# Restricted Monotonicity

Vladimir Lifschitz*

Department of Computer Sciences
and Department of Philosophy
University of Texas at Austin
Austin, TX 78712

## Abstract

A knowledge representation problem can be sometimes viewed as an element of a family of problems, with parameters corresponding to possible assumptions about the domain under consideration. When additional assumptions are made, the class of domains that are being described becomes smaller, so that the class of conclusions that are true in all the domains becomes larger. As a result, a satisfactory solution to a parametric knowledge representation problem on the basis of some nonmonotonic formalism can be expected to have a certain formal property, that we call *restricted monotonicity*. We argue that it is important to recognize parametric knowledge representation problems and to verify restricted monotonicity for their proposed solutions.

## Introduction

This paper is about the methodology of representing knowledge in nonmonotonic formalisms. A knowledge representation problem can be sometimes viewed as an element of a family of problems, with parameters corresponding to possible assumptions about the domain under consideration. When additional assumptions are made, the class of domains that are being described becomes smaller, so that the class of conclusions that are true in all the domains becomes larger. As a result, a satisfactory solution to a parametric knowledge representation problem on the basis of some nonmonotonic formalism can be expected to have a certain formal property, that we call *restricted monotonicity*.

The idea of restricted monotonicity is first illustrated here by examples. Then the precise definition of this property is given, and methods for proving it are discussed. Finally, we apply the concept of restricted monotonicity to the analysis of some of the recent work on representing action and change in nonmonotonic formalisms.

*This work was partially supported by National Science Foundation under grant IRI-9101078.

## Examples

Here is a simple knowledge representation problem involving a default:

> Formalize the assertions:
>
> *Birds normally fly.*
> *Penguins are birds.*
> *Penguins do not fly.*

We will compare two solutions, one based on default logic [Reiter, 1980], the other on circumscription [McCarthy, 1986].

The first formalization is the default theory whose postulates are the axioms

$$\forall x(Penguin(x) \supset Bird(x)) \tag{1}$$

and

$$\forall x(Penguin(x) \supset \neg Flies(x)), \tag{2}$$

and the default

$$Bird(x) : Flies(x) \,/\, Flies(x). \tag{3}$$

The second is the circumscriptive theory with the axioms (1), (2) and

$$\forall x(Bird(x) \wedge \neg Ab(x) \supset Flies(x)), \tag{4}$$

in which $Ab$ is circumscribed and *Flies* varied.

There is an important difference between the two formalizations that becomes obvious when we apply the default *birds normally fly* to specific objects. Although the postulates of the two theories do not use any object constants, let us assume that their languages include an object constant, say, *Joe*. Since Joe is not postulated to be a bird, the formula $Flies(Joe)$ is undecidable both in the default logic formalization $T_1$ and in the circumscriptive formalization $T_2$. We are interested in the theories obtained from $T_1$ and $T_2$ by adding some of the possible assumptions

$$\begin{array}{c} Bird(Joe), \neg Bird(Joe), \\ Penguin(Joe), \neg Penguin(Joe) \end{array} \tag{5}$$

to their axiom sets.

For some subsets $p$ of (5), adding $p$ to the axiom set of $T_1$ will have the same effect on the status of the formula $Flies(Joe)$ as adding $p$ to $T_2$. If, for instance, $p$ is $\{Bird(Joe), \neg Penguin(Joe)\}$, then $Flies(Joe)$ is provable both in $T_1 \cup p$ and $T_2 \cup p$. If, on the other hand, $p$ is $\{Penguin(Joe)\}$, then this formula is refutable in both theories.

The situation will be different, however, if we take $p$ to be $\{Bird(Joe)\}$. Now Joe is known to be a bird, but it is not known whether he is a penguin. The default logic formalization sanctions the conclusion that Joe flies: $T_1 \cup \{Bird(Joe)\}$ entails $Flies(Joe)$. In the corresponding circumscriptive formalization, $T_2 \cup \{Bird(Joe)\}$, the formula $Flies(Joe)$ remains undecidable.

The theories $T_1$ and $T_2$ can be viewed as somewhat different interpretations of the given set of assumptions about the ability of birds to fly. Whether or not $T_1$ is considered too strong—or $T_2$ too weak—depends on which of the two identically worded, but slightly different knowledge representation problems we had in mind in the first place.

The circumscriptive solution $T_2$ can be viewed as reasonable, and the default logic solution $T_1$ as excessively strong, if the absence of both $Penguin(Joe)$ and $\neg Penguin(Joe)$ among the axioms is supposed to indicate our willingness to take into consideration both the domains in which Joe is a penguin and the domains in which he isn't, and to sanction only the conclusions that are true in domains of both kinds. We will express this knowledge representation convention by saying that these two literals function in this example as "parameters." A parameter is an additional postulate whose presence in the axiom set is supposed to make the set of domains under consideration smaller, and thus to increase the set of conclusions sanctioned by the formalization.

The knowledge representation problem stated at the beginning of this section would be described more precisely if we specified that the ground literals containing *Bird* or *Penguin* should be treated as parameters. This statement implies that a formalization $T$ will be considered adequate only if it has the following property: *For any subsets $p$, $q$ of (5) such that $p \subset q$, each consequence of $T \cup p$ is a consequence of $T \cup q$.* In particular, if $T$ is adequate, then all theorems of $T \cup \{Bird(Joe)\}$ will be among the theorems of $T \cup \{Bird(Joe), Penguin(Joe)\}$, so that $Flies(Joe)$ will not be one of them.

This property is an example of what we call "restricted monotonicity." It is satisfied for the circumscriptive formalization $T_2$, but not for the default theory $T_1$. We will see that, in general, there is no correlation between restricted monotonicity and the choice of a nonmonotonic formalism; what matters is which postulates are included in the formalization and, in case of circumscription, what circumscription policy is applied. For instance, we will give a default logic formalization of the same example that satisfies the restricted monotonicity condition.

There is nothing wrong, of course, with a different interpretation of the flying birds problem: If the axioms do not tell us whether Joe is a penguin, we may treat the domains in which this is the case as "secondary," and be prepared to jump to the conclusion that we are not in such a domain, at least when we decide whether Joe can fly. But it is important to be clear about how the problem is interpreted before discussing the adequacy of a particular solution.

We argue in this paper that many knowledge representation problems can be described as parametric, and that, when we deal with such a problem, it is important to recognize this fact and to verify restricted monotonicity for its proposed solutions.

A class of examples that is discussed here in some detail is given by *initial conditions* in temporal projection problems. In various versions of the "Yale shooting" story [Hanks and McDermott, 1987], the initial situation is described by including some of the formulas

$$Holds(Loaded, S0),\ \neg Holds(Loaded, S0),\\ Holds(Alive, S0),\ \neg Holds(Alive, S0) \qquad (6)$$

in the axiom set. We can think of these formulas as parameters. Every consistent subset of (6) represents an instance of a "parametric problem"; the larger the subset is, the more conclusions about the values of fluents in future situations can be justified. This is again an example of restricted monotonicity.

## Definition

We would like to give a definition of restricted monotonicity applicable to many nonmonotonic formalisms. In order to make it general, we will first introduce the notion of a "declarative formalism."

A *declarative formalism* is defined by a set $S$ of symbolic expressions called *sentences*, a set $P$ of symbolic expressions called *postulates*, and a map $Cn$ from sets of postulates to sets of sentences. A set of postulates is a *theory*. A sentence $A$ is a *consequence* of a theory $T$ if $A \in Cn(T)$. The formalism is *monotonic* if $Cn$ is a monotone operator, that is, if $Cn(T) \subset Cn(T')$ whenever $T \subset T'$.

Here are some examples. Any first-order or higher-order language of classical logic can be viewed as a declarative formalism. Its postulates are identical to its sentences—they are arbitrary closed formulas of the language; $Cn(T)$ is the set of sentences that are true in all models of $T$. The use of circumscription amounts to defining $Cn(T)$ to be the set of sentences that are true in the models of $T$ which are minimal relative to some circumscription policy. In case of default theories, a sentence is a closed formula, and a postulate is either a closed formula or a default. (Note

that here $P$ differs from $S$.) For a default theory $T$, $Cn(T)$ is the intersection of the extensions of $T$.

When a declarative formalism $\langle S, P, Cn\rangle$ is used to solve a parametric knowledge representation problem, a subset $S_0$ of its sentences is designated as the set of *assertions*, and a subset $P_0$ of its postulates is designated as the set of *parameters*. The idea of a parameter was discussed above: A parameter is an additional postulate whose presence in a theory is supposed to make the class of domains under consideration smaller. An assertion is a sentence that can be interpreted as true or false in a domain described by the theory. The need to distinguish between assertions and arbitrary sentences arises when the language contains auxiliary symbols, such as $Ab$, that have no "observable" meaning in the domains under consideration. We may wish to specify, for instance, that a sentence is an assertion if it does not contain $Ab$. Some formalizations do not use auxiliary symbols; in such cases, we view every sentence as an assertion.

Let $\langle S, P, Cn\rangle$ be a declarative formalism. Let a subset $S_0$ of $S$ be designated as the set of assertions, and a subset $P_0$ of $P$ as the set of parameters. We say that a theory $T$ satisfies the *restricted monotonicity condition* if, for any sets $p, q \subset P_0$,

$$p \subset q \Rightarrow Cn(T \cup p) \cap S_0 \subset Cn(T \cup q) \cap S_0. \qquad (7)$$

In words: *If more parameters are added to $T$ as additional postulates, no assertions will be retracted.*

Note that (7) is trivially true if $Cn$ is a monotone operator. Consequently, restricted monotonicity becomes an issue only when a nonmonotonic formalism is used.

Condition (7) is weaker than the monotonicity of $Cn$ in two ways. First, it applies only to theories of the form $T \cup p$ for the subsets $p$ of $P_0$, rather than to arbitrary theories. Second, it refers not to the set of all consequences of a theory, but only to the assertions that belong to it.

## Methods for Proving Restricted Monotonicity

The mathematical apparatus required for proving restricted monotonicity will vary depending on the declarative formalism on which the solution is based. In this section we discuss some of the methods that can be used for verifying the restricted monotonicity condition in circumscriptive theories and in extended logic programs.

### Circumscriptive Theories

For simplicity, we restrict attention to finite circumscriptive theories without prioritization. Let $S$ be the set of all sentences of some first-order language, $R$ a list of distinct predicate constants of that language, and $Z$ a list of distinct function and/or predicate constants disjoint from $R$. By $Cn_{R;Z}$ we denote the consequence operator corresponding to the circumscription which circumscribes $R$ and varies $Z$. This means that, for any finite subset $T$ of $S$, $Cn_{R;Z}(T)$ is the set of sentences entailed by $\mathrm{CIRC}[\bigwedge_{A\in T} A; R; Z]$.

**Proposition 1.** *Let $T$ be a finite theory in the formalism $\langle S, S, Cn_{R;Z}\rangle$. If the set of parameters is finite, and the parameters do not contain symbols from $R$ or $Z$, then $T$ satisfies the restricted monotonicity condition.*

**Proof.** For any set $p$ of parameters, $Cn_{R;Z}(T \cup p)$ is the set of sentences entailed by

$$\mathrm{CIRC}[\bigwedge_{A\in T} A \wedge \bigwedge_{A\in p} A; R; Z].$$

Since $\bigwedge_{A\in p} A$ does not contain symbols from $R$ or $Z$, this formula is equivalent to

$$\mathrm{CIRC}[\bigwedge_{A\in T} A; R; Z] \wedge \bigwedge_{A\in p} A.$$

If $p \subset q$, then this conjunction is entailed by the corresponding condition for $q$:

$$\mathrm{CIRC}[\bigwedge_{A\in T} A; R; Z] \wedge \bigwedge_{A\in q} A.$$

It follows that $p \subset q$ implies

$$Cn_{R;Z}(T \cup p) \subset Cn_{R;Z}(T \cup q),$$

and consequently

$$Cn_{R;Z}(T \cup p) \cap S_0 \subset Cn_{R;Z}(T \cup q) \cap S_0.$$

In case of the circumscriptive theory $T_2$ defined above, $R$ is $Ab$, $Z$ is *Flies*, $P_0$ is (5), and $S_0$ is the set of sentences not containing $Ab$. By Proposition 1, the restricted monotonicity of $T_2$ follows from the fact that the parameters (5) contain neither $Ab$ nor *Flies*.

Note that the statement of Proposition 1 imposes a restriction on the set of parameters, but not on the set of assertions. It follows, in particular, that $T_2$ would have satisfied the restricted monotonicity condition even if all sentences were considered assertions. We will see below that, for some other solutions to the same knowledge representation problem, the fact that $Ab$ is not allowed in assertions is crucial for the verification of restricted monotonicity.

Consider, on the other hand, the theory $T_2'$, which differs from $T_2$ in that the predicates *Bird* and *Penguin* are allowed to vary, along with *Flies*. This theory is stronger than $T_2$, and it allows us to justify, among others, the conclusion that there are no penguins in the world: $\forall x \neg Penguin(x)$. This result may be viewed as undesirable. Peculiarities of this kind in circumscriptive theories are well known ([McCarthy, 1986], Section 5). In fact, $T_2'$ has a more fundamental defect: It does not satisfy the restricted monotonicity condition. Indeed, $\neg Penguin(Joe)$ is a consequence of $T_2'$ which is lost when $Penguin(Joe)$ is added to its axiom set. Proposition 1 does not apply here, because the predicates *Bird* and *Penguin*, varied in $T_2'$, occur in the parameters.

## Extended Logic Programs

According to [Gelfond and Lifschitz, 1991], an *extended logic program* is a set of rules of the form

$$L_0 \leftarrow L_1, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n, \quad (8)$$

where each $L_i$ is a literal, that is, an atom possibly preceded by $\neg$. ("General" logic programs are, syntactically, the special case when classical negation $\neg$ is not used.) The rule (8) has the same meaning as the default

$$L_1 \wedge \ldots \wedge L_m : \overline{L_{m+1}}, \ldots, \overline{L_n} \,/\, L_0, \quad (9)$$

where $\overline{L}$ stands for the literal complementary to $L$, so that the language of extended programs can be viewed as a subsystem of default logic. A ground literal $L$ is a *consequence* of an extended program $\Pi$ if it belongs to all extensions of $\Pi$. Extended logic programs can be viewed as theories in a declarative formalism, if ground literals are taken to be sentences, and rules of the form (8) are considered postulates.

The knowledge representation problem described at the beginning of the paper can be solved in the language of extended programs as follows:

$$\begin{array}{l} Flies(x) \leftarrow Bird(x), not\ Ab(x), \\ Bird(x) \leftarrow Penguin(x), \\ \neg Flies(x) \leftarrow Penguin(x), \\ Ab(x) \leftarrow not\ \neg Penguin(x). \end{array} \quad (10)$$

Note the last rule, which plays an important part in this program. Without it, adding the fact $Penguin(Joe)$ to the program would have made it inconsistent. Note also the use of the combination $not\ \neg$ in that rule. The simpler rule

$$Ab(x) \leftarrow Penguin(x) \quad (11)$$

would have canceled the applicability of the first rule of the program to $x$ only when $x$ is known to be a penguin; with $not\ \neg$ inserted in front of $Penguin(x)$, this is accomplished for every $x$ that is not known to satisfy $\neg Penguin(x)$. (Compare this with the discussion of the cancelation rule for *Noninertial* in Section 4 of [Gelfond and Lifschitz, 1992].) In particular, even with the fact $Bird(Joe)$ added to (10), the formula $Flies(Joe)$ remains undecidable. We see that (10) is similar in this respect to the circumscriptive formalization $T_2$, rather than to the default theory $T_1$.

Written as defaults, the rules (10) are:

$$\begin{array}{l} Bird(x) : \neg Ab(x) \,/\, Flies(x), \\ Penguin(x) \,/\, Bird(x), \\ Penguin(x) \,/\, \neg Flies(x), \\ : Penguin(x) \,/\, Ab(x). \end{array} \quad (12)$$

The first of these defaults is reminiscent of the approach to the use of default logic advocated by Morris [1988].

The theorem about restricted monotonicity in extended logic programs stated below is based on the notion of a "signing." This notion was originally defined for general logic programs [Kunen, 1989], and then extended by Turner [1993] to programs that may contain classical negation.

The *absolute value* of a literal $L$ (symbolically, $|L|$) is $L$ if $L$ is positive, and $\overline{L}$ otherwise. A *signing* for an extended logic program $\Pi$ without variables is a set $X$ of ground atoms such that

(i) for any rule (8) from $\Pi$, either

$$|L_0|, \ldots, |L_m| \in X,\ |L_{m+1}|, \ldots, |L_n| \notin X$$

or

$$|L_0|, \ldots, |L_m| \notin X,\ |L_{m+1}|, \ldots, |L_n| \in X;$$

(ii) for any atom $A \in X$, $\neg A$ does not appear in $\Pi$.

It is easy to see, for example, that the set of ground instances of $Ab(x)$ is a signing for the set of ground instances of the rules (10).

The following lemma is a special case of Theorem 1 from [Turner, 1993].

**Lemma.** *Let $\Pi_1$ be an extended program without variables, and let $X$ be a signing for $\Pi_1$. Let $\Pi_2$ be a program obtained from $\Pi_1$ by dropping some of its rules (8) such that $|L_0| \notin X$. If a ground literal $L$ is a consequence of $\Pi_2$ and $|L| \notin X$, then $L$ is a consequence of $\Pi_1$ also.*

**Proposition 2.** *Let $\Pi$ be an extended logic program, and let $X$ be a signing for the set of ground instances of the rules of $\Pi$. If all parameters and assertions are ground literals whose absolute values do not belong to $X$, then $\Pi$ satisfies the restricted monotonicity condition.*

**Proof.** Since a program has the same consequences as the set of all ground instances of its rules, we can assume, without loss of generality, that the rules of $\Pi$ do not contain variables. Let $p$ and $q$ be sets of parameters such that $p \subset q$. By applying the lemma to $\Pi \cup q$ as $\Pi_1$ and $\Pi \cup p$ as $\Pi_2$, we conclude that, for any ground literal $L$ such that $|L| \notin X$, if $L \in Cn(\Pi \cup p)$ then $L \in Cn(\Pi \cup q)$. It follows that

$$Cn(\Pi \cup p) \cap S_0 \subset Cn(\Pi \cup q) \cap S_0.$$

Proposition 2 implies, for instance, that (10) satisfies the restricted monotonicity condition.

## Restricted Monotonicity in Theories of Action

As observed above, temporal projection problems can be thought of as parametric, with initial conditions as parameters. It is interesting to look from this perspective at the existing approaches to describing actions in nonmonotonic formalisms and to see how successful they are in achieving restricted monotonicity. (For the methods based on stating frame axioms explicitly and then applying classical logic, the problem does not arise, because any theory based on a monotonic logic satisfies the restricted monotonicity condition.)

## Minimizing Change

The first attempt to solve the frame problem using circumscription ([McCarthy, 1986], Section 9) was shown by Hanks and McDermott [1987] to lead in some cases to "overweak disjunctions." The analysis of McCarthy's method from the point of view of restricted monotonicity shows that it has also another flaw.

The following key observation was made by Fangzhen Lin in connection with the temporal minimization method (personal communication, October 31, 1992). Let $A$ be an action whose effect is to make a propositional fluent $F$ false if it is currently true. The initial value of $F$ is not given. Minimizing change will lead to the conclusion that $F$ was initially false, because in this case nothing has to change as $A$ is executed.

This undesirable conclusion presents a difficulty that is perhaps even more fundamental than the one uncovered by Hanks and McDermott. Minimizing change may lead not only to conclusions that are too weak; sometimes, its results are much too strong. Since the only action considered in this example is performed in the initial situation, it does not matter whether the minimization criterion is simple or temporal, as in [Kautz, 1986], [Lifschitz, 1986] and [Shoham, 1986].

We will describe Lin's example formally and show that it can be viewed as a violation of restricted monotonicity. Instead of the situation calculus language, we will use the simpler syntax of a *theory of a single action* ([Lifschitz, 1990], Section 2). Theories of this kind include situation variables and fluent variables, but they do not have variables for actions. The function *Result* is replaced by two situation constants: $S^i$ for the initial situation, in which a certain fixed action is executed, and $S^r$ for the result of the action. If the only fluent constant in the language is $F$, then the possible initial conditions are

$$Holds(F, S^i),\ \neg Holds(F, S^i). \tag{13}$$

The assumption that the action in question makes $F$ false if executed when $F$ is true is expressed by the formula

$$Holds(F, S^i) \supset \neg Holds(F, S^r). \tag{14}$$

Minimizing change, in this simplified language, is expressed by postulating the "commonsense law of inertia" in the form

$$\neg Ab(f) \supset Holds(f, S^r) \equiv Holds(f, S^i), \tag{15}$$

and circumscribing $Ab$.

Let $T$ be the circumscriptive theory with axioms (14) and (15), in which $Ab$ is circumscribed and *Holds* varied. Take the formulas (13) to be parameters, and all closed formulas not containing $Ab$ to be assertions. Lin's observation shows that $T$ does not satisfy the restricted monotonicity condition. Indeed, $\neg Holds(F, S^i)$ is a consequence of $T$, but not a consequence of $T \cup \{Holds(F, S^i)\}$.

Mathematically, the lack of restricted monotonicity in this example is not surprising: The predicate *Holds*, varied in $T$, occurs in parameters, so that Proposition 1 does not apply.

## Other Approaches to the Frame Problem

Two other ways to apply circumscription to the frame problem are proposed in [Lifschitz, 1987] and [Baker, 1991]. Unlike McCarthy's original proposal and the temporal minimization approach, these methods have reasonable restricted monotonicity properties, although this fact does not follow from Proposition 1. A restricted monotonicity theorem for Baker-style formalizations can be proved using Theorem 3 from [Kartha, 1993].

The logic programming method of [Gelfond and Lifschitz, 1992] and [Baral and Gelfond, 1993] builds on the ideas of [Morris, 1988], [Eshghi and Kowalski, 1989], [Evans, 1989] and [Apt and Bezem, 1990]. A restricted monotonicity theorem for this method can be derived from Proposition 2.

Sandewall [1989] proposed to apply a nonmonotonic formalism to a set of axioms that does not include initial conditions ("observations") and to get first "the set of all possible developments in the world *regardless of* any observations," and then " to take that whole set and 'filter' it with the given observations." A mechanism of this kind may achieve restricted monotonicity by removing the initial conditions from the scope of the nonmonotonic consequence operator. Some of the ideas of [Lin and Shoham, 1991] seem to be in the same group.

## High-Level Languages for Describing Actions

The "high-level" language $\mathcal{A}$ [Gelfond and Lifschitz, 1992], designed specifically for describing action and change, has propositions of two kinds. "Value propositions" specify the values of fluents in particlular situations. "Effect propositions" are general statements about the effects of actions. A "domain description" is a set of propositions. The semantics of domain descriptions is defined in terms of "models." A model of a domain description $D$ consists of two components: One specifies the "initial state" of the system, and the other is a "transition function," describing how states are affected by performing actions. The effect propositions from $D$ determine what can be used as the transition function in a model of $D$. The value propositions limit possible choices of the initial state. Details of the syntax and semantics of $\mathcal{A}$ can be found in [Gelfond and Lifschitz, 1992].

A value proposition is a *consequence* of a domain description $D$ if it is true in all models of $D$. This definition allows us to treat the language $\mathcal{A}$ as a declarative formalism, with value propositions as sentences, and both value propositions and effect propositions as postulates.

This formalism is nonmonotonic. Indeed, adding an effect proposition to a domain description $D$, generally, changes the set of models of $D$ in a nonmonotonic way, so that the set of consequences of $D$ changes nonmonotonically also. However, adding value propositions to $D$ merely imposes additional constraints on the choice of the initial state in a model, so that it can only make the set of models smaller, and the set of consequences larger. If we agree to identify both parameters and assertions with value propositions, then this fact can be expressed as follows:

**Proposition 3.** *Every domain description in $\mathcal{A}$ satisfies the restricted monotonicity condition.*

Since initial conditions in a temporal projection problem are represented in $\mathcal{A}$ by value propositions, we conclude that the problem of restricted monotonicity for temporal projection is resolved in $\mathcal{A}$ in a satisfactory way.

The extensions of $\mathcal{A}$ introduced in [Baral and Gelfond, 1993] and [Lifschitz, 1993] have similar restricted monotonicity properties.

## Acknowledgements

I have benefitted from discussing the ideas presented here with Robert Causey, Michael Gelfond, G.N. Kartha, Fangzhen Lin, Norman McCain, Luis Pereira, Hudson Turner and Thomas Woo. My special thanks go to Fangzhen Lin for permission to include his unpublished counterexample.

## References

Apt, Krzysztof and Bezem, Marc 1990. Acyclic programs. In Warren, David and Szeredi, Peter, editors 1990, *Logic Programming: Proc. of the Seventh Int'l Conf.* 617–633.

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

Baral, Chitta and Gelfond, Michael 1993. Representing concurrent actions in extended logic programming. In *Proc. of IJCAI-93.* To appear.

Eshghi, Kave and Kowalski, Robert 1989. Abduction compared with negation as failure. In Levi, Giorgio and Martelli, Maurizio, editors 1989, *Logic Programming: Proc. of the Sixth Int'l Conf.* 234–255.

Evans, Chris 1989. Negation-as-failure as an approach to the Hanks and McDermott problem. In *Proc. of the Second Int'l Symp. on Artificial Intelligence.*

Gelfond, Michael and Lifschitz, Vladimir 1991. Classical negation in logic programs and disjunctive databases. *New Generation Computing* 9:365–385.

Gelfond, Michael and Lifschitz, Vladimir 1992. Representing actions in extended logic programming. In Apt, Krzysztof, editor 1992, *Proc. Joint Int'l Conf. and Symp. on Logic Programming.* 559–573.

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

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

Kautz, Henry 1986. The logic of persistence. In *Proc. of AAAI-86.* 401–405.

Kunen, Kenneth 1989. Signed data dependencies in logic programs. *Journal of Logic Programming* 7(3):231–245.

Lifschitz, Vladimir 1986. Pointwise circumscription: Preliminary report. In *Proc. AAAI-86.* 406–410.

Lifschitz, Vladimir 1987. Formal theories of action (preliminary report). In *Proc. of IJCAI-87.* 966–972.

Lifschitz, Vladimir 1990. Frames in the space of situations. *Artificial Intelligence* 46:365–376.

Lifschitz, Vladimir 1993. A language for describing actions. In *Working Papers of the Second Symposium on Logical Formalizations of Commonsense Reasoning.*

Lin, Fangzhen and Shoham, Yoav 1991. Provably correct theories of action (preliminary report). In *Proc. AAAI-91.* 349–354.

McCarthy, John 1986. Applications of circumscription to formalizing common sense knowledge. *Artificial Intelligence* 26(3):89–116. Reproduced in [McCarthy, 1990].

McCarthy, John 1990. *Formalizing common sense: papers by John McCarthy.* Ablex, Norwood, NJ.

Morris, Paul 1988. The anomalous extension problem in default reasoning. *Artificial Intelligence* 35(3):383–399.

Reiter, Raymond 1980. A logic for default reasoning. *Artificial Intelligence* 13(1,2):81–132.

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

Shoham, Yoav 1986. Chronological ignorance: Time, nonmonotonicity, necessity and causal theories. In *Proc. of AAAI-86.* 389–393.

Turner, Hudson 1993. A monotonicity theorem for extended logic programs. In *Proc. of the Tenth Int'l Conference on Logic Programming.* To appear.