, if $\phi$ follows from all possible expansions of $\psi$, then it surely follows from the prototypical expansion of $\psi$. A solution to the right-hand side of Theorem 3.1 is called a *prototypical proof*. If we fail to find a prototypical proof, we have failed, period. But if we succeed, we can then use the prototypical proof to suggest an inductive proof for the left-hand side of Theorem 3.1.

Finding a prototypical proof is a simple problem in first-order intuitionistic logic programming [McCarty, 1988a; McCarty, 1988b; Nadathur and Miller, 1988]. The goal, $\phi{\Leftarrow}\psi$, is a universally quantified implication, so we can prove it by replacing all the universally quantified variables with unique constants, asserting the antecedent, $\psi$, into a hypothetical database, and then showing that the conclusion, $\phi$, can be proven from this hypothetical database. In the mermaid example, we assert:

$$\begin{aligned}&\mathbf{P}(\mathrm{d}, \mathrm{u}_1, \mathrm{u}_2), \mathrm{A}(\mathrm{d}, \mathrm{u}_2, \mathrm{u}_4), \mathbf{P}(\mathbf{e}, \mathrm{u}_1, \mathrm{u}_3),\\ &\quad \mathrm{u}_1 < \mathrm{u}_2 < \mathrm{u}_3 < \mathrm{u}_4\end{aligned} \qquad (13)$$

and try to prove (10) using (8)–(9) and (12). It is easy to see that this query succeeds with $x = \mathrm{d}$, $y = \mathrm{e}$, and $v_i = \mathrm{u}_i$ for $i = 1, 3$. Notice, however, that we need to use (12) twice in this proof, once for each instance of

'**P**' in (13). Without the use of (12), we would only be able to prove that

$$\mathrm{A}(\mathrm{d}, \mathrm{u}_1, \mathrm{u}_2), \mathrm{A}(\mathrm{d}, \mathrm{u}_2, \mathrm{u}_4), \mathrm{A}(\mathrm{e}, \mathrm{u}_1, \mathrm{u}_3), \quad \mathrm{u}_1 < \mathrm{u}_2 < \mathrm{u}_3 < \mathrm{u}_4 \tag{14}$$

implies (10). Our task, therefore, is to "strengthen" the proof that (14) implies (10) into a proof that (13) implies (10).

There are two ways to try to do this, and each approach can be tried on each of the two instances of 'A' generated by (12) from (13). We thus encounter a nondeterministic choice in the computation, and we will simply describe here the successful branch. The first approach uses predicate completion, and tries to finish the proof by means of a *disjunctive splitting procedure* [Loveland, 1991]. Let us apply this technique to prove that

$$\mathrm{A}(\mathrm{d}, \mathrm{u}_1, \mathrm{u}_2), \mathrm{A}(\mathrm{d}, \mathrm{u}_2, \mathrm{u}_4), \mathbf{P}(\mathrm{e}, \mathrm{u}_1, \mathrm{u}_3), \quad \mathrm{u}_1 < \mathrm{u}_2 < \mathrm{u}_3 < \mathrm{u}_4 \tag{15}$$

implies (10). The completion of (8)–(9) is:

$$\begin{aligned} \mathbf{P}(x, t_1, t_2) \Rightarrow & \\ & \mathrm{A}(x, t_1, t_2) \wedge t_1 < t_2 \\ & \vee (\exists t)[\mathrm{A}(x, t_1, \mathrm{t}) \ \wedge \ \mathbf{P}(x, t, t_2) \ \wedge \ t_1 < t < t_2] \end{aligned} \tag{16}$$

We have already proven the desired result from the first disjunct in (16), and we leave it to the reader to show that the desired result also follows from the second disjunct. The proof requires the use of the linear order constraint: $t < \mathrm{u}_2 \vee t = \mathrm{u}_2 \vee \mathrm{u}_2 < t$, which in turn requires another use of the disjunctive splitting procedure. (The reader may wish to consult Figure 1 to see why this is so.)

The second approach is inductive. Returning to the original definition of '$\mathbf{P}(x, t_1, t_2)$', let us apply predicate completion to (9) alone:

$$\begin{aligned} \mathbf{P}(x, t_1, t_2) \Rightarrow & \\ & (\exists t)[\mathrm{A}(x, t_1, t) \wedge \mathbf{P}(x, t, t_2) \wedge t_1 < t < t_2] \end{aligned} \tag{17}$$

and then replace all the instances of '**P**' in (17) with a predicate variable '$X$':

$$\begin{aligned} X(x, t_1, t_2) \Rightarrow & \\ & (\exists t)[\mathrm{A}(x, t_1, t) \wedge X(x, t, t_2) \wedge t_1 < t < t_2] \end{aligned} \tag{18}$$

We call this second-order sentence the *transformation* associated with '$\mathbf{P}(x, t_1, t_2)$', and we write it in general as '$X(\mathbf{x};\mathbf{t}) \Rightarrow \Delta X(\mathbf{x};\mathbf{t})$'. Thus '$\Delta X(x, t_1, t_2)$' denotes the right-hand side of (18). Although (18) is written as an implication, it should be viewed, quite literally, as an operation that *transforms* any relation between **x** and **t** that matches its left-hand side into a relation between **x** and **t** that matches its right-hand side. In particular, it can be used to transform the prototypical relation '$P^0(\mathbf{x};\mathbf{t})$', and then used again to transform the resulting relation, and so on. Intuitively, if the proof of $\phi$ is "preserved" under all such transformations, then $\phi$ must be true.

To formalize this argument, we need an induction schema. Notice, in "strengthening" the proof of (10) from (14) into a proof of (10) from (15), that we have actually shown the following:

$$\begin{aligned} \phi \ \Leftarrow \ & \mathrm{A}(\mathrm{d}, u_1, u_2) \wedge \mathrm{A}(\mathrm{d}, u_2, u_4) \wedge \mathbf{P}(\mathrm{e}, u_1, u_3) \wedge \\ & u_1 < u_2 < u_3 < u_4. \end{aligned} \tag{19}$$

Let $\Phi$ be the schema common to (11) and (19) in which the first occurrence of '**P**' in (11) is taken to be the variable. Thus (11) would be written $\Phi(\mathbf{P})$, and (19) would be written $\Phi(\mathbf{P}^0)$. Substituting '$X$' and '$\Delta X$' into $\Phi$ as well, we have:

**Definition 3.2** *The induction schema for $\Phi(P)$ is the following sentence in second-order intuitionistic logic:*

$$\Phi(P) \ \Leftarrow \ \Phi(P^0) \ \wedge \ (\forall X)[\Phi(\Delta X) \Leftarrow \Phi(X)]$$

We now note a surprising fact: The fragment of second-order intuitionistic logic that is needed for the application of this induction schema has a complete proof procedure (see [McCarty, 1992; Miller, 1989]). The procedure is similar to the first-order proof procedure for universally quantified implications discussed in [McCarty, 1988b]. To prove the second conjunct on the right-hand side of Definition 3.2, we replace the predicate variable '$X$' with a new predicate constant '!X', we assert $\Phi(!\mathrm{X})$ into the rulebase, and we try to prove $\Phi(\Delta!\mathrm{X})$. If this proof succeeds, then we have proven the goal: $(\forall X)[\Phi(\Delta X) \Leftarrow \Phi(X)]$.

Let us see how this works in the mermaid problem. Carrying out the indicated substitutions, $\Phi(!\mathrm{X})$ is:

$$\begin{aligned} \phi \ \Leftarrow \ & !\mathrm{X}(\mathrm{d}, u_1, u_2) \wedge \\ & \mathrm{A}(\mathrm{d}, u_2, u_4) \wedge \mathbf{P}(\mathrm{e}, u_1, u_3) \wedge \\ & u_1 < u_2 < u_3 < u_4, \end{aligned} \tag{20}$$

and $\Phi(\Delta!\mathrm{X})$ is equivalent to:

$$\phi \Leftarrow \mathrm{A}(\mathrm{d}, u_1, t) \wedge !\mathrm{X}(\mathrm{d}, t, u_2) \wedge \mathrm{A}(\mathrm{d}, u_2, u_4) \wedge \mathbf{P}(\mathrm{e}, u_1, u_3) \wedge u_1 < t < u_2 < u_3 < u_4, \tag{21}$$

so we assert (20) and try to prove (21). But (21) is itself a first-order universally quantified implication, so we assert the right-hand side of (21), with variables replaced by constants, and try to prove the left-hand side. We leave it to the reader to verify that this proof goes through. Note, in particular, that rule (20) is essential to this proof, and that the predicate '!X' in (20) unifies with the predicate '!X' in (21). Putting all these pieces together – if we assume the induction schema in Definition 3.2 – we have just succeeded in proving $\Phi(\mathbf{P})$, which is simply an abbreviation of the goal $\phi \Leftarrow \psi$ in (11).

The justification for our procedure is the following soundness theorem for inductive proofs (see [McCarty, 1992]). Let $\mathcal{S}(\boldsymbol{P})$ be the set of all induction schemata, i.e., the schemata given by Definition 3.2 for all linear recursive definitions in $\mathcal{R}$ and all possible $\Phi$ . Then:

**Theorem 3.3** *If* $Comp(\mathcal{R}) \cup \mathcal{S}(\boldsymbol{P}) \models \phi \Leftarrow \psi$ *then* $Circ(\mathcal{R}(\boldsymbol{P}); \boldsymbol{P}) \models \phi \Leftarrow \psi$.

It is interesting to compare Theorem 3.1 with Theorem 3.3. If we could always "strengthen $\mathcal{P}(\boldsymbol{P})$ to $\mathcal{S}(\boldsymbol{P})$, then we would have a sound and complete proof procedure for the circumscriptive query problem. But we know from Theorem 2.3 and Theorem 2.4 that this is impossible. Thus the construction of an appropriate induction schema in Definition 3.2 is necessarily a heuristic step.

In the following section of the paper, we will establish a stronger result, for a smaller class of problems.

## 4 A Decision Procedure

We have seen in Theorem 2.3 that when dealing with order relations, the circumscriptive query problem is not decidable, even if we restrict the query $\phi$ to be basic, whereas this problem was decidable in the absence of order relations (Theorem 2.2). We will show in this section that it is possible to recover decidability of basic queries containing order relations, provided we impose a constraint on the rules $\mathcal{R}$. The program $\mathcal{R}$ is said to be *regular* if every rule in $\mathcal{R}$ is a linear rule of the form:

$$\mathbf{Q}(\mathbf{x}; t_1, t_k) \Leftarrow \bigwedge B_i(\mathbf{x}, \mathbf{y}; t_1, \ldots, t_{k-1}) \wedge \mathbf{R}(\mathbf{x}, \mathbf{y}; t_{k-1}, t_k) \wedge t_1 < \ldots < t_{k-1} < t_k \tag{22}$$

in which the $B_i$ are basic actions and $\mathbf{R}$ is an abstract action, or a rule of the same form except that the abstract action $\mathbf{R}$ is omitted. Note that (8)–(9), the set of action definitions in the mermaid problem, is regular. The main result of this section is that if the rules $\mathcal{R}$ are regular, the the circumscriptive query problem is decidable for basic queries.

The most important feature of regular action definitions is the fact that the basic actions in the body of each recursive rule occur *prior* to the abstract action. Our decision procedure exploits this fact by interleaving two operations: (i) an expansion of the abstract actions using predicate completion, and (ii) a topological sort of the order constants in each expansion. Intuitively, we are searching for a model of $\psi$ in which $\phi$ does not hold, i.e., a countermodel to the implication $\phi \Leftarrow \psi$. If no such countermodel exists, then $\phi \Leftarrow \psi$ is entailed by $Circ(\mathcal{R}(\boldsymbol{P}); \boldsymbol{P})$. Although the models of $\psi$ are potentially infinite in number, it turns out that we can determine the entailment of a basic query by checking only a *finite* set of *partial* expansions. (This approach will not work for general queries, but note that we can always expand a query containing only *nonrecursive* abstract actions into an equivalent basic query.)

We will illustrate our decision procedure with the mermaid example. The procedure is nondeterministic, and we will only explore one of the branches of the search tree. As in the proof procedure of Section 3, our starting point is the database shown in (13), which happens to be topologically sorted. Let us add a marker '$\|$' to this database, and stipulate that all the actions to the left of the marker must be basic and linearly ordered. Since '$\mathrm{u}_1$' is the unique minimal element in the linear order, we can immediately move it to the left:

$$\mathrm{u}_1 \;\|\; \mathbf{P}(\mathrm{d}, \mathrm{u}_1, \mathrm{u}_2), \mathrm{A}(\mathrm{d}, \mathrm{u}_2, \mathrm{u}_4), \mathbf{P}(\mathrm{e}, \mathrm{u}_1, \mathrm{u}_3), \mathrm{u}_2 < \mathrm{u}_3 < \mathrm{u}_4, \tag{23}$$

We now have two abstract actions on the right that contain $\mathrm{u}_1$, which is the maximal element in the linear order on the left, so we cannot shift the marker further until we have expanded these abstract actions. Let us use (16) for the expansion, choosing a disjunct nondeterministically. Suppose we select the second disjunct in each case. Then the two abstract actions in (23) are replaced by:

$$\mathrm{A}(\mathrm{d}, \mathrm{u}_1, t), \mathbf{P}(\mathrm{d}, t, \mathrm{u}_2), \mathrm{u}_1 < t < \mathrm{u}_2$$

and

$$\mathrm{A}(\mathrm{e}, \mathrm{u}_1, t'), \mathbf{P}(\mathrm{e}, t', \mathrm{u}_3), \mathrm{u}_1 < t' < \mathrm{u}_3$$

Our database is now only partially ordered, and there are two minimal elements, $t$ and $t$', on the right side of the marker. Suppose we continue the topological sort by setting $t = t'$ and mapping both of these elements to the next point on the left side of the marker. We then obtain:

$$\begin{aligned} &\mathrm{A}(\mathrm{d}, \mathrm{u}_1, t), \mathrm{A}(\mathrm{e}, \mathrm{u}_1, t), \mathrm{u}_1 < t \;\; \| \\ &\quad \mathbf{P}(\mathrm{d}, t, \mathrm{u}_2), \mathrm{A}(\mathrm{d}, \mathrm{u}_2, \mathrm{u}_4), \mathbf{P}(\mathrm{e}, t, \mathrm{u}_3), \\ &\quad \mathrm{u}_2 < \mathrm{u}_3 < \mathrm{u}_4 \end{aligned} \tag{24}$$

The two basic actions '$\mathrm{A}(\mathrm{d}, \mathrm{u}_1, t)$' and '$\mathrm{A}(\mathrm{e}, \mathrm{u}_1, t)$' have been shifted to the left here, since they are now completely contained within the sorted segment of the database.

Is it necessary to continue the search further along this branch? Notice that the constant '$\mathrm{u}_1$' in (24) occurs only on the left side of the marker. Therefore, however the process of expanding and sorting is continued, the two atoms on the left will be the only atoms containing the constant '$\mathrm{u}_1$'. Can these atoms contribute in any way to the satisfaction of the query? Clearly they do not satisfy the query on their own. Thus, if the query is ever to be satisfied using these atoms, we would need to introduce at least one more atom containing the constant '$\mathrm{u}_1$', and we have just argued that this cannot happen. Thus, the two atoms on the left can make no contribution to the satisfaction of the query, and can safely be deleted. This leaves us with the database:

$$\begin{aligned} t \;\; \| \;\; &\mathbf{P}(\mathrm{d}, t, \mathrm{u}_2), \mathrm{A}(\mathrm{d}, \mathrm{u}_2, \mathrm{u}_4), \mathbf{P}(\mathrm{e}, t, \mathrm{u}_3), \\ &\mathrm{u}_2 < \mathrm{u}_3 < \mathrm{u}_4 \end{aligned} \tag{25}$$

Notice that (25) is isomorphic to (23), which means that our procedure has looped. We can thus safely terminate the present branch of our search, and backtrack to consider alternative expansions and alternative topological sorts. We leave it to the reader to verify that the remaining branches eventually entail the query.

Although our argument for terminating the search in this example may seem *ad hoc*, it can actually be extended into a systematic procedure. This is done by means of an equivalence relation on the left halves of the partial expansions, which we now describe informally. Notice that the left halves interact with the right halves only through the constants in common to the two halves. Every model produced by continuing the expansion process from a partial expansion $L \| R$ can be written as $L \cup S$, where the order constants of $S$ are linearly ordered and "to the right" of $L$, except, possibly, for the constants $C$ in both $L$ and $R$. Consider two left halves $L$ and $L'$, which share only constants $C$ with the right halves. Say $L$ and $L'$ are equivalent with respect to $\phi$ if for every corresponding right half $S$ of a model, $L \cup S$ satisfies $\phi$ if and only if $L' \cup S$ does. We show in [Meyden, 1992 forthcoming] that for regular rules this yields an equivalence relation of finite rank on the left halves of the partial expansions, so that these can be compactly encoded using the representatives of these equivalence classes. This reduces the query problem to checking a finite number of partial expansions, and formalizes the type of loop checking illustrated above.

This analysis leads to the following result:

**Theorem 4.1** *If $\mathcal{R}$ is a set of regular action definitions and if $\phi$ is a basic query, then the problem $Circ(\mathcal{R}(\boldsymbol{P}); \boldsymbol{P}) \models \phi \Leftarrow \psi$ is decidable in deterministic space $O\left((|\psi| \,.\, |\mathcal{R}|)^{4|\phi|}\right)$.*

Notice that the space complexity of the query problem depends exponentially on the size of $\phi$, but is polynomial as a function of the size of the hypothetical assertion $\psi$ and the rules $\mathcal{R}$. Thus, if we take $\phi$ to be fixed and view $\psi$ and $\mathcal{R}$ as "data", then the *data complexity* of our decision procedure is in PSPACE. In the mermaid problem, this would correspond to taking the test for dismissal, $\phi$, as fixed, and analyzing the implications of various scenarios as Sam returns from his nap.

This complexity can be reduced to PTIME by a further reasonable restriction. Say that a conjunction of abstract actions $\psi$ has *$k$-bounded concurrency* if every model of $\psi$ obtained by expanding abstract actions using the rules and topologically sorting satisfies the following property:

> For every point $t$ there exist no more than $k$ basic actions $A(\mathbf{x}, u, v)$ with $u < t < v$.

Although this condition refers to an infinite set of expansions, it turns out to be easily decidable, as shown in [Meyden, 1992 forthcoming]. We then have the following result:

**Theorem 4.2** *For a fixed basic query $\phi$ and for abstract actions $\psi$ that have $k$-bounded concurrency and contain no more than a fixed number of object constants, $Circ(\mathcal{R}(\boldsymbol{P}); \boldsymbol{P}) \models \phi \Leftarrow \psi$ is decidable in polynomial time.*

Note that our mermaid problem has 2-bounded concurrency and contains only two object constants, and thus satisfies the conditions of Theorem 4.2 with $k = 2$.

Theorem 4.1 can be generalized in a variety of ways. As we have already mentioned, decidability still holds for queries $\phi$ containing non-recursive defined predicates. Decidability is also preserved if one allows in the body of regular rules defined predicates whose (possibly recursive) definition does not contain order variables. We will use the latter extension in the next section. We note, however, that the complexity result stated in Theorem 4.2 does not apply to these extensions.

## 5 Countermodels as Plans

The mermaid problem is not, strictly speaking, a planning problem. We can recast it as a planning problem by asking whether there is any way for Ethel and Daphne to bring about the scenario depicted in Figure 1 without violating Sam's instructions. The inductive proof procedure in Section 3 and the decision procedure in Section 4 then show that such a "plan" is impossible. However, this is an artificial interpretation of the story.

In this section, we will analyze a second example in which a countermodel to the implication $\phi \Leftarrow \psi$ is easily seen to be a plan for "doing $\psi$ and avoiding $\phi$." The example also shows how the "closed world assumption" can be incorprated into our formalism. Consider the following story:

**The Radioactive Robot**

Two robots, R1 and R2, inhabit a suite of six rooms, connected as indicated in Figure 2. Initially, robot R1 is in room D and robot R2 is in room F. We wish to swap the locations of the robots. However, robot R2 is radioactive, and if the two robots are ever in the same room at the same time, R1 will also become contaminated with radiation. It is therefore necessary to ensure that the two robots are never in the same room at the same time.

To formalize this problem, we introduce a basic predicate '$\mathrm{In}(x, y, t_1, t_2)$' which asserts that $x$ is in room $y$ for a period of time starting at $t_1$ and ending at $t_2$. We also use an abstract action '$\mathbf{Move}(x, y, z, t_1, t_2)$' which asserts that $x$ moves from room $y$ to room $z$ over the time interval from $t_1$ to $t_2$. We let the constants A, ..., F denote the individual rooms. We then define $\psi$ to be the following conjunction:

$$\mathbf{Move}(\mathrm{R1}, \mathrm{D}, \mathrm{F}, u_1, u_2) \wedge \mathbf{Move}(\mathrm{R2}, \mathrm{F}, \mathrm{D}, u_1, u_2) \tag{26}$$

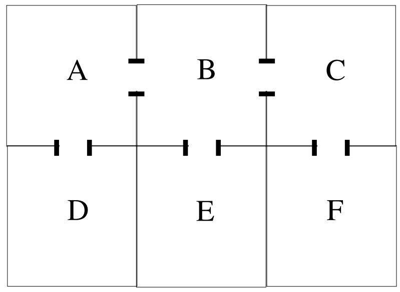


Figure 2: The Radioactive Robot

and define $\phi$ to be the following existential formula:

$$(\exists y, t_1, t_2, t'_1, t'_2, t) [\mathrm{In}(\mathrm{R1}, y, t_1, t_2) \wedge \mathrm{In}(\mathrm{R2}, y, t'_1, t'_2) \wedge t_1 \leq t \leq t_2 \wedge t'_1 \leq t \leq t'_2]. \tag{27}$$

Clearly, $\phi$ describes the situation that we want to avoid, i.e., a situation in which R1 and R2 are both in the same room at the same time. Thus a countermodel to the implication $\phi \Leftarrow \psi$, if it exists, would give us a way to swap the locations of the two robots without violating the stated constraint.

We now define the predicate '$\mathbf{Move}(x, y, z, t_1, t_2)$'. Let $\mathcal{R}(\boldsymbol{P})$ consist of the rules:

$$\mathbf{Move}(x, y, z, t_1, t_2) \Leftarrow \mathrm{In}(x, y, t_1, t) \wedge \mathrm{In}(x, z, t, t_2) \wedge \mathbf{Conn}(y, z) \wedge t_1 < t < t_2, \tag{28}$$

$$\mathbf{Move}(x, y, z, t_1, t_2) \Leftarrow \mathrm{In}(x, y, t_1, t) \wedge \mathbf{Move}(x, u, z, t, t_2) \wedge \mathbf{Conn}(y, u) \wedge t_1 < t < t_2, \tag{29}$$

together with the following rules to express the connectedness relations of Figure 2:

$$\begin{aligned} \mathbf{Conn0}(\mathrm{A}, \mathrm{B}) &\Leftarrow \\ \mathbf{Conn0}(\mathrm{B}, \mathrm{C}) &\Leftarrow \\ \mathbf{Conn0}(\mathrm{D}, \mathrm{A}) &\Leftarrow \\ \mathbf{Conn0}(\mathrm{E}, \mathrm{B}) &\Leftarrow \\ \mathbf{Conn0}(\mathrm{F}, \mathrm{C}) &\Leftarrow \end{aligned} \tag{30}$$

To ensure the symmetry of connectedness, we add the rules:

$$\begin{aligned} \mathbf{Conn}(x, y) &\Leftarrow \mathbf{Conn0}(x, y) \\ \mathbf{Conn}(x, y) &\Leftarrow \mathbf{Conn0}(y, x) \end{aligned} \tag{31}$$

We now let $\boldsymbol{P} = <\mathbf{Move}, \mathbf{Conn}, \mathbf{Conn0}>$ be the tuple of predicates in the circumscription of $\mathcal{R}(\boldsymbol{P})$.

It is the circumscription of **Conn** and **Conn0** which captures the closed world assumption for the rooms in Figure 2. For example, predicate completion applied to (30) yields:

$$\begin{aligned} \mathbf{Conn0}(x,y) \;\Rightarrow\; & [x = \mathrm{A} \wedge y = \mathrm{B}] \vee \\ & [x = \mathrm{B} \wedge y = \mathrm{C}] \vee \\ & [x = \mathrm{D} \wedge y = \mathrm{A}] \vee \\ & [x = \mathrm{E} \wedge y = \mathrm{B}] \vee \\ & [x = \mathrm{F} \wedge y = \mathrm{C}] \end{aligned} \tag{32}$$

However, to complete the specification of our problem, we need to add the "unique names" assumption for the constants $\mathrm{A}, \ldots, \mathrm{F}$. Formally, we do this by adding the theory *UN* which contains the fact '$c_i \neq c_j$' for each pair of distinct *object constants* $c_i$ and $c_j$. Note that we do not apply the "unique names" assumption to the order constants, nor to the skolem constants generated by expanding abstract actions. Our query problem can now be formulated as follows:

$$\mathit{UN} \cup \mathit{Circ}(\mathcal{R}(\boldsymbol{P}); \boldsymbol{P}) \models \phi \Leftarrow \psi \tag{33}$$

The inductive proof procedure and the decision procedure of Sections 3 and 4 can both be adapted to handle this slight modification of our original query problem.

To understand this formalization, consider the abstract action '$\mathbf{Move}(\mathrm{R2}, \mathrm{F}, \mathrm{D}, \mathrm{u}_1, \mathrm{u}_2)$'. Using rule (29), this action expands to:

$$\begin{aligned} & \mathrm{In}(\mathrm{R2}, \mathrm{F}, \mathrm{u}_1, t_1), \mathbf{Move}(\mathrm{R2}, x_1, \mathrm{D}, t_1, \mathrm{u}_2), \\ & \quad \mathbf{Conn}(\mathrm{F}, x_1), \mathrm{u}_1 < t_1 < \mathrm{u}_2, \end{aligned} \tag{34}$$

where $x_1$ and $t_1$ are skolem variables. Expanding the **Move** action in (34) using rule (28), we obtain:

$$\begin{aligned} & \mathrm{In}(\mathrm{R2}, \mathrm{F}, \mathrm{u}_1, t_1), \\ & \quad \mathrm{In}(\mathrm{R2}, x_1, \mathrm{u}_1, t_2), \mathrm{In}(\mathrm{R2}, \mathrm{D}, t_2, \mathrm{u}_2), \\ & \quad \mathbf{Conn}(\mathrm{F}, x_1), \mathbf{Conn}(x_1, \mathrm{D}), \mathrm{u}_1 < t_1 < t_2 < \mathrm{u}_2. \end{aligned} \tag{35}$$

Now notice that if we were to expand the fact $\mathbf{Conn}(\mathrm{F}, x_1)$ in (35) using **Conn0**(A,B), we would generate the equalities $\mathrm{F} = \mathrm{A}$ and $x_1 = \mathrm{B}$. But *UN* contains the assertion '$\mathrm{F} \neq \mathrm{A}$', and therefore this expansion implies a contradiction. In fact, the only expansion of $\mathbf{Conn}(\mathrm{F}, x_1)$ that does not result in an immediate contradiction is one in which $x_1 = \mathrm{C}$. However, this means that the second **Conn** fact in (35) now becomes $\mathbf{Conn}(\mathrm{C}, \mathrm{D})$. Consistently expanding this fact turns out to be impossible, since every expansion of $\mathbf{Conn}(\mathrm{C}, \mathrm{D})$ generates an equality that contradicts the "unique names" assumption.

In short, because the rules for **Conn0** contain only constants, every expansion of a fact $\mathbf{Conn}(x, y)$ will "ground out" the variables $x$ and $y$ in one of the constants $\mathrm{A}, \ldots, \mathrm{F}$. Thus the consistent expansions of an action $\mathbf{Move}(\mathrm{R}, x_1, x_n, t_1, t_n)$ have the form

$$\begin{aligned} & \mathrm{In}(\mathrm{R}, c_1, t_1, t_2), \\ & \mathrm{In}(\mathrm{R}, c_2, t_2, t_3), \\ & \vdots \\ & \mathrm{In}(\mathrm{R}, c_{n-1}, t_{n-1}, t_n), \\ & t_1 < t_2 < t_3 < \ldots < t_n \end{aligned}$$

where $c_1, c_2, \ldots, c_n$ is a sequence of rooms $\mathrm{A}, \ldots, \mathrm{F}$ such that for each $i$, $c_i$ and $c_{i+1}$ are connected according to Figure 2. For example, the goal $\psi$ for our problem has an expansion in which robot R1 moves through rooms D,A,B,C,F, starting in room D at time '$\mathrm{u}_1$' and switching rooms at times $t_1 < \ldots < t_4$, and robot R2 moves through rooms F,C,B,A,D, starting in room F at time '$\mathrm{u}_1$' and switching rooms at times $t'_1 < \ldots < t'_4$. This expansion is not yet a model, since the constants $t_i$ and $t'_j$ are only partially ordered. When we attempt to topologically sort this partial order, of course, we find that all the sorted models satisfy the condition $\phi$ that we wish to avoid. Hence this particular expansion does not yield a plan for swapping the locations of the two robots while ensuring that they are never in the same room at the same time.

However, the reader can readily verify that the expansion in which R1 moves through rooms D,A,B,E,B,C,F and R2 moves through rooms F,C,B,A,D does have topological sorts that avoid the undesirable condition $\phi$. We need only guarantee that the interval over which R2 is in room B is contained in the interval in which R1 is in room E. We have thus found an expansion of $\psi$ that does not satisfy '$\phi$', i.e., we have found a countermodel to the implication '$\phi \Leftarrow \psi$'. On the other hand, if we altered the connectivity of the rooms by deleting the rule **Conn0**(E,B) from (30), then the planning problem would no longer be solvable. In this case, the formula '$\phi \Leftarrow \psi$' would be entailed by $\mathit{UN} \cup \mathit{Circ}(\mathcal{R}(\boldsymbol{P}); \boldsymbol{P})$.

Since the decision procedure in Section 4 proceeds by attempting to construct a countermodel to the circumscriptive entailment relation in (33), it may readily be modified to output such a countermodel in case this entailment relation fails. This modification yields a procedure that solves the planning problem whenever a solution exists, and reports unsolvability whenever no solution exists. That is, the procedure is not only correct, but it also guarantees termination.

## 6 Discussion

We have presented in this paper a novel technique for reasoning about indefinite actions: We assume that abstract actions are defined over a linear temporal order by a set of Horn clauses, $\mathcal{R}$, and we ask whether a hypothetical implication, $\phi \Leftarrow \psi$, is entailed by the *circumscription* of the defined predicates in $\mathcal{R}$. If $\mathcal{R}$ is nonrecursive, this approach gives us an action language similar to the language proposed by Allen and Koomen [1983] and a set of inferences similar to the abductive inferences analyzed by Eshghi [1988]. However, if $\mathcal{R}$ is recursive, we have a significant increase in expressive power: We can define an action '$R(t_1, t_n)$' which consists of "doing the action $A(t_1, t_2)$ some finite number of times." Although the general problem $Circ(\mathcal{R}(\boldsymbol{P}); \boldsymbol{P}) \models \phi \Leftarrow \psi$ is not in RE, we have developed two techniques for solving this problem in cases that seem to be of practical importance: (1) an inductive proof procedure for linear recursive definitions, which is sound but not complete; and (2) a decision procedure, which works for basic queries and regular rules. We have also identified a natural condition under which our decision procedure has PTIME data complexity.

An alternative approach to the representation of "doing $\alpha$ some finite number of times" is provided by the $\alpha*$ construct of *dynamic logic* [Pratt, 1976; Harel, 1979]. It is interesting to note that our "regular rules" can express the class of "regular events" which corresponds to the action modality of dynamic logic, whereas ordinary temporal logic cannot express this class of events at all [Wolper, 1983]. (It is also interesting to note, in the AI literature, that Rosenschein's formulation of planning problems in dynamic logic [Rosenschein, 1981] omitted the $\alpha*$ construct!) Despite its expressiveness, dynamic logic has been criticized within the AI community for its rigid "change-of-state" semantics [Shoham and Goyal, 1988]. More closely related to our approach is the family of *process logics* proposed by Pratt [Pratt, 1979; Harel *et al.*, 1982], in which modal operators can refer to the total trajectory in the execution of an action.

We have also argued in this paper that our formalism for reasoning about indefinite actions permits an elegant statement of a certain type of planning problem: A countermodel to the hypothetical implication $\phi \Leftarrow \psi$ can be interpreted as a *plan* for "doing $\psi$ and avoiding $\phi$." The close relationship between circumscription and abduction has been studied by Konolige [1992], and our procedure for constructing countermodels could also be viewed as a procedure for generating abductive inferences. To see this, note that an expansion of the predicates in $\psi$ could be viewed as an "explanation" of $\psi$ in terms of the definitions in $\mathcal{R}$. Under this interpretation, the goal $\phi$ plays the role of a negative "integrity constraint" [Eshghi and Kowalski, 1989; Kakas and Mancerella, 1990], i.e., a sentence that must be false in the abductive explanation. (If $n$ integrity constraints $\neg I_1, \ldots, \neg I_n$ must be satisfied, constructing an abductive explanation of $\psi$ corresponds to finding a countermodel for the relation $Circ(\mathcal{R}(\boldsymbol{P}); \boldsymbol{P}) \models \phi \Leftarrow \psi$, where $\phi = I_1 \vee \ldots \vee I_n$.) The main feature that is lacking in these studies of abductive inference, however, is a procedure for determining whether an abductive explanation that satisfies the integrity constraints, i.e., a countermodel, is impossible. This is precisely the point of our inductive proof procedure in Section 3 and our decision procedure in Section 4.

Moreover, most abductive theories of planning [Eshghi, 1988; Missiaen, 1991; Denecker *et al.*, 1992] have adopted a different logical framework from the one presented here, a framework that formalizes the effects of primitive actions on the state of the world. Abductive reasoning is then used within this framework to implement a nonlinear goal-directed regression-style planner. By contrast, our approach assumes that many planning situations come equipped with standard (but abstract) actions that are known to achieve certain goals. The planning problem then reduces to the problem of choosing appropriate expansions of these abstract actions and finding an appropriate way to schedule concurrent events. Of course, this is not the complete AI planning problem. It appears that even common simplified representations of planning, such as the STRIPS formalism, cannot be conveniently encoded using only definite rules. However, ours is not a new view of planning. It dates back to the early work of Tate [1977] and Sacerdoti [1977]. This view seems to have been obscured in most recent *logical* theories of planning.

We have focussed in this paper on reasoning about abstract actions, and have considered plans to be countermodels, i.e., sets of basic actions in which the time points are linearly ordered. However, our formalism is amenable to *nonlinear planning,* in which one deals with basic actions and partially ordered time points. A non-linear plan represents a collection of linear plans, namely those obtained by topologically sorting the partial order. When dealing with non-linear plans, one needs a variety of *query modalities.* For example, the planner needs to be able to calculate whether there is some refinement of the partial order under which the plan is guaranteed to succeed. This is not the case if the query $\phi$ to be avoided holds in all the linear plans corresponding to the non-linear plan. Depending on the precise formulation of queries and actions, this may be a nontrivial problem [Chapman, 1985; Dean and Boddy, 1988]. For our formalism, even if $\psi$ contains no abstract actions, determining the validity of the query $\phi \Leftarrow \psi$ with respect to linear orders requires consideration of all topological sorts of the partial order stated in $\psi$, potentially a very large set. We refer the reader to [Meyden, 1992] for a study of the complexity of this problem, which finds surprisingly high complexities, although certain interesting cases are in PTIME.

To formulate a greater variety of planning problems, we need, in addition, various modalities over actions. We suggest the *deontic modalities* as an appropriate set: **P** (permitted, in the "free choice" sense), **F** (forbidden), **O** (obligatory), and their respective negations. We have studied these modalities in previous work [McCarty, 1983; McCarty, 1986; Meyden, 1990; Meyden, 1991]. In future work, we will show how to combine a system of deontic logic with our present techniques for the representation of indefinite actions, and how to apply such a system in legal domains [McCarty, 1989; Schlobohm and McCarty, 1989].

## References


[Allen and Koomen, 1983] J. F. Allen and J. A. Koomen. Planning using a temporal world model. In *Proceedings, International Joint Conference on Artificial Intelligence*, pages 741–747, 1983.

[Allen, 1984] J.F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–154, 1984.

[Chapman, 1985] D. Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32:333–377, 1985.

[Clark, 1978] K.L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum, 1978.

[Dean and Boddy, 1988] T. Dean and M. Boddy. Reasoning about partially ordered events. *Artificial Intelligence*, 36:375–387, 1988.

[Denecker *et al.*, 1992] M. Denecker, L. Missiaen, and M. Bruynooghe. Temporal reasoning with abductive event calculus. In *Proceedings of the European Conference on Artificial Intelligence*, 1992. to appear.

[Eshghi and Kowalski, 1989] K. Eshghi and R. Kowalski. Abduction compared with negation as failure. In G. Levi and M. Martelli, editors, *Logic Programming: Proceedings of the Sixth International Conference*, pages 234–254, Boston, MA, 1989. MIT Press.

[Eshghi, 1988] K. Eshghi. Abductive planning with event calculus. In R.A. Kowalski and K. Bowen, editors, *Logic Programming: Proceedings of the Fifth International Conference and Symposium*, pages 562–579. MIT Press, 1988.

[Georgeff and Lansky, 1985] M.P. Georgeff and A.L. Lansky. A procedural logic. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 516–523, 1985.

[Harel *et al.*, 1982] D. Harel, D. Kozen, and R. Parikh. Process logic: Expressiveness, decidability and completeness. *Journal of Computer and System Sciences*, 25, 1982.

[Harel, 1979] D. Harel. *First-Order Dynamic Logic*. Springer-Verlag, 1979. LNCS No. 68.

[Kakas and Mancerella, 1990] A.C. Kakas and P. Mancerella. Generalized stable models: A semantics for abduction. In *Proceedings of the European Conference on Artificial Intelligence*, pages 385–391, 1990.

[Konolige, 1992] K. Konolige. Abduction versus closure in causal theories. *Artificial Intelligence*, 53:255–272, 1992.

[Kowalski and Sergot, 1986] R.A. Kowalski and M.J. Sergot. A logic-based calculus of events. *New Generation Computing*, 4(1):67–95, 1986.

[Lifschitz, 1985] V. Lifschitz. Computing circumscription. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 121–127, 1985.

[Loveland, 1991] D.W. Loveland. Near-Horn Prolog and beyond. *Journal of Automated Reasoning*, 7(1):1–26, 1991.

[Manna and Waldinger, 1987] Z. Manna and R. Waldinger. How to clear a block: A theory of plans. *Jounal of Automated Reasoning*, 3:343–377, 1987.

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

[McCarty and Meyden, 1991] L.T. McCarty and R. van der Meyden. Indefinite reasoning with definite rules. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 890–896, 1991.

[McCarty, 1983] L.T. McCarty. Permissions and obligations. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 287–294, 1983.

[McCarty, 1986] L.T. McCarty. Permissions and obligations: An informal introduction. In A.A. Martino and F. Socci Natali, editors, *Automated Analysis of Legal Texts: Logic, Informatics, Law*, pages 307–337. Elsevier North-Holland, 1986. Also available as Rutgers Technical Report LRP-TR-19.

[McCarty, 1988a] L.T. McCarty. Clausal intuitionistic logic. I. Fixed-point semantics. *Journal of Logic Programming*, 5(1):1–31, 1988.

[McCarty, 1988b] L.T. McCarty. Clausal intuitionistic logic. II. Tableau proof procedures. *Journal of Logic Programming*, 5(2):93–132, 1988.

[McCarty, 1989] L.T. McCarty. A language for legal discourse. I. Basic features. In *Proceedings of the Second International Conference on Artificial Intelligence and Law*, pages 180–189. ACM Press, June 1989.

[McCarty, 1992] L.T. McCarty. Computing with prototypes. Technical report, Computer Science Department, Rutgers University, 1992. Submitted for publication.

[McDermott, 1978] D. McDermott. Planning and acting. *Cognitive Science*, 2:71–109, 1978.

[McDermott, 1982] D. McDermott. A temporal logic for reasoning about processes and plans. *Cognitive Science*, 6:101–155, 1982.

[Meyden, 1990] R. van der Meyden. The dynamic logic of permission. In *Proceedings of the Symposium on Logic in Computer Science*, pages 72–78, Philadelphia, 1990.

[Meyden, 1991] R. van der Meyden. A clausal logic for deontic action specification. In *Proceedings of the International Logic Programming Symposium*, pages 221–238, San Diego, 1991.

[Meyden, 1992] R. van der Meyden. The complexity of querying indefinite data about linearly ordered domains. In *Proceedings of the ACM Symposium on Principles of Database Systems*, pages 331–345, 1992.

[Meyden, 1992 forthcoming] R. van der Meyden. *The Complexity of Querying Indefinite Information: Defined relations, Recursion and Linear Order*. PhD thesis, Rutgers University, 1992 (forthcoming).

[Meyden, 1992 in press] R. van der Meyden. Recursively indefinite databases. *Theoretical Computer Science*, 1992 (in press). An earlier version of this paper appears in *ICDT'90: Third International Conference on Database Theory*, Springer LNCS No. 470, S. Abiteboul and P.C. Kanellakis, eds., pp. 364-378 (1990).

[Miller, 1989] D.A. Miller. Lexical scoping as universal quantification. In G. Levi and M. Martelli, editors, *Logic Programming: Proceedings of the Sixth International Conference*, pages 268–283, 1989.

[Missiaen, 1991] L. Missiaen. *Localized Abductive Planning with the Event Calculus*. PhD thesis, Katholieke Universiteit Leuven, 1991.

[Nadathur and Miller, 1988] G. Nadathur and D.A. Miller. An overview of λPROLOG. In *Proceedings, Fifth International Conference and Symposium on Logic Programming*, pages 810–827, 1988.

[Peirce, 1931] C.S. Peirce. *Collected Papers of Charles Sanders Peirce*, volume 6:522-528. Harvard University Press, 1931.

[Pratt, 1976] V. Pratt. Semantical considerations on Floyd-Hoare logic. In *Proceedings, 17th IEEE Symposium on Foundations of Computer Science*, pages 109–121, 1976.

[Pratt, 1979] V. Pratt. Process logic. In *Proceedings, 6th ACM Symposium on Principles of Programming Languages*, pages 93–100, 1979.

[Rosenschein, 1981] S. Rosenschein. Plan synthesis: A logical perspective. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, pages 331–337, 1981.

[Sacerdoti, 1977] E.D. Sacerdoti. *A Structure for Plans and Behaviour*. Elsevier, New York, 1977.

[Schlobohm and McCarty, 1989] D.A. Schlobohm and L.T. McCarty. EPS-II: Estate planning with prototypes. In *Proceedings of the Second International Conference on Artificial Intelligence and Law*, pages 1–10. ACM Press, June 1989.

[Shanahan, 1989] M. Shanahan. Prediction is deduction but explanation is abduction. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1055–1060, 1989.

[Shoham and Goyal, 1988] Y. Shoham and N. Goyal. Temporal reasoning in artificial intelligence. In H. Shrobe, editor, *Exploring Artificial Intelligence*. Morgan Kaufmann, 1988.

[Shoham, 1987] Y. Shoham. *Reasoning about Change*. MIT Press, Boston, 1987.

[Tate, 1977] A. Tate. Generating project networks. In *Proceedings, International Joint Conference on Artificial Intelligence*, pages 888–839, 1977.

[Wolper, 1983] P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56:72–93, 1983.