A Simple Formalization of Actions Using Circumscription

G. Neelakantan Kartha and **Vladimir Lifschitz**
Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712-1188, USA
{ kartha,vl }@cs.utexas.edu

Abstract

We present a simple circumscriptive method for formalizing actions with indirect effects (ramifications), and show that, in several examples, all second-order quantifiers can be eliminated from these formalizations using existing techniques for computing circumscriptions. One of the two symbolic computation methods employed here is a generalization of predicate completion and the other is based on the SCAN algorithm. The simplicity of our new approach to representing actions is due to the use of the formalism of nested abnormality theories.

1 Introduction

Solving the frame problem—the problem of representing succinctly what remains unchanged as a result of performing an action—is one of major challenges in the logical approach to Artificial Intelligence. This problem was one of the motivating factors behind the emergence of several nonmonotonic formalisms in the 1980s. It has led, in particular, to the development of circumscription in [McCarthy, 1980] and [McCarthy, 1986]. Circumscription is a syntactic transformation that expresses a minimality property of models. The idea was to solve the frame problem by postulating that, whenever an action is executed, the "difference" between the two states of the world, before and after the event, is minimal. However, the straightforward formalization based on this idea turned out to be inadequate [Hanks and McDermott, 1987].

In this paper, we present a simple formalization of actions using the framework of *nested abnormality theories* (NATs) [Lifschitz, 1995], a formalism based on circumscription. The main feature of this framework is that the effects of various circumscriptions are confined to the parts of the axiom set called "blocks." As a result, the circumscriptions that solve the frame problem become rather simple, and, in several examples, we will be able to eliminate all their second-order quantifiers using existing techniques for computing circumscriptions. These examples involve actions with indirect effects and, in one case, these effects are nondeterministic. One of the two methods for computing circumscriptions employed here is a generalization of predicate completion [Lifschitz, 1993] and the other is based on the SCAN algorithm [Gabbay and Ohlbach, 1992].

The idea behind the approach to the frame problem presented here is closest to that of Winslett [1988]. The main difference is that Winslett's formalization is in terms of theory update, and ours includes the full expressive power of the situation calculus. Combining the ideas of [Winslett, 1988] with the situation calculus is achieved by the use of NATs.

Another closely related direction of research is described in [Lin and Shoham, 1991] and [Lin and Reiter, 1994]. Central to the formalization presented there is a minimality condition formulated in terms of models. To obtain meaningful conclusions with their formalization, Lin, Shoham and Reiter need to impose certain consistency conditions and on include certain "tree axioms" (that impose a tree structure to the universe of situations). In contrast, the use of circumscription described in this paper allows us to dispense with the consistency conditions and the tree axioms.

Also, this paper differs from earlier work in that we investigate the applicability of symbolic methods to computing the circumscriptions involved in the solution to the frame problem—an issue not considered by Winslett or by Lin, Shoham and Reiter.

The rest of the paper is organized as follows. In the next two sections, we review the notion of a causal theory introduced in [Lin and Reiter, 1994] and give a few examples. Section 4 introduces the new formalization. In Section 5, we illustrate via examples how the effect of circumscriptions can be computed by syntactic manipulations and using SCAN. In Section 6, we relate this paper to action languages [Gelfond and Lifschitz, 1993] and indicate directions for future work.

For the terminology and notation related to circumscription, the reader is referred to [Lifschitz, 1993].

2 Causal Theories

For clarity, we will present the new formalization in the context of a simple class of theories called "causal." This class of theories is essentially the same as that defined in [Lin and Reiter, 1994].

The formalism is based on the situation calculus [McCarthy and Hayes, 1969]. Consider a first-order language with object variables for situations and actions, and possibly variables of other sorts. In this section, by s we will denote a situation variable, by a an action variable, and by x, y tuples of distinct variables of other sorts. The nonlogical constants of the language are

- binary function constant *Result*; $Result(a, s)$ is the situation obtained by performing action a in situation s;
- binary predicate constant *Poss*; $Poss(a, s)$ expresses that it is possible to execute a in situation s;
- function constants (some of them possibly of arity 0) called *action symbols*; for an action symbol A, $A(x)$ is an action term;
- predicate constants called *fluent symbols*; for a fluent symbol F, $F(x, s)$ is an atomic formula.

A language of this kind will be called a *causal language*.

Two groups of axioms will be allowed in a causal theory—"effect axioms" and "ramification constraints." In order to describe the syntactic form that these axioms may have, we need the following definition. A formula Φ is a *simple state formula* if every occurrence of a situation term in Φ is an occurrence of the same variable s as the last argument of a fluent symbol. Clearly, S_0, *Result* and *Poss* cannot occur in a simple state formula.

A set of axioms in a causal language is a *causal theory* if it consists of

- some *effect axioms* of the forms

  $$\begin{array}{l} Poss(a, s) \supset [\Psi(x, a, s) \supset F(x, Result(a, s))], \\ Poss(a, s) \supset [\Psi(x, a, s) \supset \neg F(x, Result(a, s))], \end{array} \tag{1}$$

  where $\Psi(x, a, s)$ is a simple state formula,
- some *ramification constraints*, that are assumed to be simple state formulas that do not contain action terms.

3 Examples

We will now illustrate the definitions introduced so far with a few examples. The first example is reproduced from [Lin and Reiter, 1994].

Example 1. Consider a blocks world domain in which the only actions possible are painting blocks with different colours. To describe this domain, we first introduce an action term $paint(x, y)$ that stands for the action of painting block x with colour y. The following effect axiom describes what this action does:

$$Poss(paint(x, y), s) \supset colour(x, y, Result(paint(x, y), s)).$$

Note that this axiom can be rewritten as

$$Poss(a, s) \supset [a = paint(x, y) \supset colour(x, y, Result(a, s))]$$

so that it will have the form of the effect axioms given in (1).

The only ramification constraint that we have for this domain is that a block can have just one colour. This constraint is expressed by the simple state formula

$$colour(x, y_1, s) \wedge colour(x, y_2, s) \supset y_1 = y_2. \tag{2}$$

Our intention is that, in view of the ramification constraint, the action of painting a block with a new colour should have the indirect effect of making the old colour disappear.

Example 2. The "murder mystery" from [Baker, 1991] can be formalized by the effect axioms

$$\begin{array}{l} Poss(Load, s) \supset Loaded(Result(Load, s)), \\ Poss(Shoot, s) \supset \\ \qquad [Loaded(s) \supset \neg Alive(Result(Shoot, s))], \\ Poss(Shoot, s) \supset \neg Loaded(Result(Shoot, s)) \end{array}$$

and the constraint

$$Walking(s) \supset Alive(s).$$

Example 3. Consider a table divided into three sectors F, G and H. A block is always in exactly one of these three locations. There is an action A, which, if performed when the block is in location F, moves it out of that location. Hence, after the action is performed, the block is in location G or H, but we do not know which. Thus, this is a domain where the indirect effects are nondeterministic.

To represent this domain as a causal theory, we introduce three fluents F, G and H. The only effect axiom is

$$Poss(A, s) \supset [F(s) \supset \neg F(Result(A, s)].$$

The constraints are

$$\begin{array}{l} F(s) \vee G(s) \vee H(s), \\ \neg F(s) \vee \neg G(s), \\ \neg F(s) \vee \neg H(s), \\ \neg G(s) \vee \neg H(s). \end{array}$$

4 Turning a Causal Theory into an Abnormality Theory

In nested abnormality theories, as defined in [Lifschitz, 1995], parts of the axiom set can be grouped into "blocks" of the form

$$\{C_1, \ldots, C_m : \Phi_1, \ldots, \Phi_n\},$$

where $C_1, \ldots, C_m$ are function and/or predicate symbols (said to be "described" by the block) and $\Phi_1, \ldots, \Phi_n$ are formulas. Typically, some of these formulas contain the predicate constant Ab. Such a block corresponds to the circumscription of Ab in $\Phi_1 \wedge \ldots \wedge \Phi_n$ with $C_1, \ldots, C_m$ varied:

$$\mathbf{CIRC}[\Phi_1 \wedge \ldots \wedge \Phi_n; Ab; C_1, \ldots, C_m].$$

There can be several such blocks in the theory. Moreover, blocks can be "nested" in the sense that each Φ_i can be itself a block. This possibility corresponds roughly to the use of priorities in traditional applications of circumscription. In this paper, we discuss abnormality theories with a particularly simple structure—the circumscription operator is applied in them only once, so that no nesting of circumscriptions is possible. The reader familiar with the idea of circumscription will find it easy to understand these examples without a detailed review of the general formalism.

Let us assume that the signature and the axiom set of the given causal theory T are finite. We will denote the set of fluent symbols of T by $\mathbf{F}$, the set of its effect propositions by $\mathbf{E}$ and the set of its ramification constraints by $\mathbf{C}$. The language of the abnormality theory T_{ab} corresponding to T includes, in addition to the variables of the sorts available in T, variables for "aspects." Aspects will serve as arguments of the abnormality predicate Ab. This device, proposed in [McCarthy, 1986], allows us to distinguish between different kinds of abnormality: an object abnormal in one "aspect" can be normal in another. For every fluent symbol F of T, we will need a new function symbol Asp_F.

For every fluent symbol F, T_{ab} will include the axiom

$$F_R(x, a, s) \equiv F(x, Result(a, s)), \tag{3}$$

where F_R ("F-Result") is a new predicate constant. These axioms can be viewed as explicit definitions of the constants F_R. Using these constants, the effect axioms (1) can be rewritten as

$$Poss(a, s) \supset [\Psi(x, a, s) \supset F_R(x, a, s)], \tag{4}$$

$$Poss(a, s) \supset [\Psi(x, a, s) \supset \neg F_R(x, a, s)]. \tag{5}$$

The set of formulas obtained in this way from the effect axioms in $\mathbf{E}$ will be denoted by $\mathbf{E}_R$.

The "commonsense law of inertia" can be expressed in this notation by the formulas

$$\neg Ab(Asp_F(x), a, s) \wedge Poss(a, s) \supset (F_R(x, a, s) \equiv F(x, s))$$

for all $F \in \mathbf{F}$, which formalize the idea of minimal change. We will denote this set of formulas by $\mathbf{I}$.

Each ramification constraint will have a counterpart, obtained from it by replacing each atomic part $F(t, s)$ (where t is a tuple of terms) with $F_R(t, a, s)$ for a fixed action variable a. For instance, the counterpart of the ramification constraint (2) is

$$colour_R(x, y_1, a, s) \wedge colour_R(x, y_2, a, s) \supset y_1 = y_2.$$

The set of formulas obtained in this way from the constraints in $\mathbf{C}$ will be denoted by $\mathbf{C}_R$.

We also introduce *unique names axioms* for actions and aspects. They are the formulas

$$A(x) \neq A'(y),$$

$$A(x) = A(y) \supset x = y,$$

$$Asp_F(x) \neq Asp_{F'}(y),$$

$$Asp_F(x) = Asp_F(y) \supset x = y$$

for arbitrary pairs of distinct action symbols A, A' and for arbitrary pairs of distinct fluent symbols F, F'. We denote this set of formulas by UNA.

The axioms of the abnormality theory T_{ab} are:

$$\begin{array}{ll}
UNA, & \\
\mathbf{C}, & \\
F_R(x, a, s) \equiv F(x, Result(a, s)) & \quad (F \in \mathbf{F}), \\
\{\mathbf{F}_R : & \\
\quad \mathbf{I}, & \\
\quad \mathbf{E}_R, & \\
\quad \mathbf{C}_R & \\
\}. &
\end{array}$$

Here $\mathbf{F}_R$ stands for the list of all predicates F_R.

Note the use of $\mathbf{F}_R$ instead of *Result* in the range of circumscription. The intuition behind this style of describing actions can roughly be explained as follows. When we use circumscription to determine the effect of an action a on a fluent in a situation s, the only two situations that are of interest are the situation s and the situation obtained by performing the action a in s—for instance, we do not want to consider the sequence of actions that lead to situation s nor do we wish to consider what happens afterward. The new formalization achieves this by "removing" the *Result* function from the range of circumscription. In this part of the axiom set, we accept the local "theory update" view, as in Winslett's work.

According to the semantics of nested abnormality theories [Lifschitz, 1995], the block at the end of the axiom set stands for the axiom formed as follows. Let Φ be the universal closure of the conjunction of all three groups of formulas included in the block. Denote the circumscription

$$\mathbf{CIRC}[\Phi; Ab; \mathbf{F}_R]$$

by $C(Ab)$. The axiom represented by the block is the formula $\exists ab C(ab)$, where ab is a predicate variable that takes the same arguments as Ab. The last step—replacing the predicate Ab by an existentially quantified variable—is motivated by the fact that, in an abnormality theory, Ab is an auxiliary constant whose value can be safely "forgotten."

For instance, the nested abnormality theory T^c_{ab} corresponding to Example 1 has the following axioms:

$$\begin{array}{l}
(i) \;\; paint(x_1, y_1) = paint(x_2, y_2) \supset \\
\qquad (x_1 = x_2 \wedge y_1 = y_2), \\
(ii) \;\; Asp_{colour}(x_1, y_1) = Asp_{colour}(x_2, y_2) \supset \\
\qquad (x_1 = x_2 \wedge y_1 = y_2), \\
(iii) \; colour(x, y_1, s) \wedge colour(x, y_2, s) \supset y_1 = y_2, \\
(iv) \;\; colour_R(x, y, a, s) \equiv colour(x, y, Result(a, s)), \\
\{colour_R : \\
\quad (v) \;\; \neg Ab(Asp_{colour}(x, y), a, s) \wedge Poss(a, s) \supset \\
\qquad colour_R(x, y, a, s) \equiv colour(x, y, s), \\
\quad (vi) \;\; Poss(a, s) \supset \\
\qquad\quad [a = paint(x, y) \supset colour_R(x, y, a, s)], \\
\quad (vii) \; colour_R(x, y_1, a, s) \wedge colour_R(x, y_2, a, s) \supset \\
\qquad\quad y_1 = y_2 \\
\}.
\end{array}$$

5 Computing Ramifications

In this section, we illustrate via examples how existing methods for computing circumscriptions can be employed in conjunction with the formalization presented above.

5.1 Completion

In many case, it is possible to compute the effects of actions using the syntactic methods for computing circumscriptions from [Lifschitz, 1993]. As an illustration, we will consider computing the circumscription from Example 1 in some detail.

We will have occasion to use the following propositions from [Lifschitz, 1993].

Lemma 1. *The circumscription* $\mathbf{CIRC}[A(P,Z);P;Z]$ *is equivalent to*

$$A(P,Z) \wedge \mathbf{CIRC}[\exists z A(P,z);P].$$

Lemma 2. *If* $A(x)$ *does not contain* p *and* $B(p)$ *is negative relative to* p *then*

$$\exists p[\forall x(A(x) \supset p(x)) \wedge B(p)]$$

is equivalent to $B(\lambda x A(x))$.

Lemma 3. *If* $F(x)$ *does not contain* P, *then the circumscription*

$$\mathbf{CIRC}[\forall x(F(x) \supset P(x));P]$$

is equivalent to $\forall x(F(x) \equiv P(x))$.

Lemma 3 shows that circumscription can be sometimes evaluated by a process similar to the predicate completion algorithm from [Clark, 1978].

Let Φ be the conjunction of the universal closures of formulas (v), (vi) and (vii) from the nested abnormality theory T^c_{ab} (Section 4). By the definition of the semantics of nested abnormality theories, the block at the end of T^c_{ab} stands for the formula $\exists ab\ C(ab)$ where $C(Ab)$ stands for $\mathbf{CIRC}[\Phi; Ab; colour_R]$. By Lemma 1, this circumscription is equivalent to the conjunction of Φ and $\mathbf{CIRC}[\exists colour_R\ \Phi; Ab]$.[1] Using Lemma 2, it is easy to eliminate the second-order quantifier in the formula $\exists colour_R\ \Phi$. After a few simplification steps, we can use Lemma 3 to show that, in the presence of axioms (i)–(iv), $C(Ab)$ is equivalent to the conjunction of Φ and the following explicit definition for Ab:

$$\begin{array}{l} Ab(z,a,s) \equiv \\ Poss(a,s) \wedge \exists xy[z = Asp_{colour}(x,y) \wedge \\ \quad ((\neg colour(x,y,s) \wedge a = paint(x,y)) \vee \\ \quad (colour(x,y,s) \wedge \exists\, y'(a = paint(x,y') \wedge y \neq y')))]. \end{array}$$

It follows that T^c_{ab} is equivalent to the conjunction of formulas (i)–(iv), (vi), (vii) and the result of substituting this expression for Ab in (v). The result of this substitution is essentially the conjunction of all "frame axioms" that are needed in conjunction with the causal theory of Example 1. We see that syntactic methods of computing circumscriptions can be employed to generate the necessary frame axioms.

The effect of the circumscriptions in Examples 2 and 3 can be computed in a similar way.

5.2 SCAN

SCAN [Gabbay and Ohlbach, 1992] is an algorithm for eliminating second-order quantifiers over predicate variables $P_1, \ldots, P_n$ in formulas of the form $\exists P_1 \ldots P_n\ \Phi$, where Φ is a first-order formula. If SCAN terminates, then the resulting formula is equivalent to the original formula.

Recall that $\mathbf{CIRC}[A;P;Z]$, the circumscription of the predicate P in formula A with the functions and/or predicates in the tuple Z varied, is the sentence

$$A(P,Z) \wedge \neg\exists pz[A(p,z) \wedge p < P].$$

Here p is a predicate variable of the same arity as P, z stands for a tuple of variables which matches the tuple Z in arities and the sorts of arguments, and $p < P$ stands for $(p \leq P) \wedge \neg(p = P)$.

Hence to compute circumscriptions using SCAN, we input the formula $\exists pz[A(p,z) \wedge p < P]$ to it, negate the resulting formula and take its conjunction with the formula $A(P,Z)$. Note that if z includes function constants, SCAN cannot be applied. For instance, since Baker's circumscriptive approach [Baker, 1991] involves varying the *Result* function, it cannot be used directly with SCAN.

An implementation of SCAN has been developed by Engel and Ohlbach[2]. Consider the application of this system to computing the circumscriptions from Section 4.

In Example 1, the following set of formulas is taken as input to SCAN; the variables p and R are eliminated.

```
(all x all y all xa all xs (
  (-p(asp(x,y),xa,xs) &  Poss(xa,xs))  ->
        (c(x,y,xs) <-> R(x,y,xa,xs)))).
(all x all y all xa all xs  (
  (Poss(xa,xs) &  (xa = paint(x,y)))   ->
                         R(x,y,xa,xs))).
(all x all y1  all y2 all xa all xs (
      ((R(x,y1,xa,xs) & R(x,y2,xa,xs)) ->
                              (y1 = y2))).
(all xf all xa all xs (p(xf,xa,xs)     ->
                         Ab(xf,xa,xs))).
(exists xf exists xa exists xs (
        -p(xf,xa,xs) &   Ab(xf,xa,xs))).
(all x1 all y1 all x2  all y2  (
        (paint(x1,y1) =  paint(x2,y2)) ->
              ((x1 = x2) & (y1 = y2)))).
(all x1 all y1 all x2  all y2  (
           (asp(x1,y1) =  asp(x2,y2))  ->
              ((x1 = x2) & (y1 = y2)))).
(all x all y1  all y2  all xs (
             (c(x,y1,xs) & c(x,y2,xs)) ->
                              (y1 = y2))).
(all x (x = x)).
```

In the above formulas, `R` stands for $colour_R$ and `c` stands for *colour*. The first three formulas are direct encodings of (v), (vi) and (vii). The next two formulas express that $p < Ab$. The next four are not a part of the circumscription formula; they are included to help SCAN simplify its output. Of these, the first three are formulas (i), (ii) and (iii) of the NAT.

SCAN produces the following 14 clauses as its output (in 1.6 seconds on a Sun 4).

```
1   paint(x,y)!=paint(z,u)|x=z.
2   paint(x,y)!=paint(z,u)|y=u.
3   asp(x,y)!=asp(z,u)|x=z.
4   asp(x,y)!=asp(z,u)|y=u.
5  -c(x,y,z)| -c(x,u,z)|y=u.
```

[1] To simplify notation, we use $colour_R$ both as a predicate constant and predicate variable.

[2] For more information on the implementation, see http://www.mpi-sb.mpg.de/guide/staff/ohlbach/scan/scan.html

```
6   Ab($c3,$c2,$c1).
7   x=x.
8  -Poss(paint(x,y),z)|c(x,y,z)|
    Ab(asp(x,y),paint(x,y),z).
9  -Poss(paint(x,y),$c1)| -Poss($c2,$c1)|
    c(x,y,$c1)|$c3!=asp(x,y)|$c2!=paint(x,y).
10 -Poss(paint(x,y),z)| -Poss(paint(x,u),z)|
    y=u|paint(x,u)!=paint(x,y).
11 -Poss(paint(x,y),z)| -c(x,u,z)|
    Ab(asp(x,u),paint(x,y),z)|y=u.
12 -Poss(paint(x,y),z)| -c(x,u,z)|
    Ab(asp(x,u),paint(x,y),z)|u=y.
13 -Poss($c2,$c1)| -c(x,y,$c1)|$c3!=asp(x,y)|
   -Poss(paint(x,z),$c1)|z=y|paint(x,z)!=$c2.
14 -Poss($c2,$c1)| -c(x,y,$c1)|$c3!=asp(x,y)|
   -Poss(paint(x,z),$c1)|y=z|paint(x,z)!=$c2.
```

In the set of clauses above, the expressions beginning with `$` are Skolem constants introduced by SCAN in the process of converting the input into clauses, and the symbol `|` stands for $\vee$.

By unskolemizing (that is, appending an existential quantifier to) this set of clauses, negating them, conjoining them with the axioms outside the block and simplifying them, we can get an explicit definition of Ab identical to the formula obtained in Section 5.1, and consequently the frame axioms.

Some of this process of simplification of the set of clauses and elimination of Ab can be automated. For instance, to eliminate the redundant clauses, SCAN can be directed to try to derive each newly generated clause from the others for a fixed amount of time. In addition, SCAN can eliminate Ab from the simplified set of clauses.

The causal theory corresponding to Example 2 is

$$
\begin{array}{l}
UNA,\\
Walking(s) \supset Alive(s),\\
Walking_R(a,s) \equiv Walking(Result(a,s)),\\
Alive_R(a,s) \equiv Alive(Result(a,s)),\\
Loaded_R(a,s) \equiv Loaded(Result(a,s)),\\
\{Walking_R, Alive_R, Loaded_R :\\
\quad Poss(a,s) \wedge \neg Ab(Aspect_{Walking}, a, s) \supset\\
\qquad\qquad Walking_R(a,s) \equiv Walking(s),\\
\quad Poss(a,s) \wedge \neg Ab(Aspect_{Alive}, a, s) \supset\\
\qquad\qquad Alive_R(a,s) \equiv Alive(s),\\
\quad Poss(a,s) \wedge \neg Ab(Aspect_{Loaded}, a, s) \supset\\
\qquad\qquad Loaded_R(a,s) \equiv Loaded(s),\\
\quad Poss(Load,s) \supset Loaded_R(Load,s),\\
\quad Poss(Shoot,s) \wedge Loaded(s) \supset \neg Alive_R(Shoot,s),\\
\quad Poss(Shoot,s) \supset \neg Loaded_R(Shoot,s),\\
\quad Walking_R(a,s) \supset Alive_R(a,s).\\
\}.
\end{array}
$$

When the corresponding formulas were input to SCAN, it produced 16 clauses in 1.8 seconds. Here again, we can further eliminate Ab and produce the frame axioms. For this example, SCAN helped in the automation of this process even better than in the previous case. For instance, consider the following clauses among the ones generated by SCAN:

```
x=L|Lo(y)| -Poss(x,y)| -Lor(x,y).
Lo(x)| -Poss(y,x)|Alr(y,x)| -Al(x).
```

In these clauses, `L` stands for *Load*, `Lo` for *Loaded*, `Lor` for $Loaded_R$, `Al` for *Alive* and `Alr` for $Alive_R$. Rewriting the first one as

$$Poss(a,s) \wedge a \neq Load \wedge \neg Loaded(s) \supset \neg Loaded(Result(a,s))$$

makes it clear that this a negative frame axiom for *Loaded*. Similarly, the second one can be rewritten as

$$Poss(a,s) \wedge \neg Loaded(s) \wedge Alive(s) \supset Alive(Result(a,s)),$$

a positive frame axiom for *Alive*. Frame axioms of a similar and easy to comprehend form were generated for the other fluents also.

In Example 3, SCAN was used to generate the frame axioms in a similar way.

6 Discussion

In [Giunchiglia *et al.*, 1995], the method for representing actions from Section 4 is used to define a translation from the high-level action language $\mathcal{AR}$ to NATs. In some ways, $\mathcal{AR}$ is more expressive than the language of causal theories from Section 2; for instance, it includes nonpropositional fluents. In other ways, it is less expressive—in $\mathcal{AR}$, actions have no parameters, such as x and y in $paint(x,y)$. The main difference between $\mathcal{AR}$ and the language of causal theories, however, is that the former has a semantics, based on the notion of a transition function. The translation from [Giunchiglia *et al.*, 1995] is complete relative to this semantics.

We have presented evidence that this approach to formalizing actions leads, in some cases, to the circumscriptions that can be evaluated using the predicate completion method or the SCAN algorithm. One of these examples (Example 3) involves a "ternary state constraint" in the sense of Section 3.1 of [Pinto, 1994] and thus is not amenable to the methods developed there for the generation of successor state axioms in the case when all constraints are binary.

Our plans for the future include the development of an experimental program that uses SCAN for the completely automated generation of frame axioms. For this purpose, the simplification procedure used currently in SCAN will need to be enhanced.

The set of formulas produced by SCAN is equivalent to the formula input to SCAN, *if the algorithm terminates*. Hence, even if we know that a given second-order formula is equivalent to some first-order formula, there is no guarantee that SCAN will ever find it. It would be interesting to isolate classes of action domains for which it can be proved that SCAN terminates when applied to the corresponding circumscriptive theory.

Acknowledgements

We would like to thank Enrico Giunchiglia, Fangzhen Lin, Norman McCain, Ray Reiter and Hudson Turner for useful discussions on the topic of reasoning about actions. Special thanks to Thorsten Engel and Hans Juergen Ohlbach for making the implementation of SCAN available to us. This research was supported in part by National Science Foundation under grant IRI-9306751.

References

[Baker, 1991] Andrew Baker. Nonmonotonic reasoning in the framework of situation calculus. *Artificial Intelligence*, 49:5–23, 1991.

[Clark, 1978] Keith Clark. Negation as failure. In Herve Gallaire and Jack Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, New York, 1978.

[Gabbay and Ohlbach, 1992] Dov Gabbay and Hans J. Ohlbach. Quantifier elimination in second-order predicate logic. In Bernhard Nebel, Charles Rich, and William Swartout, editors, *Proc. of the Third Int'l Conf. on Principles of Knowledge Representation and Reasoning*, 1992.

[Gelfond and Lifschitz, 1993] Michael Gelfond and Vladimir Lifschitz. Representing action and change by logic programs. *Journal of Logic Programming*, 17:301–322, 1993.

[Giunchiglia *et al.*, 1995] Enrico Giunchiglia, G. Neelakantan Kartha, and Vladimir Lifschitz. Actions with indirect effects (extended abstract). In *Working Notes of the Symposium on Extending Theories of Actions*, 1995.

[Hanks and McDermott, 1987] Steve Hanks and Drew McDermott. Nonmonotonic logic and temporal projection. *Artificial Intelligence*, 33(3):379–412, 1987.

[Lifschitz, 1993] Vladimir Lifschitz. Circumscription. In D.M. Gabbay, C.J. Hogger, and J.A. Robinson, editors, *The Handbook of Logic in AI and Logic Programming*, volume 3, pages 297–352. Oxford University Press, 1993.

[Lifschitz, 1995] Vladimir Lifschitz. Nested abnormality theories. *Artificial Intelligence*, 1995. To appear.

[Lin and Reiter, 1994] Fangzhen Lin and Raymond Reiter. State constraints revisited. *Journal of Logic and Computation, Special Issue on Actions and Processes*, 4(5):655–678, 1994.

[Lin and Shoham, 1991] Fangzhen Lin and Yoav Shoham. Provably correct theories of action. In *Proc. of the Ninth National Conference of Artificial Intelligence*, pages 349–354, 1991.

[McCarthy and Hayes, 1969] John McCarthy and Patrick Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 4, pages 463–502. Edinburgh University Press, Edinburgh, 1969.

[McCarthy, 1980] John McCarthy. Circumscription—a form of non-monotonic reasoning. *Artificial Intelligence*, 13(1–2):27–39, 1980.

[McCarthy, 1986] John McCarthy. Applications of circumscription to formalizing common sense knowledge. *Artificial Intelligence*, 28(1):89–116, 1986.

[Pinto, 1994] Javier A. Pinto. Temporal reasoning in the situation calculus. Ph. D. thesis, University of Toronto (available as Technical Report KRR-TR-94-1), 1994.

[Winslett, 1988] Marianne Winslett. Reasoning about action using a possible models approach. In *Proc. of AAAI-88*, pages 89–93, 1988.