On the Range of Applicability of Baker's Approach to the Frame Problem

G. Neelakantan Kartha
Honeywell Technology Center
3600 Technology Drive
Minneapolis, Minnesota 55418
kartha@src.honeywell.com

Abstract

We investigate the range of applicability of Baker's approach to the frame problem using an action language. We show that for temporal projection and deterministic domains, Baker's approach gives the intuitively expected results.

Introduction

Baker's circumscriptive approach to the frame problem (Baker 1991) has been employed in several studies of reasoning about action (for instance, (Lifschitz 1991; Kartha 1993; Shanahan 1995)) because of its simplicity and its ability to deal with domain constraints. However, it has recently been pointed out (Crawford & Etherington 1992; Kartha 1994) that Baker's approach may sometimes lead to unintuitive conclusions. This paper addresses the following question: Under what circumstances does Baker's approach give intuitively correct conclusions?

Of course, the question as phrased above is imprecise—we have not characterized what we mean by the "intuitively correct" conclusions obtainable from the action domain under consideration. To do this, we follow the approach suggested in (Gelfond & Lifschitz 1993) of defining the syntax and semantics of an action language and identifying the "intuitively correct" conclusions with those entailed by the encoding of the action domain in the language. So the question we address becomes: Can we identify classes of domains in an action language that can be faithfully encoded using Baker's approach?

We answer this question in the affirmative as follows. First we define the syntax and semantics of an action language $\mathcal{AR}^-$. Then we identify two classes of domains expressible in this language, present a translation that employs Baker's approach and prove that the translation is sound and complete for these classes. As will be seen, these two classes correspond to deterministic and temporal projection action domains.

The rest of the paper is organized as follows. We first define the action language $\mathcal{AR}^-$ and present a brief review of Baker's approach. We then give a translation from $\mathcal{AR}^-$ and characterize the two classes of domains for which the translation is sound and complete. We conclude by discussing related work and the significance of the results presented in this paper.

The Language $\mathcal{AR}^-$

In this section, we define the syntax and semantics of an action language $\mathcal{AR}^-$. The language $\mathcal{AR}^-$ is a subset of the language $\mathcal{AR}_0$ introduced in (Kartha & Lifschitz 1994)—the latter has one additional kind of propositions called release propositions.

Syntax of $\mathcal{AR}^-$

Formulae and Propositions To be precise, $\mathcal{AR}^-$ is not a single language, but rather a family of languages. A particular language in this group is characterized by

- a nonempty set of symbols, that are called *fluent names*, or *fluents*,
- a subset of fluent names, that is called the *frame*,
- a nonempty set of symbols, that are called *action names*, or *actions*.

A *formula* is a propositional combination of fluents.

There are three types of *propositions* in $\mathcal{AR}^-$—value propositions, effect propositions and constraints.

A *value proposition* is an expression of the form

$$C \textbf{ after } \overline{A}, \tag{1}$$

where C is a formula, and $\overline{A}$ is a string of actions. Informally, (1) asserts that C holds after the sequence of actions $\overline{A}$ is performed in the initial situation. For instance,

$$\mathit{OnRedBus} \textbf{ after } \mathit{BuyTicket}; \mathit{GetOnBoard}$$

is a value proposition, where *OnRedBus* is a fluent and *BuyTicket* and *GetOnBoard* are actions.

An *effect proposition* is an expression of the form

$$A \textbf{ causes } C \textbf{ if } P, \tag{2}$$

where A is an action, and C and P are formulae. Intuitively, (2) asserts that A, if executed in a situation in which the precondition P is true, makes C true. For instance,

$$\mathit{BuyTicket} \textbf{ causes } \mathit{HasTicket} \textbf{ if } \neg \mathit{HasTicket}$$

is an effect proposition, where *BuyTicket* is an action, and *HasTicket* is a fluent.

Finally, a *constraint* is a proposition of the form

$$\textbf{always } C, \tag{3}$$

where C is a formula. Intuitively, (3) asserts that C holds in all possible situations. For instance,

$$\textbf{always } \mathit{OnRedBus} \supset \mathit{HasTicket}$$

is a constraint. We will sometimes identify a constraint (3) with the formula C.

A *domain description*, or *domain*, is a set of propositions.

Notational Conventions In formulae, we will omit some parentheses, as customary in classical logic. If $\overline{A}$ in a value proposition (1) is empty, we will write this proposition as

$$\textbf{initially } C.$$

Otherwise, the members of $\overline{A}$ will be separated by semicolons. An effect proposition (2) will be written as

$$A \textbf{ causes } C$$

if P is *True*.

Examples

In this section, we give two examples to illustrate how we can describe action domains in $\mathcal{AR}^-$.

Example 1. We will first show how to formalize an extension of the "Yale Shooting" domain (Hanks & McDermott 1987) in $\mathcal{AR}^-$, since this example is likely to be familiar to the reader. The formalization is as follows.

$$\begin{array}{l}\textbf{always } \mathit{Walking} \supset \mathit{Alive},\\ \textbf{initially } \neg \mathit{Loaded},\\ \textbf{initially } \mathit{Alive},\\ \mathit{Load} \textbf{ causes } \mathit{Loaded},\\ \mathit{Shoot} \textbf{ causes } \neg \mathit{Alive} \textbf{ if } \mathit{Loaded},\\ \mathit{Shoot} \textbf{ causes } \neg \mathit{Loaded}.\end{array} \tag{4}$$

All fluents belong to the frame. Once the semantics of $\mathcal{AR}^-$ is given in the next section, we will be able show, for instance that the domain description (4) entails, for instance, the value proposition

$$\neg \mathit{Walking} \textbf{ after } \mathit{Load}; \mathit{Wait}; \mathit{Shoot}.$$

Note that the fluent *Walking* does not occur in the effect propositions. Thus any change in the value of *Walking* is an indirect effect, or ramification, of the actions that have been performed.

Example 2. We will now formalize the "Two Bus" example from (Kartha 1994). The task here is two formalize the actions of a commuter who has to take a bus to work. There are two buses that take him to work, one red and the other yellow, and the commuter catches the first bus that comes by. The commuter can board a bus only if he has a ticket. On a particular day, he does not have a ticket initially, and after purchasing the ticket and getting on board, he is on the red bus.

This domain can be formalized as follows.

$$\begin{array}{l}\textbf{always } \mathit{OnRedBus} \supset \mathit{HasTicket},\\ \textbf{always } \mathit{OnYellowBus} \supset \mathit{HasTicket},\\ \textbf{always } \neg(\mathit{OnYellowBus} \wedge \mathit{OnRedBus}),\\ \textbf{initially } \neg \mathit{HasTicket},\\ \mathit{OnRedBus} \textbf{ after } \mathit{BuyTicket}; \mathit{GetOnBoard},\\ \mathit{BuyTicket} \textbf{ causes } \mathit{HasTicket} \textbf{ if } \neg \mathit{HasTicket},\\ \mathit{GetOnBoard} \textbf{ causes } \mathit{OnRedBus} \vee \mathit{OnYellowBus} \textbf{ if}\\ \qquad \mathit{HasTicket} \wedge \neg \mathit{OnRedBus} \wedge \neg \mathit{OnYellowBus}.\end{array} \tag{5}$$

Again, all fluents belong to the frame. Note that in this domain, the action *GetOnBoard* is nondeterministic.

Semantics of $\mathcal{AR}^-$

States and Transition Functions In this section, we consider truth-valued functions on the set of fluents called *valuations*. Such a function can be extended to arbitrary formulae according to the truth tables of propositional logic.

We say that a valuation σ is a *state* of a domain D if it maps every constraint in D to T ("true").

For instance, the domain (4) has 6 states; they are the truth-valued functions on

$$\{\mathit{Loaded}, \mathit{Alive}, \mathit{Walking}\}$$

that make the formula $\mathit{Walking} \supset \mathit{Alive}$ true.

The semantics of $\mathcal{AR}^-$ shows how the effect propositions of D define a nondeterministic transition system with this set of states, whose input symbols are actions. We will describe this transition system by a function *Res* that maps an action and a state to a set of states. The elements of $\mathit{Res}(A, \sigma)$ are, intuitively, the states that differ from σ as dictated by the effect propositions for A, but, at the same time, differ from σ as little as possible. In determining the "difference" between σ and the elements of $\mathit{Res}(A, \sigma)$, we will consider only the frame fluents.

As a preliminary step, define $\mathit{Res}_0(A, \sigma)$ to be the set of states σ' such that, for each effect proposition A **causes** C **if** P in D, $\sigma'(C) = \mathsf{T}$ whenever $\sigma(P) = \mathsf{T}$. The set $\mathit{Res}(A, \sigma)$ will be defined as the subset of $\mathit{Res}_0(A, \sigma)$ whose elements are "close" to σ.

In order to make this precise, the following notation is needed. The *distance* $\rho(\sigma_1, \sigma_2)$ between states σ_1 and σ_2 is the set of fluents on which the states differ:

$$\rho(\sigma_1, \sigma_2) = \{F \mid \sigma_1(F) \neq \sigma_2(F)\}.$$

Let $\mathbf{Fr}$ denote the frame.

Now the transition function *Res* corresponding to D is defined as follows: $\mathit{Res}(A, \sigma)$ is the set of states $\sigma' \in \mathit{Res}_0(A, \sigma)$ for which $\rho(\sigma, \sigma') \cap \mathbf{Fr}$ is minimal relative to set inclusion—in other words, for which there is no $\sigma'' \in \mathit{Res}_0(A, \sigma)$ such that $\rho(\sigma, \sigma'') \cap \mathbf{Fr}$ is a proper subset of $\rho(\sigma, \sigma') \cap \mathbf{Fr}$.

Consider, for instance, the domain (4). We will represent a state σ by the set of fluents F such that $\sigma(F) = \mathsf{T}$. In this notation,

$$\begin{aligned} Res_0(Load, \{Alive\}) &= \{\{Loaded\}, \{Loaded, Alive\}, \\ &\quad \{Loaded, Alive, Walking\}\}, \\ Res(Load, \{Alive\}) &= \{\{Loaded, Alive\}\}; \end{aligned}$$

Models and Entailment A *structure* is a partial function from strings of actions to valuations whose domain is nonempty and prefix-closed. If a structure Ψ is defined on a string $\overline{A}$, we say that $\overline{A}$ is *executable* in Ψ. Thus, note that in any structure, the empty string of actions is always executable. As we will see shortly, $\Psi(\overline{A})$ in a model of D represents the state that results from the execution of the members of $\overline{A}$ sequentially from the initial state.

A value proposition (1) is *true* in a structure Ψ if $\overline{A}$ is executable in Ψ and $\Psi(\overline{A})(C) = \mathsf{T}$. A constraint **always** C is *true* in a structure Ψ if for any string $\overline{A}$ executable in Ψ, $\Psi(\overline{A})(C) = \mathsf{T}$.

A structure Ψ is a *model* of a domain D if every value proposition and every constraint in D is true in Ψ, and, for every string of actions $\overline{A}$ executable in Ψ and every action A,

- if $\overline{A}A$ is executable in Ψ, then
$$\Psi(\overline{A}A) \in Res(A, \Psi(\overline{A})),$$
- otherwise, $Res(A, \Psi(\overline{A})) = \emptyset$.

We say that a domain description is *consistent* if it has a model. Two domain descriptions are *equivalent* if they have the same models. A value proposition is *entailed* by a domain description D if it is true in every model of D.

For instance, it is easy to see that the domain (4) has two models that that differ by the initial value of *Walking*. As remarked before, it entails the value proposition

$$\neg Walking \textbf{ after } Load; Wait; Shoot.$$

A Brief Review of Baker's Approach

As is standard with circumscriptive approaches to reasoning about action, Baker [1991] starts with a theory that includes axioms encoding the effects of actions, domain constraints and the commonsense law of inertia:

$$\neg Ab(f, a, s) \supset [Holds(f, s) \equiv Holds(f, Result(a, s))].$$

Here, $Ab(f, a, s)$ is an "abnormality" predicate (McCarthy 1986) which stands for the fluent f being abnormal with respect to action a in situation s. The key innovation that Baker proposes is to change the way circumscription is applied to this theory—instead of circumscribing *Ab* while varying the predicate *Holds* as is done in (Hanks & McDermott 1987), Baker circumscribes *Ab* while varying the function *Result* and the situation constant S_0. For this to work correctly, Baker needs an existence of situations axiom, which says that corresponding to each set of fluents consistent with the domain constraints, there is a situation in which exactly these fluents hold. To this end, he introduces the following axiom:

$$\neg AbSit(\lambda) \supset \forall f[Holds(f, Sit(\lambda)) \equiv \lambda(f)].$$

Here λ is a unary predicate variable, *Sit* a function from unary predicate variables to situations and *Absit* an abnormality predicate on fluent predicates. The idea is that by circumscribing *Absit*, we would get that λ is in the extent of *Absit* iff the set of all fluents f such that $\lambda(f)$ holds violates some constraint. That is, for each subset T of the set of fluents consistent with the set of constraints, there is a situation where precisely those fluents in T hold. The following unique name axiom for *Sit* is necessary to make this work correctly:

$$Sit(\lambda_1) = Sit(\lambda_2) \supset \lambda_1 = \lambda_2.$$

Translating from $\mathcal{AR}^-$ into a Circumscriptive Theory

In this section, we present a translation from a subset of $\mathcal{AR}^-$. We restrict attention to *finite* domains, that is, to domains with finitely many fluents, actions and propositions.

We impose one additional restriction to the domains to be translated. We define a domain description D to be *executable* [1] if, for every action name A and every state σ of D, $Res(A, \sigma) \neq \emptyset$. We restrict attention to executable domain descriptions.

We will transform a finite, executable domain D in the sense of the language $\mathcal{AR}^-$ into a circumscriptive theory whose language L has variables of three sorts: for fluents, actions and situations. These variables will be denoted respectively by $f, f_1, f_2, \ldots$; $a, a_1, a_2, \ldots$; $s, s_1, s_2, \ldots$. In addition, L has predicate variables $\lambda, \lambda_1, \lambda_2, \ldots$ that range over properties of fluents.

The language has the following object constants:

- the fluents of D,
- the actions of D,
- the situation constant S_0.

The language L contains the following function and predicate constants:

- the unary predicate *FrameFluent*, whose argument is a fluent,
- the binary predicate *Holds*, whose two arguments are a fluent and a situation.
- the binary function *Result*, which takes an action and a situation and gives a situation.

[1] Similar concepts, for other action languages, were introduced in (Dung 1993) ("selfcontradicted actions") and in (Denecker & De Schreye 1993) ("e-consistency").

- the ternary predicate Ab, whose arguments are a fluent, action and a situation
- the unary predicate $AbSit$, whose argument is a unary predicate on fluents.
- the unary function Sit, which takes unary predicate on fluents as its argument and returns a situation.

Note that "formulae" as defined in the earlier section on the syntax of $\mathcal{AR}^-$ are not among the formulae of L. To avoid confusion, we will refer to the former as as "domain formulae." For any domain formula C, by $T(C,s)$ we will denote the formula of L obtained from C as the result of replacing each fluent name F by $Holds(F,s)$. For instance, $T(\neg Alive \wedge Loaded, s)$ stands for

$$\neg Holds(Alive,s) \wedge Holds(Loaded,s).$$

For any string of actions $A_1 \ldots A_m$, by $[A_1 \ldots A_m]$ we will denote the ground term

$$Result(A_m, Result(A_{m-1}, \ldots, Result(A_1, S_0), \ldots)).$$

Law of Inertia and Existence of Situations

Our translation β from $\mathcal{AR}^-$ uses two defaults. The first one is the commonsense law of inertia, which we restrict to the frame fluents (following (Lifschitz 1991)) as follows:

$$FrameFluent(f) \wedge \neg Ab(f,a,s) \supset [Holds(f, Result(a,s)) \equiv Holds(f,s)].$$

This formula will be denoted by LI.

The second default, the existence of situations principle, is expressed by the formula

$$\neg AbSit(\lambda) \supset \forall f(Holds(f, Sit(\lambda)) \equiv \lambda(f))].$$

This formula will be denoted by ES.

Translating Propositions

We first define an auxiliary translation βD of the domain D. We will define how to construct, for each proposition P in D, the corresponding formula βP. These formulae will be included in the translation βD.

If P is a value proposition C **after** $\overline{A}$, then βP is

$$T(C, [\overline{A}]).$$

For instance, the proposition

$$\neg Alive \textbf{ after } Shoot$$

is translated as

$$\neg Holds(Alive, Result(Shoot, S_0)).$$

If P is an effect proposition A **causes** C **if** P, then βP is

$$T(P,s) \supset T(C, Result(A,s)).$$

For instance, the proposition

$$Shoot \textbf{ causes } \neg Alive \textbf{ if } Loaded$$

is translated as

$$Holds(Loaded,s) \supset \neg Holds(Alive, Result(Shoot,s)).$$

Finally, if P is a constraint **always** C, then βP is $T(C,s)$. For instance,

$$\textbf{always } Walking \supset Alive$$

is translated as

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

Definition of $B(D)$

By $\mathbf{F}$ we will denote the set of fluents in the language of D, by $\mathbf{Fr}$ the set of frame fluents, by $\mathbf{A}$ the set of actions. The axioms of the auxiliary translation βD will now be given.

Group 1. Unique names axioms:

$$F_1 \neq F_2$$

for all pairs of distinct fluents $F_1, F_2 \in \mathbf{F}$,

$$A_1 \neq A_2$$

for all pairs of distinct actions $A_1, A_2 \in \mathbf{A}$, and

$$Sit(\lambda_1) = Sit(\lambda_2) \supset \lambda_1 = \lambda_2.$$

Group 2. Domain closure axioms:

$$\bigvee_{F \in \mathbf{F}} f = F,$$

$$\bigvee_{A \in \mathbf{A}} a = A.$$

Group 3. Translations of the propositions:

$$\beta P \qquad (P \in D).$$

Group 4. Characterization of $FrameFluent$:

$$FrameFluent(F) \qquad (F \in \mathbf{Fr})$$

Group 5. The Two Defaults

$$LI$$
$$ES.$$

By $B(D)$, we will denote the circumscriptive theory whose axioms are $\beta(D)$ and whose policy declarations (Lifschitz 1993) are the following:

circ *FrameFluent* **var** *Ab*,
circ *AbSit* **var** *Ab*, *Holds*, *Result*, S_0,
circ *Ab* **var** *Result*, S_0.

The following facts have been established regarding the soundness and completeness of B.

Let D be a finite, executable domain, let Ψ be a model of D, and let M be a model of $B(D)$. We say that M is *similar* to Ψ if, for every domain formula C and every string of actions $\overline{A}$, the value proposition C **after** $\overline{A}$ is true in Ψ if and only if M satisfies $T(C, [\overline{A}])$.

Theorem 1. *Let D be a finite, executable domain. For any model Ψ of D, there exists a model M of $B(D)$ similar to Ψ.*

The following corollary expresses the soundness of the translation.

Corollary. *Let D be a finite, executable domain. For any domain formula C and every string of actions $\overline{A}$, if the formula $T(C, [\overline{A}])$ is entailed by $B(D)$ then the value proposition C* **after** *$\overline{A}$ is entailed by D.*

There are two important cases when the translation is complete.

One is the case of "temporal projection." A domain D is a *temporal projection domain* if every value proposition in D is of the form **initially** C. For instance, the action domain (4) is a temporal projection domain. Obviously, the domains that contain no value propositions are temporal projection domains.

Theorem 2. *Let D be a finite, consistent, executable, temporal projection domain. For any model M of $B(D)$, there exists a model Ψ of D such that M is similar to Ψ.*

The following corollary expresses the completeness of the translation for temporal projection domains.

Corollary. *Let D be a finite, consistent, executable, temporal projection domain. For any domain formula C and every string of actions $\overline{A}$, if the value proposition C* **after** *$\overline{A}$ is entailed by D then the formula $T(C, [\overline{A}])$ is entailed by $B(D)$.*

The second case is the case of "deterministic" domains. A domain D is *deterministic* if, for every action A and every state σ, $Res(A, \sigma)$ is a singleton. For instance, the domain (4) is deterministic. Obviously, every deterministic domain is executable.

Theorem 3. *Let D be a finite, consistent, deterministic domain. For any model M of $B(D)$, there exists a model Ψ of D such that M is similar to Ψ.*

Corollary. *Let D be a finite, consistent, deterministic domain. For any domain formula C and every string of actions $\overline{A}$, if the value proposition C* **after** *$\overline{A}$ is entailed by D then the formula $T(C, [\overline{A}])$ is entailed by $B(D)$.*

Discussion

Two papers that are closely related to the work presented in this paper are (Lifschitz 1991; Kartha 1993). In (Lifschitz 1991), Baker's method is applied to a class of problems characterized in terms of their syntactic form. In (Kartha 1993), a soundness and completeness result is presented for a translation from an action language $\mathcal{A}$. The results presented in this paper can be seen as extensions of the work presented in these two papers. For instance, the soundness and completeness result from (Kartha 1993) is subsumed by Theorems 1 and 3, because $\mathcal{A}$ domains can be thought of as a special case of deterministic $\mathcal{AR}^-$ domains where there are no constraints.

Note that we have restricted attention here to executable domains. To adequately deal with domains that are not executable, the symbol *Result* in the situation calculus should be viewed as representing a partial function. This can be technically achieved, for example, using the "possibility" predicate, as in (Reiter 1991). Baker [1991] treats *Result* as total, and we chose not to modify his approach in this respect.

Roughly, why Baker's approach works on temporal projection domains and deterministic domains can be explained as follows. First of all, it is important to note a crucial difference in the role that value propositions play in $\mathcal{AR}^-$ and in the circumscriptive theory $B(D)$. In the definition of the semantics of $\mathcal{AR}^-$, value propositions play no role in determining the transition function *Res*—they are used only in "filtering" structures to determine whether they are models (Sandewall 1989). However, the set of axioms that are circumscribed to obtain the theory $B(D)$ include the translations of the value propositions and hence they play a role in determining the *Result* function.

Due to this difference in the role that value propositions play in determining the effects of actions in $\mathcal{AR}^-$ and in $B(D)$, a correspondence between these action representations is possible only when the value propositions are suitably restricted. In the case of temporal projection domains, the value propositions are restricted to the initial situation. In the case of deterministic domains, they are again restricted due to the requirement that they be consistent with values allowed by the deterministic transition function *Res*. In either case, the restrictions suffice to guarantee that the translations of the value propositions do not not interfere with the circumscriptions in $B(D)$ in unforeseen ways.

Consider now the "Two Bus" domain (5). It is easy to show that this domain entails

$$HasTicket \wedge \neg OnRedBus \wedge \neg OnYellowBus \textbf{ after } BuyTicket.$$

However, it can be shown (Kartha 1994) that for this domain, Baker's circumscription does not entail the corresponding formula

$$Holds(HasTicket, Result(BuyTicket, S_0)) \wedge \neg Holds(OnRedBus, Result(BuyTicket, S_0)) \wedge \neg Holds(OnYellowBus, Result(BuyTicket, S_0)).$$

Note that (5) is neither a deterministic nor a temporal projection domain. This observation, in conjunction with the soundness and completeness results for temporal projection and deterministic domains gives us a sharp characterization of the class of action domains where Baker's method is applicable.

Acknowledgement: I am grateful to Vladimir Lifschitz for discussions related to the topic of this paper.

References

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

Crawford, J., and Etherington, D. 1992. Formalizing reasoning about change: A qualitative reasoning approach. In *Proc. AAAI-92*, 577–583.

Denecker, M., and De Schreye, D. 1993. Representing incomplete knowledge in abductive logic programming. In Miller, D., ed., *Logic Programming: Proceedings of the 1993 Int'l Symposium*, 147–163.

Dung, P. M. 1993. Representing actions in logic programming and its applications in database updates. In *Logic Programming: Proceedings of the Tenth Int'l Conf. on Logic Programming*, 222–238.

Gelfond, M., and Lifschitz, V. 1993. Representing action and change by logic programs. *Journal of Logic Programming* 17:301–322.

Hanks, S., and McDermott, D. 1987. Nonmonotonic logic and temporal projection. *Artificial Intelligence* 33(3):379–412.

Kartha, G. N., and Lifschitz, V. 1994. Actions with indirect effects. In Doyle, J.; Sandewall, E.; and Torasso, P., eds., *Proc. of the Fourth Int'l Conf. on Principles of Knowledge Representation and Reasoning*, 341–350.

Kartha, G. N. 1993. Soundness and completeness theorems for three formalizations of action. In *Proc. of IJCAI-93*, 724–729.

Kartha, G. N. 1994. Two counterexamples related to Baker's approach to the frame problem. *Artificial Intelligence* 69:379–391.

Lifschitz, V. 1991. Towards a metatheory of action. In Allen, J.; Fikes, R.; and Sandewall, E., eds., *Proc. of the Second Int'l Conf. on Principles of Knowledge Representation and Reasoning*, 376–386.

Lifschitz, V. 1993. Circumscription. In Gabbay, D.; Hogger, C.; and Robinson, J., eds., *The Handbook of Logic in AI and Logic Programming*, volume 3. Oxford University Press. 297–352.

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

Reiter, R. 1991. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In Lifschitz, V., ed., *Artificial Intelligence and Mathematical Theory of Computation*. Academic Press. 359–380.

Sandewall, E. 1989. Combining logic and differential equations for describing real-world systems. In Brachman, R.; Levesque, H.; and Reiter, R., eds., *Proc. of the First Int'l Conf. on Principles of Knowledge Representation and Reasoning*, 412–420.

Shanahan, M. 1995. *Solving the Frame Problem.* Draft.