A STUDY OF NONMONOTONIC REASONING

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

By
Fangzhen Lin
August 1991

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Yoav Shoham
(Principal Adviser)

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

John McCarthy

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Vladimir Lifschitz
Department of Computer Science
University of Texas at Austin

Approved for the University Committee on Graduate Studies:

Dean of Graduate Studies

Abstract

This thesis contains several essays on nonmonotonic reasoning, which touch on a number of key issues in that area. It begins with the introduction of the logic GK of Grounded Knowledge as a uniform semantic basis for fixed-point nonmonotonic logics. These logics include Reiter's default logic and Moore's autoepistemic logic, and to our knowledge GK is the first semantic unification of the two. Similarly to circumscription, GK is based on the notion of logical minimization, and thus provides a bridge between circumscription and fixed-point nonmonotonic logics, an outstanding problem in nonmonotonic logics. As an application of this bridge, we propose a formalization of logic programs with negation-as-failure in circumscription. Together with the unique names assumption, the formalization provides a concise representation for the facts that are true in every answer set of a logic program.

We then introduce the notion of argument systems as a proof theory for nonmonotonic reasoning. By reformulating some major existing nonmonotonic logics as argument systems, we show that most nonmonotonic reasoning can be captured with the aid of a generalized negation-as-failure rule. We describe an implemented system for default logic based on this idea.

Finally, we apply nonmonotonic logic to the formalization of inheritance and of theories of action, the two areas that have to a large degree motivated research in nonmonotonic reasoning. We show that various intuitions about nonmonotonic inheritance hierarchies can be conveniently formalized in logic programs with negation-as-failure. For reasoning about action, we first propose a formal yet intuitive criterion by which to evaluate the adequacy of theories of action. We then formulate a class

of monotonic theories that satisfy the criterion by using explicit frame axioms. Finally for a significant subclass of the monotonic theories, we provide provably-correct nonmonotonic counterparts which avoid explicit frame axioms.

Acknowledgements

I would like to thank the members of my reading committee: Yoav Shoham, John McCarthy, and Vladimir Lifschitz. I thank Yoav for his support, both financial and intellectual, throughout my four years at Stanford: the first year as a visitor, and the last three years as a Ph.D. student. Yoav has taught me not only how to do research, but how to write a research paper. His influence on me is everywhere. In fact, most of the dissertation was originally published as joint work with him. I am especially grateful to Yoav and John for their help during my early days at Stanford. Without them, my stay at Stanford would have never been possible. I also thank Vladimir and John for many fruitful discussions we had about AI in general, and nonmonotonic reasoning in particular.

Besides the members of my reading committee, I have had many helpful discussions with Jun-Ichi Akahani, Tom Dean, Kurt Konolige, Jean-Francois Lavignon, Hector Levesque, and many others. I thank them all.

The Stanford Nobotics group, and the Computer Science Department have been a stimulating environment to work in. I'd like to thank my fellow students in the Nobotics group: Nita Goyal, Eyal Mozes, Anton Schwartz, Dominique Snyers, and Becky Thomas. I especially thank Becky and Anton for their helpful comments on an earlier draft of the dissertation.

Last but not the least, I would like to thank my parents, my other family members, and my friends for their love, support, and encouragements. Without them, my life would be meaningless. Especially, I would like to thank my American family, Jim and Lois Dewart. They have always made me feel at home far away from home. Thanks.

Contents

Chapter 1

Introduction

It is well recognized today that it is much harder to program a computer to understand human commonsense than to perform arithmetic operations[1]. The logic approach to Artificial Intelligence (AI), pioneered by John McCarthy, attempts to capture commonsense knowledge and reasoning in formal logic.

There are many difficulties in using classical logic to formalize commonsense reasoning. One of them is that classical logic is *monotonic*, while commonsense reasoning is *nonmonotonic*. Intuitively, in monotonic reasoning, the more assumptions we have, the more conclusions we are able to draw. This in turn implies that monotonic reasoning is deductive, and deductive conclusions never have to be withdrawn. In comparison, in nonmonotonic reasoning, conclusions we drew earlier may have to be withdrawn in the face of new evidence. For example, if we are told that Tweety is a bird, then since typical birds we see every day fly, and we have no further information to the contrary, we would like to say that Tweety flies. However, if later we learn that Tweety is a dead bird, then since dead birds never fly, we will have to withdraw our earlier conclusion.

In response to this nonmonotonic aspect of commonsense reasoning, a class of so-called *nonmonotonic logics* have been proposed. They formalize nonmonotonic reasoning from various different, but closely related, angles.

There are three directions one can take in studying nonmonotonic logics. First,

[1]Computer commonsense?

there are foundational issues such as the semantics of the logics and the relationships among the logics. Second, there are questions of how the logics can be implemented. Finally, there are questions of how the logics can best be applied to formalizing commonsense reasoning.

This dissertation reports our work in all of the three directions. It begins with the introduction of the logic GK of *Grounded Knowledge* as a uniform semantic basis for fixed-point nonmonotonic logics such as Reiter's default logic and Moore's autoepistemic logic. To our knowledge, GK for the first time unifies the two logics semantically. As with circumscription, GK is based on the notion of logical minimization, and thus provides a bridge between circumscription and fixed-point nonmonotonic logics. As an application of this bridge, we propose a formalization of logic programs with negation-as-failure in circumscription. Together with the unique names assumption, the formalization provides a concise representation for the facts that are true in every answer set of a logic program.

We then introduce the notion of *argument systems* as a proof theory for nonmonotonic reasoning. By reformulating some major existing nonmonotonic logics as argument systems, we show that most nonmonotonic reasoning can be captured with the aid of a generalized negation-as-failure rule. We describe an implemented system for default logic based on this idea.

Finally, we apply nonmonotonic logic to the formalization of inheritance and of theories of action, the two areas that have to a large degree motivated research in nonmonotonic reasoning. We show that various intuitions about nonmonotonic inheritance hierarchies can be conveniently formalized in logic programs with negation-as-failure. For reasoning about action, we first propose a formal yet intuitive criterion by which to evaluate the adequacy of theories of action. We then formulate a class of monotonic theories that satisfy this criterion by using explicit frame axioms. Finally for a significant subclass of the monotonic theories, we provide provably-correct nonmonotonic counterparts which avoid explicit frame axioms.

Chapter 2

A Brief Review of Existing Nonmonotonic Logics

In this chapter, we briefly review major existing nonmonotonic logics. In the later chapters, we shall study the relationships among them, their semantics, and their applications. We shall begin with Shoham's general preferential semantics for nonmonotonic logics [Shoham 1987], then move on to circumscription [McCarthy 1980, 1986, Lifschitz 1985], default logic [Reiter 1980], autoepistemic logic [Moore 1983, Konolige 1988], and negation-as-failure [Clark 1978, Gelfond and Lifschitz 1988, and others]. We shall only give basic definitions. The reader is referred to the original publications for more details.

2.1 Preferential semantics

Preferential semantics was proposed by Shoham [1987] as a general semantic framework for nonmonotonic logic. Beginning with any given logic $\mathcal{L}$, and a strict partial order $\sqsubset$ (called a *preference order*) over the set of the interpretations of $\mathcal{L}$, Shoham constructs a nonmonotonic logic $\mathcal{L}_\sqsubset$ (called a *preference logic*) as follows.

The syntax of $\mathcal{L}_\sqsubset$ is the same as that of $\mathcal{L}$. The semantical entailment of $\mathcal{L}_\sqsubset$ is defined as follows. An interpretation M *preferentially satisfies* a set of sentences Σ if:

1. M satisfies Σ in $\mathcal{L}$.

2. There is no M' such that M' satisfies Σ in $\mathcal{L}$, and $M' \sqsubset M$.

A set of sentences Σ *preferentially entails* a sentence φ, written $\Sigma \models_{\sqsubset} \varphi$, if for any interpretation M, M satisfies φ in $\mathcal{L}$ whenever M *preferentially satisfies* Σ.

We say that an interpretation M is a *preferred* model of a sentence A if M preferentially satisfies $\{A\}$. Thus M is a preferred model of A if it is a model of A, and is minimal according to $\sqsubset$.

2.2 Circumscription

Circumscription [McCarthy 1980] is one of the oldest and best-known general formalisms for nonmonotonic reasoning. Viewed syntactically, it is a second-order sentence. Viewed semantically, it is a kind of minimization.

Let P be a tuple of predicate symbols, and x a tuple of object variables. Let $A(P)$ and $E(P,x)$ be formulas of second-order logic such that if a predicate in P appears in A or E, then it appears there free, and if a variable in x appears in E, then it appears there free. The *circumscription* of $E(P,x)$ in $A(P)$ with P allowed to vary is the following formula, written $Circum(A(P); E(P,x))$:

$$A(P) \wedge \forall P'(A(P') \supset \neg(E(P',x) < E(P,x)), \tag{2.1}$$

where P' is a tuple of predicate symbols, and similar to P, $E(P',x) < E(P,x)$ is the formula:

$$\forall x(E(P',x) \supset E(P,x)) \wedge \neg\forall x(E(P,x) \supset E(P',x)).$$

For brevity we are writing $\forall x$ instead of $\forall x_1 ... x_n$, where $x = (x_1, ..., x_n)$. We notice here that $A(P)$ may contain predicate symbols that are not in P. $A(P)$ may also contain free variables. Similarly, $E(P,x)$ may contain free predicate symbols not in P, and free variables not in x. Notice that predicate symbols not in P are not allowed to vary.

The semantics of $Circum(A(P); E(P,x))$ can be defined in terms of preferential semantics as follows. Let P, x, and $E(P,x)$ are as those in the above definition of circumscription. For any structures M_1, M_2, and any variable assignment σ, we define $M_1, \sigma \sqsubseteq_{E(P,x)} M_2, \sigma$ if:

1. M_1 and M_2 have the same domain, and they interpret everything the same except for the predicate symbols in P.
2. For any variable assignment σ' that agrees with σ on every variable except those in x, if $M_1, \sigma' \models E(P, x)$, then $M_2, \sigma' \models E(P, x)$.

We write $M_1, \sigma \sqsubset_{E(P,x)} M_2, \sigma$ if $M_1, \sigma \sqsubseteq_{E(P,x)} M_2, \sigma$, but $M_2, \sigma \not\sqsubseteq_{E(P,x)} M_2, \sigma$.

Proposition 2.1 *A structure M and a variable assignment σ satisfy*

$$Circum(A(P); E(P, x))$$

iff M, σ is a preferred (minimal) model of A according to $\sqsubset_{E(P,x)}$.

Our definition of circumscription is basically adopted from [McCarthy 1986] and [Lifschitz 1985]. Our definition is more general in the sense that it allows $A(P)$ to contain free variables, and $E(P, x)$ to contain free variables other than those in x. When P is the tuple of the free predicates in E, and x is the tuple of the free individual variables in E, then we shall abbreviate $Circum(A(P); E(P, x))$ as $Circum(A; E)$.

2.3 Default logic

Default logic [Reiter 1980] enjoys the same popularity as circumscription. In some applications such as logic programs, default logic has been shown to be more successful [Gelfond and Lifschitz 1990].

A *default* is an expression of the form:

$$\alpha : \beta_1, ..., \beta_n/\gamma, \tag{2.2}$$

where $\alpha, \beta_1, ..., \beta_n, \gamma$ are first-order formulas. α is called the *prerequisite*, γ the *consequence*, and $\beta_1, ..., \beta_n$ the *assumptions* of the default. A default is *closed* if none of the formulas in it contain a free variable. Otherwise, it is called *open*. An open default is usually equated with the set of closed defaults obtained by replacing the free variables with ground terms.[1] In the following, we shall consider only closed defaults.

[1]Lifschitz [1990a] recently proposed a theory in which variables in open defaults are considered to be genuine variables.

A *default theory* is a pair (W, D), where W is a set of sentences, and D a set of defaults. A default theory (W, D) is *closed* if every default in D is closed.

Let $\Delta = (W, D)$ be a closed default theory. A set of sentences E is a *default extension* of Δ if $E = \Gamma(E)$, where Γ is the following operator. For any set of sentences S, $\Gamma(S)$ is the smallest set satisfying the following three properties:

1. $W \subseteq \Gamma(S)$.

2. $\Gamma(S)$ is closed under classical first-order deduction.

3. If $(\alpha : \beta_1, ..., \beta_n/\gamma) \in D$, $\alpha \in \Gamma(S)$, and $\neg\beta_1, ..., \neg\beta_n \notin S$, then $\gamma \in \Gamma(S)$.

The following proposition seems central for default logic:

Proposition 2.2 (Reiter 1980) *A set of sentences E is a default extension of the closed default theory $\Delta = (W, D)$ iff*

$$E = \bigcup_{i=0}^{\infty} E_i,$$

where the E_i's are defined as follows: $E_0 = W$, and for any $i \geq 0$,[2]

$$E_{i+1} = Th(E_i) \cup \{\gamma \mid (\alpha : \beta_1, ..., \beta_n/\gamma) \in D, \alpha \in E_i, \neg\beta_1, ..., \neg\beta_n \notin E\},$$

where for any set S, $Th(S)$ is the logical closure of S under classical first-order deduction.

2.4 Autoepistemic logic

Autoepistemic logic was proposed by Moore [1983] as a logic for reasoning about one's own beliefs. Unlike circumscription and default logic, it uses a modal language.

Following Moore, we shall consider only the propositional case. The extension to the first-order case without quantifying-in is straightforward [Konolige 1988]. Let $\mathcal{L}$ be a propositional language augmented with a modal operator L. Intuitively, for any

[2]Notice the appearance of E in the definition of E_{i+1}.

sentence P, LP means that P is believed. An *autoepistemic sentence* is a sentence in $\mathcal{L}$, and an *autoepistemic theory* is a set of autoepistemic sentences.

Let W be an autoepistemic theory. A set of sentences T is a *stable expansion* of W if

$$T = Th(W \cup \{LP \mid P \in T\} \cup \{\neg LP \mid P \notin T\}),$$

where for any set of sentences S, $Th(S)$ is the tautological closure of S.

Konolige [1988] shows that autoepistemic logic satisfies weak S5 modal logic [Hughes and Cresswell 1972]. More precisely, if W_1 and W_2 are equivalent in weak S5, then for any set T, it is a stable expansion of W_1 iff it is a stable expansion of W_2. In particular, for any autoepistemic theory W, there is an equivalent theory W' such that every member of W' has the following form:

$$(LP_1 \wedge \neg LP_2 \wedge ... \wedge \neg LP_n) \supset P_0,$$

where $n \geq 0$, and the P_i's do not contain the epistemic operator L.

Konolige [1988] also shows that the default (2.2) corresponds to the following autoepistemic sentence:

$$(L\alpha \wedge \neg L\neg\beta_1 \wedge ... \wedge \neg L\neg\beta_n) \supset \gamma.$$

2.5 Negation-as-failure

The negation-as-failure rule has been used in answering queries for deductive data bases and logic programs for many years [Clark 1978]. Recently, it has been formally defined in various ways [van Gelder 1986, Gelfond 1987, Gelfond and Lifschitz 1988, Bidoit and Froidevaux 1988, Lin and Shoham 1989, and others].

A *logic program* is a set of rules of the form:

$$p_0 \leftarrow p_1, ..., p_m, \neg p_{m+1}, ..., \neg p_n, \tag{2.3}$$

where p_i, $0 \leq i \leq n$, is an atomic formula, and $0 \leq m \leq n$. If (2.3) contains variables, then it is an *open* rule, otherwise, it is a *closed* rule. Usually, an open rule is considered to be the set of closed rules obtained by replacing the variables with the closed terms. In the following, we consider only logic programs with closed rules.

The following definition of answer sets is taken from [Gelfond and Lifschitz 1988].[3]

Let P be a logic program in which no rules contain $\neg$. A set of ground atoms (atomic sentences) E is an *answer set* of P if it is the smallest set satisfying the following property:

For any rule $p_0 \leftarrow p_1, ..., p_m \in P$, if $p_1, ..., p_m \in E$, then $p_0 \in E$.

In general, a set of atoms E is an *answer set* of a logic program P if E is an answer set of $P(E)$, where $P(E)$ is the smallest set of rules satisfying the following property:

If $p_0 \leftarrow p_1, ..., p_m, \neg p_{m+1}, ..., \neg p_n \in P$, and $p_{m+1}, ..., p_n \notin E$, then

$$p_0 \leftarrow p_1, ..., p_m \in P(E),$$

where $p_0, ..., p_n$ are atoms.

Answer sets can also be defined by a translation from logic programs to default logic [Bidoit and Froidevaux 1988, Lin and Shoham 1989]. The rule (2.3) corresponds to the following default:

$$(p_1 \wedge \ldots \wedge p_m : \neg p_{m+1}, \ldots, \neg p_n)/p_0.$$

Thus we can characterize answer sets as follows. A set of atoms E is an answer set of a logic program P iff $\Gamma(E) = E$, where for any set S, $\Gamma(S)$ is the smallest set of atoms satisfying the following property:

If $p_0 \leftarrow p_1, \ldots, p_m, \neg p_{m+1}, \ldots, \neg p_n \in P$, $p_1, ..., p_m \in \Gamma(S)$, and $p_{m+1}, ..., p_n \notin S$, then $p_0 \in \Gamma(S)$.

[3]Actually, they are called stable sets there. In their more recent writings, the authors use the term "answer sets," which we adopt here.

Chapter 3

A Semantic Foundation: Grounded Knowledge

We have briefly reviewed major existing nonmonotonic logics[1]. We can divide them roughly into two kinds. One is based on fixed points, such as Reiter's default logic and Moore's autoepistemic logic. The other is based on minimal models, such as McCarthy's circumscription and the logic of minimal knowledge ([Halpern and Moses 1984], [Shoham 1987], and [Lin 1988]).

Although Shoham [1987] shows that preferential semantics provide a natural and uniform semantic basis for minimal-model nonmonotonic logics, a similar one for fixed-point nonmonotonic logics was still missing untill recently. In this chapter, we shall propose such a basis. More specifically, we shall provide possible-world epistemic semantics for default logic as well as for autoepistemic logic. Among other things, the semantics clearly show the relationship, both similarities and differences, between the two logics. Furthermore, our semantics are essentially based on the notion of minimal models, and thus provide a bridge between minimal-model and fixed-point nonmonotonic logics. In fact, they are largely motivated by the failure of Shoham's early attempt to provide preference semantics for default logic [Shoham 1988]. Shoham's key idea is to translate a closed default

$$p : q/r$$

[1]An extended abstract of this chapter appeared as [Lin and Shoham 1990].

into something like

$$Kp \wedge \neg K \neg q \supset Kr$$

with knowledge (K) minimized. Intuitively speaking, the translation fails due to its wrong interpretation of the above default as "if p is known and $\neg q$ is not known to be true, then r is also known to be true." This wrong interpretation does not meet the requirement that knowledge be not only minimal but also *grounded.*

For instance, consider the simple default theory $\Delta = (\emptyset, \{: p/\neg p\})$, where p is a primitive proposition. It can be easily seen that Δ has no extensions. However, under Shoham's translation, it corresponds to the sentence $\neg K \neg p \supset K \neg p$, which has a unique minimal model where $K \neg p$ is true.

Our new semantics can be thought of as accounting for groundedness of knowledge by the following informal reading of the same default rule: "if p is known to be true and $\neg q$ is not *assumed* to be true, then r is known to be true – provided the assumption can eventually be justified."

This suggests employing *two* epistemic modalities, K (for knowledge) and A (for assumption), which will be related to one another. Indeed, we will translate the above default rule into the sentence

$$Kp \wedge \neg A \neg q \supset Kr,$$

and the process of justifying the assumption that q is consistent will correspond to first minimizing the knowledge K with A fixed, and then comparing the resulting knowledge and assumptions to see whether they agree.

Consider again the default theory Δ. This time it is translated into the sentence $\neg A \neg p \supset Kp$. There are two assumptions we can make about $\neg p$: $A \neg p$ or $\neg A \neg p$. If we assume $A \neg p$, then $\neg A \neg p \supset K \neg p$ is automatically true. Thus we shall know nothing except tautologies when knowledge is minimized. In particular, $\neg p$ is not known, but this violates our earlier hypothesis that $A \neg p$ is true. If we assume $\neg A \neg p$, then $\neg A \neg p \supset K \neg p$ implies that $\neg p$ must be known, again a violation of the hypothesis. Therefore, there is no way we can justify our assumption if we minimize knowledge in $\neg A \neg p \supset K \neg p$. This agrees with the fact that Δ has no default extension.

Interestingly, the treatment of autoepistemic logic will be almost identical. It was shown in [Konolige 1988] (see also Section 2.4) that any autoepistemic theory can be transformed into an equivalent one in which every sentence has the form

$$Lp \wedge \neg L\neg q_1 \wedge \ldots \wedge \neg L\neg q_n \supset r.$$

We will translate each such sentence into

$$Ap \wedge \neg A\neg q_1 \wedge \ldots \wedge \neg A\neg q_n \supset Kr,$$

and keep the rest as in the default logic translation. Thus the only change required when moving from default logic to autoepistemic logic is the replacement of the first K by an A!

In recent years, we have seen some related work. Siegel [1990] also explicitly distinguishes between knowledge and assumption. However, Siegel defines assumption in terms of knowledge, and uses a different minimization strategy. As a result, Siegel is only able to account for default logic. There are a number of related papers by researchers at the University of Kentucky [Marek and Truszczynski 1990, Marek *et al.* 1991, and Truszczynski 1991]. The tool used in these papers is McDermott and Doyle's fixed-point construction [McDermott and Doyle 1980]. They show how default logic and autoepistemic logic, among others, can be captured by varying underlying monotonic logics. An interesting open question is how to capture this class of logics in epistemic semantics like GK.

This chapter is organized as follows. We first define our basic logical language, which is simply a propositional one augmented by two epistemic modalities. Then we define the logic GK of Grounded Knowledge, which is the result of imposing an augmented version of preference semantics on our basic language. We then offer translations of both default logic and autoepistemic logic into GK, translations that are similar but subtly different. We prove the correctness of the translations, and show how the common framework explicates the relationship between the two logics. Finally, as an application of the bridge GK provides between minimal-model and fixed-point nonmonotonic logics, we show a formalization of logic programs in circumscription.

3.1 The basic logical language

Our language is a propositional one, augmented with two modalities K and A, which are at this point mutually unconstrained. Well-formed formulas are defined as usual. Intuitively $K\varphi$ means that φ is known or believed (the distinctions between the two are not important in this paper) to be true, while $A\varphi$ means that φ is assumed to be true.

A *Kripke structure* is a tuple (W, π, R_K, R_A), where W is a nonempty set, $\pi(w)$ a truth assignment to the primitive propositions for each $w \in W$, and R_K, R_A are binary relations over W (the accessibility relations for K and A, respectively). A *Kripke interpretation* M is a pair $(w, (W, \pi, R_K, R_A))$, where $w \in W$ and (W, π, R_K, R_A) is a Kripke structure.

We have not placed restrictions on the two accessibility relations so far. Indeed, our results are surprisingly insensitive to the properties of these relations.[2] In particular, the reader may assume any of the major systems that have been used to capture epistemic notions – $S5$, $K45$, $KD45$ or $S4$.

The satisfaction relation "$\models$" between Kripke interpretations and formulas is defined as follows:

1. $(w, (W, \pi, R_K, R_A)) \models p$ iff $\pi(w)(p) = 1$, where p is a primitive proposition.

2. $M \models \varphi_1 \vee \varphi_2$ iff $M \models \varphi_1$ or $M \models \varphi_2$.

3. $M \models \neg\varphi$ iff it is not the case that $M \models \varphi$.

4. $(w, (W, \pi, R_K, R_A)) \models K\varphi$ iff $(w', (W, \pi, R_K, R_A)) \models \varphi$ for any $w' \in W$ such that $(w, w') \in R_K$.

5. $(w, (W, \pi, R_K, R_A)) \models A\varphi$ iff $(w', (W, \pi, R_K, R_A)) \models \varphi$ for any $w' \in W$ such that $(w, w') \in R_A$.

[2] Many people have commented on this fact. Joham van Benthem (private communication) noted that our results are insensitive to the underlying modal logic because we only consider non-nested modal sentences on which the usual hierarchies of modal logics collape. We agree.

We say that a Kripke interpretation M is a *model* of a set of formulas S if M satisfies every member of S.

We conclude this section with two definitions that will be used throughout the remainder of the chapter: For each Kripke interpretation M,

$$K(M) = \{\varphi \mid M \models K\varphi, \varphi \text{ is a base formula}\},$$

and

$$A(M) = \{\varphi \mid M \models A\varphi, \varphi \text{ is a base formula}\},$$

where a *base formula* is one that does not contain modal operators.

3.2 Preference semantics and the logic GK

In this section we modify the semantics of our basic language, defining the logic GK. We start by endowing our basic language with a preference semantics (see section 2.1 or [Shoham 1987]).

To do so we define a preference relation on the Kripke interpretations defined in the previous section. The following relation "$\sqsubset$" has the effect of minimizing knowledge with assumptions fixed.

Definition 3.1 *Let M_i, $i = 1, 2$, be two Kripke interpretations. We say that M_1 is preferred over M_2, written $M_1 \sqsubset M_2$, if:*

1. $A(M_1) = A(M_2)$.
2. $K(M_1) \subset K(M_2)$.

M is a *minimal* model of S if M is a model of S and there is no other model M' of S such that $M' \sqsubset M$.

These semantics, however, are not sufficient for our purposes, as they do not capture the property of "groundedness" of knowledge. As was said in the introduction, the assumptions made by the agent should be justified. We capture this justification by an added requirement of the minimal models – that the assumptions coincide with the knowledge. This naturally leads to the following definition:

Definition 3.2 *Let S be a set of formulas, and M a Kripke interpretation. We say that M is a* preferred model *of S if:*

1. *M is a minimal model of S.*

2. *$K(M) = A(M)$.*

Definition 3.3 *The logic GK is defined as follows:*

Syntax: The syntax of our basic language.

Semantic entailment: $\Phi \models_{GK} \varphi$ iff φ holds in all preferred models of Φ.

Example 3.1 In the following, p, q, r are primitive propositions.

$S_1 = \{Kp\}$ has a unique preferred model in the sense that M is a preferred model of $\{Kp\}$ iff $K(M)$ is the tautological closure of p, and $A(M) = K(M)$.

$S_2 = \{\neg Ap \supset Kp\}$ has no preferred models. For let M be a preferred model of S_2. If Ap is true in M, then Kp must be false because of the minimality of M. But then $K(M) \neq A(M)$ and M can not be a preferred model of S_2. Now if Ap is false in M, then Kp must be true in M because it is a model of S_2, and this again will make $K(M) \neq A(M)$.

As we will see in the next section, S_1 corresponds to the default theory $(\{p\}, \emptyset)$, and S_2 to $(\emptyset, \{: \neg p/p\})$.

$S_3 = \{Kp \vee Kq, Kp \wedge \neg A\neg r \supset Kr, Kq \wedge \neg A\neg r \supset Kr\}$. We can distinguish two kinds of preferred models of S_3: One in which $Kp \wedge \neg Kq$ and Kr are true; and the other in which $Kq \wedge \neg Kp$ and Kr are true. Notice that Kr is true in all preferred models.

There is no default theory that corresponds to S_3. In fact, S_3 is an instance of a disjunctive default theory [Gelfond *et al.* 1991]. For instance, we can interpret p as "Tweety is a penguin", q as "Tweety has a broken wing", and r as "Tweety does not fly".

■

We conclude this section by proving a generalization of Reiter's Theorem 2.1 in [Reiter, 1980] (Proposition 2.2 in Section 2.3).

Theorem 1 *Let $S = S_1 \cup S_2$ (either S_1 or S_2 may be empty) be a set of formulas such that every member of S_1 has the form Kp and every member of S_2 the form*

$$Kp \wedge Aq \wedge \neg Ar_1 \wedge \ldots \wedge \neg Ar_n \supset Ks,$$

where p, q, r_1, ..., r_n, s are base formulas, $n \geq 0$, and both Kp and Aq may be absent. Then a Kripke interpretation M is a preferred model of S iff:

1. $K(M) = A(M)$.
2. $K(M) = E_1 \cup E_2 \cup ...$, *where E_i, $i = 1, 2, ...$, are defined inductively as follows:*
   (a) $E_1 = \{p \mid Kp \in S_1\}$.
   (b) $E_{i+1} = Th(E_i) \cup \{s \mid Kp \wedge Aq \wedge \neg Ar_1 \wedge \ldots \wedge \neg Ar_n \supset Ks \in S_2$, *where* $p \in E_i, q \in A(M)$ *and* $r_1, r_2, ..., r_n \notin A(M)\}$, *where $i > 0$ and $Th(E_i)$ is the tautological closure of E_i.*

Proof: In the following, let $E = E_1 \cup E_2 \cup ....$

($\Leftarrow$): By the definition of E_i, $i = 1, 2, ...$, it is easy to see that M is a model of S. M must be minimal for if M' is a model of S such that $A(M) = A(M')$, then it is easy to prove by induction on i that $E_i \subseteq K(M')$ for $i > 0$; thus $K(M) = E \subseteq K(M')$. Thus M is a preferred model of S.

($\Rightarrow$): Since M is a model of S, as we have proved in the ($\Leftarrow$) part, we have $E \subseteq K(M)$. Thus E is a tautologically consistent set of sentences. Therefore we can construct a Kripke interpretation M' such that $A(M') = A(M)$ and $K(M') = E$. By the definition of the E_i's, M' must be a model of S. Thus by the minimality of M, we must have $K(M) = K(M') = E$. ■

3.3 Fixed-point nonmonotonic logics

The two major fixed-point nonmonotonic logics in the AI literature are Reiter's default logic [Reiter, 1980] and Moore's autoepistemic logic [Moore, 1985]. By the above

Theorem 1 and Theorem 2.1 in [Reiter, 1980] (also Proposition 2.2 in Section 2.3), it is straightforward to translate default theories into sets of formulas in our logic GK. As we shall show in this section, a closely related but subtly different translation also exists for autoepistemic theories.

3.3.1 Default logic

Recall from section 2.3 that default logic was originally defined with respect to a first-order language. For the sake of consistency with autoepistemic logic, the default logic used here is with respect to a propositional language. It is straightforward to extend the results to the first-order case (this is true because open defaults are defined as shorthands for sets of closed defaults). In this subsection, we define sentences as base formulas.

Recall that a default theory, adapted to a propositional language, is a pair (W, D), where W is a set of sentences and D a set of defaults, which are expressions of the form

$$p : q_1, \ldots, q_n / r,$$

where $p, q_1, \ldots, q_n, r$ are sentences.

A default theory $\Delta = (W, D)$ is translated into the following set of formulas Δ_{GK} in GK:

1. If $p \in W$, then $Kp \in \Delta_{GK}$.

2. If $(p : q_1, \ldots, q_n / r) \in D$, then

$$Kp \wedge \neg A \neg q_1 \wedge \ldots \wedge \neg A \neg q_n \supset Kr \in \Delta_{GK}.$$

Theorem 2 *A consistent set of sentences E is a default extension of Δ iff there is a preferred model M of Δ_{GK} such that $E = K(M)$.*

Proof: By Theorem 1 and Theorem 2.1 in [Reiter, 1980], and the fact that if E is consistent, then there is a Kripke interpretation M such that $K(M) = A(M) = E$. ■

3.3.2 Autoepistemic logic

Recall from section 2.4 that the language of autoepistemic logic is a propositional one augmented by a modal operator L for belief. In the following, by a L-sentence we mean one that may contain the modal operator L but not K nor A.

Without loss of generality, let S be a set of L-sentences of the form (see section 2.4):

$$Lp \wedge \neg Lq_1 \wedge \ldots \wedge \neg Lq_n \supset r,$$

where $p, q_1, ..., q_n, r$ are base formulas, $n \geq 0$, and Lp may be absent.[3]

For any such S, we define S_{GK} by the following equation:

$$S_{GK} = \{Ap \wedge \neg Aq_1 \wedge \ldots \wedge \neg Aq_n \supset Kr \mid Lp \wedge \neg Lq_1 \wedge \ldots \wedge \neg Lq_n \supset r \in S\}.$$

Parallel to Theorem 2, we have the following result:

Theorem 3 *A consistent stable set of L-sentences E is a stable expansion of S iff there is a preferred model M of S_{GK} such that $K(M) = Base(E)$, where $Base(E)$ is the set of base formulas in E.*

By Theorem 1, the theorem is equivalent to the following proposition which was announced in [Lin and Shoham 1989] and independently proved in [Marek and Truszczynski 1989]:

Proposition 3.1 *A stable set E is a stable expansion of S iff $Base(E) = E_2$ where*

1. $E_1 = \{p \mid p \in S \textit{ is a base formula}\}$.
2. $E_2 = Th(E_1 \cup \{r \mid Lp \wedge \neg Lq_1 \wedge \ldots \wedge \neg Lq_n \supset r \in S, \textit{ where } p \in Base(E) \textit{ and } q_1, ..., q_n \notin Base(E)\})$.

[3]Theoretically, it is not necessary to use the *normal form*. Under the assumption that K and A satisfy the weak S5 system, an arbitrary L-sentence can be translated into a sentence in our language by inserting K in front of the L-sentence and replacing every L by A. The following Theorem 3 will also be true for this transformation.

Proof:

($\Rightarrow$): First it is easy to see that $E_i \subseteq E$, $i = 1, 2$. Thus $E_2 \subseteq Base(E)$. On the other hand, if $A \in Base(E)$, then A is a tautological consequence of

$$S \cup \{L\varphi \mid \varphi \in E\} \cup \{\neg L\varphi \mid \varphi \notin E\}.$$

Because of the normal form of S, A must be a tautological consequence of

$$S \cup \{L\varphi \mid \varphi \in Base(E)\} \cup \{\neg L\varphi \mid \varphi \notin Base(E)\},$$

thus a tautological consequence of

$$\{r \mid Lp \wedge \neg Lq_1 \wedge \ldots \wedge \neg Lq_n \supset r \in S, \text{ where } p \in Base(E) \text{ and } q_1, ..., q_n \notin Base(E)\}.$$

Hence A must be in E_2. Therefore $Base(E) = E_2$.

($\Leftarrow$): Without loss of generality, we can assume that E is tautologically consistent. First we prove that $S \subseteq E$. Let

$$\varphi = Lp \wedge \neg Lq_1 \wedge \ldots \wedge \neg Lq_n \supset r \in S.$$

There are two cases:

1. $p \in E$ and $q_1, \ldots, q_n \notin E$: Since E is a stable set and $Base(E) = E_2$, we have $r \in E$ and $\varphi \in E$.

2. Otherwise, for instance, $p \notin E$, then $\neg Lp \in E$. But $\neg Lp \vdash \varphi$, thus $\varphi \in E$. Other cases are similar.

Let

$$E' = Th(S \cup \{L\varphi \mid \varphi \in E\} \cup \{\neg L\varphi \mid \varphi \notin E\}).$$

We have that $E' \subseteq E$. Suppose $\varphi \in E$, there are several cases:

1. φ is a base sentence. Then we have $\varphi \in Base(E) = E_2 \subseteq E'$.

2. φ has the form $L\phi$. Since E is a stable set, thus $\phi \in E$. Therefore $L\phi \in E'$.

3. φ has the form $\neg L\phi$. Since E is consistent, we have $\phi \notin E$. Thus $\neg L\phi \in E'$.

4. φ is a disjunction of sentences of the forms in the above three cases. Then one of the disjuncts must be in E, and thus in E'. Therefore $\varphi \in E'$.

5. Finally for arbitrary φ, it is tautologically equivalent to a conjunction of sentences of the form in the above case. Thus all of the conjuncts must be in E, and hence in E'. Therefore φ must be in E'.

Therefore $E \subseteq E'$. Thus $E = E'$ and E is a stable expansion of S. ∎

In summary, by Theorem 2, we have the following correspondence between GK and default logic:

$$p : q_1, \ldots, q_n/r \Leftrightarrow Kp \wedge \neg A \neg q_1 \wedge \ldots \wedge \neg A \neg q_n \supset Kr,$$

and by Theorem 3, we have the following correspondence between GK and autoepistemic logic:

$$Lp \wedge \neg L \neg q_1 \wedge \ldots \wedge \neg L \neg q_n \supset r \Leftrightarrow Ap \wedge \neg A \neg q_1 \wedge \ldots \wedge \neg A \neg q_n \supset Kr.$$

Thus the difference between default and autoepistemic logics lies in their different interpretations of the premise p. While default logic treats the premise p and the conclusion r in the same way (as knowledge, in our terminology), autoepistemic logic treats the premise p and the consistency assumptions q's in the same way (as assumptions, in our terminology). The effect of this difference is that the notion of default extensions is stronger than that of stable expansions, that is, under Konolige's transformation [Konolige 1988]

$$p : q_1, \ldots, q_n/r \Leftrightarrow Lp \wedge \neg L \neg q_1 \wedge \ldots \wedge \neg L \neg q_n \supset r,$$

if E is a default extension of a default theory and T is a stable set such that $Base(T) = E$, then T is also a stable expansions of the corresponding autoepistemic theory, but the converse is not true in general [Konolige 1988]. There is, however, a special case in which the converse is also true. If Δ is a default theory without premises, that is, if every default in Δ has the form

$$: q_1, \ldots, q_n/r,$$

then Konolige's transformation is exact, that is, a set of base sentence E is a default extension of Δ iff there is a stable set T such that $Base(T) = E$ and T is a stable expansion of the corresponding autoepistemic theory. This result is also proved in [Lin and Shoham 1989], and now follows trivially from Theorem 2 and Theorem 3.

3.4 Negation-as-failure as circumscription

We now consider an application of the logic GK. As with circumscription, our logic GK is based on minimization. In fact, if we reify propositions, then GK can be captured by the following conjunction of a circumscription with a first-order sentence:

$$Circum(S; Known) \land \forall x(Known(x) \equiv Assumed(x)).$$

In the general case, the price of reification seems too high[4]. However, in logic programs, where the sentences in a rule are literals, reification is easy. In this case, the above connection between GK and circumscription yields an interesting relationship between logic programs and circumscription. In this section, we shall explore this relationship. Our work can be considered an extension of [Lifschitz 1987a] and [Przymusinski 1989], which provide semantics for stratified logic programs in circumscription.

3.4.1 From logic programs to circumscription

Recall that logic programs consist of rules of the following form:

$$p_0 \leftarrow p_1, ..., p_m, \neg p_{m+1}, ..., \neg p_n, \tag{3.1}$$

where $p_0, ..., p_n$ are atoms. The rule corresponds to the following default (see Section 2.5):

$$(p_1 \land \ldots \land p_m : \neg p_{m+1}, \ldots, \neg p_n)/p_0,$$

which is translated into the following sentence in GK (see Section 3.3.1):

$$(Kp_1 \land ... \land Kp_m \land \neg Ap_{m+1} \land ... \land \neg Ap_n) \supset Kp_0,$$

which is "reified" into the following first-order sentence:

$$p_1 \land ... \land p_m \land \neg p'_{m+1} \land ... \land \neg p'_n \supset p_0, \tag{3.2}$$

[4]John McCarthy pointed out that for propositional logic, we only need about five axioms. But the effects of circumscribing predicates in the axioms seem hard to capture, witness the mechanisms needed in [Lin 1988]. In any event, the advantages that circumscription can bring us in the general case by reification are less clear than in the special case of logic programs.

where for any first-order formula F, F' is the result of replacing every predicate P in F by a corresponding new predicate P' of the same arity. Instead of minimizing the operator K with A fixed, we minimize all predicates in the original language with the new predicates fixed, and instead of equating K and A after minimization, for every predicate P, we equate it with the corresponding P'. Let us see a simple example.

Example 3.2 Let $\mathcal{P}$ be a logic program with the following rules:

$$Q(b),$$
$$Q(a) \leftarrow Q(b), \neg Q(c).$$

$E = \{Q(b), Q(a)\}$ is the unique answer set of $\mathcal{P}$. Now let F be the conjunction of the following sentences:

$$a \neq b \neq c,$$
$$Q(b) \wedge (Q(b) \wedge \neg Q'(c) \supset Q(a)).$$

It can be shown that $Circum(F; Q(x))$ is equivalent to

$$F \wedge [Q'(c) \supset \forall x(Q(x) \equiv x = b)] \wedge [\neg Q'(c) \supset \forall x(Q(x) \equiv x = b \vee x = a)].$$

Thus the sentence $\forall x(Q(x) \equiv Q'(x)) \wedge Circum(F; Q(x))$ is equivalent to the following one:

$$F \wedge \forall x(Q(x) \equiv Q'(x)) \wedge \forall x(Q(x) \equiv x = b \vee x = a),$$

which implies the set of ground atoms given by E. ■

In general, for any given finite logic program $\mathcal{P}$, let $C(\mathcal{P})$ be the conjunction of the sentences obtained by translating every rule (3.1) in $\mathcal{P}$ into the universal closure of (3.2). If $\mathcal{P}$ contains only closed rules, then let $EQ(\mathcal{P})$ be the unique names assumption for the closed terms in $\mathcal{P}$.

Theorem 4 *Let $\mathcal{P}$ be a finite logic program with only closed rules. For any set E of ground atoms, E is an answer set of $\mathcal{P}$ iff there is a model M of the following sentence*

$$\bigwedge_{P \in \mathcal{P}} \forall x(P(x) \equiv P'(x)) \wedge Circum(EQ(\mathcal{P}) \wedge C(\mathcal{P}); \bigwedge_{P \in \mathcal{P}} P(x)) \tag{3.3}$$

such that E is the set of the ground atoms in $\mathcal{P}$ satisfied in M.

Proof: Suppose E is an answer set of $\mathcal{P}$. Then E will contain only ground atoms that appear in $\mathcal{P}$. Construct an Herbrand model M satisfying the following property

> For any predicate $P \in \mathcal{P}$, any tuple of closed terms t, $M \models P(t)$ iff $M \models P'(t)$ iff $P(t) \in E$.

We show that M satisfies (3.3). Since M is an Herbrand model, thus it satisfies $EQ(\mathcal{P})$. $M \models C(\mathcal{P})$ follows easily from the construction of M, and the assumption that E is an answer set. We now show that M is minimal. If not, suppose there is a M' such that $M' \models C(\mathcal{P})$, and $M' \sqsubset_{\bigwedge_{P\in\mathcal{P}} P(x)} M$. Let E' be the set of ground atoms that are true in M' and appear in $\mathcal{P}$. It can be seen that E' satisfies the property for $\Gamma(E)$ in section 2.5. Thus $\Gamma(E) \subseteq E' \subset E = \Gamma(E)$, a contradiction. Therefore, M must be minimal, and satisfy (3.3).

Suppose M is a model of (3.3). Let E be the set of the ground atoms that are true in M and appear in $\mathcal{P}$. We show that E is an answer set of $\mathcal{P}$, i.e. $E = \Gamma(E)$. It is easy to see that E satisfies the property for $\Gamma(E)$. Thus $\Gamma(E) \subseteq E$. Now let M' be just like M except that for any ground atoms p in $\mathcal{P}$, $M' \models p$ iff $p \in \Gamma(E)$. This is a sound definition since M is a model of $EQ(\mathcal{P})$. It is easy to check that M' is a model of $C(\mathcal{P})$, and $M' \sqsubseteq_{\bigwedge_{P\in\mathcal{P}} P(x)} M$. Thus $M = M'$, and $E = \Gamma(E)$. ■

Corollary 4.1 *Under the assumptions of the theorem, a ground atom in $\mathcal{P}$ is in every answer set of $\mathcal{P}$ iff it is a logical consequence of (3.3).*

The proof of Theorem 4 also gives the following result:

Theorem 5 *Let $\mathcal{P}$ be a finite logic program. A set E is an answer set of $\mathcal{P}$ iff there is an Herbrand model M such that (1) for any ground atom p in the language of $\mathcal{P}$, $p \in E$ iff $M \models p$, and (2) M is a model of the following sentence:*

$$\bigwedge_{P\in\mathcal{P}} \forall x(P(x) \equiv P'(x)) \wedge Circum(C(\mathcal{P}); \bigwedge_{P\in\mathcal{P}} P(x)).$$

For finite logic programs with free variables, we may still have the corresponding unique names assumption for them. In this case, Theorem 4 will be applicable.

Example 3.3 [Gelfond and Lifschitz 1988] Consider the following program $\mathcal{P}$:

$$\begin{aligned} &P(1,2), \\ &Q(x) \quad \leftarrow \quad P(x,y), \neg Q(y). \end{aligned}$$

The unique names assumption for $\mathcal{P}$ is $1 \neq 2$. Let A be the following sentence

$$1 \neq 2 \wedge P(1,2) \wedge \forall xy(P(x,y) \wedge \neg Q'(y) \supset Q(x)).$$

Then $Circum(A; P(x,y) \wedge Q(z))$ is equivalent to

$$A \wedge \forall xy[P(x,y) \equiv (x = 1 \wedge y = 2)] \wedge \forall x[Q(x) \equiv (x = 1 \wedge \neg Q'(2))]$$

Thus $\mathcal{P}$ is captured by the following sentence:

$$\forall xy(P(x,y) \equiv P'(x,y)) \wedge \forall x(Q(x) \equiv Q'(x)) \wedge Circum(A; P(x,y) \wedge Q(z)),$$

which is equivalent to the following sentence as far as P and Q are concerned:

$$\forall xy(P(x,y) \equiv x = 1 \wedge y = 2) \wedge \forall x(Q(x) \equiv x = 1) \wedge 1 \neq 2,$$

which concisely summarize the unique answer set of $\mathcal{P}$: $\{P(1,2), Q(1)\}$. ■

It is particularly interesting to note that if a logic program has more than one answer set, then the circumscription of the program gives us the intersection of the answer sets.

Example 3.4 Consider the following logic program $\mathcal{P}$:

$$\begin{aligned} P(a) &\leftarrow \neg P(b), \\ P(b) &\leftarrow \neg P(a). \end{aligned}$$

The circumscription of P in

$$(\neg P'(a) \supset P(b)) \wedge (\neg P'(b) \supset P(a)) \wedge a \neq b$$

is

$$\forall x(x \neq a \wedge x \neq b \supset \neg P(x)) \wedge (\neg P'(a) \equiv P(b)) \wedge (\neg P'(b) \equiv P(a)) \wedge a \neq b.$$

Together with $\forall x(P(x) \equiv P'(x))$, it gives

$$P(a) \equiv \neg P(b) \wedge \forall x(x \neq a \wedge x \neq b \supset \neg P(x)).$$

■

3.4.2 Disjunctive data bases

In this subsection, we show that our formalization of logic programs in circumscription can be generalized to logic programs with disjunctive heads (see, for example, [Przymusinski 1990] and [Gelfond and Lifschitz 1991]).

A *disjunctive data base* is a set of rules of the following form:

$$p_0|p_1|...|p_l \leftarrow p_{l+1}, ..., p_m, \neg p_{m+1}, ..., \neg p_n, \tag{3.4}$$

where $p_0, ..., p_n$ are atoms, and $0 \leq l \leq m \leq n$. Intuitively, $p_0|p_1|...|p_l$ is $p_0 \vee p_1 \vee ... \vee p_l$, i.e., one of p_i, $0 \leq i \leq l$, must be true.

The notion of answer sets can be extended to disjunctive data bases [Przymusinski 1990] and [Gelfond and Lifschitz 1991]. Let $\mathcal{P}$ be a disjunctive data base with only closed rules.[5] A set of atoms E is an answer set of $\mathcal{P}$ if $E = \Gamma(E)$, where for any set of atoms S, $\Gamma(S)$ is the smallest set satisfying the following condition:

> For any rule (3.4) in $\mathcal{P}$, if $p_{l+1}, ..., p_m \in \Gamma(S)$, and $p_{m+1}, ..., p_n \notin S$, then one of $p_0, ..., p_l$ must be in $\Gamma(S)$.

Again, for any finite disjunctive data base $\mathcal{P}$, let $C(\mathcal{P})$ be the disjunction of the sentences obtained by translating every rule (3.4) in $\mathcal{P}$ into the universal closure of the following sentence:

$$p_{l+1} \wedge ... \wedge p_m \wedge \neg p'_{m+1} \wedge ... \wedge \neg p'_n \supset p_0 \vee ... \vee p_l, \tag{3.5}$$

where for any first-order formula F, F' is the result of replacing every predicate P in F by a corresponding new predicate P' of the same arity. If $\mathcal{P}$ contains only closed rules, then let $EQ(\mathcal{P})$ be the unique names assumption for closed terms in $\mathcal{P}$.

Theorem 4, Corollary 4.1, and Theorem 5 can be extended straightforwardly to disjunctive data bases.

Example 3.5 Consider $\mathcal{P}_1 = \{P(a)|P(b) \leftarrow\}$. There are two answer sets, $E_1 = \{P(a)\}$, and $E_2 = \{P(b)\}$.

[5] Again, an open rule is considered to be a shorthand for the set of the closed rules obtained by replacing the variables by the closed terms.

$C(\mathcal{P}_1)$ is $P(a) \vee P(b)$, and $EQ(\mathcal{P}_1)$ is $a \neq b$. The circumscription of P in $C \wedge EQ$ is

$$a \neq b \wedge [\forall x(P(x) \equiv x = a) \vee \forall x(P(x) \equiv x = b)],$$

which summarizes E_1 and E_2.

Let $\mathcal{P}_2 = \{P(a), P(x)|Q(x) \leftarrow\}$. $\{P(a)\}$ is the unique answer set of $\mathcal{P}_2$. $C(\mathcal{P}_2)$ is

$$P(a) \wedge \forall x(P(x) \vee Q(x)).$$

The circumscription of P and Q in $C(\mathcal{P}_2)$ is

$$P(a) \wedge \forall x(P(x) \equiv \neg Q(x)).$$

■

3.5 Summary

We have defined the logic GK, of Grounded Knowledge, and provided two very similar transformations from default logic and autoepistemic logic into GK. The transformations provide for the first time a uniform epistemic semantics for the two logics, and thus a common semantic background against which they can be clearly compared.

Using a relationship between GK and circumscription, we have shown that logic programs and disjunctive data bases can be formalized in circumscription by introducing, for each predicate, a new one with the same arity.

Recently, Lifschitz [1991] has extended a version of GK to first-order case, and shown that it can be used to formalize disjunctive data bases with classical negation, and to define epistemic queries to logic programs. We are sure that many other applications of GK will come in the days ahead.

Chapter 4

A Proof-Theoretic Foundation: Argument Systems

In this chapter[1], we study nonmonotonic reasoning from a proof-theoretic point of view. Our study will shed some new light on the way in which nonmonotonic logic may be implemented.

A way of classifying formal theories of nonmonotonic reasoning is to see whether they are *sentence-based* or *argument-based*. Sentence-based nonmonotonic logics include various forms of circumscription [McCarthy 1986, Lifschitz 1987], McDermott and Doyle's nonmonotonic logic [McDermott and Doyle 1980], Reiter's default logic [Reiter 1980],[2] Moore's autoepistemic logic [Moore 1983], Shoham's chronological ignorance [Shoham 1986], and our logic GK in Chapter 3. Argument-based systems includes logic programs with negation-as-failure, nonmonotonic inheritance hierarchies [Touretzky 1986, Horty et al. 1987, and others], and defeasible reasoning [Pollock 1987, Loui 1987a, and others]. Argument-based systems focus most of their attention on the structure of arguments (proofs) and on accounting for choices between conflicting arguments, while these notions are often implicit, if they exist at all, in most sentence-based systems. As a result, most of the reasoning in argument-based systems tends to be closer to human commonsense reasoning, and most sentence-based

[1] Part of this chapter appeared as [Lin and Shoham 1989].

[2] Perhaps it is more reasonable to classify default logic in both categories.

systems are mathematically simpler, and closer to classical logic.

In this chapter, we propose the notion of *argument systems* as a general proof theory for nonmonotonic logic. Compared with other argument-based systems, our framework has the following features. Although we explicitly introduce the notion of arguments (thus diverging from sentence-based systems), we do not explicitly state when an argument is better than (or overrides) another. This is motivated by our desire to have a system that is close to human commonsense reasoning, and which at the same time has a simple and intuitive definition. Our early efforts showed that the definitions of when an argument is better than another are not only mathematically complicated, but clearly not unique, and it is not clear how to make a choice among the alternatives.[3] Our results show that the notion of when an argument is better than another one may be both theoretically and practically unnecessary. For example, we have shown that default logic can be naturally considered to be a special case of our framework, and that various intuitions about nonmonotonic inheritance hierarchies can be conveniently formalized in logic programs with negation-as-failure, which are special argument systems. Of course this does not mean that the notion of when an argument overrides another is worthless. On the contrary, we do find that it is often intuitively very plausible to talk about priorities among conflicting arguments.

4.1 Basic definitions

Throughout this section we fix a language $\mathcal{L}$ so that in the following when we say, for example, that φ is a sentence, we mean that φ is a sentence in $\mathcal{L}$. The only requirement we have of $\mathcal{L}$ is that it has a special operator "$\neg$" so that if φ is a sentence, then $\neg\varphi$ is also a sentence. At the moment no logical properties are assigned to the operator $\neg$. In particular, no inherent connection between φ and $\neg\neg\varphi$ is assumed. The sole function of $\mathcal{L}$ is to provide us with a set of sentences.

We first introduce the notion of *inference rules*. There are three kinds of them:

Definition 4.1 *An* (inference) rule *is an expression of one of the following forms:*

[3] This point is well manisfested in the "clash of intuition" paper by Touretzky *et al.* [1987].

1. *A, where A is a sentence. A rule of this form will be called a* base fact.

2. *$A_1, \ldots, A_n \rightarrow B$, where $n > 0$, and the A's and B are sentences. A rule of this form will be called* monotonic.

3. *$A_1, \ldots, A_n \Rightarrow B$, where $n > 0$, and the A's and B are sentences. A rule of this form will be called* nonmonotonic.

Intuitively, a base fact represents our explicit knowledge about a specific domain. For example, the fact that Tweety is a bird can be represented as a base fact *bird(Tweety)*. A monotonic rule $A_1, \ldots, A_n \rightarrow B$ represents our deductive knowledge about the domain. It always enables us to conclude the B from the A's. For example, the knowledge "if a is a penguin, then it is also a bird" can be represented as the monotonic rule $penguin(a) \rightarrow bird(a)$. A nonmonotonic rule $A_1, \ldots, A_n \Rightarrow B$ represents our commonsense knowledge about the domain. It usually enables us to conclude the B from the A's, but there may be exceptions. For example, the statement "if a is a bird, then it flies" can be represented as the nonmonotonic rule $bird(a) \Rightarrow fly(a)$.

In the following, we assume that for any sentences $A_1, ..., A_n, A'_1, ..., A'_n$, if $(A_1, ..., A_n)$ is a permutation of $(A'_1, ..., A'_n)$, then $A_1, ..., A_n \rightarrow B$ is the same as $A'_1, ..., A'_n \rightarrow B$, and $A_1, ..., A_n \Rightarrow B$ is the same as $A'_1, ..., A'_n \Rightarrow B$.

Definition 4.2 *An* argument system *is a pair (R, C), where R is a set of rules, and C a set of sentences called* completeness conditions.

Intuitively, C is the set of sentences about which our knowledge has to be complete, i.e. for every member of C, either it or its negation has to be known. Typically, C will be a set of instances of McCarthy's abnormality (ab) predicate.

Next we define *argument structures*, which are to argument systems as extensions are to default theories. To this end, we first define *arguments*,[4] which are trees constructed from rules, and are used to establish propositions.

Definition 4.3 *Let $\mathcal{A} = (R, C)$ be an argument system. An argument in $\mathcal{A}$ is a tree, and is defined inductively as follows:*

[4]We remark here that we only consider what Pollock calls *linear* arguments [Pollock 1987].

1. *If $A \in R$ is a base fact, then the tree consisting of A as the only node is an argument.*

2. *If $p_1, \ldots, p_n$ are arguments whose roots are $A_1, \ldots, A_n$, respectively, and $A_1, \ldots, A_n \rightarrow B$ (or $A_1, \ldots, A_n \Rightarrow B$) is a rule in R such that B is not a node in any one of the trees $p_1, \ldots, p_n$, then the tree p with B as its root and $p_1, \ldots, p_n$ as its immediate subtrees is an argument. p is said to be* formed *from $p_1, \ldots, p_n$ by using the monotonic rule $A_1, \ldots, A_n \rightarrow B$ (or by using the nonmonotonic rule $A_1, \ldots, A_n \Rightarrow B$).*[5]

An argument p is said to be *for* φ, or φ is said to be *supported* by p, if φ is the root of p. Notice that according to our definition of arguments, if the argument p is for φ, then φ can only appear as the root of p, that is, arguments are well founded. This is in order to prevent us from generating an infinite number of redundant arguments. For example, A is the only argument that can be constructed from the rules A and $A \rightarrow A$. $A \rightarrow A$, $A \rightarrow A \rightarrow A$, etc. are not arguments.

Grouping arguments together we get the notion of *argument structures.* These can be viewed as sets of logically consistent arguments held by an agent.

Definition 4.4 *Let $\mathcal{A} = (R, C)$ be an argument system. A set T of arguments in $\mathcal{A}$ is an* argument structure *of $\mathcal{A}$ if the following conditions are satisfied:*

1. *If p is a base fact in R, then $p \in T$.*

2. *T is* structurally closed*, that is, for any $p \in T$ if p' is a subtree of p, then $p' \in T$.*

3. *T is* monotonically closed*, that is, if p is formed from $p_1, \ldots, p_n$ in T by a monotonic rule, then p is also in T.*

4. *T is* consistent*, that is, it does not contain arguments for both φ and $\neg\varphi$, for any sentence φ.*

[5]From this definition we see that, actually, an argument is more than a tree. It also contains information about how the tree is formed. But this information can be uniquely determined from the set R of rules under the assumption that if $A_1, ..., A_n \rightarrow B$ is in R, then $A_1, ..., A_n \Rightarrow B$ is not in R.

5. T *is* complete*, that is, for any* $\varphi \in C$*, either there is an argument in* T *for* φ *or there is an argument in* T *for* $\neg\varphi$*.*

Notice that the notion of completeness plays a role similar to that of minimality in circumscription or fixed-point in default logic.

Another notion that will be used throughout this paper is the set of sentences *supported* by an argument structure.

Definition 4.5 *Let* T *be an argument structure. The set of sentences supported by* T*, written* $\mathrm{Sup}(T)$*, is defined as follows:*

$$\mathrm{Sup}(T) = \{\varphi \mid \text{ there is a } p \in T \text{ such that } \varphi \text{ is supported by } p.\}.$$

Example 4.1 [Penguins do not fly] Let R be the set of following 8 rules:

$$\begin{aligned}
r_1 &: \ True, \\
r_2 &: \ Penguin(a), \\
r_3 &: \ Penguin(a) \to Bird(a), \\
r_4 &: \ Penguin(a), \neg ab(a, penguin, flyless) \to \neg Fly(a), \\
r_5 &: \ Bird(a), \neg ab(a, bird, flyness) \to Fly(a), \\
r_6 &: \ Penguin(a) \to ab(a, bird, flyness), \\
r_7 &: \ True \Rightarrow \neg ab(a, penguin, flyless), \\
r_8 &: \ True \Rightarrow \neg ab(a, bird, flyness).
\end{aligned}$$

Let $C = \{ab(a, penguin, flyless), ab(a, bird, flyness)\}$. There are also 8 arguments in $\mathcal{A} = (R, C)$:

$$\begin{aligned}
p_1 &: \ True, \\
p_2 &: \ Penguin(a), \\
p_3 &: \ p_1 \overset{r_7}{\Rightarrow} \neg ab(a, penguin, flyless), \\
p_4 &: \ p_1 \overset{r_8}{\Rightarrow} \neg ab(a, bird, flyness), \\
p_5 &: \ p_2 \overset{r_3}{\to} Bird(a),
\end{aligned}$$

$$\begin{aligned} p_6 &: \ p_2 \xrightarrow{r_6} ab(a, bird, flyness), \\ p_7 &: \ p_3, p_2 \xrightarrow{r_4} \neg Fly(a), \\ p_8 &: \ p_4, p_5 \xrightarrow{r_5} Fly(a). \end{aligned}$$

There is a unique argument structure of $\mathcal{A}$:

$$T = \{p_1, p_2, p_5, p_6, p_3, p_7\},$$

and

$$\begin{aligned} \mathrm{Sup}(T) \ = \ & \{True, Penguin(a), Bird(a), \neg ab(a, penguin, flyless), \\ & ab(a, bird, flyness), \neg Fly(a)\}. \end{aligned}$$

Intuitively, our intention for introducing the "artificial" predicate ab is that unless we know that $ab(x, y, z)$ is true, we will assume by default that $\neg ab(x, y, z)$ is true. Thus it is natural to expect that we shall have a complete knowledge about the ab predicate, and this is how we have defined the set C. ■

Notice that in this example, the nonmonotonic rules are of the simplest form:

$$True \Rightarrow B,$$

where B is a sentence. As we shall see later on, nonmonotonic rules of this form are of particular importance to us.

4.2 Default logic as argument systems

In this section, we show that default logic can be naturally considered a special case of argument systems. In this section, we suppose the underlying language to be the ordinary first-order one, augmented with a second-order predicate ab, and our special operator "$\neg$" is the classical negation operator.

Let R_d be the following set of rules:

1. $True$ is a base fact.

2. If $A_1, \ldots, A_n$, and B are first-order sentences and B is a consequence of $A_1, \ldots, A_n$ in first-order logic, then $A_1, \ldots, A_n \rightarrow B$ is a monotonic rule.[6]

3. If A is a first-order sentence, then $\neg A \rightarrow ab(A)$ is a monotonic rule.

4. If B is a first-order sentence, then $True \Rightarrow \neg ab(B)$ is a nonmonotonic rule.

Let $\Delta = (W, D)$ be a closed default theory. Define $R(\Delta)$ to be the set of the following rules:

1. If $A \in W$, then A is a base fact of $R(\Delta)$.

2. If $A : B_1, \ldots, B_n / C$ is a default in D, then

$$A, \neg ab(B_1), \ldots, \neg ab(B_n) \rightarrow C$$

   is a monotonic rule.

Let $Ab = \{ab(A) \mid A \text{ is a first-order sentence}\}$.

We have the following result:

Theorem 6 *Let E be a consistent set of first-order sentences. E is an extension of Δ iff there is an argument structure T of $(R_d \cup R(\Delta), Ab)$ such that E is the set of first-order sentences supported by T.*

Proof: Let E be a consistent set of first-order sentences. According to Reiter's result (Proposition 2.2), E is an extension of Δ iff $E = E_0 \cup E_1 \cup \ldots \cup E_n \cup \ldots$, where E_i, $i \geq 0$, is defined as follows:

$$\begin{aligned} E_0 &= W, \\ E_{i+1} &= Th(E_i) \cup \{C \mid A : B_1, \ldots, B_n/C \in D \\ &\qquad \text{where } A \in E_i \text{ and } \neg B_1, \ldots, \neg B_n \notin E\}, \end{aligned}$$

where $Th(E_i)$ is the closure of E_i under first-order consequence.

[6] Instead of using the "rule schema," we can use a set of axioms plus modus ponens to capture first-order deduction. We think it is conceptually simpler to use the rule schema.

Suppose E is an extension of Δ, let $E' = E \cup \{ab(B) | \neg B \in E\} \cup \{\neg ab(B) | \neg B \notin E\}$. Obviously, for any $ab(B) \in Ab$, either $ab(B)$ or $\neg ab(B)$ is in E', and E is the restriction of E' to the set of first-order sentences. We now prove that there is an argument structure T of $(R_d \cup R(\Delta), Ab)$ such that $E' = \text{Sup}(T)$.

First we prove that for any $\varphi \in E'$ there is an argument for φ. To this end, we inductively prove that for any $i \geq 0$, if $\varphi \in E_i$, then there is an argument for φ. If $i = 0$, then $\varphi \in W$, thus φ is a base fact of $R_d \cup R(\Delta)$. Inductively, suppose we have done the proof for i. Let $\varphi \in E_{i+1}$, then φ is either in $Th(E_i)$ or there is a default $A : B_1, \ldots, B_n / \varphi$ in D such that $A \in E_i$. The former case is obvious. The later case is also easy to prove by using the inductive assumption and the fact that $A, \neg ab(B_1), \ldots, \neg ab(B_n) \rightarrow \varphi$ and $True \rightarrow \neg ab(B_k)$, $1 \leq k \leq n$, are rules. Therefore by Reiter's result, if $\varphi \in E$, then there is an argument for φ. From this by the definition of E', it is easy to see that if $\varphi \in E'$, then there is an argument for φ.

Now let T be the set of those arguments such that all sentences in them (as nodes) are in E'. Obviously $E' = \text{Sup}(T)$. We prove that T is an argument structure of $(R_d \cup R(\Delta), Ab)$.

1. T contains all base facts: trivial.

2. T is structurally closed: trivially follows from the definition of T.

3. T is consistent: follows easily from the fact that E is consistent (in the sense of first-order logic).

4. T is monotonically closed: this is proved by checking all monotonic rules. For example, if p is formed from $p_1, \ldots, p_n \in T$ by using the monotonic rule:

$$A, \neg ab(B_1), \ldots, \neg ab(B_n) \rightarrow C,$$

then $A \in E$ and $\neg B_1, \ldots, \neg B_n \notin E$, therefore $C \in E$, thus all sentences in p are in E', so $p \in T$.

5. T is complete: Trivial.

Conversely, let T be an argument structure of $(R_d \cup R(\Delta), Ab)$ such that E is the restriction of $\mathrm{Sup}(T)$ to the set of first-order sentences. We prove that E is an extension of Δ, that is, $E = \cup E_i$. By induction on i, using the fact that for any first-order sentence A, $ab(A) \in \mathrm{Sup}(T)$ iff $\neg A \in \mathrm{Sup}(T)$, it is easy to see that $E_i \subseteq E$ for each i, that is, $\cup E_i \subseteq E$. Now let $\varphi \in E$, then there is an argument $p \in T$ for φ. By induction on the number of rules used in p, we prove that there is an $i \geq 0$ such that $\varphi \in E_i$. If p is a base fact, then $p \in W$, that is, $p \in E_0$. Inductively, suppose that for any proper subtree p' of p, if p' is for a first-order sentence, then the sentence is in $\cup E_i$. If p is formed by using first-order deduction rule, then trivially, $\varphi \in \cup E_i$. If p is formed by using the rule $A, \neg ab(B_1), \ldots, \neg ab(B_n) \rightarrow \varphi$, then by inductive assumption, there is an i such that $A \in E_i$. But $p \in T$, therefore $\neg ab(B_1), \ldots, \neg ab(B_n) \in \mathrm{Sup}(T)$, so $B_1, \ldots, B_n \notin E$, thus $\varphi \in E_{i+1}$. This concludes the induction proof, thus we have proved that $E = \cup E_i$ and E is a default extension of Δ. ∎

Example 4.2 Consider $\Delta_1 = (W_1, D_1)$, where

$$D_1 = \{: A/A, : B/B, : C/C\}, \qquad W_1 = \{B \supset \neg A \wedge \neg C\}.$$

This is example 2.1 of Reiter [1980]. It has two extensions given by $E_1 = Th(W \cup \{A, C\})$, and $E_2 = Th(W \cup \{B\})$, where Th is the closure operator given by classical deduction.

$R(\Delta_1)$ is the following set of rules:

$$B \supset \neg A \wedge \neg C,$$
$$\neg ab(A) \rightarrow A,$$
$$\neg ab(B) \rightarrow B,$$
$$\neg ab(C) \rightarrow C.$$

Suppose an argument structure contains an argument for A, then it can not contain an argument for B. But it must contain one for C, since there are no arguments for $ab(C)$ when arguments for B are excluded. Similarly, if an argument structure contains an argument for B, then it can not contains one for A or for C. Thus there are two argument structures of $(R_d \cup R(\Delta_1), Ab)$. One supports A and C, the other supports B.

Now consider $\Delta_2 = (\{: A/\neg A\}, \{\emptyset\})$. This is example 2.6 of Reiter [1980]. It has no extension. $R(\Delta_2) = \{\neg ab(A) \rightarrow \neg A\}$. There is no argument structure of $(R_d \cup R(\Delta_2), Ab)$. Suppose there is a one. Then it must either support $ab(A)$ or $\neg ab(A)$. If it supports $\neg ab(A)$, then it must also support $\neg A$, and thus support $ab(A)$, a contradiction. If it supports $ab(A)$, then there must be an argument for $ab(A)$. But any such argument must contain $\neg A$, and thus $\neg ab(A)$, again a contradiction. ■

4.3 Negation-as-failure

In this section,[7] let our language be a first-order one. Let our special operator "$\neg$" be classical negation operator, which will be interpreted by negation-as-failure rule.

Let $\mathcal{P}$ be a logic program that contains only closed rules. Let $R(\mathcal{P})$ be the set of the following rules:

1. $True$ is a base fact.

2. For any clause $p_0 \leftarrow p_1, \ldots, p_n$ in W, if $n = 0$ then p_0 is a base fact, otherwise $p_1, \ldots, p_n \rightarrow p_0$ is a monotonic rule.

3. For any ground atom p, $True \Rightarrow \neg p$ is a nonmonotonic rule.

Let $\mathcal{A}$ be the set of ground atoms. Then $\mathcal{P}$ is captured by the argument system $(R(\mathcal{P}), \mathcal{A})$.

Theorem 7 *A set E is an answer set of $\mathcal{P}$ iff it is the set of ground atoms supported by an argument structure of $(R(\mathcal{P}), \mathcal{A})$.*

[7]Historically, in [Lin and Shoham 1989], we first formulated logic programs with negation-as-failure as special argument systems. Then by studying the relationship of these argument systems with the ones for default theories, we obtained a transformation from logic programs to default theories. The transformation turns out to be exactly the same as the one given by Bidoit and Froidevaux [1988].

Example 4.3 Let $\mathcal{P}$ be the following set of rules:

$$p \leftarrow \neg q,$$
$$q \leftarrow \neg p,$$

where p, q are the only ground atoms in the language. There are two answer sets for $\mathcal{P}$, given by $E_1 = \{p\}$, and $E_2 = \{q\}$. $R(\mathcal{P})$ is the set of following rules:

$$True,$$
$$\neg q \rightarrow p,$$
$$\neg p \rightarrow q,$$
$$True \Rightarrow \neg p,$$
$$True \Rightarrow \neg q.$$

There are two argument structures; one supports $\{p, \neg q\}$, the other supports $\{q, \neg p\}$. ■

The theorem can be easily proved by using the following lemma, which is obvious from the definitions:

Lemma 4.1 *A set E is the support set of an argument structure of $(R(\mathcal{P}), \mathcal{A})$ iff E is the support set of the argument structure of $(R_0 \cup E_1, \mathcal{A})$, where R_0 is the set of base facts and monotonic rules in $R(\mathcal{P})$, and $E_1 = \{\neg p \mid \neg p \in E$, p is a ground atom$\}$.*

In fact, the argument systems for logic programs can be translated to argument systems for default theories under the correspondence

$$\neg p \leftrightarrow \neg ab(\neg p),$$

which gives us the translation of logic programs to default theories in section 2.5.

4.4 Negation-as-failure systems

As we have seen from the last two sections, argument systems corresponding to default theories and logic programs have some special properties. For example, in the ones that correspond to default theories, all nonmonotonic rules have the form:

$$True \Rightarrow \neg ab(A).$$

If $ab(A)$ is a completeness condition, then this rule says that, intuitively, we apply negation-as-failure to $ab(A)$. This leads to the idea of implementing default logic using a *generalized negation-as-failure* rule.

Let us first extend the special argument systems for default theories and logic programs, and define the following notion:

Definition 4.6 *An argument system* (R, C) *is a* negation-as-failure system *(NAFS) if* R *contains the base fact* $True$*, and the set of the nonmonotonic rules in* R *is* $\{True \Rightarrow \neg\varphi \mid \varphi \in C\}$.

For example, let R be the following set of rules:

$$\begin{aligned} &True, \\ &P, \\ &P, \neg Q \rightarrow R, \\ &True \Rightarrow \neg Q, \end{aligned}$$

and let $C = \{Q\}$, then (R, C) is a negation as failure system. In fact, it corresponds to the logic program: $\{P, (R \leftarrow P, \neg Q)\}$.

Not surprisingly, there are close relationships between NAFS and truth maintenance system (TMS). In fact, there is a transformation between NASF and Junker and Konolige's TMS [Junker and Konolige 1990] under which these two systems are equivalent[8].

Let $\mathcal{A} = (R, C)$ be an argument system, and φ a sentence. We denote by $\mathcal{A}(\varphi)$ the set of the arguments for φ in $\mathcal{A}$.

[8] We thank Kurt Konolige for asking us a question about the relationship.

Given a negation-as-failure system $\mathcal{A} = (R, C)$, and a query A (a sentence), we have the following "decision procedure" using the generalized negation-as-failure rule:

> If $\mathcal{A}(A) = \emptyset$, then answer "no" to the query A. If there is a $p \in \mathcal{A}(A)$ such that for any node $\neg\varphi$ in p that is generated by a nonmonotonic rule, the answer to the query φ is "no" (thus the term negation as failure), then answer "yes" to the query A; otherwise, answer "no" to the query A.[9]

Notice that since $\mathcal{A}$ is a negation-as-failure system, if a node φ in an argument p is generated by a nonmonotonic rule, then it must be generated by the nonmonotonic rule $True \Rightarrow \neg\phi$, where $\phi \in C$ and $\varphi = \neg\phi$.

The decision procedure may not terminate in some cases. Obviously it may not terminate if R is infinite. When R is finite, it still may not terminate if (R, C) has multiple argument structures. For example, it runs into cycle with the system $(\{\neg P \rightarrow Q,\ \ \neg Q \rightarrow P\}, \{P, Q\})$[10] and the query Q. In fact, like the negation as failure rule in logic programming, the procedure works by pretending the underlying system has a unique argument structure.

It may not be "correct" in some cases. For example, it will answer "yes" to the query P for the negation-as-failure system $(\{P, \neg Q \rightarrow Q\}, \{Q\})$, although the system has no argument structure. In fact, the decision procedure uses a rule of locality in the sense that it explores the underlying argument system only as much as necessary in order to answer the query.

Generally, suppose $\mathcal{A} = (R, C)$ is a consistent negation-as-failure system in the sense that there is an argument structure for $(R, \emptyset)$. If the decision procedure terminates for the query A, then there is a $C' \subseteq C$ such that A is supported by the unique argument structure of (R, C') iff the answer to the query A is "yes". In particular, if $\mathcal{A}$ has a unique argument structure, and the decision procedure terminates for the query A, then A is supported by the unique argument structure iff the answer to the query is 'yes'.

It can also be shown that if a negation-as-failure system $\mathcal{A}$ is finite, and has a unique extension, then the decision procedure always terminates.

[9] Strictly speaking, $\mathcal{A}(A)$ has to be sorted. Also the nodes in an argument need to be ordered.

[10] Notice that we have omitted the nonmonotonic rules $True \Rightarrow \neg P$ and $True \Rightarrow \neg Q$.

Inadequate as the decision procedure may appear, we hope that, like the negation as failure rule in logic programming, the decision procedure will prove useful in practice.

4.5 Default reasoning as negation-as-failure

Since argument systems corresponding to default theories are negation-as-failure system, we can use the "decision procedure" in the last section to answer queries for default theories.

Example 4.4 $\Delta = (D, W)$, where $D = \{: C/\neg D, : D/\neg E, : E/\neg F\}$ and $W = \emptyset$. This is example 2.2 of Reiter [1980]. This theory has a unique extension given by $E = Th(\{\neg D, \neg F\})$. Let $\mathcal{A} = (R(\Delta), Ab)$. The three rules correspond to the default rules are

$$\neg ab(C) \rightarrow \neg D,$$
$$\neg ab(D) \rightarrow \neg E,$$
$$\neg ab(E) \rightarrow \neg F.$$

We have

$$\mathcal{A}(\neg D) = \{True \Rightarrow \neg ab(C) \rightarrow \neg D\}, \qquad \mathcal{A}(ab(C)) = \mathcal{A}(\neg C) = \emptyset.$$

Thus the decision procedure in last section will answer 'no' to the query $ab(C)$, and 'yes' to $\neg D$. Similarly, it will answer 'yes' to the query $ab(D)$, and 'no' to $\neg E$. ■

Apparently, the key to the success of the decision procedure is how to compute $\mathcal{A}(A)$. A common method used in logic programming is backward chaining which basically works as follows. To construct an argument for A (the main goal), find a rule with the form $A_1, ..., A_n \rightarrow A$, and reduce the problem to the AND subgoals: $A_1, ..., A_n$. Repeat the process until base facts are reached.

The obvious adaptation of this method does not work for negation-as-failure systems corresponding to default theories. For example, for any sentences A and B, if

B follows from A in first-order logic, then $A \rightarrow B$ is a rule. Therefore we can always "reduce" the goal A to $A \wedge A$ to $A \wedge B$, and so on. Fortunately, there seems a way out of this. The following idea is similar to the one used in [Junker and Konolige 1990].

Given a set of defaults

$$D = \{(A_1 : B_1, ..., B_i/C_1), \quad ..., \quad (A_n : B_j, ..., B_k/C_n)\}$$

we define $Consequence(D)$ to be the set of consequences of the defaults in D, i.e.,

$$Consequence(D) = \{C_1, ..., C_n\}.$$

For any sentence A, we can proceed to construct arguments for A in $(R(\Delta), Ab)$ as follows:

1. If A is $ab(B)$, then recursively call the procedure to construct arguments for $\neg B$.
2. If A is a first-order sentence, then find a minimal set of defaults

   $$d_1, ..., d_n$$

   such that $W \cup Consequence(\{d_1, ..., d_n\})$ infers A in first-order logic, and recursively call the procedure to construct arguments for $A_1, ..., A_n$, where A_i is the prerequisite of d_i, $1 \leq i \leq n$.

This is the best we can get for general default theories. There are special cases where we can do much better. For example, if we consider only literals (atoms and negated atoms), then first-order deduction will be reduced to membership checking. In fact, in this case we have what Gelfond and Lifschitz call logic programs with classical negation [Gelfond and Lifschitz 1991].

Combining the "decision procedure" for negation-as-failure system and the procedure for constructing arguments for default logic, we have the following algorithm: Given a default theory (W,D), and a query Q, we proceed as follows.

1. If $W \cup Consequence(D_1) \vdash Q$, then return with "yes", where D_1 is the set of defaults in D that have been confirmed. Initially, none of the defaults in D are confirmed.

2. If $W \cup Consequence(D_2) \not\vdash Q$, then return with "no", where D_2 is the set of defaults in D that have not been refuted so far. Initially, none of the defaults in D are refuted.

3. Find a minimal set of defaults in D_2:

$$D_3 = \{d_1, ..., d_n\}$$

such that $W \cup Consequence(D_3) \vdash A$. Let D_m be the set of defaults in D_3 that have not been confirmed.

4. While $D_m \neq \emptyset$ do

   Choose a default $d_l = A_l : B_p, ..., B_q/C_l$ from C_m, and run the algorithm with the queries $A_l, \neg B_p, ..., \neg B_q$. If the answer to A_l is "no" or one of the answers to $\neg B_p, ..., \neg B_q$ is "yes", then mark the default as being refuted, and go back to step 1; otherwise mark the default as being confirmed, and let $D_m = D_m - \{d_l\}$.

5. Return with "yes".

We say that a default has been *considered* if it was chosen in step 4. It is clear that in order for the algorithm to terminate, each default in D can be considered at most once. Let us define the *size* of a default to be the number of sentences in it, and the size of D, $size(D)$, to be

$$|D| \times max\{size(d) \mid d \in D\},$$

where $|D|$ is the cardinal of D, and $size(d)$ is the size of d. Since D_1, D_2, D_3, and D_m can all be computed in $O(size(D)) * FOL$ time, thus it is easy to see that if the algorithm terminates, then it will terminate in $O(size(D)^2) * FOL$, where FOL is the cost of doing a first-order deduction. This compares with mostly negative results in [Kautz and Selman 1989].

Of course, for the algorithm to be useful, there should be large class of default theories for which the algorithm terminates. In fact, similar to stratified logic programs in logic programming, we can define *stratified default theories*.

Definition 4.7 *A default theory* $\Delta = (W, D)$ *is stratified if there is a sequence of disjoint sets* $D_1, ..., D_n$ *such that*

1. $D = D_1 \cup ... \cup D_n$,
2. *For any* $1 \leq i \leq n$, *any default* $A : B_1, ..., B_k/C \in D_i$, *any set* $D' \subseteq D_1 \cup ... \cup D_{i-1}$, *and any* $\varphi \in \{A, \neg B_1, ..., \neg B_k\}$, *if*

$$\{W\} \cup Consequence(D') \not\vdash \varphi$$

then

$$\{W\} \cup Consequence(D' \cup D_i \cup ... \cup D_n) \not\vdash \varphi.$$

It can be easily seen that for stratified default theories, the algorithm always terminates.[11]

Example 4.5 $\Delta = (W, D)$:

$$W = \{\forall x(bird(x) \supset animal(x)), \forall x(bird(x) \supset ab2(x)), bird(a)\}$$

and

$$D = \{bird(a) : \neg ab1(a)/fly(a),\ animal(a) : \neg ab2(a)/\neg fly(a)\}.$$

Δ is stratified with $D_1 = D$. ■

We have actually implemented the algorithm in C using OTTER 2.0 [McCune 1990] resolution theorem prover. The system is called DNAF, and it answers queries for default theories. Currently, DNAF works only for closed default theories.

4.5.1 Input of DNAF

The input of DNAF consists of the following items:

1. Comments, which begin with %, and end at the end of the line.
2. A first-order theory, which has the following form:

[11]For this to be true, of course, we have to properly constraint the way we find minimal sets in step 3, and the way we choose defaults in step 4.

```
theory.
```

A_1.

...

A_k.

```
end-theory.
```

Where $A_1, ..., A_k$ are sentences, which will be defined below. Notice that the period "." is required after each sentence, and after the key words `theory` and `end-theory`.

3. A set of closed defaults, which has the following form:

```
default.
```

$A_1 : B_{11}, ..., B_{1i}/C_1$.

...

$A_k : B_{k1}, ..., B_{kj}/C_k$.

```
end-default.
```

where $A_1, ..., A_k, B_{11}, ..., B_{kj}, C_1, ..., C_k$ are sentences.

4. A query which has the form:

```
query.
```

A.

```
end-query.
```

where A is a sentence.

DNAF will try to prove the query from the default theory, and it will output defaults that have been used (either proved or disproved) in the process.

Our syntax for sentences is the same as the one used in OTTER [McCune 1990]. Inductively, sentences are defined as follows [McCune 1990]:

1. A *name*, which is an alphanumeric string, is a sentence.

2. If F and G are sentences, then (F `<->` G) and (F `->` G) are sentences.

3. If $F_1, ..., F_n$ are sentences, then (F_1 `| ... |` F_n) and (F_1 `& ... &` F_n) are sentences.

4. If $x_1, ..., x_n$ are names, and F a sentence, then

$$(Q_1\ x_1\ ...\ Q_n\ x_n\ F)$$

   is a sentence, where Q_i is either `all` (universal quantifier) or `exists` (existential quantifier).

5. If F is a non-negated sentence, then `-`F is a negated sentence.

The symbols have their expected meanings: `-` stands for "not", `<->` for "if and only if", `->` for "implies", `|` for "or", and `&` for "and".

Notice that all parentheses and white spaces are required around the symbols. Thus `(p & q -> r)` is not a sentence (not enough parentheses), `(p|q)` is not a sentence (not enough white space). Notice also that `--p` is not a sentence (double negation).

4.5.2 Sample runs of DNAF

DNAF is non-interactive. It is called by:

dnaf [input-file]

If there is no input-file, then DNAF reads from the standard input.

Example 4.6 [Birds fly] Suppose we have the following default theory:

```
theory.
(all x (bird(x) -> animal(x))).
bird(a).
```

```
(all x (bird(x) -> ab2(x))).
end-theory.
default.
bird(a) :  -ab1(a) / fly(a).
animal(a) :  -ab2(a) / -fly(a).
end-default.
query.
fly(a).
end-query.
```

DNAF runs for about one second, and outputs the following message:

```
The query follows from the default theory.
The following defaults are used in the process:
bird(a) :  -ab1(a) / fly(a).  True
```

If instead of `fly(a)`, `-fly(a)` is queried, then DNAF will output the following:

```
The query does not follow from the default theory.
The following defaults are used in the process:
animal(a) :  -ab2(a) / -fly(a).  False
```

■

Example 4.7 [Penguin and Pelican] This example is from [Davis 1991].

```
theory.
(all x (pelican(x) -> bird(x))).
(all x (penguin(x) -> bird(x))).
(all x (penguin(x) -> -fly(x))).
(penguin(Tweety) | pelican(Tweety)).
```

```
end-theory.

default.

bird(Tweety) :  fly(Tweety) / fly(Tweety).

end-default.

query.

fly(Tweety).

end-query.
```

DNAF outputs the following message within seconds:

```
The query follows from the default theory.

The following defaults are used in the process:

bird(Tweety) :  fly(Tweety) / fly(Tweety).  True
```

DNAF gives the same output if $-penguin(Tweety)$ is queried.

This example is used by Davis to show that default logic sometime forces us to accept a disjunct in order not to violate a default rule. ■

4.6 Concluding remarks

We have defined the notion of argument systems. We started with the primitive notion of inference rules and defined those of arguments and argument structures. Arguments are used to establish propositions, and argument structures are logically consistent sets of arguments that are complete with respect to a set of sentences.

Argument systems can be considered to be extended axiom systems. A classical axiom system consists of a set of axioms and a set of inference rules, and can be considered to be a special argument system with the empty set of nonmonotonic rules and completeness conditions. Argument systems may also be viewed as extended logic programs.

We have shown that the notion of argument systems includes default logic and negation-as-failure as special cases. In [Lin and Shoham 1989] we show that it can be used to reformulate autoepistemic logic and circumscription as well. We have introduced a "decision procedure" using a generalized negation-as-failure rule to answer queries for a special class of argument systems called negation-as-failure systems, and apply it in an implementation of default logic.

The framework of argument systems provides a convenient way to construct non-monotonic logics based on non-classical monotonic logics. For example, argument systems for default theories have a collection of monotonic rules which corresponds to first-order deduction:

> If $A_1, \ldots, A_n$, and B are first-order sentences and B is a consequence of $A_1, \ldots, A_n$ in first-order logic, then $A_1, \ldots, A_n \to B$ is a monotonic rule.

If we delete these rules from the argument systems, we essentially get extended logic programs with classical negation in the sense of [Gelfond and Lifschitz 1991]. We can also replace them by the rules corresponding to, for example, a relevance logic, and get a default relevance logic.

Chapter 5

Application I: Formalizing Inheritance

In this chapter and the next one, we study two applications of nonmonotonic logics. In this chapter, we formalize nonmonotonic inheritance in logic programs. In the next one, we study formal theories of actions.

5.1 Introduction

Reasoning about inheritance was one of the earliest applications of nonmonotonic logics. It has also been one of the motivations for developing such logics. Unfortunately, early attempts (e.g., [Etherington and Reiter 1983] and [McCarthy 1986]) in formalizing it using general purpose nonmonotonic logics have not been as successful as the ones using *ad hoc* methods (e.g., [Touretzky 1986] and [Horty *et al.* 1987]). This raises an important question: Are the existing general-purpose nonmonotonic logics, such as default logic and circumscription, suitable for the task? Answering this question positively is one of the motivations behind this work. Specifically, we first propose and study in detail two simple and well-behaved theories of inheritance in logic programs with negation as failure. Then we show how the same approach can be used to formalize other intuitions about inheritance.

The second motivation behind this work is to address the issue of the so-called conflicting intuitions about inheritance [Touretzky *et al.* 1987]. As Touretzky *et al.* pointed out, there are conflicting intuitions about inheritance, and they cannot be resolved by purely theoretical considerations. We see this as a call for a general methodology for formalizing inheritance. This work provides such a methodology by presenting a uniform approach to formalizing in logic programs various intuitions about inheritance.

This chapter is organized as follows. In the next section we review some background notions about inheritance systems. Then in section 3 we study in detail a simple credulous theory of inheritance. In section 4, we show how the credulous theory in section 3 can easily be revised to become a skeptical theory, and we study the relationship between the two theories. In section 5, we show how to use our approach to formalizing existing theories of inheritance. In particular, we formalize the skeptical theory in [Horty *et al.* 1987]. Finally in section 6 we discuss some related work and make some concluding remarks.

5.2 Basic definitions

We assume a two-sorted first-order language $\mathcal{L}$: object sort and property sort. We use $x, x_1, ...$ to denote variables that range over objects; $p, q, p_1, q_1, ...$ to denote variables that range over properties; $a, a_1, ...$ to denote object constants; $P, Q, P_1, Q_1, ...$ to denote property constants; and $t, t_1, ...$ to denote property terms. We assume that for any property term t, if t is of the form $neg(t_1)$, then $\overline{t}$ is t_1, otherwise $\overline{t}$ is $neg(t)$, where neg is a unary function on properties.

Following Horty *et al.* [1987], we call an inheritance system a net. We consider nets with only nonmonotonic links. Formally a *net* is a directed graph such that

1. Each node is either an object constant (call an *object node*) or a property constant (call a *property node*).

2. Each link has a weight either 1 (call a *positive link*) or -1 (call a *negative link*).

3. There are no links that end at object nodes.

In the following, a positive link from P to Q will be denoted by the atomic sentence $P \to Q$, and a negative link from P to Q by the atomic sentence $P \to neg(Q)$. In the following, we shall identify a net with the set of links in it. Paths can be defined as usual. A *positive path* is one that consists of only positive links.

5.3 A credulous theory of inheritance

Our first formalization of inheritance is perhaps the simplest one. It is based on two principles. The first one, which is often credited to Touretzky [1986], says that arguments based on more specific classes will override conflicting ones based on more general classes. The second principle which is in a sense complementary to the first one and a trademark of credulous theories, says that if two conflicting arguments can not be resolved by the first principle, then either one of them can be used to exclude the other. Based on the two principles, we have the following theory of inheritance in a logic program:

$$p < q \leftarrow p \to q. \tag{5.1}$$
$$p < r \leftarrow p \to q, q < r. \tag{5.2}$$

$$Has(x, p) \leftarrow x \to p. \tag{5.3}$$
$$Has(x, q) \leftarrow Has(x, p), p \to q, \neg ab(x, p, q). \tag{5.4}$$

$$ab(x, p_2, q) \leftarrow Has(x, p_1), p_1 \to neg(q), p_1 < p_2, \neg p_2 < p_1. \tag{5.5}$$
$$ab(x, p_2, neg(q)) \leftarrow Has(x, p_1), p_1 \to q, p_1 < p_2, \neg p_2 < p_1. \tag{5.6}$$

$$ab(x, p, q) \leftarrow Has(x, neg(q)). \tag{5.7}$$
$$ab(x, p, neg(q)) \leftarrow Has(x, q). \tag{5.8}$$

We use $P < Q$ to mean that P is more specific than Q, and according to rules (5.1) and (5.2), it is true iff there is a positive path from P to Q.[1] $Has(a,t)$ means that the object a has the property t, and (5.3) and (5.4) define the meaning of various links in a net. Rules (5.5) and (5.6) formalize the aforementioned Touretzky's principle. The rules (5.7) and (5.8) achieves two things. First, combined with the rule (5.3), they assure that direct links from objects to properties always have higher priorities than compound paths. Secondly, they formalize the aforementioned second principle.

For any net N, let $R1(N)$ be the union of N (as the set of links in it) and the rules (5.1) – (5.8).

Theorem 8 *For any net N, there always exists an answer set of $R1(N)$.*

Proof: Without loss of generality, we assume that there is only one object node a in the net N.

We first define a partial order over the set of property nodes in N. For any property nodes P_1 and P_2, we define $P_1 \sqsubset P_2$ iff $P_1 < P_2$ holds but $P_2 < P_1$ does not hold, that is, there is a positive path from P_1 to P_2 but not the other way around. Clearly, $\sqsubset$ is a strict partial order. Let Max be the set of sequences $(P_1, P_2, ..., P_k)$ such that

1. $P_i \sqsubset P_{i+1}$, $1 \leq i < k$, $k > 0$.
2. If $Q_i \sqsubset Q_{i+1}$ for $0 < i < h$, and $\{P_i \mid 1 \leq i \leq k\} \subseteq \{Q_i \mid 1 \leq i \leq h\}$, then $(P_1, ..., P_k) = (Q_1, ..., Q_h)$.

In other words, Max is the set of maximal $\sqsubset$-chains. For any set S of nodes and any set M of sequences, we define M/S to be the set of sequences obtained from M by deleting nodes not in S. Thus if $S = \{P_1, P_2\}$, and $M = \{(P_1, Q, P_2, P_3), (Q, P_2, Q_1)\}$, then $M/S = \{(P_1, P_2), (P_2)\}$.

We then construct an answer set for $R1(N)$ according to the following procedure:

1. Set $M = Max$.

[1] According to the rule (5.1), $P < neg(Q)$ if there is a negative link from P to Q. This seems a little odd. Fortunately, it does no damage. We will ignore this kind of sentence in the rest of the chapter.

2. Mark positive all property nodes P such that $a \to P \in N$, and mark negative all property nodes P such that $a \to neg(P) \in N$. Let S to be the set of nodes that have been marked positively.

3. For any property nodes P and Q, if Q has been marked positively, then set $ab(a, P, neg(Q))$; if Q has been marked negatively, then set $ab(a, P, Q)$.

4. Until $M/S = \emptyset$, select a property node P such that it only appears as the first element, if it appears at all, in the members of M/S (see the below for a proof of the existence of such P), and do the following steps:

   4.1 For any Q, if $P \to Q \in N$ and $ab(a, P, Q)$ has not been set, then mark Q positively.

   4.2 For any Q, if $P \to neg(Q) \in N$ and $ab(a, P, neg(Q))$ has not been set, then mark Q negatively.

   4.3 For any P_1 and t such that $P \sqsubset P_1$, and $P \to t \in N$, set $ab(a, P_1, \bar{t})$.

   4.4 Do step 3 above.

   4.5 Delete P from all members of M.

   4.6 Add to S all the nodes that have been marked positively in step 4.1.

The existence of the property node P in step 4 can be easily proved. For example, let $M/S = \{l_1, l_2\}$, and P_1 and P_2 be the first element of l_1 and l_2, respectively. Since $P_1 \sqsubset P_2$ and $P_2 \sqsubset P_1$ cannot be true at the same time, either $P_1 = P_2$ or one of the members of M/S dose not contain both P_1 and P_2. Thus either P_1 or P_2 satisfies the condition for P in step 4.

We notice that the above procedure is nondeterministic because of step 4. Fix an output of the procedure and let E be the following sets:

1. If P is marked positively, then $Has(a, P) \in E$.

2. If P is marked negatively, then $Has(a, neg(P)) \in E$.

3. If $ab(a, P, t)$ is set then $ab(a, P, t) \in E$.

4. If there is a positive path from P to Q, then $P < Q \in E$.[2]

5. $N \subseteq E$.

We prove that E is an answer set for $R1(N)$, that is, $\Gamma(E) = E$. We notice that we only have to prove that $\Gamma(E)$ and E agree on the predicates Has and ab. In order to do this, we first prove the following lemma about the procedure:

Lemma 5.1 *We have:*

(a) If P is marked positively (negatively), then it cannot later be marked negatively (positively), where P is any property constant.

(b) If P is marked positively, then it will be selected at some point in step 4.

(c) If Q is marked positively (negatively) in step 4.1 (step 4.2) by using the fact that $ab(a, P, Q)$ ($ab(a, P, neg(Q))$) has not been set, then it cannot be set later.

Proof of the Lemma: (a) and (b) follow directly from the procedure. We prove (c) for the case of $ab(a, P, Q)$. The case for $ab(a, P, neg(Q))$ is similar. If $ab(a, P, Q)$ is later set, then this happens either at step 3 or at step 4.3. The former case is impossible since Q has already been marked positively. Assuming the latter case, that is, there is a property node P_1 such that $P_1 \sqsubset P$, $P_1 \rightarrow neg(Q) \in N$, and P_1 is selected later than P. Since P_1 is marked positively, there is a sequence $(a, P_n, P_{n-1}, ..., P_1)$ such that $a \rightarrow P_n \in N$, $P_i \rightarrow P_{i-1} \in N$, $ab(a, P_i, P_{i-1})$ is not set, and each P_i is marked positively, where $n \geq i > 1$ and $n \geq 1$. By $P_1 \sqsubset P$, we have $P_i \sqsubset P$ for every $n \geq i \geq 1$. This means that P_n has to be selected before P, which in turn means that P_{n-1} has to be selected before P, and so inductively, P_1 has to be selected before P, a contradiction. **End of the proof of the Lemma.**

We now prove that $\Gamma(E) = E$. We first prove $\Gamma(E) \subseteq E$ by showing that E satisfies the condition for $\Gamma(E)$. We consider only nontrivial rules:

1. Rule (5.4): Let $Has(a, P) \in E$, $P \rightarrow t \in E$, and $ab(a, P, t) \notin E$. We have to prove that $Has(a, t) \in E$. By the definition of E, the assumptions mean that

[2] As we noted earlier, we ignore irrelevant sentences like $P < neg(Q)$.

P is marked positively, and $ab(a, P, t)$ has not been set. Thus by step2 4.1 and 4.2, if $t = Q$, then Q is marked positively; if $t = neg(Q)$, then Q is marked negatively. In any cases, $Has(a, t) \in E$.

2. Rule (5.5): Let $Has(a, P_1) \in E$, $P_1 \rightarrow neg(Q) \in E$, and $P_1 \sqsubset P_2$. We have to prove that $ab(a, P_2, Q) \in E$. This follows easily from the step 4.3, Lemma 5.1, and the definition of E.

3. Rule (5.6): Similar to the last case.

4. Rules (5.7) and (5.8): Follow easily from step 3 and the definition of E.

Thus E satisfies the condition of $\Gamma(E)$, and $\Gamma(E) \subseteq E$.

We now prove that $E \subseteq \Gamma(E)$. We prove this by induction on the number of times that the iterative step (4) is executed. First it is easy to see that before step (4) is reached, if P is marked positively (negatively), then $Has(a, P) \in \Gamma(E)$ ($Has(a, neg(P)) \in \Gamma(E)$); and if $ab(a, P, t)$ is set, then it must be in $\Gamma(E)$. The inductive step can be easily proved by using Lemma 5.1. ■

The logic program captures a theory of inheritance which has a simple mathematical definition.

Definition 5.1 *The* arguments *in the net N are defined as follows:*

1. *If $a \rightarrow t \in N$, then it is an argument for $Has(a, t)$, where a is an object constant, and t a property term.*

2. *If α is an argument for $Has(a, P)$, and $P \rightarrow t \in N$, then $\alpha \rightarrow t$ is an argument for $Has(a, t)$, where a is an object constant, P a property constant, and t a property term.*

Let T be a set of arguments in N. We say that T *supports* a sentence A if there is an argument for A in T.

Definition 5.2 *A set T of arguments in N is a* theory *iff*

1. *$N \subseteq T$.*

2. *For any object constant a and property term t, T does not support both $Has(a,t)$ and $Has(a,\bar{t})$.*

3. *If $\alpha \to t \in T$, and α is an argument, then $\alpha \in T$.*

4. *If $a \to \alpha \to P \in T$ and $P \to t \in N$, then T supports $Has(a,t)$ unless T supports $Has(a,\bar{t})$, where α may be empty.*

5. *If $a \to \alpha \to P_1 \in T$, $P_1 \to t \in N$, and there is a positive path from P_1 to P_2 but none from P_2 to P_1, then no argument of the form $a \to \beta \to P_2 \to \bar{t}$ will be in T, where α and β may be empty.*

Following tradition, we say that N is *consistent* if it does not contain both $a \to P$ and $a \to neg(P)$, for any object node a and property node P. Let S be a set of sentences of the form $Has(a,t)$. We say that a theory T *supports* S iff S is the set of sentences supported by T. We say that an answer set E supports S iff S is the restriction of E to the sentences of the form $Has(a,t)$.

Theorem 9 *Let N be a consistent net, and S a set of sentences of the form $Has(a,t)$. Then S is supported by a theory of N iff it is supported by an answer set of $R1(N)$.*

Proof: Let E be an answer set of $R1(N)$, and let E support S. We prove that there is a theory T that supports S. Define T inductively as follows:

1. $N \subseteq T$.

2. If $a \to \alpha \to P \in T$, $P \to t \in N$, and $ab(a,P,t) \notin E$, then $a \to \alpha \to P \to t \in T$, where a is any object constant, P any property constant, t any property term, and α may be empty.

By the construction of T and the rules (5.3) and (5.4), it is easy to see that T supports S. We now prove that T is a theory:

1. $N \subseteq T$: By the construction of T.

2. T does not support both $Has(a, t)$ and $Has(a, \bar{t})$: By the consistency of N and the rules (5.7) and (5.8).

3. If $\alpha \to t \in T$, and α is an argument, then $\alpha \in T$: By the construction of T.

4. Let $a \to \alpha \to P \in T$, $P \to t \in N$, and T does not support $Has(a, \bar{t})$. We prove that T supports $Has(a, t)$. There are two cases. If $ab(a, P, t) \notin E$, then the result easily follows from the definition of T. Otherwise, since $Has(a, \bar{t}) \notin E$, there must be a P_1 such that $P_1 \sqsubset P$ (see the proof of the last theorem for the meaning of $\sqsubset$), $Has(a, P_1) \in E$, and $P_1 \to \bar{t} \in N$. This means that $ab(a, P_1, \bar{t}) \in E$. If $ab(a, P_1, \bar{t})$ is deduced using the rule (5.7) or (5.8), then $Has(a, t) \in E$, thus T supports $Has(a, t)$. Otherwise, since N is finite and $\sqsubset$ is a strict partial order, there must be a P_2 such that $P_2 \sqsubset P_1$, $Has(a, P_2)$, $P_2 \to t$, and $ab(a, P_2, t) \notin E$. This means that $Has(a, t)$ is in E, thus is supported by T.

5. Let $a \to \alpha \to P_1 \in T$, $P_1 \to t \in N$, and $P_1 \sqsubset P_2$. We prove that no argument of the form $\beta \to P_2 \to \bar{t}$ will be in T. By the assumption, $Has(a, P_1) \in E$. Thus by rule (5.6), $ab(a, P_2, \bar{t}) \in E$. Thus by the construction of T, no argument of the form $\beta \to P_2 \to \bar{t}$ will be in T.

Thus T is a theory.

Conversely, let T be a theory and let T support S. We prove that there is an answer set E of $R1(N)$ such that E supports S. Define E as follows:

1. $N \subseteq E$.

2. $P_1 < P_2 \in E$ provided there is a positive path from P_1 to P_2.

3. $Has(a, t) \in E$ provided $Has(a, t) \in S$.

4. $ab(a, P, t) \in E$ provided $Has(a, \bar{t}) \in E$.

5. $ab(a, P_1, t) \in E$ provided that there is a P_2 such that $Has(a, P_2) \in E$, $P_2 \sqsubset P_1$, and $P_2 \to \bar{t} \in N$.

Obviously, E supports S. We prove that E is an answer set of $R1(N)$, that is, $E = \Gamma(E)$.

We prove $\Gamma(E) \subseteq E$ by showing that E satisfies the condition (β). We prove this by considering the rules in $R1(N)$ one by one:

1. The case for the rules (5.1) and (5.2) is obvious.

2. Let $Has(a, P) \in E$, $P \to t \in E$, and $ab(a, P, t) \notin E$. We prove that $Has(a, t) \in E$. Since $Has(a, P) \in E$, thus T supports it. Therefore by the fact that T is a theory, T supports $Has(a, t)$ unless it supports $Has(a, \bar{t})$. But T can not support $Has(a, \bar{t})$ otherwise $ab(a, P, t) \in E$. Thus $Has(a, t) \in S \subseteq E$.

3. The case for the rules (5.5) – (5.8) is obvious from our definition of E.

Next we prove that $E \subseteq \Gamma(E)$. It is easy to see that we only have to prove that for any sentence of the form $Has(a, t)$, if it is in E, then it is also in $\Gamma(E)$. Since E supports S, we only have to prove that if there is an argument α in T for $Has(a, t)$, then $Has(a, t) \in \Gamma(E)$. We prove this by induction on the length of α. It is trivial if the length of α is 1. Inductively, suppose that the claim is true for all arguments of length $< n$ in T. Let α be of length n and of the form $a \to \beta \to P \to t$. We prove that $Has(a, t) \in \Gamma(E)$. First we notice that $a \to \beta \to P \in T$. Thus by the inductive assumption, $Has(a, P) \in \Gamma(E)$. We claim that $ab(a, P, t) \notin E$. Otherwise, $ab(a, P, t) \in E$. Then by the definition of E, either $has(a, \bar{t}) \in E$ or there must be a P_1 such that $Has(a, P_1) \in E$, $P_1 \sqsubset P$, and $P_1 \to \bar{t} \in N$. The first case is impossible by condition (2) in the definition of theory, and the second case is impossible by condition (5). Therefore by rule (5.4), we have $Has(a, t) \in \Gamma(E)$. ■

From the theorem, we have the consistency and stability of our theory:

Corollary 9.1 *Let E be an answer set of $R1(N)$. Then (1) E contains both $Has(a, t)$ and $Has(a, \bar{t})$ for some a and t iff N is not consistent; (2) If $a \to t_1 \to t_2 \to \ldots \to t_n$ is an argument in N, and $ab(a, t_i, t_{i+1})$ is not in E for $1 \leq i < n$, then for any $1 \leq i < j \leq n$ there is an answer set E' for $R1(N \cup \{t_i \to t_j\})$ such that E and E' support the same set of sentences.*

Proof: (1) is trivial. For the proof of (2), notice that there is a theory T such that $a \to t_1 \to \ldots \to t_n \in T$, and T supports the same set of sentences with E. It is easy to see that T is also a theory for $N \cup \{t_i \to t_j\}$. ■

Many credulous theories of inheritance in the literature are similar to the one we have just defined. These include the ones outlined in [Morris 1988] and [Gelfond 1989]. The closest one is that in [Gelfond and Przymusinska 1989]. Although Gelfond and Przymusinska use autoepistemic logic, as they point out, the autoepistemic theory that they use can be translated into logic programs. Our overall methodology seems in accordance with theirs. There are a few technical differences. The most important one is manifested in their formalization of the aforementioned two principles. For example, according to Gelfond and Przymusinska, Touretzky's principle is formalized (in our terms) as follows (they only consider acyclic nets):

$$ab(x, p_2, q) \leftarrow Has(x, p_1), Has(x, p_2), p_1 \to neg(q), p_2 \to q, p_1 < p_2, \neg ab(x, p_1, neg(q)).$$

The main difference between the above rule and our rule (5.5) is the extra

$$\neg ab(x, p_1, neg(q))$$

in the right hand side of the above rule. We suspect that this makes no difference for acyclic nets, which are the nets considered by Gelfond and Przymusinska. It is interesting to note that for arbitrary nets, our proofs for the above theorems will no longer be valid under Gelfond and Przymusinska's formalization, and we do not know any alternatives.

5.4 A skeptical theory of inheritance

The theory in the last section is credulous in the sense that the reasoner always maintains consistency by arbitrarily breaking a deadlock, or in other words, the reasoner tries to conclude as much as possible while maintaining consistency. The skeptical reasoner on the other hand, in the words of Touretzky *et al.* [1987] from whom we borrow the terms "skeptical" and "credulous," "refuses to draw conclusions in ambiguous situations." More precisely, in the case of inheritance hierarchies, when two conflicting arguments can not be resolved by using Touretzky's principle, then instead of arbitrarily chosing one and rejecting the other, skeptical reasoners reject both.

Thus a natural skeptical theory of inheritance that corresponds to the credulous theory in the last section can be obtained by adding the following two rules to the logic programs in the last section:

$$\begin{aligned} ab(x, p_1, q) &\leftarrow Has(x, p_1), Has(x, p_2), p_1 \rightarrow q, p_2 \rightarrow neg(q), \\ &\quad \neg p_1 < p_2, \neg p_2 < p_1. \qquad (5.9) \\ ab(x, p_2, neg(q)) &\leftarrow Has(x, p_1), Has(x, p_2), p_1 \rightarrow q, p_2 \rightarrow neg(q), \\ &\quad \neg p_1 < p_2, \neg p_2 < p_1. \qquad (5.10) \end{aligned}$$

It is interesting to notice that rules (5.5) and (5.9) can be combined into the following one:[3]

$$ab(x, p_1, q) \leftarrow Has(x, p_1), Has(x, p_2), p_1 \rightarrow q, p_2 \rightarrow neg(q), \neg p_1 < p_2. \qquad (5.11)$$

Similarly, rules (5.6) and (5.10) can also be combined into a single rule.

For any net N, let $R2(N)$ be $R1(N)$ plus rules (5.9) and (5.10). For any acyclic net N, $R2(N)$ is well-behaved and has a unique answer set. We show this by first presenting an equivalent mathematical definition that has the same inductive flavor as that in [Horty *et al.* 1987]. In the following, let N be an acyclic net.

Let α be an argument for $Has(a, P)$ or $Has(a, neg(P))$ in N. We define the *degree* of α, written $deg(\alpha)$, to be the number of links in the longest path from a to P (notice that the path may contain negative links). Notice that according to the definition, all arguments for $Has(a, P)$ or $Has(a, neg(P))$ have the same degree. Since N is acyclic, this notion of degree is well-defined. For any argument α, inductively define whether α is supported by N, written $N \vdash \alpha$, as follows:

1. If $\alpha \in N$, then $N \vdash \alpha$.

[3] Actually, rule (5.5) allows $ab(a, P_1, t)$ to be inferred even when $Has(a, P_1)$ is not established or when there is no link from P_1 to t. But in these cases, it does not matter whether $ab(a, P_1, t)$ is true or not.

2. Inductively, suppose that for any argument α with $deg(\alpha) < n$, $n > 1$, the question of whether $N \vdash \alpha$ has been settled. For any $\alpha = a \rightarrow \beta_1 \rightarrow P_1 \rightarrow t$ with $deg(\alpha) = n$, $N \vdash \alpha$ iff

   (a) $N \vdash a \rightarrow \beta_1 \rightarrow P_1$.

   (b) $a \rightarrow \overline{t} \notin N$.

   (c) For any argument of the form $a \rightarrow \beta_2 \rightarrow P_2 \rightarrow \overline{t}$ such that $N \vdash a \rightarrow \beta_2 \rightarrow P_2$, $P_1 < P_2$ holds, that is, there is a positive path from P_1 to P_2.

For any object a and property t, we say that N *supports* $Has(a,t)$ iff it supports an argument for $Has(a,t)$. We have the following theorem:

Theorem 10 *Let N be an acyclic net. Then there is a unique answer set of $R2(N)$, and for any object constant a and any property t, $Has(a,t)$ is in the answer set of $R2(N)$ iff $Has(a,t)$ is supported by N.*

Proof: Without loss of generality, we assume that there is only one object node a in N. We first prove the uniqueness of the answer set of $R2(N)$. Let E_1 and E_2 be two answer sets of $R2(N)$. We prove that $E_1 = E_2$. First, notice that if for any property term t, $Has(a,t) \in E_1$ iff $Has(a,t) \in E_2$, then $E_1 = E_2$. For any property term t, define the degree of t, written $deg(t)$, to be the degree of arguments for $Has(a,t)$ or $Has(a,\overline{t})$, if there are any, and to be 1, otherwise. We prove by induction on $deg(t)$ that E_1 and E_2 agree on $has(a,t)$.

If $deg(t) = 1$, then $Has(a,t) \in E_1$ iff $a \rightarrow t \in N$ iff $Has(a,t) \in E_2$. Inductively, suppose that for any t with $deg(t) < n$, $n > 1$, the result holds. Let t be a term such that $deg(t) = n$. Assume that E_1 and E_2 do not agree with each other on $Has(a,t)$, for example, suppose that $Has(a,t) \in E_1$ and $Has(a,t) \notin E_2$. Since $Has(a,t) \in E_1$, there must be an argument $a \rightarrow \alpha \rightarrow P_1 \rightarrow t$ for $Has(a,t)$ such that $Has(a,P_1) \in E_1$ and $ab(a,P_1,t) \notin E_1$. By the inductive assumption, $Has(a,P_1) \in E_2$. Thus $ab(a,P_1,t) \in E_2$. Since $a \rightarrow \overline{t} \notin N$, there must be a node P_2 such that $Has(a,P_2) \in E_2$, $P_2 \rightarrow \overline{t} \in N$, and $P_1 < P_2 \notin E_2$. It follows that $deg(P_2) < n$, thus $Has(a,P_2) \in E_1$. This contradicts with the fact that $ab(a,P_1,t) \notin E_1$.

Next we prove the rest of the theorem by proving that the following set E is an answer set for $R2(N)$:

1. $N \subseteq E$.
2. If there is a positive path from P to Q, then $P < Q$, where P and Q are property constants.
3. If N supports $Has(a,t)$, then $Has(a,t) \in E$, where t is any property term.
4. If $Has(a,t) \in E$, then $ab(a,P,\overline{t}) \in E$, where P is a property constant and t a property term.
5. If $Has(a,P_1) \in E$, $P_1 \to t \in N$, and there is a positive path from P_1 to P_2, then $ab(a,P_2,\overline{t}) \in E$, where P_1 and P_2 are object constants, and t a property term.
6. For any argument $\alpha = a \to \beta \to P \to t$, if $N \vdash a \to \beta \to P$, but $N \not\vdash \alpha$, then $ab(a,P,t) \in E$.

We prove that $E = \Gamma(E)$ for $R2(E)$. Again we show that $\Gamma(E) \subseteq E$ by proving that E satisfies the condition (γ). We only check the nontrivial cases:

1. Rule (5.4): Let $Has(a,P) \in E$, $P \to t \in N$, and $ab(a,P,t) \notin E$. We have to show that $Has(a,t) \in E$. Since $Has(a,P) \in E$, there is an argument α for it such that $N \vdash \alpha$. Thus $N \vdash \alpha \to t$, for otherwise $ab(a,P,t) \in E$ by the construction of E, a contradiction.
2. Rules (5.9) and (5.10): Let $Has(a,P_1) \in E$, $Has(a,P_2) \in E$, $P_1 \to t \in N$, $P_2 \to \overline{t} \in N$, $P_1 < P_2 \notin E$, and $P_2 < P_1 \notin E$. We have to show that both $ab(a,P_1,t)$ and $ab(a,P_2,\overline{t})$ are in E. By the assumptions, there are two arguments α and β for $Has(a,P_1)$ and $Has(a,P_2)$, respectively, such that $N \vdash \alpha$ and $N \vdash \beta$. It is easy to see that $N \not\vdash \alpha \to t$ and $N \not\vdash \beta \to \overline{t}$ hold. Thus by the construction of E, $ab(a,P_1,t)$ and $ab(a,P_2,\overline{t})$ are in E.

We now prove $E \subseteq \Gamma(E)$. We only have to show this for the Has and ab predicates. Let α be an argument for $Has(a,t)$. We prove by induction on $deg(\alpha)$ that if $N \vdash \alpha$, then $Has(a,t) \in \Gamma(E)$. The base case is easy to see. Suppose the result holds for any α with $deg(\alpha) < n$. Let $deg(\alpha) = n$, $\alpha = a \rightarrow \alpha_1 \rightarrow P \rightarrow t$, and $N \vdash \alpha$. By the inductive assumption, $Has(a,P) \in \Gamma(E)$. Thus we only have to prove that $ab(a,P,t) \notin E$. Suppose $ab(a,P,t) \in E$, then there three cases:

1. N supports $Has(a,\bar{t})$: Since N already supports $Has(a,t)$, this case is possible only if $a \rightarrow \bar{t} \in N$, but this contradicts the fact that $N \vdash \alpha$.

2. There is a node P_1 such that $P_1 \rightarrow \bar{t} \in N$, $Has(a,P_1)$ is supported by N, and $P_1 < P$: This again contradicts with the fact that $N \vdash \alpha$.

3. There is an argument β for $Has(a,P)$ such that $N \vdash \beta$ but $N \not\vdash \beta \rightarrow t$: This would mean that $N \not\vdash \alpha$, a contradiction.

Finally we prove that if $ab(a,P,t) \in E$, then $ab(a,P,t) \in \Gamma(E)$. The only nontrivial case is that if $P \rightarrow t \in N$, and there is an argument α for $Has(a,P)$ such that $N \vdash \alpha$ but $N \not\vdash \alpha \rightarrow t$, then $ab(a,P,t) \in \Gamma(E)$. Suppose the assumption is true, then there are two cases:

1. $a \rightarrow \bar{t} \in N$: Then $ab(a,P,t) \in \Gamma(E)$ by the rules (5.7) and (5.8).

2. There is an argument of the form $a \rightarrow \beta \rightarrow P_1 \rightarrow \bar{t}$ such that $N \vdash a \rightarrow \beta \rightarrow P_1$ and $P < P_1 \notin E$: Then $Has(a,P_1) \in \Gamma(E)$, and $P_1 \rightarrow \bar{t} \in N$. Thus $ab(a,P,t) \in \Gamma(E)$ according to the rules (5.5) and (5.6) if $P_1 < P$, and according to rules (5.9) and (5.10) otherwise.

Thus $E = \Gamma(E)$, that is, E is an answer set of $R2(N)$. ■

Again the skeptical theory is consistent and stable in the following sense:

Theorem 11 *Let N be an acyclic net, and E the answer set of $R2(N)$. Then (1) E contains both $Has(a,t)$ and $Has(a,\bar{t})$ for some a and t iff N is not consistent; (2) If $N \vdash a \rightarrow t_1 \rightarrow \ldots \rightarrow t_n$, then for any $1 \leq i < j \leq n$ and any a and t, $E \cup \{t_i \rightarrow t_j\}$ is the answer set of $R2(N \cup \{t_i \rightarrow t_j\})$.*

Proof: (1) follows easily from the rules (5.7) and (5.8). (2) follows from the fact that for any property nodes P_1 and P_2, $P_1 < P_2$ is true in N iff it is true in $N \cup \{t_i \to t_j\}$, and $Has(a, t_l) \in E$ for any $1 \leq l \leq n$. ■

Since $R2(N)$ is obtained from $R1(N)$ by adding the rules (5.9) and (5.10), thus in the cases where the two rules have no effects, $R1(N)$ and $R2(N)$ should be the same. More precisely, we have the following theorem which shows the relationship between the credulous theory in the last section and the skeptical theory in this section:

Theorem 12 *Let N be an acyclic net, and E_2 the answer set of $R2(N)$. Then $R1(N)$ has a unique answer set E_1 iff for any a and t, $Has(a, t) \in E_1$ iff $Has(a, t) \in E_2$.*

Proof: For any object constant a and property term t, let $Pr(a, t)$ be the set of arguments of the form $a \to \alpha \to P \to t$ such that $N \vdash a \to \alpha \to P$, where P is a property constant. We prove the theorem by showing that $R1(N)$ has a unique answer set iff for any a and t, one of the following conditions holds:

1. Either $a \to t \in N$ or $a \to \bar{t} \in N$.

2. Either $Pr(a, t)$ or $Pr(a, \bar{t})$ is empty.

3. There is an argument $\alpha \to P \to t$ in $Pr(a, t)$ such that for any argument $\beta \to P_1 \to \bar{t}$ in $Pr(a, \bar{t})$, there is a positive path from P to P_1.

4. There is an argument $\alpha \to P \to \bar{t}$ in $Pr(a, \bar{t})$ such that for any argument $\beta \to P_1 \to t$ in $Pr(a, t)$, there is a positive path from P to P_1.

Suppose the claim holds. Then it is not hard to see that for any answer set E_1 of $R1(N)$, if $Has(a, t)$ is supported by N, then $Has(a, t) \in E_1$. Thus $R1(N)$ has a unique answer set. Conversely, suppose the claim does not hold. Let $a \to \alpha \to t$ be a shortest argument such that the above condition for a and t does not hold. Then it is easy to see that there are two answer sets of $R1(N)$ such that one includes $Has(a, t)$, and the other $Has(a, \bar{t})$. ■

So far in this section, we have only considered acyclic nets. Unfortunately, for arbitrary nets, $R2(N)$ does not behave well. In the general case, $R2(N)$ can have multiple answer sets or none at all. It seems to us that the only skeptical theory that behaves well in the general case is the one that takes the intersection of the all answer sets for $R1(N)$, and we do not know how to characterize it by a logic program.

5.5 Capturing other theories of inheritance

As we pointed out in the introduction, one of our goals is to provide an approach to formalizing various intuitions about inheritance in a uniform way. In this section, we show how the approach outlined in the last two sections can be used to formalize other intuitions about inheritance. In particular, we show how to capture Horty-Thomason-Touretzky's skeptical theory [Horty *et al.* 1987] in logic programs. A similar way can be used to capture Touretzky's mathematical theory [Touretzky 1986].

For the sake of coherence, we first redefine the relevant notions of [Horty *et al.* 1987] in our terms. In the following, let N be an acyclic net, as required in [Horty *et al.* 1987]. For any argument α, inductively define whether α is supported by N in Horty-Thomason-Touretzky's sense, written $N \vdash_{htt} \alpha$, as follows:

1. If $\alpha \in N$, then $N \vdash_{htt} \alpha$.
2. Inductively, suppose that for any argument α with $deg(\alpha) < n$ (see the definition in the last section), $n > 1$, the question of whether $N \vdash_{htt} \alpha$ has been settled. For any $\alpha = a \to \beta_1 \to P_1 \to t$ with $deg(\alpha) = n$, $N \vdash_{htt} \alpha$ iff
   (a) $N \vdash_{htt} a \to \beta_1 \to P_1$.
   (b) $a \to \bar{t} \notin N$.
   (c) Either $a \to t \in N$ or for any argument of the form $a \to \beta_2 \to P_2 \to \bar{t}$ such that $N \vdash_{htt} a \to \beta_2 \to P_2$, there exists an argument of the form $\beta = a \to \beta_3 \to P_3 \to \beta_4 \to P_2$ such that $P_3 \to t \in N$ and $N \vdash_{htt} \beta$.

It is interesting to note that $\vdash_{htt}$ is very similar to $\vdash$. The essential difference is in the interpretation of the statement "P is more specific than Q." For us, the notion of specificity is a monotonic property about a net, and the above statement is true iff there is a positive path from P to Q. But according to Horty *et al.*, the notion of specificity is a nonmonotonic property about a net, and the above statement is true iff there is an argument such that it passes through P and ends at Q, and is supported (in the sense of Horty *et al.*). Thus if $N = \{a \to P_1, a \to neg(P_2), a \to P_3, P_1 \to P_2, P_2 \to P_3, P_1 \to P_4, P_3 \to neg(P_4)\}$, then according to the theories in the last two sections, P_1 is more specific than P_3. But this is not true according to Horty-Thomason-Touretzky's theory. Thus $N \vdash a \to P_1 \to P_4$, but $N \not\vdash_{htt} a \to P_1 \to P_4$.

We can formalize Horty-Thomason-Touretzky's theory in logic programs by directly introducing paths into the language.

In the following, we will use $[p, q]$ as a shorthand for $list(p, q)$, $[p_1, p_2, p_3]$ as a shorthand for $list(list(p_1, p_2), p_3)$, and so on, where $list$ is a function over properties. We will use $S(a, [P_1, P_1, P_2, ..., P_n])$[4] to mean that the path $(a, P_1, P_2, ..., P_n)$ is supported, and $less(a, Q, P)$ to mean that there is a node P' such that P' is more specific than P (w.r.t. a and Q, in Horty *et al.*'s sense). The following program is a direct translation of the definition for $\vdash_{htt}$:

$$S(x, [p, p]) \leftarrow x \to p \tag{5.12}$$
$$S(x, [p_1, p_2, q]) \leftarrow S(x, [p_1, p_2]), p_2 \to q, \neg ab(x, [p_1, p_2, q]) \tag{5.13}$$

$$ab(x, [p_1, q]) \leftarrow S(x, [p_2, neg(q)] \tag{5.14}$$
$$ab(x, [p_1, neg(q)]) \leftarrow S(x, [p_2, q]) \tag{5.15}$$
$$ab(x, [p_1, p_2, q]) \leftarrow S(x, [q_1, q_2]), q_2 \to neg(q), \neg less(x, q, q_2) \tag{5.16}$$
$$ab(x, [p_1, p_2, neg(q)]) \leftarrow S(x, [q_1, q_2]), q_2 \to q, \neg less(x, neg(q), q_2) \tag{5.17}$$

$$less(x, q, p) \leftarrow x \to q \tag{5.18}$$
$$less(x, q, p) \leftarrow S(x, [[q_1, p_1], [q_2, p]]), p_1 \to q \tag{5.19}$$
$$less(x, q, p) \leftarrow S(x, [q_1, p_1, p]), p_1 \to q \tag{5.20}$$

[4]The double occurrences of P_1 looks a little odd. It is just a way to make the following program more uniform.

$$S(x,[p_1,[p_2,p_3]]) \leftarrow p_2 \rightarrow p_3, S(x,[[p_1,p_2],p_3]). \tag{5.21}$$

$$S(x,[p_1,[[p_2,p_3],p_4]]) \leftarrow S(x,[[p_1,p_2],[p_3,p_4]]). \tag{5.22}$$

For any net N, let $R3(N)$ be the union of the above rules with N. We have the following theorem:

Theorem 13 *For any acyclic net N and any argument $\alpha = a \rightarrow P_1 \rightarrow \ldots \rightarrow P_n \rightarrow t$ in N, $N \vdash_{htt} \alpha$ iff $S(a,[P_1,...,P_n,t])$ is in the unique answer set of $R3(N)$.*

Proof: Similar to the proof of Theorem 10, by using the induction on $deg(\alpha)$. ■

5.6 Discussion

We have formalized some intuitions about inheritance in logic programs with negation as failure. We believe that our approach is general enough to be used to formalize other intuitions about inheritance.

Our basic ideas grew out of the work in [Lin and Shoham 1989]. There are also many similar ideas in the literature. We have mentioned earlier that aside from some technical differences, the approach used in [Gelfond and Przymusinska 1989] is basically the same as ours.

Obviously, our work is strongly influenced by Horty, Thomason, and Touretzky's. In a sense, our work can be considered a logical formalization of their mathematical theories (or algorithms).

Of course, we owe the *ab* predicate to McCarthy [1986]. Furthermore, our methodology seems quite in the spirit of [McCarthy 1986].

The use of the *ab* predicate makes our work appear different from that in [Etherington and Reiter 1983] on the surface. A more important difference is in Etherington and Reiter's choice of semi-normal default theories as the target representation.

Some points in [Ginsberg 1989] are worth mentioning. Basically, we share with Ginsberg the concern about the right level of transformation. At one extreme we have some logics specially designed for inheritance [Thomason and Horty 1989], which eliminate the need for translation. At another extreme we can imagine trying to formalize nonmonotonic inheritance in first-order logic. Any attempt to do this must unavoidably use a transformation that somehow incorporates all the nonmonotonicities in it. Logic programs with negation as failure as used in this chapter seem to be a good choice.

Finally there are some probability-based theories of inheritance [Geffner and Pearl 1987, Bacchus 1987]. The exact relationship of our work with these theories is still under investigation.

In summary, we have presented a methodology for formalizing various intuitions about inheritance. Most of the theories that we formalized above are not new. Informally and formally, they have been discussed by many authors. Rather, we emphasize the proposed methodology, which has its roots in [McCarthy 1986]. Although using this methodology it is possible to invent some *new* theories about inheritance, we see no point in doing this without proper applications in mind. We generally agree with Touretzky, Horty, and Thomason [1987] that there are many equally sound intuitions about inheritance, and they cannot be rated by purely theoretical considerations. Instead we should have a general methodology ready for use so that for any particular application we can devise the most appropriate theory for it. Indeed we consider applying formal theories about nonmonotonic inheritance to real application programs (for example, in natural language understanding systems) one of the more important problems.

Chapter 6

Application II: Provably Correct Theories of Action

The histories of the frame problem [McCarthy and Hayes 1969][1], and of the particular Yale Shooting Problem (YSP) which has become its best known illustration [Hanks and McDermott 1986], have followed a disturbing pattern. The frame problem itself, although introduced in the context of formalizing common sense, was never formally defined, and was only illustrated through suggestive examples. This is an initial disturbing factor.

A second disturbing factor is that, despite the lack of a formal definition, arguments were made that a particular collection of formal tools, namely nonmonotonic logics, would 'solve' the problem. Again, no formal analysis was provided, and the claim was based only on sketchy examples.

A third disturbing factor is that the response in the community was not that the above claim is ill defined, but that it's false. In particular, the Yale Shooting Problem was proposed as an illustration of the falseness.

Given these shaky foundations, it is not surprising that subsequent research on the topic became increasingly splintered and controversial. From the outset there were arguments that the YSP is not a problem at all [Loui 1987]. Simultaneously there were several proposed solutions to it [Shoham 1986, Lifschitz 1986, Kautz 1986,

[1]An extended abstract of this chapter will appear as [Lin and Shoham 1991]

Lifschitz 1987, Haugh 1987]. Then there were counter-arguments that each of these solutions was 'wrong,' in that it didn't solve other problems such as the 'qualification problem' or the 'ramification problem,' or that that it supported 'prediction' but not 'explanation.' New examples were then devised, with names such as the Stolen Car Problem [Kautz 1986] and the Stanford Murder Mystery [Baker 1989]. The responses again varied, including dismissals of some of the complaints, as well as new solutions to the YSP that allegedly avoided some of these problems [Morgenstern and Stein 1988, Lifschitz and Rabinov 1989, Baker and Ginsberg 1989, etc.]. Each solution has attracted some measure of criticism.

The lack of precise criteria against which to evaluate theories of action does not mean that the research has been worthless; quite the contrary. It is widely recognized that the frame problem is real and that its identification was insightful, even if it has not yet been formally defined. Similarly, the YSP led to major improvements in the understanding of nonmonotonic logics and their applications, as well as to better understanding of formal temporal reasoning.

Nevertheless, in order to better understand what we have achieved so far, it is important to arrive at precise criteria for the adequacy of theories of action. In this chapter we take a step in that direction. Specifically, we identify a formal yet intuitive adequacy criterion, prove a certain class of monotonic theories of action adequate relative to this criterion, and then show an equivalent nonmonotonic counterpart for a significant subclass of the theories. To our knowledge this is the first instance of provably-correct nonmonotonic temporal reasoning.

This chapter is structured as follows. In the following section we illustrate our approach through an analogy with data bases and closed-world assumption. We then start a technical development; after short logical preliminaries in section 3, in section 4 we define the epistemological adequacy of a (monotonic or nonmonotonic) theory of action. In section 5 we illustrate the notion through the YSP. In section 6 we define a wide class of monotonic theories of action, and show those to be epistemologically adequate. In section 7 we show that a version of circumscription captures a subclass of the theories. In section 8 we point to our future work and make some concluding remarks.

6.1 The approach

Recall the following intuitive explanation of the frame problem. Suppose we are trying to formalize the effects of actions. Usually, an action causes only a small number of changes. For example, when we paint a block, only the color of the block changes. Most of the other facts, such as the location of the block, the smell of the paint, *et cetera*, do not change. The frame problem is the problem of representing concisely these numerous facts that are unaffected by an action.

Our approach to making the problem precise is conceptually very simple, and perhaps best illustrated by an analogy with simple data bases. Consider a data base of flight connections between pairs of cities. One way to structure the data base is by a set of assertions of the form $Flight(x, y)$ and $\neg Flight(x, y)$, where for each pair of cities A, B exactly one of $Flight(A, B)$ and $\neg Flight(A, B)$ appears in the data base. The semantics of this data base are those of classical logic. This is an *epistemologically complete* representation since for any pair of cities it tells one whether the two are connected. However, while epistemologically adequate, the representation is pragmatically inadequate: it requires representation of all pairs of cities, whereas the connectivity graph is usually quite sparse.

The solution is, of course, to omit all the $\neg Flight(x, y)$ assertions, and infer $\neg Flight(A, B)$ 'by default' in the absence of $Flight(A, B)$. This is a simple application of the so-called *closed world assumption (CWA)* [Reiter 1978], and the equivalent monotonic formulation can be regenerated from the abbreviated representation through *data base completion* [Clark 1978]. This concise representation is epistemologically complete since it too entails $Flight(A, B)$ or $\neg Flight(A, B)$ for all pairs of cities A and B, albeit nonmonotonically. Furthermore, the nonmonotonic version is epistemologically correct in a stronger sense: it is sound and complete relative to the monotonic version, since they entail the same facts.

Thus there are two criteria for evaluating the epistemological adequacy of a theory. Both monotonic and nonmonotonic theories can be tested for their epistemological completeness; this is an absolute criterion. In addition, nonmonotonic theories can be tested for equivalence to a given, monotonic, often better understood, and typically much larger, theory; this is a relative criterion.

In principle our treatment of theories of actions will be identical; we will require them to be complete, and furthermore evaluate a nonmonotonic theory relative to an equivalent and larger monotonic one. The complications will arise from a more complex definition of epistemological completeness, resulting difficulty in determining whether a given theory is indeed epistemologically complete, and a nonmonotonic mechanism that is more complex than CWA.

6.2 Logical preliminaries

We shall base our presentation on situation calculus [McCarthy and Hayes 1969], although we believe that a similar treatment is possible in other frameworks such as temporal logics. The standard situation calculus, which we adopt here, precludes the representation of certain notions such as concurrent actions. In future publications we will address those. In this section we review the language for discussing the situation calculus. We do this briefly and almost apologetically since we realize that the situation calculus is very well known; however, we feel that in this chapter it is important to be precise about the language.

Our language $\mathcal{L}$ is a three-sorted first-order one with equality. Its three sorts are:

1. Situation sort: with situation constants $S_1, S_2, \ldots$, and situation variable $s, s_1, s_2, \ldots$. We will use $S, S', \ldots$ as meta-variables for ground situation terms.

2. Action sort: with action constants $A_1, A_2, \ldots$, and action variables $a, a_1, a_2, \ldots$. We will use $A, A', \ldots$ as meta-variables for action constants.

3. Propositional fluent sort: with fluent constants $P_1, P_2, \ldots$, and fluent variables $p, p_1, p_2, \ldots$. We will use $P, P', \ldots$ as meta-variables for fluent constants.

We have a binary function *result* whose first argument is of action sort, second argument is of situation sort, and whose value is of situation sort. Thus for any action term A, any situation term S, $result(A, S)$ is a situation term. Intuitively, $result(A, S)$ is the resulting situation when action A is performed in the situation S. We also have a binary predicate *holds* whose first argument is of fluent sort, and second argument is of situation sort. Intuitively, $holds(P, S)$ means that the fluent P

is true in the situation S.

As with any other language, we may interpret $\mathcal{L}$ classically, assuming the standard notion of entailment, or nonmonotonically, using some form of nonmonotonic entailment.

6.3 Epistemologically complete theories of action

Suppose we want to use our language to state that action *toggle* changes the truth value of the fluent P_1. We may wish to use the following axiom:

$$\forall s.(holds(P_1, s) \equiv \neg \mathrm{holds}(P_1, result(toggle, s))). \tag{6.1}$$

Intuitively speaking, this axiom alone is not enough. For example, it tells us nothing about the effects of *toggle* on P_2; for that we would need to add the following so-called frame axiom:

$$\forall s.(holds(P_2, s) \equiv \mathrm{holds}(P_2, result(toggle, s))). \tag{6.2}$$

Clearly we need such a frame axiom for every fluent that is different from P_1. But, are those frame axioms enough? In other words, do these axioms together completely formalize our knowledge about *toggle*? For this simple example it is easy to convince oneself that the above axioms indeed do completely formalize action *toggle*. In general, however, the answer may not be obvious, and it is essential for us to have a precise definition of when a first-order theory is a complete formalization of an action.

In this chapter we are only concerned with deterministic actions. Intuitively speaking, a theory T is a complete formalization of a deterministic action A if, given a complete description of the initial situation, it enables us to deduce a complete description of the resulting situation after A is performed. We now proceed to make this intuition precise. First, we notice that in actual applications, it is most convenient to talk about whether a description of a situation is complete with respect to a set of fluents in which we are interested. Thus we shall define conditions under which a theory is complete *about an action* and *with respect to a set of fluents*. This fixed set of fluents plays a role similar to that of the *Frame* predicate in [Lifschitz 1990]. In the following, let $\mathcal{P}$ be a fixed set of fluent constants.

Definition 6.1 *A set SS is a* state of the situation S *(with respect to $\mathcal{P}$) if there is a subset $\mathcal{P}'$ of $\mathcal{P}$ such that*

$$SS = \{holds(P, S) \mid P \in \mathcal{P}'\} \cup \{\neg holds(P, S) \mid P \in \mathcal{P} - \mathcal{P}'\}.$$

Therefore, if SS is a state of S, then for any $P \in \mathcal{P}$, either $holds(P, S) \in SS$ or $\neg holds(P, S) \in SS$. Intuitively, states completely characterize situations with respect to the fluents in $\mathcal{P}$. Thus we can say that a first-order theory T is epistemologically complete about action A (with respect to $\mathcal{P}$) if it is consistent, and for any ground situation term S, any state SS of S, and any fluent $P \in \mathcal{P}$, either $T \cup SS \models holds(P, result(A, S))$ or $T \cup SS \models \neg holds(P, result(A, S))$, where $\models$ is classical first-order entailment.

However, as we said earlier, the notion of epistemological completeness is not limited to monotonic first-order theories. In general, for any given monotonic or non-monotonic entailment $\models_{\sqsubset}$, we can define when a theory is epistemologically complete about an action according to the entailment $\models_{\sqsubset}$:

Definition 6.2 *A theory T is* epistemologically complete *about action A (with respect to $\mathcal{P}$, and according to $\models_{\sqsubset}$) if $T \not\models_{\sqsubset} False$, and for any ground situation term S, any state SS of S, and any fluent $P \in \mathcal{P}$, there is a finite subset SS' of SS such that either $T \models_{\sqsubset} \bigwedge SS' \supset holds(P, result(A, S))$ or $T \models_{\sqsubset} \bigwedge SS' \supset \neg holds(P, result(A, S))$.*

We note that for any sets T, SS, and formula φ, $T \cup SS \models \varphi$ if there is a finite subset SS' of SS such that $T \models \bigwedge SS' \supset \varphi$. Thus if we replace $\models_{\sqsubset}$ in Definition 6.2 by classical entailment $\models$, we get the same definition we have earlier for monotonic first-order theories.

Referring back to the data base example from section 2, we note that requiring a theory to be epistemologically complete is analogous to requiring a representation for $Flight$ to tell us, for any (A, B), whether $Flight(A, B)$ is true.

In the following, we shall say that T is a monotonic (nonmonotonic) theory if the entailment relation associated with T is classical (nonmonotonic). Thus if T is an epistemologically complete theory according to classical entailment $\models$, then we say that T is an epistemologically complete monotonic theory.

In the later sections of the chapter, we first identify a broad class of epistemologically complete monotonic theories of action. These monotonic theories may contain a large number of explicit frame axioms. Then for a class of the monotonic theories, we develop equivalent nonmonotonic ones that do not appeal to explicit frame axioms.

However, we first re-examine the Yale Shooting Problem [Hanks and McDermott 1986] as an extended example of the foregoing definitions. The YSP turns out to be a very special case of the class of theories we consider later.

6.4 The Yale shooting problem revisited

In the YSP we consider three actions: $Shoot$, $Load$, and $Wait$. After $Load$ is performed, the gun is loaded, and if the gun is loaded, then after $Shoot$ is performed, Fred is dead. Thus we have the following two 'causal rules':

$$\forall s.holds(Loaded, result(Load, s)). \tag{6.3}$$

$$\forall s(holds(Loaded, s) \supset holds(Dead, result(Shoot, s))). \tag{6.4}$$

This theory is, of course, insufficient to capture fully the effects of the three actions.

6.4.1 Monotonic completion

One way to achieve epistemological completeness is to supply frame axioms. Let $\mathcal{P} = \{Dead, Loaded\}$. For action $Shoot$, we have that for each $P \in \mathcal{P}$,

$$\forall s(\neg holds(Loaded, s) \supset (holds(P, s) \equiv holds(P, result(Shoot, s)))), \tag{6.5}$$

$$\forall s(holds(Loaded, s) \equiv holds(Loaded, result(Shoot, s))). \tag{6.6}$$

For action $Load$, we have

$$\forall s(holds(Dead, s) \equiv holds(Dead, result(Load, s))). \tag{6.7}$$

For $Wait$, we have that for any $P \in \mathcal{P}$:

$$\forall s(holds(P, s) \equiv holds(P, result(Wait, s))). \tag{6.8}$$

Let $T_1 = \{(6.3), (6.4), (6.5), (6.6), (6.7), (6.8)\}$. It is possible to show that the monotonic theory T_1 is epistemologically complete about the actions $Wait$, $Shoot$, and $Load$ with respect to $\mathcal{P}$. Using first-order logic only, we can answer queries about the theory. As a 'temporal projection' example, we have

$$T_1 \models \forall s.holds(Dead, result(Shoot, result(Wait, result(Load, s)))),$$

that is, $Dead$ holds after the $Load$, $Wait$, and $Shoot$ actions are performed in sequence in any situation. As an example of 'temporal explanation', we have

$$T_1 \models \forall s[holds(Dead, result(Shoot, s)) \wedge \neg holds(Dead, s) \supset holds(Loaded, s)],$$

that is, in any situation, if $Dead$ is not true initially but becomes true after the $Shoot$ action, then $Loaded$ must be true initially.

6.4.2 Nonmonotonic completion

Although the above monotonic theory is epistemologically complete, it does not solve the frame problem since it contains the explicit frame axioms. We now provide an equivalent nonmonotonic theory that avoids them.

It was implied in [McCarthy 1986] that the frame axioms can be replaced by the single

$$\forall pas(\neg ab(p, a, s) \supset (holds(p, s) \equiv holds(p, result(a, s)))), \tag{6.9}$$

and minimizing the abnormality predicate ab with $holds$ allowed to vary. The main technical result of [Hanks and McDermott 1986] is that this does not work. We mentioned in the introduction the slew of proposed solutions, all criticized on some grounds or others. Surprisingly, most of them are actually correct relative to the above monotonic theory. We pick as an example chronological minimization [Shoham 1986], although other proposals, such as [Lifschitz 1986] and [Kautz 1986], would work as well.

For the full definition of chronological minimization the reader is referred to the above publication; we only remind the reader that in this framework the preferred models are those in which the minimized predicate is true as *late* as possible, rather than as *infrequently* as possible. In our framework, the obvious (partial) temporal ordering on situations is

$$S < result(Shoot, S) < result(Wait, result(Shoot, S)) < ...$$

and so on. Like circumscription, we also need unique names assumptions for chronological minimization:

$$Loaded \neq Dead \neq Shoot \neq Load \neq Wait. \tag{6.10}$$

We now simply take T_2 to be the conjunction of (6.3), (6.4), (6.9), and (6.10), and chronologically minimize ab in T_2. It is now possible to show that $T_1 \cup \{(6.10)\}$ and T_2 are equivalent: for any φ in the language of T_1, $T_1 \cup \{(6.10)\} \models \varphi$ iff $T_2 \models_{\sqsubset} \varphi$, where $\models$ is classical entailment and $\models_{\sqsubset}$ is the nonmonotonic entailment. In particular, we have that the nonmonotonic theory T_2 is epistemologically complete.

We observe that previous arguments against chronological minimization and other solutions, for example by appeal to temporal explanation, involved incorrect use of the theories. For example, instead of T_2, such arguments would use

$$T_2 \cup \{holds(Dead, result(Shoot, S_0))\}.$$

Other arguments referred not to the YSP but to extended or modified examples; most of those are covered by the general treatment in the following sections.

We remark here that we have avoided formally claiming that T_2 solves the frame problem for YSP. The reason is that we do not yet have a formal criterion to decide when a representation is *concise* enough to qualify as a solution to the frame problem. Until we have one, the frame problem will continue to contain an informal factor. However, this does not affect our claim about provable correctness of theories of action.

6.5 Causal theories

In this section we introduce the notion of causal theories, and study their monotonic and nonmonotonic completions.

In reasoning about action, our knowledge can be generally divided into two kinds. First we have knowledge about the environment where the actions are taken place, commonly referred to as domain constraints. Second we have knowledge about effects of actions themselves, usually called causal rules. Causal rules only tell us the direct effects of the actions. In different environments, an action may have different side effects. Side effects are determined by the direct effects and the domain constraints.

Definition 6.3 *Let $C(s)$, $R_i(s)$, $i = 1, ..., n$, $n \geq 0$, be formulas with a free variable s. The* causal theory *of action A with domain constraint C, and direct effects $P_1, ..., P_n$ under preconditions $R_1, ..., R_n$, respectively, is the following set of sentences. The domain constraint:*

$$\forall s.C(s). \tag{6.11}$$

For each $1 \leq i \leq n$, the causal rules:

$$\forall s(R_i(s) \supset holds(P_i, result(A, s))). \tag{6.12}$$

Note that the above definition is for single action. The causal theory for a set of actions is the union of the causal theory for each individual action.

Of course, causal theories are generally incomplete. As with our solution to YSP, we can provide both monotonic and nonmonotonic completions to causal theories.

6.5.1 A monotonic completion of causal theories

As we said earlier, there are two kinds of changes an action can cause. One is direct effects, which are determined by causal rules. The other is side effects, which are determined by the direct effects and domain constraints. Facts that are neither direct effects nor side effects are assumed to be unchanged by the action.

Essentially, making causal theories epistemologically complete amounts to supplying frame axioms that capture facts which are not changed by actions. When the frame axioms are explicitly listed, the completions are called monotonic. Again in the following, we fix a set of fluents $\mathcal{P}$.

Definition 6.4 *Let T be the causal theory about action A with domain constraint $C(s)$, and direct effects $P_1, ..., P_n$ under preconditions $R_1, ..., R_n$, respectively. The* monotonic completion *of T, written $Comp(T)$, is the union of T with the following frame axioms. For any subset M of $N = \{1, ..., n\}$, and any $P \in \mathcal{P}$, if*

$$\not\models \forall s (\bigwedge_{i \in M} holds(P_i, s) \wedge C(s) \supset holds(P, s))$$

and

$$\not\models \forall s (\bigwedge_{i \in M} holds(P_i, s) \wedge C(s) \supset \neg holds(P, s)),$$

then the following is a frame axiom:

$$\forall s.(\bigwedge_{i \in N-M} \neg R_i(s) \supset (holds(P, s) \equiv holds(P, result(A, s)))). \tag{6.13}$$

Example 6.1 The causal theory T of action $Wait$ with domain constraint $C(s)$, and no direct effect is $\forall s.C(s)$. For each $P \in \mathcal{P}$, if

$$\not\models \forall s (C(s) \supset holds(P, s))$$

and

$$\not\models \forall s (C(s) \supset \neg holds(P, s)),$$

then the following sentence is a frame axiom:

$$\forall s.(holds(P, s) \equiv holds(P, result(Wait, s))).$$

Thus the monotonic completion of T, $Comp(T)$, is equivalent to the following set of axioms:

$$\forall s.C(s),$$
$$\forall s.(holds(P, s) \equiv holds(P, result(Wait, s))).$$

where P is any fluent in $\mathcal{P}$. ∎

It is interesting to notice here that our formulation of $Comp(T)$ is related to the forms of axioms proposed by Reiter [1991]. The major difference is that domain constraints are integrated into axioms in [Reiter 1991], while they are separated from causal rules and frame axioms in our treatment.

We now formulate a sufficient condition under which the monotonic completions of causal theories are epistemologically complete.

Let T be a causal theory about action A. For any state SS of S, let

$$\begin{aligned} RES(A, SS) &= \{holds(P, S') \mid P \in \mathcal{P}, SS \cup T \models holds(P, S')\} \\ &\cup \{\neg holds(P, S') \mid P \in \mathcal{P}, SS \cup T \models \neg holds(P, S')\} \\ &\cup \{holds(P, S') \mid holds(P, S) \in SS, SS \cup T \not\models \neg holds(P, S')\} \\ &\cup \{\neg holds(P, S') \mid \neg holds(P, S) \in SS, SS \cup T \not\models holds(P, S')\}, \end{aligned}$$

where $S' = result(A, S)$. We see that $RES(A, SS)$ is a state of S' if $SS \cup T$ is consistent.

Theorem 14 *Let T be the causal theory about action A with the domain constraint C, and direct effects $P_1, ..., P_n$ under preconditions $R_1, ..., R_n$, $n \geq 0$, respectively. $Comp(T)$ is an epistemologically complete monotonic theory about A with respect to $\mathcal{P}$ if the following conditions are satisfied:*

Condition 1. *$C(s), R_1(s), ..., R_n(s)$ do not contain any situation term other than s.*

Condition 2. *For any state SS of S, either $SS \models \varphi(S)$ or $SS \models \neg\varphi(S)$, where $\varphi(S) \in \{C(S), R_1(S), ..., R_n(S)\}$.*

Condition 3. *$\forall s.C(s)$ is consistent.*

Condition 4. *For any SS of S, if $SS \models C(S)$, then $RES(A, SS)$ is consistent, and $RES(A, SS) \models C(result(A, S))$.*

Proof: First we show that for any situation S, and any state SS of S, $Comp(T) \cup SS \models \varphi$ for any $\varphi \in RES(A, SS)$. This can be proved by noting the fact that by Condition 1 and 2, for any P,

$$SS \cup T \vdash Qholds(P, result(A, S))$$

iff

$$F_0 \wedge \bigwedge_{1 \leq i \leq n} F_i \vdash Qholds(P, result(A, S)),$$

where Q is either empty or $\neg$, F_0 is $False$ if $SS \vdash \neg C(S)$ and $C(result(A, S))$ otherwise, and for $1 \leq i \leq n$, F_i is $True$ if $SS \models \neg R_i(S)$, and $holds(P_i, result(A, S))$ if $SS \models R_i(S)$.

Thus $Comp(T)$ is complete if it is consistent. Since $\forall s.C(s)$ is consistent, and $C(s)$ has no situation terms other than s, thus there is a model M of $\forall s.C(s)$ such that the situation domain of M is the set of the closed situation terms. Construct an interpretation M_1 which agrees with M on everything except the interpretation of the predicate $holds$. For any situation constant S, and any P^* in the fluent domain, $M_1 \models holds(P^*, S)$ iff $M \models holds(P^*, S)$. Inductively, for any $result(A, S)$, and any P^* in the fluent domain, if there is no $P \in \mathcal{P}$ such that P is interpreted as P^*, then $M_1 \models holds(P^*, result(A, S))$ iff $M \models holds(P^*, result(A, S))$; if $P \in \mathcal{P}$ is interpreted as P^*, then $M_1 \models holds(P^*, result(A, S))$ iff $holds(P, result(A, S)) \in RES(A, M_1(S))$, where $M_1(S)$ is the state determined by M_1 at S. Formally

$$M_1(S) = \{Qholds(P, S) \mid P \in \mathcal{P}, M_1 \models Qholds(P, S)\},$$

where Q is either empty or $\neg$. The model M_1 is well-defined since by Condition 4, it can be proved inductively that $M_1 \models \forall s.C(s)$, and $RES(A, M_1(S))$ is a state of $result(A, S)$. We show that M_1 is a model of $Comp(T)$.

1. $M_1 \models \forall s.(R_i(s) \supset holds(P_i, result(A, s)))$, $i = 1, ..., n$: Suppose $M_1 \models R_i(S)$, then $holds(P_i, result(A, S)) \in RES(A, M_1(S))$. Thus $M_1 \models holds(P_i, result(A, S))$.

2. For any subset M of N, and any $P \in \mathcal{P}$, M_1 satisfies the frame axiom (6.13) under the conditions for it to be in $Comp(T)$: Let S be any situation. Suppose $M_1 \models \bigwedge_{i \in N-M} \neg R_i(S)$. If $M_1 \models holds(P, S)$, and $M_1 \models \neg holds(P, result(A, S))$ then

$$M_1(S) \cup \{F_0, F_1, ..., F_n\} \vdash \neg holds(P, result(A, S)),$$

where $F_0, ..., F_n$ are defined in terms of $M_1(S)$. Thus

$$C(result(A, S)) \wedge \bigwedge_{i \in M} holds(P_i, result(A, S)) \vdash \neg holds(P, result(A, S)),$$

therefore

$$\vdash \forall s.(C(s) \wedge \bigwedge_{i \in M} holds(P_i, s) \supset \neg holds(P, s)),$$

a contradiction with one of the conditions for (6.13) to be in $Comp(T)$. Thus $M_1 \models holds(P, result(A, S))$. Similarly, if $M_1 \models \neg holds(P, S)$, then

$$M_1 \models \neg holds(P, result(A, S)).$$

Thus we have proved that M_1 is a model of $Comp(T)$. Therefore $Comp(T)$ is a complete theory about A with respect to $\mathcal{P}$. ■

Although Theorem 14 is about a single action, it can easily be extended to multiple actions. This is because the five conditions guarantee that all of the actions must be independent. For example, Condition 1 excludes any action terms from appearing in $C(s)$, $R_1(s)$, ..., and $R_n(s)$.

The class of the causal theories that satisfy the four conditions includes most of the blocks world examples found in the literature. If we ignore the predicates *frame* and *possible* in [Lifschitz 1990a], then our class includes the causal theories in the main theorem of [Lifschitz 1990a].

Example 6.2 The Extended Yale Shooting Problem [Baker 1989]: We add one more fluent *Alive* into YSP. Let $\mathcal{P} = \{Dead, Alive, Loaded\}$. The domain constraint $\forall s C(s)$ is:

$$\forall s(holds(Alive, s) \equiv \neg holds(Dead, s)).$$

The causal rules are (6.3) and (6.4). Let us check the five conditions for the example.

1. Conditions 1, 2, and 3: Trivial.

2. Condition 5: Let SS be a state of S. Suppose $SS \models C(S)$, i.e.

$$holds(Dead, S) \in SS \quad \text{iff} \quad \neg holds(Alive, S) \in SS.$$

We can show that

$$holds(Dead, result(Shoot, S)) \in RES(Shoot, SS)$$

iff

$$\neg holds(Alive, result(Shoot, S)) \in RES(Shoot, SS),$$

that is, $RES(Shoot, SS) \models C(result(Shoot, S))$. The cases for $Wait$ and $Load$ are trivial.

■

6.5.2 A nonmonotonic completion

We now proceed to provide a nonmonotonic completion of causal theories. We shall use a simple version of circumscription.

Again, we fix a set of fluents $\mathcal{P}$. We assume that the sentence $p \in \mathcal{P}$ can be formalized by a first-order formula. Formally, we assume that $Frame(p)$ is a formula with a free variable p such that for any interpretation M, and any P^* in the fluent domain of M, $M \models Frame(P^*)$ iff there is a $P \in \mathcal{P}$ such that P is interpreted as P^*. For example, if $\mathcal{P} = \{P_1, P_2\}$, then $Frame(p)$ can be $p = P_1 \vee p = P_2$.

Let $ab(p)$ be the abbreviation of the following formula

$$Frame(p) \wedge (holds(p, s) \equiv \neg holds(p, result(a, s))).$$

Our circumscriptive policy will be that for any given situation S and any given action A, we minimize $ab(p)(s/S, a/A)$ as a unary formula of p with $holds(p, S)$ fixed but $holds(p, S')$ allowed to vary for every S' that is different from S, where $ab(p)(s/S, a/A)$ is the result of replacing s and a in $ab(p)$ by S and A, respectively. In order to do that, we extend our language $\mathcal{L}$ to include a new predicate $holds'$ that is similar to $holds$. Then for any formula W, we minimize $ab(p)$ in the following formula with $holds'$ fixed and $holds$ allowed to vary:

$$\forall p(holds(p, s) \equiv holds'(p, s)) \wedge W.$$

Also, in order to use circumscription, we need to have some unique names axioms. We suppose there is an axiom U_1 that captures the unique names assumption for fluents in $\mathcal{P}$. We also suppose U_2 is the following axiom:

$$\forall as.earlier(s, result(a, s)) \wedge$$
$$\forall ss_1s_2(earlier(s, s_1) \wedge earlier(s_1, s_2) \supset earlier(s, s_2)) \wedge$$
$$\forall ss_1(earlier(s, s_1) \supset s \neq s_1),$$

where *earlier* is a new binary predicate. The purpose of U_2 is to capture the following unique names axiom:

$$S \neq result(A, S) \neq result(A, result(A, S)) \neq ...$$

Now we can have the following result:

Theorem 15 *Under the assumptions and conditions in Theorem 14, for any formula* φ *in* $\mathcal{L}$ *(our original language without holds′ and earlier),* $Comp(T) \cup \{U_1, U_2\} \models \varphi$ *iff* $\forall s.Circum(W; ab(p)(a/A); holds) \models \varphi$*, where* W *is the following formula*

$$\forall p(holds(p, s) \equiv holds'(p, s)) \wedge U_1 \wedge U_2 \wedge (\bigwedge T).$$

Proof: Suppose M_0 is a model of $Comp(T) \cup \{U_1, U_2\}$. Let M_1 be the extension of M_0 with $holds'$ interpreted identically with $holds$. We show that M_1 is a model of $\forall s.Circum(W; ab(p)(a/A); holds)$. Suppose otherwise; since M_1 is a model of $\forall s.W$, there is an S^* in the situation domain of M_1, and a model M_2 of $W(S^*)$ such that

1. M_1 and M_2 have the same domains, and they interpret everything except *holds* the same.

2. The extension of $ab(p)(s/S^*, a/A)$ in M_2 is a proper subdomain of that in M_1.

We first notice that for any P^* in the fluent domain, $M_1 \models holds(P^*, S^*)$ iff $M_2 \models holds(P^*, S^*)$. Suppose P^* is an element of the fluent domain such that $M_1 \models ab(P^*)(s/S^*, a/A)$ but $M_2 \models \neg ab(P^*)(s/S^*, a/A)$. Thus there is a $P \in \mathcal{P}$ such that it is interpreted as P^*. Suppose $M_1 \models holds(P, S^*)$. Then $M_2 \models holds(P, S^*)$. Thus $M_1 \models \neg holds(P, result(A, S^*))$ and $M_2 \models holds(P, result(A, S^*))$. Let $M \subseteq N$ be such that

$$M_1 \models \bigwedge_{i \in M} R_i(S^*) \wedge \bigwedge_{i \in N-M} \neg R_i(S^*).$$

Since M_1 is a model of $Comp(T)$, thus from the frame axiom (6.13), we have that either

$$\forall s(C(s) \wedge (\bigwedge_{i \in M} holds(P_i, s)) \supset holds(P, s))$$

or

$$\forall s(C(s) \wedge (\bigwedge_{i \in M} holds(P_i, s)) \supset \neg holds(P, s)).$$

But the first contradicts with $M_1 \models \neg holds(P, result(A, S^*))$, and the second case with $M_2 \models holds(P, result(A, S^*))$ (recall that M_2 is a model of $W(S^*)$). Similarly, it is impossible for the existence of a P^* in the fluent domain such that $M_1 \models \neg ab(P^*)(s/S^*, a/A)$ but $M_2 \models ab(P^*)(s/S^*, a/A)$.

Thus we have shown that the existence of such M_2 is impossible, and thus M_1 is a model of $\forall s.Circum(W; ab(p)(a/A); holds)$.

Now suppose that M_0 is a model of $\forall s.Circum(W; ab(p)(a/A); holds)$. We want to show that M_0 is also a model of $Comp(T)$. This is proved by showing that for any S^* in the situation domain of M_0, $M_0(result(A, S^*)) = RES(A, M_0(S^*))$. Suppose there is a S^* such that this is not true. Then construct M_1 such that M_1 is exactly like M_0 but for any $n > 0$, $M_1(result(A^n, S^*)) = RES(A^n, M_1(S^*))$, where

1. $result(A^1, S^*) = result(A, S^*)$, and $RES(A^1, M_1(S^*)) = RES(A, M_1(S^*))$;

2. $result(A^{i+1}, S^*) = result(A, result(A^i, S^*))$, and
   $RES(A^{i+1}, M_1(S^*)) = RES(A, RES(A^i, M_1(S^*)))$.

Since M_0 satisfies $U_1 \wedge U_2$, thus M_1 is well-defined. As in the proof of Theorem 14, we can prove that M_1 is a model of $W(S^*)$, and $M_1 \sqsubset_{ab(p)} M_0$ according to the circumscriptive policy, a contradiction. ■

Thus if we define $\models_{\sqsubset}$ such that $\{W\} \models_{\sqsubset} \varphi$ iff $\forall s.Circum(W; ab(p)(a/A); holds) \models \varphi$, then we have the following corollary

Corollary 15.1 *Under the assumptions in Theorem 15, if W is classically consistent, then $\{W\}$ is an epistemologically complete nonmonotonic theory about A according to $\models_{\sqsubset}$.*

6.6 Future work and concluding remarks

We have argued that a useful way to tackle the frame problem is to consider a monotonic theory with explicit frame axioms first, and then to show that a succinct and provably equivalent representation using, for example, nonmonotonic logics, captures the frame axioms concisely. The idea is not startling. A similar idea is used in verifying negation-as-failure [Clark 1978]. It seems that several researchers have entertained the idea of verifying a nonmonotonic theory of action against a monotonic one (for example, it is listed as future work in [Winslett 1988]), even if no one to date has actually followed this course. Central to our project is the definition of an epistemological completeness condition for deterministic actions, whose intuitive purpose is to determine whether sufficient axioms are included in a theory. Our main technical contribution is in formulating a wide class of epistemologically complete monotonic causal theories, and showing that for each complete causal theory, there is a succinct representation using a version of circumscription.

We notice here that our nonmonotonic completions of the causal theories in circumscription do not address the qualification problem. This can be easily done by using the method in [Lifschitz 1987].

There are many directions for future work. At the present time, the most important one is to extend our work to allow concurrent actions. Just as fluents can partially describe a situation, we may have action descriptions that partially describe the set of actions taken in any situation. As a result we will be able to infer by default not only what is true in situations, but also what actions have been taken; in the current framework that is not even expressible.

Chapter 7

Summary and Future Work

We began by introducing the logic GK of Grounded Knowledge as a uniform semantic basis for fixed-point nonmonotonic logics such as Reiter's default logic and Moore's autoepistemic logic. To our knowledge, GK for the first time unifies the two logics semantically. As with circumscription, GK is based on the notion of logical minimization, and thus provides a bridge between circumscription and fixed-point nonmonotonic logics. As an application of this bridge, we proposed a formalization of logic programs with negation-as-failure in circumscription. Together with the unique names assumption, the formalization provides a concise representation for the facts that are true in every answer set of a logic program.

We then introduced the notion of argument systems as a proof theory for nonmonotonic reasoning. By reformulating some major existing nonmonotonic logics as argument systems, we showed that most nonmonotonic reasoning can be captured with the aid of a generalized negation-as-failure rule. We described an implemented system for default logic based on this idea.

Finally, we applied nonmonotonic logic to the formalization of inheritance and of theories of action, the two areas that have to a large degree motivated research in nonmonotonic reasoning. We showed that various intuitions about nonmonotonic inheritance hierarchies can be conveniently formalized in logic programs with negation-as-failure. For reasoning about action, we first proposed a formal yet intuitive criterion by which to evaluate the adequacy of theories of action. We then formulated a class of monotonic theories that satisfy this criterion by using explicit frame axioms. Finally for a significant subclass of the monotonic theories, we provided provably-correct nonmonotonic counterparts which avoid explicit frame axioms.

According to McCarthy and Hayes [1969], AI problems can be divided into two parts – epistemological part and heuristic part. According to McCarthy, one of the reasons for this division is that AI problems are generally very hard, and dividing them into manageably smaller parts helps.

Without doubt, this divide-and-conquer methodology is very important, and fruitful. But sometimes, working on subproblems for too long may make one lose sight of what the original problem was, especially when the subproblems themselve are very hard. Having worked on nonmonotonic reasoning, a subarea of the AI epistemology, for many years, I wish to list integrated AI systems as part of my future work.

Bibliography

[Bacchus 1987] Bacchus, F., A heterogeneous inheritance system based on probabilities, Technical Report 87–03, Alberta Centre for Machine Intelligence and Robotics, University of Alberta, Canada, 1987.

[Baker 1989] Baker, A. B., A simple solution to the Yale shooting problem, *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, 1989.

[Baker and Ginsberg 1989] Baker, A. B. and M. L. Ginsberg, Temporal projection and explanation, *Proceedings of IJCAI–89*, 1989.

[Bidoit and Froidevaux 1988] Bidoit, N. and C. Froidevaux, Negation by default and nonstratifiable logic programs, Technical Report 437, Universite Paris XI, September 1988.

[Clark 1978] Clark, K., Negation as failure, in *Logics and Databases*, eds. by Gallaire and Minker, Plenum Press, 1978.

[Davis 1991] Davis, E., Physical idealization as plausible inference, *Working Notes of the AAAI Spring Symposium on Logical Formalizations of Commonsense Reasoning*, 1991, Stanford, CA.

[Etherington 1987] Etherington, D.W., More on inheritance hierarchy with exceptions: default theory and inference distance, *Proceedings of AAAI–87*, Seattle.

[Etherington 1987a] Etherington, D.W., Relating default logic and circumscription, *Proceedings of IJCAI–87*, Milan, Italy.

[Etherington 1987b] Etherington, D.W., A semantics for default logic, *Proceedings of IJCAI–87*, Milan, Italy.

[Etherington and Reiter 1987] Etherington, D.W. and R. Reiter, On inheritance hierarchies with exception, *Proceedings of AAAI–83*, Morgan Kaufmann, 1983.

[Geffner and Pearl 1987] Geffner, H. and J. Pearl, Sound defeasible inference, Technical Report, University of California, Los Angeles, Computer Science Department, (UCLACS) CSD-870058, 1987.

[Gelfond 1987] Gelfond, M., On stratified autoepistemic theories, *Proceedings of AAAI–87*.

[Gelfond 1989] Gelfond, M., Autoepistemic logic and formalization of common-sense reasoning, in M. Reinfrank, J. de Kleer, M. Ginsberg, and E. Sandewall, editors, *Non-Monotonic Reasoning: 2nd International Workshop (Lecture Notes in Artificial Intelligence 346)*, Springer-Verlag, 1989.

[Gelfond and Lifschitz 1988] Gelfond, M. and V. Lifschitz, The stable model semantics for logic programming, in R. Kowalski and K. Bowen, editors, *Logic Programming: Proc. of the Fifth International Conference*, 1988.

[Gelfond and Lifschitz 1989] Gelfond, M. and V. Lifschitz, Compiling circumscriptive theories into logic programs, in M. Reinfrank, J. de Kleer, M. Ginsberg, and E. Sandewall, editors, *Non-Monotonic Reasoning: 2nd International Workshop (Lecture Notes in Artificial Intelligence 346)*, Springer-Verlag, 1989.

[Gelfond and Lifschitz 1991] Gelfond, M. and V. Lifschitz, Classical negation in logic programs and disjunctive databases, *New Generation Computing*, 1991. To appear.

[Gelfond and Przymusinska 1990] Gelfond, M. and H. Przymusinska, Formalization of inheritance reasoning in autoepistemic logic, *Fundamenta Informaticae*, 1990.

[Gelfond *et al.* 1991] Gelfond, M., V. Lifschitz, H. Przymusinska, and M. Truszczynski, Disjunctive defaults, *Proceedings of KR-91*.

[Ginsberg 1989] Ginsberg, M. L., A local formalization of inheritance: Preliminary report, *Preprints of the Third International Workshop on Nonmonotonic Reasoning*, 1989.

[Gregoire 1989] Gregoire, E., Skeptical theories of inheritance and nonmonotonic logics, in *Methodologies for Intelligent Systems*, edited by Z. W. Ras, 1989.

[Halpern and Moses 1984] Halpern, J. Y. and Y. O. Moses, Towards a theory of knowledge and ignorance: preliminary report, IBM Technical Report RJ 4448 (48136), 1984.

[Hanks and McDermott 1986] Hanks, S. and D. McDermott, Nonmonotonic logics and temporal projection, *Artificial Intelligence*, 33 (1987).

[Haugh 1987] Haugh, B., Simple causal minimizations for temporal persistence and projection, *Conference Proceedings of AAAI-87.*

[Horty *et al.* 1987] Horty, J. F., R. H. Thomason, and D. S. Touretzky, A skeptical theory of inheritance in nonmonotonic semantic networks, *Conference Proceedings of AAAI–87*.

[Horty and Thomason 1988] Horty, J. F. and R. H. Thomason (1988), Mixing strict and defeasible inheritance, *Conference Proceedings of AAAI–88*.

[Hughes and Cresswell 1972] Hughes, G. E. and M. J. Cresswell, *An Introduction to Modal Logic*, Methuen and Co., London, 1972.

[Junker and Konolige 1990] Junker, U. and K. Konolige, Computing the extensions of autoepistemic and default logics with a truth maintenance system, *Proceedings of AAAI–90.*

[Kautz 1986] Kautz, H., The logic of persistence, *Proceedings of AAAI–1986.*

[Kautz and Selman 1989] Kautz, H. and B. Selman, Hard problems for simple default theories, *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, Toronto, 1989.

[Konolige 1988] Konolige, K., On the relation between default logic and autoepistemic logic, *Artificial Intelligence*, 35 (1988).

[Levesque 1989] Levesque, H., All I know: a study in autoepistemic logic, Technical Report KRR-TR-89-3, University of Toronto, 1989.

[Lifschitz 1985] Lifschitz, V., Computing circumscription, *Proceedings of IJCAI–85.*

[Lifschitz 1986] Lifschitz, V., Pointwise circumscription, *Proceedings of AAAI – 1986.*

[Lifschitz 1987] Lifschitz, V., Formal theories of action, in *Proceedings of IJCAI–87.*

[Lifschitz 1987a] Lifschitz, V., On the declarative semantics of logic programs with negation, in *Readings in Nonmonotonic Reasoning*, Morgan Kaufmann, 1987, ed. by M. Ginsberg.

[Lifschitz 1990] Lifschitz, V., Frames in the Space of Situations, *Artificial Intelligence*, 46 (1990) 365-376.

[Lifschitz 1990a] Lifschitz, V., On open defaults, *Proceedings of the Symposium on Computational Logic*, Brussels, 1990.

[Lifschitz 1991] Lifschitz, V., Nonmonotonic databases and epistemic queries, in IJCAI – 91.

[Lifschitz 1991] Lifschitz, V., Toward a Meta-theory of Action, *Proceedings of Second International Conference on Principles of Knowledge Representation and Reasoning*, 1991.

[Lifschitz and Rabinov 1989] Lifschitz, V. and A. Rabinov, Miracles in formal theories of action, *Artificial Intelligence*, 38 (1989).

[Lin 1988] Lin, F., Circumscription in a modal logic, *Proceedings of the Second Conference on Theoretical Aspects of Reasoning about Knowledge*, Asilomar, California, 1988.

[Lin 1990] Lin, F., Formalizing various intuition about inheritance in logic programs," in *Preprints of the Third International Workshop on Nonmonotonic Reasoning*, South Lake Tahoe, CA, May 1990.

[Lin and Shoham 1989] Lin, F. and Y. Shoham, Argument systems: a uniform basis for nonmonotonic reasoning, *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, Toronto, 1989.

[Lin and Shoham 1990] Lin, F. and Y. Shoham, Epistemic semantics for fixed-points nonmonotonic logics, *Proceedings of the Third Conference on Theoretical Aspects of Reasoning about Knowledge*, Asilomar, California, 1990.

[Lin and Shoham 1991] Lin, F. and Y. Shoham, Provably correct theories of action, to appear in *Conference Proceedings of AAAI–91.*

[Loui 1987] Loui, R., Response to Hanks and McDermott: temporal evolution of beliefs and beliefs about temporal evolution, *Cognitive Science* 11 (1987).

[Loui 1987a] Loui, R., Defeat among arguments: a system of defeasible inference, *Computational Intelligence 3*, 1987.

[McCarthy 1980] McCarthy, J., Circumscription — A form of non-monotonic reasoning, *Artificial Intelligence* 13(1980).

[McCarthy 1986] McCarthy, J., Applications of circumscription to formalizing commonsense knowledge, *Artificial Intelligence* 28 (1986).

[McCarthy and Hayes 1969] McCarthy, J. and P. Hayes, Some philosophical problems from the standpoint of artificial intelligence, *Machine Intelligence* 4, Meltzer, B and Michie, D. (eds), Edinburgh University Press.

[McCune 1990] McCune, W. W., OTTER 2.0 user's guide, Technical Report ANL-90/9, Mathematics and Computer Science Division, Argonne National Laboratory, 1990.

[McDermott and Doyle 1980] McDermott, D. and J. Doyle, Nonmonotonic logic I, *Artificial Intelligence* 13(1980).

[Marek and Truszczynski 1989] Marek, W. and M. Truszczynski, Relating autoepistemic and default logics, *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, Toronto, 1989.

[Marek and Truszczynski 1990] Marek, W. and M. Truszczynski, Modal logic for default reasoning, *Annals of Mathematics and Artificial Intelligence,* 1:275-302, 1990.

[Marek *et al.* 1991] Marek, W., G. F. Shvarts, and M. Truszczynski, Classification of expansions in modal nonmonotonic logics, *Proceedings of the Second Conference on the Principles of Knowledge Representation and Reasoning*, 1991.

[Moore 1983] Moore, R., Semantical considerations on nonmonotonic logic, *Conference Proceedings of IJCAI–83*.

[Morgenstern and Stein 1988] Morgenstern, L. and L. A. Stein, Why things go wrong: A formal theory of causal reasoning, *Proceedings of AAAI–1988*.

[Morris 1988] Morris, P. H., The anomalous extension problem in default reasoning, *Artificial Intelligence* 35 (1988).

[Nute 1987] Nute, D., Defeasible reasoning, *Proceedings of 20th HICSS*, 1987.

[Pollock 1987] Pollock, J. L., Defeasible reasoning, *Cognitive Science*, 11 (1987).

[Przymusinski 1989] Przymusinski, T., On the declarative and procedural semantics of logic programs, *Journal of Automated Reasoning*, 5:167-205, 1989.

[Przymusinski 1990] Przymusinski, T., Extended stable semantics for normal and disjunctive programs, in D. Warren and P. Szeredi (eds.), *Logic Programming: Proc. of the Seventh International Conference*, 1990.

[Reiter 1978] Reiter, R., On closed world data bases, in H. Gallaire and J. Minker (eds.), *Logics and Data Bases*, Plenum Press, New York, 1978.

[Reiter 1980] Reiter, R., A logic for default reasoning, *Artificial Intelligence* 13(1980).

[Reiter 1991] Reiter, R., The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression, *Working Notes of the AAAI Spring Symposium on Logical Formalizations of Commonsense Reasoning*, 1991, Stanford, CA.

[Sandewall 1986] Sandewall, E., Non-monotonic inference rules for multiple inheritance with exceptions. *Proceedings of IEEE*, vol.74, 1986, pp. 1345–1353.

[Shoham 1986] Shoham, Y., Chronological ignorance: time, nonmonotonicity and necessity, *Proceedings of AAAI–86.*

[Shoham 1987] Shoham, Y., A semantical approach to nonmonotonic logics, *Proceedings of IJCAI–1989.*

[Shoham 1988] Shoham, Y., *Reasoning About Change: Time and Causation from the Standpoint of Artificial Intelligence*, MIT Press, 1988.

[Shoham and McDermott 1988] Shoham, Y. and D. McDermott, Problems in formal temporal reasoning, *Artificial Intelligence*, 36 (1988).

[Siegel 1990] Siegel, P., A modal language for nonmonotonic logic, *Proceedings of Workshop DRUMS/CEE,* Marseille 24-28 February, 1990.

[Thomason and Horty 1989] Thomason, D. and J. Horty, Logics for inheritance theory, in M. Reinfrank, J. de Kleer, M. Ginsberg, and E. Sandewall, editors, *Non-Monotonic Reasoning: 2nd International Workshop (Lecture Notes in Artificial Intelligence 346)*, Springer-Verlag, 1989.

[Touretzky 1984] Touretzky, D., Implicit ordering of defaults in inheritance systems, *Conference Proceedings of AAAI–84*.

[Touretzky 1986] Touretzky, D., *The Mathematics of Inheritance Systems*, Morgan Kaufmann, 1986.

[Touretzky *et al.* 1987] Touretzky, D., J. Horty, and R. Thomason, A clash of intuition: the current state of nonmonotonic multiple inheritance systems, *Proceedings of AAAI-87.*

[Truszczynski 1991] Truszczynski, M., Modal interpretations of default logic, to appear in *Proceedings of IJCAI - 91.*

[Van Gelder 1986] Van Gelder, A., Negation as failure using tight derivations for general logic programs, *Proceedings of IEEE Conference on Logic Programming*, 1986.

[Winslett 1988] Winslett, M., Reasoning about actions using a possible models approach, *Conference Proceedings of AAAI–88.*