Nonmonotonic logics—recent advances

Mirek Truszczynski

ESSLLI 2008
20th European Summer School in Logic, Language and Information
4–15 August 2008
Freie und Hansestadt Hamburg, Germany

Programme Committee. Enrico Franconi (Bolzano, Italy), Petra Hendriks (Groningen, The Netherlands), Michael Kaminski (Haifa, Israel), Benedikt Löwe (Amsterdam, The Netherlands & Hamburg, Germany) Massimo Poesio (Colchester, United Kingdom), Philippe Schlenker (Los Angeles CA, United States of America), Khalil Sima'an (Amsterdam, The Netherlands), Rineke Verbrugge (**Chair**, Groningen, The Netherlands).

Organizing Committee. Stefan Bold (Bonn, Germany), Hannah König (Hamburg, Germany), Benedikt Löwe (**chair**, Amsterdam, The Netherlands & Hamburg, Germany), Sanchit Saraf (Kanpur, India), Sara Uckelman (Amsterdam, The Netherlands), Hans van Ditmarsch (**chair**, Otago, New Zealand & Toulouse, France), Peter van Ormondt (Amsterdam, The Netherlands).

`http://www.illc.uva.nl/ESSLLI2008/`
`esslli2008@science.uva.nl`

INSTITUTE FOR LOGIC, LANGUAGE AND COMPUTATION

ESSLLI 2008 is organized by the Universität Hamburg under the auspices of the *Association for Logic, Language and Information* (FoLLI). The *Institute for Logic, Language and Computation* (ILLC) of the *Universiteit van Amsterdam* is providing important infrastructural support. Within the Universität Hamburg, ESSLLI 2008 is sponsored by the Departments *Informatik*, *Mathematik*, *Philosophie*, and *Sprache, Literatur, Medien I*, the *Fakultät für Mathematik, Informatik und Naturwissenschaften*, the *Zentrum für Sprachwissenschaft*, and the *Regionales Rechenzentrum*. ESSLLI 2008 is an event of the *Jahr der Mathematik 2008*. Further sponsors include the *Deutsche Forschungsgemeinschaft* (DFG), the Marie Curie Research Training Site GLoRiClass, the European Chapter of the Association for Computational Linguistics, the *Hamburgische Wissenschaftliche Stiftung*, the Kurt Gödel Society, Sun Microsystems, the Association for Symbolic Logic (ASL), and the European Association for Theoretical Computer Science (EATCS). The official airline of ESSLLI 2008 is Lufthansa; the book prize of the student session is sponsored by *Springer Verlag*.

Mirek Truszczynski

Nonmonotonic logics—recent advances

Course Material. 20th European Summer School in Logic, Language and Information (ESSLLI 2008), Freie und Hansestadt Hamburg, Germany, 4–15 August 2008

Nonmonotonic logics—recent advances

Lecture notes for ESSLLI-08; slides available at
www.cs.uky.edu/ai/esslli08Slides.pdf

Mirosław Truszczyński
Department of Computer Science
University of Kentucky
Lexington, KY 40506-0046
{mirek}@cs.engr.uky.edu

1 Introduction

In the late 1970s, the need for effective knowledge representation methods brought attention to rules of inference that admit *exceptions* and are used under the assumption of normality or, to put it differently, when things are "as expected."

For instance, a knowledge base concerning a university should support an inference that, given no information that might indicate otherwise, if Dr. Jones is a professor at that university, then Dr. Jones teaches. Such conclusion might be sanctioned by an inference rule stating that *normally* university professors teach. In commonsense reasoning rules with exceptions are ubiquitous.

The problem is that such rules do not lend themselves in any direct way to formalizations in terms of first-order logic, unless *all* exceptions are known and explicitly represented — an unrealistic expectation in practice. The reason is that standard logical inference is *monotone*, and proofs cannot be *defeated* when additional facts become available. However, in commonsense reasoning most arguments *are* defeasible, as they are conditioned on implicit assumptions (most often, precisely assumptions of normality), which may turn out incorrect once we learn more about the specific situation about which we reason.

Such reasoning, where additional information may invalidate conclusions, is called *nonmonotonic*. As we have just noted, it is common. It has been a focus of extensive studies by the knowledge representation community and resulted in a rich field of *nonmonotonic logics*.

First nonmonotonic logics were introduced in the late 70s and were based on quite simple ideas. Reiter [Rei78] introduced *Closed World Assumption* (CWA, for short), an inference rule that allows us to derive an atom a from a theory T, if T does not entail the negation of a. CWA is defeasible (if T entails $\neg a$ under CWA, $T \cup \{a\}$ clearly does not), and formalizes the basic database query-answering principle. McCarthy [McC77] introduced an early variant of *circumscription*, called *minimal entailment*, in which entailment is based on minimal models only. As some nonminimal models (which are excluded) may turn out to be relevant once we learn more about the world, the minimal entailment

is defeasible.

These early proposals drew much attention and in 1980 *Artificial Intelligence Journal* published a celebrated volume dedicated to nonmonotonic reasoning. That volume contained three fundamental papers that introduced default logic (Reiter [Rei80]), circumscription (McCarthy [McC80]) and modal nonmonotonic logic (McDermott and Doyle [MD80]). This last logic turned out to have flaws. To address the problems pointed out by the community, McDermott [McD82] introduced an entire family of modal nonmonotonic logics, each based on a standard normal modal logic. In another effort to design a modal logic for nonmonotonic reasoning, Moore [Moo84, Moo85] introduced autoepistemic logic (see also [Lev90]).

In about the same time, the logic programming community was struggling with the problem of establishing a declarative semantics for programs with *negation-as-failure* (we refer to [Kow74, Kow79, Llo84, Apt90, Doe94] for details on mainstream logic programing research and additional references). The problem was that programs with negation-as-failure do not behave as theories in first-order logic. In fact, they behave "nonmonotonically." Indeed, in the absence of any information about an atom p, we infer *not* p (we will write *not* to denote the negation-as-failure operator, to distinguish it from the classical negation operator $\neg$). However, as soon as we include a unit rule p in the program, *not* p no longer holds.

Thus, apparently unaware of the knowledge representation community efforts, the logic programming community was actively pursuing similar research objectives. In a major milestone, in 1978 Clark [Cla78] proposed the *completion semantics* based on a simple rewriting of a logic program as a first-order theory, which views logic program rules as *definitions*. Ultimately, the research on the semantics of negation-as-failure resulted in the *stable* semantics [GL88] and the *well-founded semantics* [VRS88, VRS91]. These semantics are now commonly accepted as providing a correct formalization of the intuitive meaning of logic programs. Interestingly, around the time the stable semantics was introduced, the connections between logic programming and knowledge representation efforts were finally discovered. First, the stable semantics itself was strongly motivated by a certain representation of a logic program as a modal theory and the interpretation of the latter given by the semantics of autoepistemic logic [Gel87, Gel89]. Second, it turned out that an even more direct connection to knowledge representation exists when [BF87, MT89b] proved that logic programming with the stable semantics is really nothing else but a fragment of Reiter's default logic.

Since the time of the confluence of the efforts of the two communities, the area of nonmonotonic logics has grown and matured significantly. Among the most important developments are: the emergence of the effective computational support for nonmonotonic reasoning [CMMT99, SNS02, LPF$^+$06, GKNS07, GLN$^+$07] and of *answer-set programming* [MT99, Nie99], the basic paradigm of computing with nonmonotonic logics; the discovery of deep connections between stable models and logics such as the logic here-and-there [Pea97, FL05] and modal logic S4F [Tru07]; the concept of strong and uniform equivalence of programs, which are fundamental for modular logic programming, and results on extensions, generalizations and characterizations of these notions of equivalence [LPV01, EFW07, Wol07]; the discovery of important connections to propositional satisfiability through the notion of a loop formula [LZ02, FLL06]; and the establishment of general algebraic foundations to nonmonotonic reasoning, which offer a unified view of main nonmonotonic formalisms [DMT00a, DMT03].

Our goal in the tutorial is to introduce the basic formalisms (we will focus on default logic and logic programming and only briefly mention autoepistemic logic) and to review some of these major developments we mentioned above. These notes contains most of the necessary definitions, key results,

and an extensive list of references.

2 Operators and their basic properties

We will often consider mappings of a special type, called *operators*. They are of interest as many properties of default logic and logic programming can be formulated in terms of operators.

Let H be a set. An *operator* is simply a function defined on $\mathcal{P}(H)$ and with values in $\mathcal{P}(H)$. An operator T is *monotone* if it preserves inclusion. That is, for all subsets X_1, X_2 of H

$$(X_1 \subseteq X_2) \Rightarrow (T(X_1) \subseteq T(X_2)).$$

An operator T is *antimonotone* if it reverses inclusion. That is, for all subsets X_1, X_2 of H

$$(X_1 \subseteq X_2) \Rightarrow (T(X_2) \subseteq T(X_1)).$$

Given an operator T, its *iterations* are defined inductively:

$T^0 = \emptyset$
$T^{n+1} = T(T^n)$
$T^\omega = \bigcup_{n\in\omega} T^n$

This definition can be extended to arbitrary ordinals but it is not necessary for our purposes.

We note that when an operator T is monotone,

$$\emptyset \subseteq T(\emptyset) \subseteq T^2(\emptyset) \subseteq \ldots \subseteq T^\omega(\emptyset).$$

Furthermore, if T is monotone, then for all n, T^n is monotone, too. For antimonotone operators the situation is different: for even n, T^n is monotone. For odd n, T^n is antimonotone.

Given an operator T, a subset $X \subseteq H$ is called *prefixpoint* of T if $T(X) \subseteq X$. Similarly, X is a *fixpoint* of T if $T(X) = X$. The following result is due to Tarski and Knaster [Tar55].

Theorem 2.1 *Every monotone operator T possesses a least prefixpoint and a least fixpoint, and the two coincide.*

Theorem 2.1 does not extend to operators that are not monotone. Yet our remark about the powers of antimonotone operators allows us to handle that case to some extent.

Theorem 2.2 *Let T be an antimonotone operator. Then T^2 possesses a least fixpoint F. Moreover, for every fixpoint X of T: $F \subseteq X \subseteq T(F)$*

There is one more interesting property of antimonotone operators. Let T be an antimonotone operator, and M_1 and M_2 be its fixpoints. If $M_1 \subseteq M_2$, then $M_2 = T(M_2) \subseteq T(M_1) = M_1$. Hence, we have the following result.

Theorem 2.3 *Fixpoints of an antimonotone operator form an antichain.*

3 Introduction to Default Logic

Default logic is a knowledge representation mechanism allowing for reasoning in the presence of incomplete information. It handles the logical aspects of modalities such as "normally", "usually", etc.

Syntactically, default logic extends the first order logic (however, in this tutorial we will focus on the propositional case) by introducing new entities called *default rules* or, simply, *defaults*. A default rule is a construct of the form

$$r = \frac{\varphi : \psi_1, \ldots, \psi_m}{\vartheta}$$

where $\varphi, \psi_1, \ldots, \psi_k, \vartheta$ are propositional formulas (given our focus on propositional case). The formula φ is called the *premise* or *prerequisite* of r and is denoted by $p(r)$. The set $\{\psi_1, \ldots, \psi_k\}$ is called the set of *justification* of r and is denoted by $j(r)$. The formula ϑ is called the *conclusion* or *consequent* of r and is denoted $c(r)$.

Justifications are used in default logic to explicitly represent *exceptions*, conditions blocking applicability of defaults. That is, application of a default is qualified by the *absence* of information that would imply inconsistency of one of the justifications of the rule. Put in yet another way, a default is applicable if its premise has already been established and all its justifications are consistent, that is, their negations are not provable. It is precisely that presence of justifications that allows us to model modalities such as "normally" and "usually" within default logic.

In our format, a default rule has just one premise. This is an immaterial restriction since we assume the usual rules of logic anyway.

Default logic deals with *default theories*, that is, pairs (D, W), where D is a collection of defaults and W is a collection of formulas.

Defaults can be viewed as generalized inference rules, with standard inference rules of the form

$$\frac{\varphi}{\vartheta}$$

being special defaults (defaults with no justifications).

Given a default d, by $[d]$ we denote the standard inference rule obtained from d by "stripping" it of its justifications. We extend this notation to sets of defaults in a standard way.

A default $\varphi : \psi_1, \ldots, \psi_m/\vartheta$ is *S-enabled* if $S \not\models \neg\psi_i$, for $i = 1, \ldots, m$. Enabled defaults are those defaults for which justification premises hold, if we assume that S is a theory representing our belief set. For a set D of defaults, we write D_S for the set of S-enabled defaults in D.

A default $\varphi : \psi_1, \ldots, \psi_m/\vartheta$ is *S-applicable* if it is S-enabled and $S \models \varphi$. S-applicable defaults are also called S-generating. For a set D of defaults, we write $D(S)$ for the set of S-applicable defaults in D.

The basic idea behind the semantics of default logic is to associate with a default theory $\Delta = (D, W)$ a collection of theories representing possible belief sets. There are two steps that determine this collection: first we need to *guess* a putative belief set S (which amounts to making assumptions on consistency of justifications) and then we have to justify the selection. There are at least two different ways that could be used in the second step. First, we can justify the choice of S by showing that S is precisely what can be derived from W and $D(S)$ in propositional logic (or, that the negation of no justification assumed to be consistent can be derived from W and $D(S)$). Second, we can justify S, by showing it is precisely what we can derive from S in propositional calculus extended by standard rules

derived from all S-enabled defaults, that is, by the rules in $[D_S]$ (or, that no justification assumed to be consistent can be derived in such a way).

The first approach yields the notion of an *expansion* of a default theory (also referred to as a weak extension). It was introduced in [MT89a] (see also [MT93a]). Thus, S is an expansion of a default theory (D, W) if

$$S = Cn(W \cup D(S)).$$

The second approach yields the notion of an *extension* of a default theory [Rei80], the fundamental notion of default logic. Thus, S is an extension of a default theory (D, W) if

$$S = Cn^{[D_S]}(W),$$

where Cn^B stands for the consequence operator of the formal system extending the proof system of propositional logic with inference rules in B.

Example 3.1 *Let $W = \emptyset$ and $D = \{\frac{p:\, q}{p}\}$. We will consider two contexts: $S_1 = Cn(\emptyset)$ and $S_2 = Cn(p)$. Clearly, $D(S_1) = \emptyset$ and $S_1 = Cn(W \cup c(D(S_1)))$. Similarly, $D(S_2) = D$, $c(D(S_2)) = \{p\}$, and $W = Cn(W \cup c(D(S_2)))$. Thus, S_1 and S_2 are expansions. We note that S_2 is self-justified. Indeed, the presence of p in S requires using the consequent of $\frac{p:\, q}{p}$, which we can only do if we already believe in p.*

In the case of extensions, the situation is different.

Example 3.2 *For the same default theory as above, we have $D_{S_1} = \emptyset$, $D_{S_2} = \{\frac{p:\, q}{p}\}$ and $[D_{S_2}] = \{\frac{p}{p}\}$. Clearly, $S_1 = Cn^{[D(S_1)]}(W)$ and so, S_1 is an extension. However, S_2 is not. The rule $\frac{p}{p}$ will never be applied when we attempt to justify p (it requires that we already have p derived from W by means of propositional inference extended with rules in $[D_{S_2}]$, which is impossible). Thus, $S_2 \neq Cn^{[D(S_2)]}(W)$.*

These two examples illustrate a key difference between expansions and extensions. Extensions are based on a stronger notion of a justification that disallows circular arguments.

Here is one more larger example.

Example 3.3 *Let $W = \{p\}$ and*

$$D = \left\{ \frac{p : \neg q}{r},\ \ \frac{p : \neg r, \neg s}{q},\ \ \frac{r : \neg s}{s} \right\}.$$

There is only one extension, $S = Cn(p, q)$. Indeed, to justify it, we can use p (which belongs to W) and inference rules $\frac{p}{q}$ and $\frac{r}{s}$, which belong to $[D_S]$. Clearly, what can be derived are precisely the propositional consequences of $\{p, q\}$, that is, $Cn(p.q)$.

In the same time, $S = Cn(p, r)$ is not an extension. To justify S we can use p and the rules $\frac{p}{r}$ and $\frac{r}{s}$, which allows us to justify s, even though we do not assume it!

Our definition of extensions is different from the original one provided by Reiter. For the sake of completeness, we will now present this original definition.

Let (D, W) be a default theory. We observe that for every set S, there is a least set U such that:

1. $W \subseteq U$
2. $Cn(U) = U$
3. Whenever $\frac{\varphi:\psi_1,\ldots,\psi_m}{\vartheta}$ is a default rule in D, $\varphi \in U$ and $\neg\psi_1, \ldots, \neg\psi_m \notin Cn(S)$ then $\vartheta \in U$

We denote this set by $\Gamma_{(D,W)}(S)$. We say that S is an extension of (D, W) if

$$S = \Gamma_{(D,W)}(S)$$

3.1 Basic properties of default logic

Having introduced the notion of extension, we will now discuss its elementary properties. First, we note that our definition of extension and the Reiter's one coincide.

Theorem 3.1 *Let (D, W) be a default theory. Let S be any theory. Then $Cn^{[D_S]}(W) = \Gamma_{(D,W)}(S)$. Consequently, S is an extension of (D, W) if and only if $S = \Gamma_{(D,W)}(S)$.*

The operator Γ (we drop the subscript (D, W) from the notation, when no ambiguity arises) is antimonotone. Indeed, the larger S is, the fewer defaults are applicable. Consequently, the operator Γ^2 is monotone and has the least fixpoint. This fixpoint can be used to define a version of *well-founded semantics* for default logic. We discuss this matter later in the tutorial. We also use this approach to define well-founded semantics for logic programs in Section 5.4.

Since the operator Γ is antimonotone, its fixpoints cannot be included one in the other (it is a general property of antimonotone operators; see Theorem 2.3). Consequently we have the following result.

Proposition 3.2 *Extensions of a default theory (D, W) form an antichain. That is, if T_1, T_2 are extensions of (D, W), and $T_1 \subseteq T_2$ then $T_1 = T_2$.*

Next, we note that extensions of a default theory (D, W) are expansions of (D, W).

Proposition 3.3 *If T is an extension of (D, W) then T is an expansion of (D, W) and so, satisfies $T = Cn(W \cup c(D(T)))$*

In particular, it follows that every extension of a default theory (D, W) is of the form $Cn(W \cup c(D'))$, for some set of defaults $D' \subseteq D$. This property is useful for the design of algorithms to compute extensions as it constrains the space of candidate theories. In particular, in the "larger" example in the previous section, there are 8 candidate theories for an extension. They are of the form $Cn(\{p\} \cup U)$, where $U \subseteq \{r, q, s\}$. Checking each of them in the way presented there, one can verify that $Cn(\{p, q\})$ is indeed the only extension of that default theory.

We will now establish yet another characterization of extensions, this time in terms of sets of valuations (possible-world structures) rather than provability operators. The characterization in the propositional case is not particularly deep and can be obtained from proof-theoretic characterizations of extensions by a simple application of the completeness theorem. We describe it here because it leads to the interesting extension of default logic to the predicate case discovered by Lifschitz [Lif90]. It is also relevant to an algebraic treatment of default logic, we discuss later.

Let v be a valuation. Define

$$Th(v) = \{\varphi\colon v(\varphi) = \mathbf{t}\}$$

and, for a set V of valuations,

$$Th(V) = \{\varphi\colon \ v(\varphi) = \mathbf{t}, \ \text{ for every } \ v \in V\}.$$

Clearly,

$$Th(V) = \bigcap\{Th(v)\colon v \in V\}.$$

Finally, for a theory $S \subseteq \mathcal{L}$, define

$$\mathit{Mod}(S) = \{v\colon v(\varphi) = \mathbf{t} \ \text{ for every } \ \varphi \in S\}.$$

It follows directly from the definitions of the operators Th and *Mod*, and from the definition of the operator Cn that for every $W \subseteq \mathcal{L}$,

$$Th(\mathit{Mod}(W)) = Cn(W).$$

Hence, if W is closed under propositional provability,

$$Th(\mathit{Mod}(W)) = W.$$

In order to characterize extensions of default theories in terms of valuations, for a default theory (D, W) we introduce an *operator* $\Sigma_{D,W}$, which assigns sets of valuations to sets of valuations. Our definition relies on the following relationship between sets of valuations and theories.

Theorem 3.4 *Let (D, W) be a default theory. For every set of valuations V the set $\mathit{Mod}(\Gamma_{D,W}(Th(V)))$ is a largest set V' of valuations satisfying the following conditions:*

1. $V' \subseteq \mathit{Mod}(W)$.
2. *For every $d \in D$, if $p(d) \in Th(V')$ and, for every $\beta \in j(d)$, $\neg\beta \notin Th(V)$, then $c(d) \in Th(V')$.*

Let (D, W) be a default theory and let V be a set of valuations. We define $\Sigma_{D,W}(V)$ to be the largest set V' of valuations satisfying the conditions (1) and (2) of Theorem 3.4. That is,

$$\Sigma_{D,W}(V) = \mathit{Mod}(\Gamma_{D,W}(Th(V))).$$

We have the following characterization of extensions in terms of fixpoints of the operator $\Sigma_{D,W}$.

Theorem 3.5 *Let (D, W) be a default theory. Then a theory S is an extension for (D, W) if and only if $S = Th(V)$, for some set of valuations V such that $V = \Sigma_{D,W}(V)$.*

3.2 Normal default logic and related modes of reasoning

In this section we discuss a fragment of default logic with a desirable property that every default theory has an extension. This is so-called *normal default logic*. A *normal* default rule is a rule of the form

$$r = \frac{\varphi : \psi}{\psi}$$

A *normal default theory* is a default theory (D, W) such that D consists of normal defaults only. Normal default theories have several useful properties. We will list them now. First, a normal default theory always possesses an extension.

Theorem 3.6 *A normal default theory (D, W) always possesses an extension. If, in addition, W is consistent, then all extensions of (D, W) are consistent.*

In Section 3.1 we proved that extensions of a default theory form an antichain. Normal default theories enjoy a stronger property.

Theorem 3.7 *If (D, W) is a normal default theory and T_1, T_2 are distinct extensions of (D, W) then $T_1 \cup T_2$ is inconsistent.*

Finally, normal default theories have a coherence property that, in effect, tells us that the forward chaining construction for normal default theories never leads astray. This is called *semimonotonicity*.

Theorem 3.8 *Let D_1, D_2 be collections of normal defaults. Then whenever T_1 is an extension of (D_1, W) then there exists T such that T is an extension of $(D_1 \cup D_2, W)$ and $T_1 \subseteq T$.*

Normal default theories may, of course, possess many extensions.

Example 3.4 Let $W = \{\neg p \vee \neg q\}$. Let $D = \{\frac{:p}{p}, \frac{:q}{q}\}$ Then (D, W) possesses two extensions: $Cn(\{p, \neg q\})$ and $Cn(\{q, \neg p\})$.

If, however, $W \cup \{c(r) : r \in D\}$ is consistent, then (D, W) possesses a unique extension: $Cn(W \cup \{c(r) : r \in D\})$.

We will now discuss a mode of reasoning closely related to normal default logic. This is so-called *Closed World Reasoning*, which we mentioned in the introduction. Given a propositional theory W consider

$$\textit{CWA}(W) = Cn(W \cup \{\neg p : \ p \text{ is an atom and } W \nvdash p\})$$

We say that W is *CWA-consistent* if *CWA*(W) is consistent.

The motivation for *CWA* comes from database considerations. Specifically, whenever we use database and on the absence of information about some atomic fact in database we claim that that fact is false, we perform closed world assumption. Notice that that *CWA*(W) is *always* a complete theory. It may be inconsistent, though.

Example 3.5 Let $W = \{p \vee q\}$. Then *CWA*$(W) = Cn(\{p \vee q, \neg p, \neg q\}$. Thus W is *CWA*-inconsistent.

Proposition 3.9 *If W is a consistent Horn theory then W is CWA-consistent. In fact, CWA(W) is precisely the theory of the least model of W.*

Closed World Assumption is related to normal default logic. Define

$$D_{CWA} = \left\{ \frac{: \neg p}{\neg p} : p \in At \right\}$$

We then have the following result.

Theorem 3.10 *Let W be a set of formulas of $\mathcal{L}$. Then W is CWA-consistent if and only if*

1. *W is consistent, and*
2. *(D_{CWA}, W) possesses a unique extension.*

Extensions of (D_{CWA}, W) are always complete. A complete consistent theory T in the propositional language can be identified with a valuation. Indeed, a valuation v_T defined by

$$v_T(p) = \begin{cases} 1 & \text{if } p \in T \\ 0 & \text{otherwise} \end{cases}$$

is a unique model of T. This in turn can be identified with the set of atoms p which belong to T. The following result ties (D_{CWA}, W) with minimal models (and, hence, with propositional *circumscription*). Define

$$T_M = Cn(\{p : p \in M\} \cup \{\neg p : p \notin M\}).$$

Proposition 3.11 *A set of atoms M is a minimal model of W if and only if T_M is an extension of (D_{CWA}, W).*

We note that normal defaults used to represent CWA are *supernormal*, that is, they do not have prerequisites. Such defaults are closely related to studies of nonmonotonic inference relations [Poo88, Leh89, KLM90, LM92, Pea90].

Less secure than normal default logic is default logic where all the defaults are *seminormal*. Semi-normal defaults are defaults of the form:

$$r = \frac{\varphi : \psi \wedge \vartheta}{\psi}$$

In a sense, seminormal defaults are more cautious than normal ones. They derive their conclusions (ψ) out of the fact that a stronger statement is possible ($\psi \wedge \vartheta$). Perhaps surprisingly, semi-normal default theories do not have all the properties of the normal ones. For instance, we will now show an example of a seminormal default theory which has no extensions.

Example 3.6 Let $W = \emptyset$, and

$$D = \left\{ \frac{: (p \wedge \neg q)}{p}, \frac{: (q \wedge \neg r)}{q}, \frac{: (r \wedge \neg p)}{r} \right\}$$

Then (D, W) has no extensions. Indeed, by our comments above, there are 8 possible theories to consider and none satisfies the equality defining extensions.

3.3 Complexity of reasoning with default logic

The results of this section were proved in [Got92, Sti92]. When nonmonotonic logics were first introduced, one of the expectations was that reasoning with nonmonotonic logics will be more efficient. Unfortunately, these complexity results imply that reasoning with nonmonotonic logics is, in fact, more computationally complex than reasoning with propositional logic (assuming that polynomial hierarchy does not collapse on one of its lower levels). However, we point out to [CDS94, GKPS95, ST96] for a somewhat different perspective.

We will introduce now basic reasoning tasks associated with nonmonotonic formalisms. These are:

EXISTENCE Given a finite default theory (D, W), decide if (D, W) has an extension;

IN-SOME Given a finite default theory (D, W) and a formula φ, decide if φ is in some extension for (D, W) (*credulous reasoner model*);

NOT-IN-ALL Given a finite default theory (D, W) and a formula φ, decide if there is an extension for (D, W) not containing φ;

IN-ALL Given a finite default theory (D, W) and a formula φ, decide if φ is in all extensions of (D, W) (*skeptical reasoner model*).

We have the following result.

Theorem 3.12 *The problems* EXISTENCE*,* IN-SOME *and* NOT-IN-ALL *are* Σ_2^P*-com-plete. The problem* IN-ALL *is* Π_2^P*-complete.*

The complexity remains the same even under substantial syntactic restrictions. In particular, the complexity remains the same if we restrict our attention to semi-normal default theories. For the normal default theories, the situations is similar. While normal default theories always have at least one extension (and, hence, EXISTENCE problem is trivially in P), the complexity of all other problems remains the same as specified in Theorem 3.12. The problem to decide whether a normal theory has a *consistent* extension is NP-complete.

We also have the following related result.

Corollary 3.13 *The problem of deciding whether a finite default theory* (D, W) *possesses at least one* consistent *extension is* Σ_2^P*-complete.*

4 Autoepistemic Logic

In this section, we discuss autoepistemic logic introduced by Moore [Moo84, Moo85] in a reaction to an earlier modal nonmonotonic logic of McDermott and Doyle [MD80]. We follow closely the presentation proposed in [BNT08].

Autoepistemic logic was introduced to provide an account of a way in which an *ideally rational* agent forms *belief* sets given some initial assumptions. It is a formalism in the modal language $\mathcal{L}_K$ generated from a set of propositional atoms, At, by means of boolean connectives and a (unary) modal

operator K. Intuitively, a formula $K\varphi$ stands for "φ is believed." Subsets of $\mathcal{L}_K$ are *modal theories*. Formulas without K are *modal-free* or *propositional*. The language consisting of all modal-free formulas is denoted by $\mathcal{L}$.

Let us consider a situation in which we have a rule that Professor Jones, being a university professor, normally teaches. To capture this rule in modal logic, we might say that if we do not believe that Dr. Jones does not teach (that is, if it is possible that she does), then Dr. Jones does teach, and write it as:

$$K prof_J \wedge \neg K \neg teaches_J \supset teaches_J. \tag{1}$$

Knowing only $prof_J$ (Dr. Jones is a professor) a rational agent should build a belief set containing $teaches_J$.

We see here a similarity with default logic, where the same rule is formalized by a default

$$\frac{prof(J) : teaches(J)}{teaches(J)}. \tag{2}$$

In default logic, given $W = \{prof(J)\}$, the conclusion $teaches(J)$ is supported as the default theory

$$\left(\left\{ \frac{prof(J) : teaches(J)}{teaches(J)} \right\}, W \right)$$

has exactly one extension and it does contain $teaches(J)$.

The correspondence between the formula (1) and the default (2) is intuitive and compelling. But the autoepistemic logic interpretation of (1) is *not* the same as the default logic interpretation of (2). We will return to this question later.

We will not review the area of modal logics. Instead, for a good introduction, we refer to [Che80, HC84]. But we mention that many modal logics are defined by a selection of modal axioms such K, T, D, 4, 5, etc. For instance, the axioms K, T, 4 and 5 yield the well-known modal logic $\mathrm{S5}$. The consequence operator for a modal logic $\mathcal{S}$, say $Cn_{\mathcal{S}}$, is defined syntactically in terms of the corresponding provability relation for $\mathcal{S}$.

We note that the consequence operator $Cn_{\mathcal{S}}$ can often be described by a class of *Kripke models*, say $\mathcal{C}$: $A \in Cn_{\mathcal{S}}(E)$ if and only if for every Kripke model $M \in \mathcal{C}$ such that $M \models_K E$, $M \models_K A$, where $\models_K$ stands for the relation of satisfiability of a formula or a set of formulas in a Kripke model. For instance, the consequence operator in the modal logic $\mathrm{S5}$ is characterized by *universal* Kripke models (models with the total accessibility relation).

Let us come back to autoepistemic logic. What is an *ideally rational agent* or, more precisely, which modal theories could be taken as belief sets of such agents? Stalnaker [Sta80] argued that to be a belief set of an ideally rational agent a modal theory $E \subseteq \mathcal{L}_K$ must satisfy three closure properties. First, E must be closed under the propositional consequence operator Cn:

B1: $Cn(E) \subseteq E$.

We note that modal logics offer consequence operators which are stronger than the operator Cn. One might argue that closure under one of these operators might be a more appropriate for the condition (B1). As it turns out later, it does not matter.

Next, Stalnaker postulated that theories modeling belief sets of ideally rational agents must be closed under *positive introspection*: if an agent believes in A, then the agent believes she believes A. Formally

B2: if $A \in E$, then $KA \in E$.

Finally, Stalnaker postulated that theories modeling belief sets of ideally rational agents must also be closed under *negative introspection*: if an agent does not believe A, then the agent believes she does not believe A:

B3: if $A \notin E$, then $\neg KA \in E$.

Stalnaker's postulates have become commonly accepted as the defining properties of belief sets of an ideally rational agent. Thus, we refer to modal theories satisfying conditions (B1)–(B3) simply as *belief sets*. The original term used by Stalnaker was a *stable* theory.

Belief sets have a rich theory [MT93a]. We cite here only just two results. The first one shows that given (B2) and (B3) the choice of the consequence operator for the condition (B1) becomes essentially immaterial.

Proposition 4.1 *If $E \subseteq \mathcal{L}_K$ is a belief set, then E is closed under the consequence relation in the modal logic* S5.

The second result shows that belief sets are determined by their modal-free formulas. This property yields to a representation result for belief sets.

Proposition 4.2 *Let $T \subseteq \mathcal{L}$ be closed under propositional consequence. Then $E = Cn_{\mathrm{S5}}(T \cup \{\neg KA \mid A \in \mathcal{L} \setminus T\})$ is a belief set and $E \cap \mathcal{L} = T$. Moreover, if E is a belief set then $T = E \cap \mathcal{L}$ is closed under propositional consequence and $E = Cn_{\mathrm{S5}}(T \cup \{\neg KA \mid A \in \mathcal{L} \setminus T\})$.*

Modal nonmonotonic logics are meant to provide formal means to study mechanisms by which an agent forms belief sets starting with a set T of initial assumptions. These belief sets must contain T but may also satisfy some additional properties. A precise mapping assigning to a set of modal formulas a family of belief sets is what determines a modal nonmonotonic logic.

An obvious possibility is to associate with a set $T \subseteq \mathcal{L}_K$ *all* belief sets E such that $T \subseteq E$. This choice, however, results in a formalism which is *monotone*. Namely, if $T \subseteq T'$, then every belief set for T' is a belief set for T. Consequently, the set of "safe" beliefs — beliefs that belong to every belief set associated with T — grows monotonically as T gets larger. In fact, this set of safe beliefs based on T coincides with the set of consequences of T in the logic S5. As we aim to capture nonmonotonic reasoning, this choice is not of interest to us here.

Another possibility is to employ a minimization principle. Minimizing entire belief sets is of little interest as belief sets are incomparable with respect to inclusion and so, each of them is inclusion-minimal. Thus, this form of minimization does not eliminate any of the belief sets containing T, and so, it is equivalent to the approach discussed above.

A more interesting direction is to apply the minimization principle to modal-free fragments of belief sets (cf. Proposition 4.2, which implies that there is a one-to-one correspondence between belief sets and sets of modal-free formulas closed under propositional consequence). The resulting logic is in fact nonmonotonic and it received some attention [HM85].

The principle put forth by Moore when defining the autoepistemic logic can be viewed as yet another form of minimization. The conditions (B1)–(B3) imply that every belief set E containing T satisfies the inclusion

$$Cn(T \cup \{KA \mid A \in E\} \cup \{\neg KA \mid A \notin E\}) \subseteq E.$$

Belief sets, for which the inclusion is proper contain beliefs that do not follow from initial assumptions and from the results of "introspection" and so, are undesirable. Hence, Moore [Moo85] proposed to associate with T only those belief sets E, which satisfy the *equality*:

$$Cn(T \cup \{KA \mid A \in E\} \cup \{\neg KA \mid A \notin E\}) = E. \tag{3}$$

In fact, when a theory satisfies (3), we no longer need to assume that it is a belief set — (3) implies that it is.

Proposition 4.3 *For every $T \subseteq \mathcal{L}_K$, if $E \subseteq \mathcal{L}_K$ satisfies (3) then E satisfies (B1)–(B3), that is, it is a belief set.*

Moore called belief sets defined by (3) *stable expansions* of T. We refer to them simply as *expansions* of T. We formalize our discussion in the following definition.

Definition 4.1 *Let T be a modal theory. A modal theory E is an* expansion *of T if E satisfies the identity (3).*

Belief sets have an elegant semantic characterization in terms of possible-world structures. Let Int be the set of all 2-valued interpretations (truth assignments) of At. *Possible-world structures* are subsets of Int. Intuitively, a possible-world structure collects all interpretations that *might* be describing the actual world and leaves out those that definitely do not.

A possible-world structure is essentially a Kripke model with a total accessibility relation [Che80, HC84]. The difference is that the universe of a Kripke model is required to be nonempty, which guarantees that the *theory* of the model (the set of all formulas true in the model) is consistent. Some modal theories consistent with respect to the propositional consequence relation determine inconsistent sets of beliefs. Allowing possible-world structures to be empty is a way to capture such situations and differentiate them from those situations, in which a modal theory determines no belief sets at all.

Possible-world structures interpret modal formulas, that is, assign to them truth values.

Definition 4.2 *Let $Q \subseteq Int$ be a possible-world structure and $I \in Int$ a two-valued interpretation. We define the* truth function $\mathcal{H}_{Q,I}$ *inductively as follows:*

1. $\mathcal{H}_{Q,I}(p) = I(p)$*, if p is an atom.*
2. $\mathcal{H}_{Q,I}(A_1 \wedge A_2) = \mathbf{t}$ *if* $\mathcal{H}_{Q,I}(A_1) = \mathbf{t}$ *and* $\mathcal{H}_{Q,I}(A_2) = \mathbf{t}$*. Otherwise,* $\mathcal{H}_{Q,I}(A_1 \wedge A_2) = \mathbf{f}$.
3. *Other boolean connectives are treated similarly.*
4. $\mathcal{H}_{Q,I}(KA) = \mathbf{t}$*, if for every interpretation $J \in Q$,* $\mathcal{H}_{Q,J}(A) = \mathbf{t}$*. Otherwise,* $\mathcal{H}_{Q,I}(KA) = \mathbf{f}$.

It follows directly from the definition that for every formula $A \in \mathcal{L}_K$, the truth value $\mathcal{H}_{Q,I}(KA)$ does not depend on I. It is fully determined by the possible-world structure Q and we will denote it by $\mathcal{H}_Q(KA)$, dropping I from the notation.

The *theory* of a possible-world structure Q is the set of all modal formulas that are *believed* in Q. We denote it by $Th(Q)$. Thus, formally,

$$Th(Q) = \{A \mid \mathcal{H}_Q(KA) = \mathbf{t}\}.$$

We now present a characterization of belief sets in terms of possible-world structures, which we promised earlier.

Theorem 4.4 *A set of modal formulas $E \subseteq \mathcal{L}_K$ is a belief set if and only if there is a possible-world structure $Q \subseteq Int$ such that $E = Th(Q)$.*

Expansions of a modal theory can also be characterized in terms of possible-world structures. The underlying intuitions arise from considering a way to revise possible-world structures, given a set T of initial assumptions. The characterization is also due to Moore. Namely, for every modal theory T, Moore [Moo84] defined an operator D_T on $\mathcal{P}(Int)$ (the space of all possible-world structures) by setting

$$D_T(Q) = \{I \mid \mathcal{H}_{Q,I}(A) = \mathbf{t}, \text{ for every } A \in T\}.$$

The operator D_T specifies a process to revise belief sets encoded by the corresponding possible-world structures. Given a modal theory $T \subseteq \mathcal{L}_K$, the operator D_T revises a possible-world structure Q with a possible-world structure $D_T(Q)$. This revised structure consists of all interpretations that are *acceptable* given the current structure Q and the constraints on belief sets encoded by T. Specifically, the revision consists precisely of those interpretations that make all formulas in T true with respect to Q.

Fixed points of the operator D_T are of particular interest. They represent "stable" possible-world structures (and so, belief sets) — they cannot be revised any further. This property is behind the role they play in the autoepistemic logic.

Theorem 4.5 *Let $T \subseteq \mathcal{L}_K$. A set of modal formulas $E \subseteq \mathcal{L}_K$ is an expansion of T if and only if there is a possible-world structure $Q \subseteq \mathcal{I}$ such that $Q = D_T(Q)$ and $E = Th(Q)$.*

This theorem implies a systematic procedure for constructing expansions of *finite* modal theories (or, to be more precise, possible-world structures that determine expansions). Let us continue our "Professor Jones" example and let us look at a theory

$$T = \{prof_J, Kprof_J \wedge \neg K \neg teaches_J \supset teaches_J\}.$$

There are two propositional variables in our language and, consequently, four propositional interpretations:

$I_1 = \emptyset$ (neither $prof_J$ nor $teaches_J$ is true)
$I_2 = \{prof_J\}$
$I_3 = \{teaches_J\}$
$I_4 = \{prof_J, teaches_J\}$.

There are 16 possible-world structures one can build of these four interpretations. Only one of them, though, $Q = \{prof_J, teaches_J\}$, satisfies $D_T(Q) = Q$ and so, generates an expansion of T. We skip the details of verifying it, as the process is long and tedious, and we present a more efficient method in the next section. We note however, that for the basic "Professor Jones" example autoepistemic logic gives the same conclusions as default logic.

We close this section by noting that the autoepistemic logic can also be obtained as a special case of a general fixed point schema to define modal nonmonotonic logics proposed by McDermott [McD82]. In this schema, we assume that an agent uses some modal logic $\mathcal{S}$ (extending propositional logic) to capture her basic means of inference. We then say that a modal theory $E \subseteq \mathcal{L}_K$ is an *$\mathcal{S}$-expansion* of a modal theory T if

$$E = Cn_{\mathcal{S}}(T \cup \{\neg KA \mid A \notin E\}). \tag{4}$$

In this equation, $Cn_{\mathcal{S}}$ represents the consequence relation in the modal logic $\mathcal{S}$. If E satisfies (4), then E is closed under the propositional consequence relation. Moreover, E is closed under the necessitation rule and so, E is closed under positive introspection. Finally, since $\{\neg KA \mid A \notin E\} \subseteq E$, E is closed under negative introspection. It follows that solutions to (4) are belief sets containing T. They can be taken as models of belief sets of agents reasoning by means of modal logic $\mathcal{S}$ and justifying what they believe on the basis of initial assumptions in T and *assumptions* about what *not* to believe (negative introspection). By choosing different monotone logics $\mathcal{S}$, we obtain from this schema different classes of $\mathcal{S}$-expansions of T.

If we disregard inconsistent expansions, autoepistemic logic can be viewed as a special instance of this schema, with $\mathcal{S} = \mathrm{KD45}$, the modal logic determined by the axioms K, D, 4 and 5 [HC84, MT93a]. Namely, we have the following result.

Theorem 4.6 *Let $T \subseteq \mathcal{L}_K$. If $E \subseteq \mathcal{L}_K$ is consistent, then E is and expansion of T if and only if E is a KD45-expansion of T, that is,*

$$E = Cn_{\mathrm{KD45}}(T \cup \{\neg KA \mid A \notin E\}).$$

5 Introduction to logic programming

A logic program is a declarative specification of one or more relational systems. The underlying language is that of first-order logic. However, the semantics is restricted to that of Herbrand models. Thus, semantically, there is no difference between a logic program P and its grounding *ground*(P). Consequently, from now on we focus almost entirely on propositional logic programs, with atoms from a fixed countable set At. We note however, that effective programming requires full language (or, at least, its function-free fragment) and several interesting questions concerning the complexity and expressive power make only sense for programs with variables. We refer to [Llo84, Apt90, Doe94] for more detailed in-depth presentations.

5.1 Basic syntax and semantics

A *program rule* or *clause* is an expression of the form

$$C = p \leftarrow q_1, \ldots, q_m, \mathit{not}\ r_1, \ldots, \mathit{not}\ r_n$$

where $p, q_1, \ldots, q_m, r_1, \ldots, r_n$ are atoms. The atom p is called the *head* of rule C and is denoted by *hd*(C). The expression $q_1, \ldots, q_m,$ *not* $r_1, \ldots,$ *not* r_n is called the *body* of C and is denoted by *bd*(C).

A *program* is a set of rules (possibly infinite). For a program P, by *hd*(P) we denote the set of the heads of all rules in P.

A rule is called *definite* (or *Horn program*) if $n = 0$. That is, Horn rules are of the form

$$p \leftarrow q_1, \ldots, q_m$$

A definite (or Horn) program is a set of Horn program rules.

Given a program P, its *Herbrand base* is the set of all atoms occurring in P. The Herbrand base of P is denoted by *At*(P). Since we are dealing with the propositional case only, the Herbrand base of P is also denoted by $At(P)$, as it consists of propositional atoms. In this tutorial, we prefer the latter notation. For instance, if the program P is

$p \leftarrow q,$ *not* r
$q \leftarrow s$
s
$t \leftarrow r$

then its Herbrand base *At*(P) is $\{p, q, r, s, t\}$.

An *interpretation* of the Herbrand base of program P is a mapping v : *At*$(P) \rightarrow \{0, 1\}$. An interpretation asserts some atoms to be true and some atoms to be false. Such mapping is uniquely determined by the set of atoms on which v takes value 1. We will typically think and write about interpretations in these terms. We note that the set of interpretations viewed in this way possesses a natural ordering given by inclusion.

An interpretation M satisfies a literal *not* a if $a \notin M$. It satisfies the body of a rule r, if it satisfies all literals in the body. Next, M satisfies a rule if it satisfies the head of the rule whenever it satisfies the body of a rule. Finally, M is a model of P if M is a model of every rule in P. We write $\models$ for the satisfiability relation. We note, that under this concept of satisfiability there is no difference between logic program rules and the corresponding propositional interpretations.

We will now associate operators with logic programs. Thus let P be a program. We define an operator T_P on interpretations (subsets of At) [vEK76], by setting

$$T_P(M) = \{\mathit{hd}(r) \colon r \in P, \;\; M \models \mathit{bd}(r)\}$$

The operator T_P is not, in general, monotone. If, however, the program P is a Horn program then it is monotone. Thus we can use Theorem 2.1 to assert that it has a least fixpoint. Moreover, we can show that this least fixpoint is given by $T^\omega(\emptyset)$.

Theorem 5.1 *If P is a Horn program then T_P is monotone and has a least fixpoint. This least fixpoint is given by $T^\omega(\emptyset)$.*

Let us look at an example.

Example 5.1 The program P is

$a \leftarrow$
$b \leftarrow$
$c \leftarrow a, b$
$a \leftarrow d$

The Herbrand base *At*(P) is $\{a, b, c, d\}$. Clearly,

$T_P^0(\emptyset) = \emptyset$
$T_P^1(\emptyset) = \{a, b\}$
$T_P^2(\emptyset) = \{a, b, c\}$

All the remaining iterations are equal to $T^2(\emptyset)$. The least fixpoint of T_P is $\{a, b, c\}$.

The importance of the operators for logic programming lies in the following result of van Emden and Kowalski [vEK76].

Theorem 5.2 *Let P be a program. Then models of the program are precisely prefixpoints of the operator T_P. Moreover, if the program P is a Horn program, then it possesses a least model which is the least fixpoint of the operator T_P and is equal to $T^\omega(\emptyset)$.*

We denote the least model of a Horn program P by $LM(P)$.

5.2 Supported models

Since prefixpoints of T_P are models of P, so are *fixpoints* of T_P. We call fixpoints of T_P *supported* models of P.

Given an interpretation M, we say that all elements of $T_P(M)$ have *support* in P and M. Thus, M is a supported model of P if M is the set of all elements that have support in P and M. Informally, M is a supported model of P if M supports itself through P.

Clearly, all supported models of a program P are contained in $At(P)$. Also, if P is a Horn program, the least model of P (as a fixpoint of T_P) is a supported model of P.

Example 5.2 *Let P be the following program:*

$a \leftarrow b$
$b \leftarrow a, \mathit{not}\ c$

Then $\emptyset$ and $\{a, b\}$ are supported models of P.

The second supported model in Example 5.2 is of interest as it is truly self-supported (circularly supported). Indeed, a is supported by b and b is supported by a (and the absence of c).

Supported models are related to the notion of *program completion* [Cla78]. Let r be a program rule. By $bd^\wedge(r)$ we denote the (propositional) conjunction of literals in the body of r (with *not* replaced by $\neg$. We now define:

$$cmpl^\leftarrow(P) = \{bd^\wedge(r) \rightarrow hd(r) \colon r \in P\}.$$

Thus, $cmpl^{\leftarrow}(P)$ is nothing else but a theory obtained by interpreting program rules as propositional implications.

We will view all rules in P with the head a as a defnition of a (listing all cases when a is true). Thus, we define

$$def_P(a) = \bigvee\{bd^{\wedge}(r)\colon hd(r) = a\}$$

to denote a formula defining a. From this perspective, if a holds $def_P(a)$ must hold too. This is captured by the formula

$$cmpl^{\rightarrow}(P) = \{a \rightarrow def_P(a)\colon a \in At\}$$

Thus, to capture a program P in propositional logic we could use the theory

$$cmpl(P) = cmpl^{\leftarrow}(P) \cup cmpl^{\rightarrow}(P)$$

This theory is known as the *completion* of P [Cla78]. We have the following result connection the completion with supported models [MS92].

Theorem 5.3 *Let P be a program. A set $M \subseteq At$ is a supported model of P if and only if M is a model of $cmpl(P)$.*

In particular, it follows that supported models of a program can be computed by SAT solvers.

5.3 Stable model semantics

Stable model semantics [GL88] is one of the most commonly accepted semantics for logic programs with negation. In this section, we will introduce this notion and describe some of its most important properties.

Let P be a propositional logic program over a set of atoms *At*. Let $M \subseteq At(P)$. By the *Gelfond-Lifschitz reduct of P with respect to M*, denoted by P^M, we mean the logic program obtained from P by:

1. removing from P all rules with a literal *not* a in the body for some $a \in M$
2. removing all negative literals from all other rules in P.

For example, consider a propositional logic program P consisting of the following rules:

(1) $p \leftarrow q, \mathit{not}\ r$
(2) $p \leftarrow \mathit{not}\ p$
(3) $q \leftarrow$
(4) $r \leftarrow t, \mathit{not}\ s$
(5) $s \leftarrow \mathit{not}\ q$.

Let $M = \{p, q\}$. Then, P^M consists of the following rules:

$p \leftarrow q$
$q \leftarrow$
$r \leftarrow t$

The rule $p \leftarrow$ *not* p is eliminated because $p \in M$. Similarly, rule $s \leftarrow$ *not* q is eliminated because $q \in M$. Since all negative literals in all the remaining rules are of the form *not* a for some $a \notin M$, the rules are not eliminated but their negative literals are!

Clearly, for every logic program P and for every set of atoms M, P^M is a Horn program. Consequently, this logic program has its least model $LM(P^M)$.

Definition 5.1 *We say that a set of atoms M is a* stable model *of a propositional logic program P if $M = LM(P^M)$.*

From this definition it is not at all clear that if M is a stable model of P then it is a model. This is, however, the case.

Proposition 5.4 *If M is a stable model of a logic program P, then M is a model of P. Moreover, M is a minimal model of P.*

Let us look again at program P consisting of rules (1) - (5). For $M = \{p, q\}$, the reduct P^M is described above and it is clear that $LM(P^M) = M$. Hence, $M = \{p, q\}$ is a stable model of P. On the other hand, $M = \{q, r\}$ is not. Indeed. in this case, P^M consists of

$p \leftarrow$
$q \leftarrow$
$r \leftarrow t$

The least model of P^M is $\{p, q\}$. Since it is different from $M = \{q, r\}$, M is not a stable model.

Next, we observe that stable models are supported.

Proposition 5.5 *If M is a stable model of a logic program P, then M is a supported model of P.*

The converse is not true in general. We have seen that $\{a, b\}$ is a supported model of the program P from Example 5.2. Clearly, $P^{\{}a, b\}$ consists of the rules $a \leftarrow b$ and $b \leftarrow a$. Its least model is $\emptyset$ and it is different from $\{a, b\}$. Thus, $\{a, b\}$ is not a stable model of P.

We will now describe another characterizations of stable models of propositional logic programs. For each such a program, we define $\overline{P}$ to be a Horn logic program obtained from P by treating all negative literals *not* a as distinct propositional atoms. Let *At* be a set of propositional variables appearing in a program P. For a set $N \subseteq$ *At*, define the program $P(N)$ as follows:

$$P(N) = \overline{P} \cup \{\textit{not } a \leftarrow : a \in N\}$$

(where literals *not* a, for $a \in N$, are treated as propositional atoms.

For instance, for the program P consisting of the rules (1) - (5) and for the set of atoms $N = \{r, s, t\}$, $P(N)$ consists of:

$p \leftarrow q,$ *not* r
$p \leftarrow$ *not* p
$q \leftarrow$

$r \leftarrow t, \mathit{not}\ s$
$s \leftarrow \mathit{not}\ q$
$\mathit{not}\ r \leftarrow$
$\mathit{not}\ s \leftarrow$
$\mathit{not}\ t \leftarrow$

It is easy to see that the least model of this program is

$$LM(P(N)) = \{p, q, \mathit{not}\ r, \mathit{not}\ s, \mathit{not}\ t\}.$$

Its "positive" part, $\{p, q\}$, as we saw earlier, is a stable model of P, and it is also a "complement" of the "negative" part $\{\mathit{not}\ r, \mathit{not}\ s, \mathit{not}\ t\}$ of $LM(P(N))$. It is not coincidental. We have the following result.

Theorem 5.6 *Let P be a propositional logic program and let At be the set of atoms appearing in P. A set $M \subseteq$ At is a stable model of P if and only if $M \cup \{\mathit{not}\ a\colon a \in \mathit{At} \setminus M\}$ is the least model of the Horn program $P(\mathit{At} \setminus M)$.*

Next, we describe some simple properties of stable model semantics. First, we will show that stable model semantics extends the least model semantics for Horn programs.

Proposition 5.7 *If P is a Horn program then P has exactly one stable model. It coincides with the least model of P, $LM(P)$.*

Next, we observe that the operator

$$GL_P(M) = LM(P^M)$$

(the *Gelfond-Lifschitz operator*) is antimonotone. That is, if $M_1 \subseteq M_2$ then

$$GL_P(M_2) \subseteq GL_P(M_1).$$

Hence (Theorem 2.3), we have the following result.

Proposition 5.8 *Let P be a logic program. Then, the family of its stable models forms an antichain with respect to inclusion.*

Our results on stable models for logic programs parallel, in many cases, the results on extensions of default theories. This is not coincidental. Logic programming with stable model semantics can be regarded as a special case of default logic.

A *default interpretation* of a logic program rule

$$C = p \leftarrow q_1, \ldots, q_m, \mathit{not}\ r_1, \ldots, \mathit{not}\ r_n$$

is the default

$$dl(C) = \frac{q_1 \wedge \ldots \wedge q_m\colon \neg r_1, \ldots, \neg r_n}{p}.$$

By the *default interpretation* of a logic program P we mean the default theory $(dl(P), \emptyset)$, where

$$dl(P) = \{dl(C)\colon C \in P\}.$$

We have the following result relating logic programming and default logic.

Theorem 5.9 *Let P be a logic program. A set of atoms M is a stable model for P if and only if $Cn(M)$ is an extension of $(dl(P), \emptyset)$. Conversely, every extension of $(dl(P), \emptyset)$ is of the form $Cn(M)$, for some stable model M for P.*

We have a similar connection to autoepistemic logic. We interpret a rule C given above by the following modal formula:

$$ael(C) = \neg Kr_1 \wedge \ldots \wedge \neg Kr_n \supset (q_1 \wedge \ldots \wedge q_m \supset p)$$

By the *autoepistemic interpretation* of a logic program P we mean the modal theory $ael(P)$, where

$$ael(P) = \{ael(C) \colon C \in P\}.$$

For an modal theory E, we denote by $At(E)$ the set of propositional atoms in E.

Theorem 5.10 *Let P be a logic program. A set $M \subseteq At$ is a stable model of P if and only if there is an expansion E of $ael(P)$ such that $M = At(E)$.*

5.4 Well-founded semantics

We will now describe the so-called well-founded semantics of a program [VRS91].

Let P be a propositional program. We will consider the operator GL_P whose fixpoints are stable models of P. We recall that $GL_P(M) = LM(P^M)$. As we observed in Section 5.3, the operator GL_P is antimonotone. Thus, its second iteration, GL_P^2 is a monotone operator. According to Theorem 2.2, the operator GL_P^2 possesses a least fixpoint. We will denote it by $T(P)$. We also write $M(P) = GL_P(T(P))$. One can check that $M(P)$ is the largest fixpoint of GL_P^2. These fixpoints oscillate, that is,

$$GL_P(T(P)) = M(P) \quad \text{and} \quad GL_P(M(P)) = T(P).$$

Moreover, all the fixpoints of GL_P (stable models) include $T(P)$ and are themselves included in $M(P)$. Thus the least fixpoint of GL_P^2 approximates from below the intersection of all stable models, whereas the largest fixpoint approximates from above the union of stable models of P. It is important to see that the approximation is all we get.

Example 5.3 Let P be this program

$p \leftarrow$ *not* q
$q \leftarrow$ *not* (p)

The least fixpoint of GL_P^2 is empty set, whereas the largest fixpoint is $\{p.q.r\}$. The intersection of all stable models is $\{r\}$ and the union of all stable models is $\{p, q, r\}$.

Well-founded semantics is three valued. Atoms in $T(P)$ are interpreted as *true*, atoms in $M(P)$ are interpreted as *possibly* true, and atoms not in $M(P)$ (we will denote the set of such atoms by $F(P)$) are interpreted as *false*.

5.5 Complexity

Stable model semantics has a major drawback. It is computationally complex [MT91].

Theorem 5.11 *The following problems are NP-complete:*

1. *Given a finite propositional logic program P, decide whether P has a stable model*
2. *Given a finite propositional logic program P and an atom a, decide whether there is a stable model M of P such that $a \in M$*

The following problem is co-NP-complete:

3 *Given a finite propositional logic program P and an atom a, decide whether a is in all stable models of P*

This theorem remains true even under fairly restrictive conditions imposed on the rules. For instance, the assertion remains true for the class of programs in which each rule has no positive atoms and at most one negative literal in the body.

In contrast, well-founded semantics has very good computational properties. In fact, the algorithm follows directly from the definition of well-founded semantics. First, let us recall that the least model of a finite propositional Horn program can be computed in time $O(size(P))$, where $size(P)$ denotes the total length of all rules in P. Consequently, given a finite propositional logic program P and a subset M of the set of atoms appearing in P, $GL_P(M)$ can be computed in time $O(size(P))$. According to the definition of well-founded semantics, sets T and F can be computed by iterating the operator GL_P starting with the empty set of atoms. Every two iterations the head of at least one rule is added to T, or the computation stops. So, the computation terminates in $O(|P|)$ iterations. Thus, we obtain the following result.

Theorem 5.12 *There is an algorithm that, given a finite propositional logic program P, computes the well-founded semantics for P in time $O(|P| \times size(P))$, where $|P|$ denotes the number of rules in P and $size(P)$ denotes the total length of all rules in P.*

For more details on well-founded semantics computation we refer to [BSJ95, LT00].

5.6 Stratification and splitting

The efficiency of stable model computation can be improved by exploiting the concept of *stratification*. While most of the concepts presented in this section can be generalized to the infinite case, we will restrict our discussion to the case of finite propositional programs.

Let P be a logic program. Let $P_1, \ldots, P_k$ be nonempty disjoint subprograms of P such that $P_1 \cup \ldots \cup P_k = P$. We say that $P_1, \ldots, P_k$ is a *relaxed stratification* of P if for every $1 \leq i < j \leq k$, $Var(P_i) \cap hd(P_j) = \emptyset$. Given a relaxed stratification of a logic program P, we will compute stable models for P by computing stable models for P_1, then extending them to stable models of $P_1 \cup P_2$, then extending them to stable models of $P_1 \cup P_2 \cup P_3$, and so on. The intuitive explanation of the

correctness of this method rests on an observation that in iteration i we can only determine the status of the heads of the rules in P_i, and they have no effect on the semantics of $P_1 \cup \ldots \cup P_{i-1}$. Now, we have the following result.

Theorem 5.13 *Let P_1, P_2 be relaxed stratification of a logic program P. A set of atoms M is a stable model for P if and only if there is a subset M_1 of M such that*

1. *M_1 is a stable model of P_1*
2. *M is a stable model of $P_2 \cup M_1$*

This theorem can be extended by induction to the case of arbitrary relaxed stratifications.

The role of this theorem in stable model computation is now clear. It allows us to replace the task of computing stable models for P with similar tasks but for simpler programs P_1 and $P_2 \cup M$, where M is a stable model for P_1. This leads to substantial pruning of the search space.

This theorem also implies a stronger result for the class of *stratified* programs. A relaxed stratification $P_1, \ldots, P_k$ of a logic program P is a *stratification* of P if for each i, and for each atom a, if *not* a appears in the body of a rule from P_i then a does not appear as the head of a rule from P_i.

While every program P has relaxed stratification, there are programs that do not admit stratification. Stratified programs can be viewed as an extension of Horn programs, which allows for the use of negation in the body of rules but preserves some key properties of Horn programs.

Theorem 5.14 *Let P be stratified logic program. Then P has a unique stable model and this model can be computed in time $O(size(P))$.*

5.7 Tight programs, Fages lemma

We noted that stable models are supported but the converse is not true in general. We will now present a syntactic condition on programs that guarantees which guarantees that supported models as stable. The results we present here are due to Fages [Fag94], and Erdem and Lifschitz [EL03].

We define a *positive dependency graph* $G^+(P)$ for a program P as follows. Elements of *At*(P) as the vertices of $G^+(P)$, and (a, b) is an edge in $G^+(P)$ if for some $r \in P$, *hd*$(r) = a$, and $b \in bd^+(r)$ (*bd* $+ (r)$ is the set of non-negated atoms in the body of r). A program P is *tight* if $G^+(P)$ is acyclic. Alternatively, a program P is tight if there is a labeling of atoms with non-negative integers ($a \mapsto \lambda(a)$) s.t. for every rule $r \in P$

$$\lambda(hd(r)) > \max\{\lambda(b) \colon b \in bd^+(r)\}$$

Theorem 5.15 *If a program P is tight then every supported model is stable.*

The assmption of tightness can be relaxed. Let $X \subseteq$ *At*(P). Then, P is *tight on* X if the program consisting of rules $r \in P$ such that $bd^+(r) \subseteq X$ is tight.

Theorem 5.16 *Let P be a logic program. If P is tight on X and M is a supported model of P such that $M \subseteq X$, then M is stable.*

We noted that SAT solvers can be used to compute supported models of programs as they are models of the completion of the program. The results we presented here allow us, in some cases, to compute stable models by means of SAT solvers. Clearly, there is no problem if the input program is tight — stable and supported models coincide, so models of the completion are stable,

If the input program is not tight, but is tight on X, we can run a SAT solver on the theory $cmpl(P) \cup \{\neg a \colon a \notin X\}$. If this theory has a model, it is a stable model of P. This method is effective if we are only interested in stable models of a program that are contained in some set X on which P is tight.

5.8 Loop formulas

Loop formulas were introduced in [LZ02]. They allow us to transform a logic program into a propositional theory so that stable models correspond to models. This transformation is the basis of highly effective algorithms for computing stable models that utilize SAT solvers and take advantage of major advances in SAT technology that have taken place in recent years (we refer to `http://www.satlive.org/` for a wealth of information on the topic and relevant references). In presenting loop formulas and their properties, we follow [FLL06].

Let P be a logic program and $Y \subseteq At(P)$. We define the *external support formula for* Y as the disjunction of all formulas $bd^{\wedge}(r)$, where $r \in P$ satisfies:

1. $hd(r) \in Y$
2. $bd^{+}(r) \cap Y = \emptyset$.

The external support formula for Y captures the following idea: if Y is to be a part of a stable model, Y must not be self-supported through positive recursion. To put it more formally, there must be at least one element $a \in Y$ with positive support outside of Y, that is, with a rule $r \in P$ such that $a = hd(r)$ and $bd^{+}(r) \cap Y = \emptyset$.

We note that $ES_P(\emptyset) = \top$ (the emptyset is always externally supported). We also observe that $ES_P(\{a\}) = def_P(a)$. In other words, the external support formula for a singleton set is the same as the defining formula for its element.

We now have the following theorem.

Theorem 5.17 *Let P be a logic program. The following conditions are equivalent:*

1. *X is a stable model of P*
2. *X is a model of $cmpl^{\leftarrow}(P) \cup \{Y^{\wedge} \rightarrow ES_P(Y) \colon Y \subseteq At(P)\}$*
3. *X is a model of $cmpl^{\leftarrow}(P) \cup \{Y^{\vee} \rightarrow ES_P(Y) \colon Y \subseteq At(P)\}$*
4. *X is a model of $cmpl(P) \cup \{Y^{\wedge} \rightarrow ES_P(Y) \colon Y \subseteq At(P)\}$*
5. *X is a model of $cmpl(P) \cup \{Y^{\vee} \rightarrow ES_P(Y) \colon Y \subseteq At(P)\}$*

To see why (2) and (3) are equivalent to (4) and (5), respectively, we note that

$$cmpl^{\rightarrow}(P) \subseteq \{Y^{\wedge} \rightarrow ES_P(Y) \colon Y \subseteq At(P)\}$$

and

$$cmpl^{\rightarrow}(P) \subseteq \{Y^{\vee} \rightarrow ES_P(Y) \colon Y \subseteq At(P)\}$$

This result gives a first representation of programs as propositional theories. However, it is clear that $cmpl^{\leftarrow}(P) \cup \{Y^{\wedge} \rightarrow ES_P(Y) \colon Y \subseteq At(P)\}$ (and all other theories given in the theorem) are exponential in the size of P. Thus, we do not have yet an effective way to compute stable models with SAT solvers.

To improve Theorem 5.17, we restrict the class of sets $Y \subseteq At(P)$, for which the formulas $Y^{\wedge} \rightarrow ES_P(Y)$ (or $Y^{\vee} \rightarrow ES_P(Y)$) need to be added to $cmpl^{\leftarrow}(P)$.

Definition 5.2 *A* loop *is a set $Y \subseteq At(P)$ that induces in $G^{+}(P)$ a strongly connected subgraph*

We note that, in particular, all singleton sets are loops. Formulas of the form $\{Y^{\wedge} \rightarrow ES_P(Y)$ are *conjunctive* loop formulas, Formulas of the form $\{Y^{\vee} \rightarrow ES_P(Y)$ are *disjunctive* loop formulas,

Theorem 5.18 *Let P be a logic program. The following conditions are equivalent:*

1. *X is a stable model of P*
2. *X is a model of $cmpl^{\leftarrow}(P) \cup \{Y^{\wedge} \rightarrow ES_P(Y) \colon Y$ – a loop$\}$*
3. *X is a model of $cmpl^{\leftarrow}(P) \cup \{Y^{\vee} \rightarrow ES_P(Y) \colon Y$ – a loop$\}$*
4. *X is a model of $cmpl(P) \cup \{Y^{\wedge} \rightarrow ES_P(Y) \colon Y$ – a loop$\}$*
5. *X is a model of $cmpl(P) \cup \{Y^{\vee} \rightarrow ES_P(Y) \colon Y$ – a loop$\}$*

This theorem shows hat stable models can be computed as models of smaller propositional theories. However, even under the restriction to loops, the size may be exponential in the size of P. The key to more practical algorithms is an observation [LZ02] that loop formulas can be added incrementally. We start a SAT solver on the theory $cmpl(P)$. When a model is found, it is a supported model of P. If it is stable, we are done. If it is not stable, it gives rise to a loop formula that is not satisfied by it. We add this loop formula and start the SAT solver again. We continue until we find a stable model or the program terminates without finding models. Here also, in the worst case, we may need to add exponentially many loop formulas before we terminate. However, in many practical situations, there are either few loops or a stable model is found only after just a small number of loop formulas are added.

5.9 Strong and uniform equivalence of programs

It is commonly accepted that modular program (or knowledge base) design is a fundamental to facilitate development, verification and maintenance. For instance, to improve performance, one might want to focus on a single module and replace its present implementation with an optimized one, before moving on to the next module. However, it is important that the replacement does not change the overall meaning of the program or knowledge base. Thus, deciding when two programs (or knowledge bases are *equivalent for substitution* is a fundamental problem in or declarative programming and knowledge representation.

If a knowledge base is a theory in propositional logic, equivalence for substitution coincides with the standard logical equivalence: theories P and Q are equivalent for substitution if and only if they are logically equivalent.

In nonmonotonic logics, the situation is more complex. In particular, in logic programming with the semantics of stable models [GL88], having the same stable models does not guarantee equivalence for substitution. Before we demonstrate this, we will formally define the notion. Following [LPV01], where it was introduced, we use the term *strong equivalence* instead of *equivalence for substitution*.

Definition 5.3 *Logic programs P and Q are* strongly equivalent *if for every logic program R, $P \cup R$ and $Q \cup R$ have the same stable models.*

To show that standard nonmonotonic equivalence (having the same stable models) is not enough to ensure strong equivalence, let us consider programs:

$$P = \{p\} \text{ and } Q = \{p \leftarrow \textit{not } (q)\}.$$

They the same stable models (each program has $\{p\}$ as its *only* stable model). However, $P \cup \{q\}$ and $Q \cup \{q\}$ have *different* stable models. The only stable model of $P \cup \{q\}$ is $\{p, q\}$ and the only stable model of $Q \cup \{q\}$ is $\{q\}$. Similarly, $P \cup \{q \leftarrow \textit{not } (p)\}$ has one stable model, $\{p\}$, and $Q \cup \{q \leftarrow \textit{not } (p)\}$ has two stable models $\{p\}$ and $\{q\}$.

[LPV01] presented a characterization of strong equivalence of nested logic programs by exploiting properties of the logic *here-and-there* [Hey30]. [Tur01, Lin02, Tur03] continued these studies and obtained simple characterizations of strong equivalence in terms of *se-models*, without explicit references to the logic *here-and-there*.

[EF03] introduced one more notion of equivalence, the *uniform equivalence* of logic programs with answer-set semantics.

Definition 5.4 *Logic programs P and Q are* uniformly equivalent *if for every set R of* facts, *$P \cup R$ and $Q \cup R$ have the same stable models.*

[EF03] presented a characterization of *uniform equivalence* in terms of *se-models* and then, for finite programs, in terms of *ue-models*, which are se-models with some additional properties.

We will now present key notions and results. First, given a program P we say that a pair (X, Y), with X, Y sets of atoms, is an *se-model* of P if

1. $X \subseteq Y$
2. $Y \models P$
3. $X \models P^Y$.

We denote by $SE(P)$ the set of all se-models of P. The following characterization is due to Turner [Tur03].

Theorem 5.19 *Programs P and Q are strongly equivalent if and only if $SE(P) = SE(Q)$.*

Se-models can also be used to characterize uniform equivalence. The following result comes from [EF03].

Theorem 5.20 *Let P and Q be programs. Then P and Q are uniformly equivalent if and only if*

1. *for every $Y \subseteq At$, Y is a model of P if and only if Y is a model of Q*
2. *for every $(x, y) \in SE(P)$ such that $X \subset Y$, there is $U \subseteq At$ such that $X \subseteq U \subset Y$ and $(U, Y) \in SE(Q)$*
3. *for every $(x, y) \in SE(Q)$ such that $X \subset Y$, there is $U \subseteq At$ such that $X \subseteq U \subset Y$ and $(U, Y) \in SE(P)$*

For finite programs we have a simpler characterization. An se-model (X, Y) of P is a *ue-model* of P if for every U such that $X \subset U \subseteq Y$, $(U, Y) \in SE(P)$ implies $U = Y$. We write $UE(P)$ for the set of ue-models of P.

Theorem 5.21 *Finite programs P and Q are uniformly equivalent if and only if $UE(P) = UE(Q)$.*

We note that all these results have extensions to the case of disjunctive logic programs (in fact, even general logic programs). We also note that it is coNP-complete to decide strong equivalence or uniform equivalence for normal (non-disjunctive) logic programs. When we move up to disjunctive programs, the complexity of deciding strong equivalence remains the same but the complexity of deciding uniform equivalence goes up to Π_2^P-complete.

5.10 General logic programs

In this section, we follow [FL05]. All results we provide come from that paper. We also refer to [LTT99] for a slightly different perspective (closely related but, in some aspects, more general).

Formulas are build from atoms and the symbol $\bot$ ("false") by means of the connectives $\wedge$, $\vee$ and $\rightarrow$. Thus, the language with which we work here is a *restricted* language of propositional logic. We introduce other connectives as shorthands:

1. $\neg F ::= F \rightarrow \bot$
2. $\top ::= \bot \rightarrow \bot$
3. $F \leftrightarrow G ::= (F \rightarrow G) \wedge (G \rightarrow F)$

As in other places, we consider sets of atoms as interpretations and define the satisfiability relation $\models$ in a standard propositional logic way.

An occurrence of an atom a in a formula F is *positive*, if the number of implications containing this occurrence of a in the antecedent is even. Otherwise, it is *negative*. An occurrence of a in F is *strictly positive* if no implication contains this occurrence of a in the antecedent. In particular, $\neg F$, being actually $F \rightarrow \bot$ has no strict occurrences of any atom.

We will be interested here in the *stable-model* semantics of theories in the language we described. To this end, we define first the notion of the *reduct*.

Definition 5.5 *The* reduct *of a formula F with respect to a set X of atoms is the formula F^X obtained by replacing in F each maximal subformula of F that is not satisfied by X with $\bot$.*

Example 5.4 *Let $F = (\neg p \to q) \wedge (\neg q \to p)$ and $X = \{p\}$. We observe that:*

1. *$\neg p = p \to \bot$, and $X \models \neg p \to q$. Thus, $\neg p$ is a maximal subformula not satisfied by X*
2. *$\neg q = q \to \bot$, $X \not\models q$, $X \models \neg q$. Thus, q is a maximal subformula not satisfied by X*

Thus, $F^X = (\bot \to q) \wedge ((\bot \to \bot) \to p)$. It is classically equivalent to p.

The following properties facilitate the computation of the reduct:

1. $\bot^X = \bot$
2. For a an atom, if $a \in X$, $a^X = a$; otherwise, $a^X = \bot$
3. If $X \models F \circ G$, $(F \circ G)^X = F^X \circ G^X$; otherwise, $(F \circ G)^X = \bot$ ($\circ$ stands for any of $\wedge$, $\vee$, $\to$)
4. If $X \models F$, $(\neg F)^X = \bot$; otherwise, $(\neg F)^X = (F \to \bot) = (\bot \to \bot) = \top$.

Now, we define stable models.

Definition 5.6 *Let F be a formula. A set X of atoms is a* stable model *of a formula F if X is a minimal model of F.*

One can verify that stable models of a program P (according to the original definition) coincide with stable models (according to this definition) of the representation of P as the conjunction of the implications corresponding to its rules: $\bigwedge\{cmpl^{\leftarrow}(r)\colon r \in P\}$. Thus, the language of propositional formulas can be regarded as a generalization of logic programs (assuming for each formalism we use the corresponding stable-model semantics).

We note the following properties of the formalism of general logic programs that extend the familiar properties we discussed earlier. First, we define an atom a to be a *head* atom of a formula F if at least one occurrence of a in F is strictly positive.

Theorem 5.22 *If X is a stable model of a formula F then X consists of head atoms of F.*

Theorem 5.23 *A Horn theory (conjunction of definite Horn clauses given as implications) F has a unique stable model. It is the least model of F.*

Formulas of the form $\neg F$ are *constraints*.

Theorem 5.24 *A set X is a stable model of a formula $F \wedge \neg G$ if and only if X is a stable model of F and $X \models \neg G$.*

We say that a formula F is strongly equivalent to a formula F' if for every formula G, $F \wedge G$ and $F' \wedge G$ have the same stable models.

We say that (X, Y) is an *se-model* of F if $Y \subseteq At$, $X \subseteq Y$, $Y \models F$ and $X \models F^Y$.

Theorem 5.25 *The following conditions are equivalent:*

1. *Formulas F and G are strongly equivalent*
2. *For every set X of atoms, F^X and G^X are equivalent in classical logic*
3. *F and G have the same se-models*
4. *F and G are equivalent in the logic here-and-there (details later)*

Finally, we mention a generalization of the splitting theorem.

Theorem 5.26 *Let F and G be formulas such that F does not contain any of the head atoms of G. A set X is a stable model of $F \wedge G$ if and only if there is a stable model Y of F such that X is a stable model of $G \wedge \bigwedge Y$.*

6 Modal Logics and Modal Nonmonotonic Logic S4F

In this section, we follow [Tru07]. Our overall goal in this section is to show that nonmonotonic modal logic S4F can be regarded as central to nomonotonic reasoning. We start with a brief overview of basic concepts related to modal logics and modal nonmonotonic logics.

We refer to [HC84, MT93a] for a detailed discussion of topics related to modal logics and a general discussion of modal nonmonotonic logics. We consider the propositional modal language determined by a set At (possibly infinite) of propositional atoms, a constant $\bot$, the usual boolean connectives $\neg$, $\vee$, $\wedge$, $\rightarrow$, and a single modal operator K. The constant $\bot$ represents a "generic" *contradiction* and K is read as "known". An inductive definition of a formula, given in the BNF notation, is as follows:

$$\varphi ::= \bot \,|\, p \,|\, K\varphi \,|\, \neg\varphi \,|\, \varphi \vee \varphi \,|\, \varphi \wedge \varphi \,|\, \varphi \rightarrow \varphi,$$

where $p \in At$. We denote the language consisting of such formulas by $\mathcal{L}_K$ (we drop references to At from the notation as it is fixed). We write $\mathcal{L}$ for the set of K-free (modal-free) formulas in $\mathcal{L}_K$.

Modal logics differ in the properties of the modality K. A modal logic $\mathcal{S}$ is defined semantically by its *entailment* relation $\models_{\mathcal{S}}$, specified by *Kripke interpretations*, or proof-theoretically by means of proofs based on a set of modal axioms of $\mathcal{S}$.

Let $\mathcal{S}$ be a modal logic with the entailment relation $\models_{\mathcal{S}}$. For every theory $I \subseteq \mathcal{L}_K$, a theory $T \subseteq \mathcal{L}_K$ is an *$\mathcal{S}$-expansion* of I [MD80, McD82] if

$$T = \{\varphi \in \mathcal{L}_K \colon I \cup \neg K\overline{T} \models_{\mathcal{S}} \varphi\},$$

where $\neg K\overline{T} = \{\neg K\varphi \colon \varphi \in \mathcal{L}_K \setminus T\}$. The *nonmonotonic logic* $\mathcal{S}$ is a formalism, in which the semantics of a theory $I \subseteq \mathcal{L}_K$ is given by its $\mathcal{S}$-expansions. This definition coincides with the one we gave earlier, which relied on the equivalent proof-theoretic presentation of modal logics that used the consequence operator $Cn_{\mathcal{S}}$.

If $A \subseteq \mathcal{L}$ is a propositional theory then A has a *unique* S5-expansion, where S5 is a well-known modal logic whose entailment relation is given by Kripke interpretations with the universal accessibility relation. According to Proposition 4.2, this unique expansion is given by

$$Cn_{\mathrm{S5}}(A \cup \{\neg K\varphi \colon \varphi \in \mathcal{L} \setminus A\})$$

We will denote this unique expansion by $ST(A)$[1]. We use this notation in the following result, which gives us a way to represent expansions.

Theorem 6.1 *If $\mathcal{S}$ is a modal logic contained in S5 and $A \subseteq \mathcal{L}$, then $ST(A)$ is the unique $\mathcal{S}$-expansion of A. Moreover, for every $T \subseteq \mathcal{L}_K$, if E is an $\mathcal{S}$-expansion of T, then $E = ST(A)$, where $A = E \cap \mathcal{L}$.*

6.1 Logic S4F

The modal logic S4F is fundamental to nonmonotonic reasoning [Seg71, Voo91, MT93a, ST94]. The nonmonotonic logic S4F captures, under some direct and intuitive encodings [Tru91a, ST94], the (disjunctive) logic programming with the stable-model (answer-set) semantics [GL91], the (disjunctive) default logic [Rei80, GLPT91], the logic of grounded knowledge [LS90], the logic of minimal belief and negation as failure [Lif94] and the logic of minimal knowledge and belief [ST94].

The logic S4F is a modal logic with the semantics given by *Kripke* S4F*-interpretations* (or simply, S4F*-interpretations*), that is, tuples $\langle V, W, \pi \rangle$, where

1. V and W are *nonempty* and disjoint sets of *worlds*, and
2. π is a function assigning to each world $w \in V \cup W$ a set of atoms $\pi(w)$, representing a *propositional* truth valuation for w.

Given an S4F-interpretation $\mathcal{M} = \langle V, W, \pi \rangle$, we define the *satisfaction relation* $\mathcal{M}, w \models \varphi$, where $w \in V \cup W$ and $\varphi \in \mathcal{L}_K$, as follows:

1. $\mathcal{M}, w \not\models \bot$
2. $\mathcal{M}, w \models p$ if $p \in \pi(w)$ (for $p \in At$)
3. If $w \in V$, then $\mathcal{M}, w \models K\varphi$ if $\mathcal{M}, v \models \varphi$ for every $v \in V \cup W$
4. If $w \in W$, then $\mathcal{M}, w \models K\varphi$ if $\mathcal{M}, v \models \varphi$ for every $v \in W$
5. The induction over boolean connectives is standard. For instance, $\mathcal{M}, w \models \varphi \wedge \psi$ if $\mathcal{M}, w \models \varphi$ and $\mathcal{M}, w \models \psi$.

An S4F-interpretation $\mathcal{M} = \langle V, W, \pi \rangle$ is an S4F-model of $\varphi \in \mathcal{L}_K$, written $\mathcal{M} \models \varphi$, if for every $w \in V \cup W$, $\mathcal{M}, w \models \varphi$. We write $\varphi \models_{\mathrm{S4F}} \psi$ if every S4F-model of φ is an S4F-model of ψ. The notation extends in a standard way to *modal theories*, that is, subsets of $\mathcal{L}_K$. We note that $\models_{\mathrm{S4F}}$ has a proof-theoretic characterization based on the necessitation inference rule and axiom schemata K, T, 4 and F [Seg71, MT93a].

From now on we focus on the nonmonotonic logic S4F. We start with a result characterizing S4F-expansions (a slight restatement of a result from [Sch92b]). For an S4F-interpretation $\mathcal{M} = \langle V, W, \pi \rangle$, we write $L_{\mathcal{M}}$ ($U_{\mathcal{M}}$) for the set of all formulas from $\mathcal{L}$ (propositional formulas) that hold in every truth assignment $\pi(v)$, where $v \in V$ ($\pi(w)$, where $w \in W$, respectively).

[1]The notation reflects the fact that expansions are *stable* theories [Sta80, McD82], cf. also our earlier discussion of autoepistemic logic.

Theorem 6.2 *Let $I \subseteq \mathcal{L}_K$. A theory $T \subseteq \mathcal{L}_K$ is an* S4F*-expansion of I if and only if there is an* S4F*-model $\mathcal{M}$ of I such that $T = ST(U_\mathcal{M})$; $L_\mathcal{M} = U_\mathcal{M}$; and for every* S4F*-model $\mathcal{N}$ of I with $U_\mathcal{N} = U_\mathcal{M}$, $U_\mathcal{N} \subseteq L_\mathcal{N}$.*

6.2 Modal defaults

We will now use the nonmonotonic logic S4F to generalize default logic.

A *modal default* is defined inductively (in the BNF notation) as:

$$\varphi ::= K\psi \mid K\varphi \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \varphi \rightarrow \varphi,$$

where $\psi \in \mathcal{L}$. Informally, modal defaults are built according to standard rules for boolean connectives and K of formulas $K\psi$, where $\psi \in \mathcal{L}$. A *modal default theory* is set of modal defaults.

When we restrict to modal defaults and theories the semantics simplifies. An S4F*-pair* is a pair $\langle L, U\rangle$, where $L, U \subseteq \mathcal{L}$ are propositional theories closed under propositional entailment.

For an S4F*-pair* $\langle L, U\rangle$ and a modal default φ, we define two satisfiability relations $\langle L, U\rangle \models_l \varphi$ and $\langle L, U\rangle \models_u \varphi$ inductively as follows:

1. For $\varphi = K\psi$, where $\psi \in \mathcal{L}$, we define $\langle L, U\rangle \models_u \varphi$ if $\psi \in U$; and we define $\langle L, U\rangle \models_l \varphi$ if $\psi \in L \cap U$
2. We handle boolean connectives in the standard way. For instance, for $\varphi = \neg\psi$, where ψ is a modal default, we define $\langle L, U\rangle \models_u \varphi$ if $\langle L, U\rangle \not\models_u \psi$; and $\langle L, U\rangle \models_l \varphi$ if $\langle L, U\rangle \not\models_l \psi$
3. For $\varphi = K\psi$, where ψ is a modal default, we define $\langle L, U\rangle \models_u \varphi$ if $\langle L, U\rangle \models_u \psi$; and we define $\langle L, U\rangle \models_l \varphi$ if $\langle L, U\rangle \models_l \psi$ and $\langle L, U\rangle \models_u \psi$

We write $\langle L, U\rangle \models_{md} \varphi$ if $\langle L, U\rangle \models_l \varphi$ and $\langle L, U\rangle \models_u \varphi$.

If $\mathcal{M}$ is an S4F-interpretation then $\langle L_\mathcal{M}, U_\mathcal{M}\rangle$ is an S4F-pair. Also, for every S4F-pair $\langle L, U\rangle$ there is an S4F-interpretation $\mathcal{M} = \langle V, W.\pi\rangle$ such that $L_\mathcal{M} = L$ and $U_\mathcal{M} = U$.

There is a close connection between $\models_{\mathrm{S4F}}$ and the relations $\models_l$, $\models_u$ and $\models_{md}$.

Proposition 6.3 *Let $\mathcal{M} = \langle V, W, \pi\rangle$ be an* S4F*-interpretation and φ a modal default.*

1. *For every $v \in V$:*
   $\mathcal{M}, v \models \varphi$ if and only if $\langle L_\mathcal{M}, U_\mathcal{M}\rangle \models_l \varphi$
2. *For every $w \in W$:*
   $\mathcal{M}, w \models \varphi$ if and only if $\langle L_\mathcal{M}, U_\mathcal{M}\rangle \models_u \varphi$
3. *$\mathcal{M} \models \varphi$ if and only if $\langle L_\mathcal{M}, U_\mathcal{M}\rangle \models_{md} \varphi$.*

And now we have a simpler characterization of S4F-expansions that works for modal default theories.

Theorem 6.4 *Let $I \subseteq \mathcal{L}_K$ be a modal default theory. A theory $T \subseteq \mathcal{L}_K$ is an* S4F*-expansion of I if and only if there is $U \subseteq \mathcal{L}$ such that U is closed under propositional entailment, $T = ST(U)$, $\langle U, U\rangle \models_{md} I$, and for every* S4F*-pair $\langle L, U\rangle$, $\langle L, U\rangle \models_{md} I$ implies $U \subseteq L$.*

An *se-interpretation* is an S4F-pair $\langle L, U\rangle$ such that $L \subseteq U$. If I is a modal default theory then an se-interpretation $\langle L, U\rangle$ such that $\langle L, U\rangle \models I$ is an *se-model* of I. It turns out that that for modal default theories se-interpretations suffice to characterize the S4F-entailment. For a modal default theory I and a modal default φ we write $I \models_{se} \varphi$ if every se-model of I is an se-model of φ.

Theorem 6.5 *Let $I \subseteq \mathcal{L}_K$ be a modal default theory and let $\varphi \in \mathcal{L}_K$ be a modal default. Then $I \models_{\mathrm{S4F}} \varphi$ if and only if $I \models_{se} \varphi$.*

Corollary 6.6 *Let $I \subseteq \mathcal{L}_K$ be a modal default theory. A theory $T \subseteq \mathcal{L}_K$ is an* S4F*-expansion of I if and only if there is $U \subseteq \mathcal{L}$ such that U is closed under propositional entailment, $T = ST(U)$, $\langle U, U\rangle$ is an se-model for I, and for every se-model $\langle L, U\rangle$ for I, $U = L$.*

Let $I \subseteq \mathcal{L}_K$. An se-interpretation $\langle U, U\rangle$ is an *se-expansion* of I if $\langle U, U\rangle \models I$ and for every se-model $\langle L, U\rangle$ of I, $L = U$. Our results show that there is a one-to-one correspondence between S4F-expansions and se-expansions.

Corollary 6.7 *Let $I \subseteq \mathcal{L}_K$ be a modal default theory. A theory $T \subseteq \mathcal{L}_K$ is an* S4F*-expansion of I if and only if there is an se-expansion $\langle U, U\rangle$ of I such that $T = ST(U)$.*

6.3 Strong equivalence for the nonmonotonic logic S4F

We will now characterize strong equivalence of modal default theories I and I' (modal default theories I, I' are *strongly equivalent* if for every modal default theory J, $I \cup J$ and $I' \cup J$ have the same S4F-expansions, or equivalently, the same se-expansions.

Theorem 6.8 *Let $I, I' \subseteq \mathcal{L}_K$ be modal default theories. The following conditions are equivalent:*

1. *I and I' are strongly equivalent*
2. *I and I' have the same se-models.*

6.4 Uniform equivalence for the nonmonotonic logic S4F

We will use the following notation for a set X of (modal) formulas: $KX = \{K\varphi\colon \varphi \in X\}$. Let $P, Q \subseteq \mathcal{L}_K$ be modal theories. We say that P and Q are *uniformly equivalent* if for every set $X \subseteq \mathcal{L}$, $P \cup KX$ and $Q \cup KX$ have the same S4F-expansions. If P and Q are modal default theories then for every $X \subseteq \mathcal{L}$, $P \cup KX$ and $Q \cup KX$ are modal default theories, too. It turns out that se-interpretations can also be used to characterize uniform equivalence. Namely, we have the following theorem.

Theorem 6.9 *Default modal theories $P, Q \subseteq \mathcal{L}_K$ are uniformly equivalent if and only if the following three conditions hold:*

1. *for every se-interpretation $\langle U, U\rangle$, $\langle U, U\rangle \models_{md} P$ if and only if $\langle U, U\rangle \models_{md} Q$*
2. *for every se-interpretation $\langle L, U\rangle$, if $L \subset U$ and $\langle L, U\rangle \models_{md} P$ then there is an se-interpretation $\langle L', U\rangle$ such that $L \subseteq L' \subset U$ and $\langle L', U\rangle \models_{md} Q$*

3. *for every se-interpretation $\langle L, U\rangle$, if $L \subset U$ and $\langle L, U\rangle \models_{md} Q$ then there is an se-interpretation $\langle L', U\rangle$ such that $L \subseteq L' \subset U$ and $\langle L', U\rangle \models_{md} P$*

As in the case of logic programs, we can introduce the notion of a *ue-model* and, derive from Theorem 6.9 a simpler characterization of uniform equivalence of *finite* modal default theories in terms of ue-models.

6.5 Modal programs

We will now consider *modal programs*, a special class of modal default theories consisting of *modal rules*. As it will become clear later, modal programs are equivalent to general logic programs. The two formalisms can be viewed as notational invariants of one another.

A formal inductive definition of a *modal rule*, given in the BNF notation, is as follows:

$$\varphi ::= Kp \,|\, K\varphi \,|\, \neg\varphi \,|\, \varphi \vee \varphi \,|\, \varphi \wedge \varphi \,|\, \varphi \rightarrow \varphi,$$

where $p \in At \cup \{\bot\}$. A *modal program* is a set of modal rules.

If $X \subseteq \mathcal{L}$, we write $Cn(X)$ for the set of all propositional consequences of X. A *simple se-interpretation* is any se-interpretation of the form $\langle Cn(L), Cn(U)\rangle$, where $L, U \subseteq At$. To characterize S4F-expansions of modal programs it suffices to restrict to simple se-interpretations. Indeed, the following results state the key property of modal rules.

Theorem 6.10 *If φ is a modal rule and $\langle L, U\rangle$ is an se-interpretation then $\langle L, U\rangle \models \varphi$ if and only if $\langle Cn(L \cap At), Cn(U \cap At)\rangle \models \varphi$.*

Corollary 6.11 *If I and I' are modal programs, then I and I' have the same se-models if and only if they have the same simple se-models.*

These results allow us to strengthen the characterization of strong equivalence in the case of modal programs.

Corollary 6.12 *Let $I, I' \subseteq \mathcal{L}_K$ be modal programs. The following conditions are equivalent:*

1. *I and I' are strongly equivalent*
2. *I and I' have the same simple se-models.*

Using a similar argument we can also strengthen the characterization of the uniform equivalence of modal programs (Theorem 6.9) by consistently replacing se-interpretations with simple se-interpretations.

6.6 The Logic Here-and-There

There is a strong connection between simple se-interpretations and models of non-classical propositional logic known as the logic *here-and-there* [Hey30, Pea97, FL05].

We will introduce the logic here-and-there and mention its connection to general logic programs. We will also note that it can be embedded in the logic S4F.

The language of the logic here-and-there has three primitive binary connectives $\wedge$, $\vee$ and $\rightarrow$, and a constant $\bot$ to represent a generic contradiction (thus, it is the same as the language of general logic programs). And we use the same shorthands as before.

The semantics of the logic here-and-there is given by HT-interpretations. An *HT-interpretation* is a pair $\langle L, U\rangle$, where $L \subseteq U \subseteq At$ are sets of atoms. We define the satisfiability relation $\langle L, U\rangle \models_{ht} \varphi$, where $\varphi \in \mathcal{L}_{ht}$, by induction as follows:

1. $\langle L, U\rangle \not\models \bot$
2. For $\varphi = p$, where $p \in At$, we define $\langle L, U\rangle \models_{ht} p$ if $p \in L$
3. $\langle L, U\rangle \models_{ht} \varphi \wedge \psi$ if $\langle L, U\rangle \models_{ht} \varphi$ and $\langle L, U\rangle \models_{ht} \psi$
4. $\langle L, U\rangle \models_{ht} \varphi \vee \psi$ if $\langle L, U\rangle \models_{ht} \varphi$ or $\langle L, U\rangle \models_{ht} \psi$
5. $\langle L, U\rangle \models_{ht} \varphi \rightarrow \psi$ if (i) $\langle L, U\rangle \not\models_{ht} \varphi$ or $\langle L, U\rangle \models_{ht} \psi$; and (ii) $U \models \varphi \rightarrow \psi$ (in standard propositional logic).

An HT-interpretation $\langle U, U\rangle$ is an *equilibrium model* of $A \subseteq \mathcal{L}_{ht}$ if $\langle U, U\rangle \models_{ht} A$ and for every $L \subseteq U$, if $\langle L, U\rangle \models_{ht} A$ then $L = U$ [Pea97]. It is easy to see that equilibrium models correspond to *stable* models of general logic programs [FL05] (cf. our earlier discussion). This, there is a direct connection between the two formalisms.

We will now show that the logic here-and-there can be embedded in the logic S4F. To this end, for every propositional formula $\varphi \in \mathcal{L}_{ht}$ we define a formula $\varphi_{\neg K\neg K}$ to be a modal rule obtained from φ by replacing each $a \in At \cup \{\bot\}$ in φ with $\neg K\neg Ka$ (intuitively, $\neg K\neg K$ represents a modality exhibiting properties of the *belief* modality). We note that all formulas $\varphi_{\neg K\neg K}$ are modal rules.

Next, for every propositional formula $\varphi \in \mathcal{L}_{ht}$ we define the corresponding modal rule φ_{mp} inductively as follows:

1. $a_{mp} = Ka$ for $a \in At \cup \{\bot\}$
2. $(\varphi \wedge \psi)_{mp} = \varphi_{mp} \wedge \psi_{mp}$ and $(\varphi \vee \psi)_{mp} = \varphi_{mp} \vee \psi_{mp}$
3. $(\varphi \rightarrow \psi)_{mp} = (\varphi_{mp} \rightarrow \psi_{mp}) \wedge (\psi \rightarrow \varphi)_{\neg K\neg K}$.

We extend this notation to sets of formulas: for a set $A \subseteq \mathcal{L}_{ht}$, we define $A_{mp} = \{\varphi_{mp} | \varphi \in A\}$.

We have the following result establishing the connection between the logic here-and-there and the logic S4F.

Theorem 6.13 *Let $A \subseteq \mathcal{L}_{ht}$ and $\varphi \in \mathcal{L}_{ht}$. The following conditions are equivalent:*

1. $A \models_{ht} \varphi$
2. $A_{mp} \models_{se} \varphi_{mp}$
3. $A_{mp} \models_{\mathrm{S4F}} \varphi_{mp}$.

Corollary 6.14 *Let $A \subseteq \mathcal{L}_{ht}$ and $U \subseteq At$. The following conditions are equivalent:*

1. *U is a stable model of A*
2. *$\langle U, U\rangle$ is an equilibrium model of A*
3. *$\langle Cn(U), Cn(U)\rangle$ is an se-expansion of A*
4. *$ST(U)$ is an $\mathrm{S4F}$-expansion of A_{mp}.*

6.7 Logic of nested defaults

Let $\frac{\alpha:\ \beta_1,\ldots,\beta_m}{\gamma_1|\ldots|\gamma_n}$ be a disjunctive default [GLPT91]. By encoding it with a modal default

$$K\alpha \wedge K\neg K\beta_1 \wedge \ldots \wedge K\neg K\beta_k \rightarrow K\gamma_1 \vee \ldots \vee K\gamma_n$$

we obtain an embedding of (disjunctive) default theories in $\mathcal{L}_K$ which establishes a one-to-one correspondence between extensions and $\mathrm{S4F}$-expansions [Tru91a]. Thus, the class of modal default theories with the semantics of $\mathrm{S4F}$-expansions (or se-expansions) can be regarded as a generalization of the disjunctive default logic. In fact, we can regard it as a general default logic of nested defaults as it covers, for instance, the case of formulas of the form

$$K\alpha \wedge K\neg K\beta_1 \wedge \ldots \wedge K\neg K\beta_k \rightarrow K\gamma_1 \vee \ldots \vee K\gamma_n$$

where α, β_i and γ_i are arbitrary modal defaults rather than formulas from $\mathcal{L}$.

We also note that by exploiting the embedding given above, our results on strong equivalence of modal default theories can be specialized to results on strong equivalence of disjunctive default theories, first obtained by Turner [Tur03]. Our results on uniform equivalence of modal default theories generalize those obtained by Truszczyński [Tru06].

7 Algebraic approach to nonmonotonic reasoning

This section closely follows [DMT00a],

7.1 Preliminaries from lattice theory

A *lattice* is a partially ordered set $\langle L, \leq\rangle$ such that every two element set $\{x, y\} \subseteq L$ has a *least upper bound*, $lub(x, y)$, and a *greatest lower bound*, $glb(x, y)$. A lattice $\langle L, \leq\rangle$ is *complete* if every subset of L has both least upper and greatest lower bounds. Consequently, a complete lattice has a least element ($\bot$) and a greatest element ($\top$).

Earlier, we discussed operators on families of sets. Now, we generalize the discussion to a more abstract setting. If $\langle L, \leq\rangle$ is a lattice, functions from L to L are *operators* on the lattice. An operator O on L is *monotone* if for every $x, y \in L$, $x \leq y$ implies $O(x) \leq O(y)$. An operator O on L is *antimonotone* if for every $x \leq y$, $O(y) \leq O(x)$.

We have the following two simple properties.

Proposition 7.1 *If the operators $O_1 : L \to L$, $O_2 : L \to L$ are antimonotone, then the operator $O_1 \circ O_2$ is monotone.*

Proposition 7.2 *If an operator $O : L \to L$ is monotone and antimonotone then it is constant.*

Tarski-Knaster theorem [Tar55] also generalizes to the abstract setting of operators on lattices. An element $x \in L$ is a *prefixpoint* of a lattice operator L, if $O(x) \leq x$, and a *fixpoint* of L, if $O(x) = x$.

Theorem 7.3 *Let O be a monotone operator on a complete lattice $\langle L, \leq \rangle$. Then, O has a least prefixpoint and a least fixpoint and the two coincide.*

We denote the least fixpoint of an operator O by $lfp(O)$.

An element $z \in L$ is *approximated* by a pair $(x, y) \in L^2$ if $x \leq z \leq y$. Approximations of the form (x, x) are called *exact*. There is a straightforward one-to-one correspondence between L and the set of exact elements of L^2.

The set L^2 can be given with two orderings. First, the *lattice* ordering $\leq$ is given by

$$(x, y) \leq (x_1, y_1) \; if \; x \leq x_1 \; and \; y \leq y_1.$$

Second, the *information* ordering, $\leq_i$, which captures the intuition of increased precision of the approximation, is given by

$$(x, y) \leq_i (x_1, y_1) \; if \; x \leq x_1 \; and \; y_1 \leq y.$$

With each of these two orderings L^2 is a complete lattice.

A pair $(x, y) \in L^2$ is *consistent* if $x \leq y$. Otherwise, it is called *inconsistent*. Consistent pairs can be viewed as descriptions of, in general, incomplete knowledge about elements from L that they approximate. The information ordering when applied to consistent pairs measures their precision, when applied to inconsistent pairs measures the "degree of inconsistency".

The collection of consistent pairs does not form a sublattice of L^2. Indeed, each element of the form (x, x) is a maximal consistent element of L^2. By allowing inconsistent approximations, we obtain a duality between consistent and inconsistent pairs, and between the degree of precision and the degree of inconsistency. Consequently, we obtain a richer algebraic structure and a more elegant theory.

A pair of elements $x, y \in L$ is an *oscillating pair* for an operator O on L if $y = O(x)$ and $x = O(y)$. An oscillating pair (x, y) is *extreme* if for every oscillating pair (x', y') for O, $(x, y) \leq_i (x', y')$ and $(x, y) \leq_i (y', x')$ (or equivalently, $x \leq x', y' \leq y$). If (x, y) is an extreme oscillating pair then $x \leq y$. Moreover, if an extreme oscillating pair exists, it is unique.

Theorem 7.4 *Let O be an antimonotone operator on a complete lattice $\langle L, \leq \rangle$. Then, O^2 has a least fixpoint and a greatest fixpoint and $(lfp(O^2), O(lfp(O^2)))$ is the unique extreme oscillating pair of O.*

Let A be an operator on L^2. Let us denote by A^1 and A^2 the functions from L^2 to L such that

$$A(x, y) = (A^1(x, y), A^2(x, y)).$$

We say that A is *symmetric* if $A^1(x, y) = A^2(y, x)$. If an operator $A : L^2 \to L^2$ is symmetric then for every $x \in L$, $A^1(x, x) = A^2(x, x)$. In the remainder of this document, we consider symmetric operators only.

We now have several useful characterizations and properties of symmetric operators that are monotone wrt $\leq_i$ and/or $\leq$.

Proposition 7.5 *A symmetric operator $A : L^2 \to L^2$ is $\leq_i$-monotone if and only if for every $y \in L$, $A^1(\cdot, y)$ is monotone and for every $x \in L$, $A^1(x, \cdot)$ is antimonotone (or equivalently, if and only if for every $y \in L$, $A^2(\cdot, y)$ is antimonotone and for every $x \in L$, $A^2(x, \cdot)$ is monotone).*

Proposition 7.6 *A symmetric operator $A : L^2 \to L^2$ is $\leq$-monotone if and only if for every $x, y \in L$, $A^1(x, \cdot)$ and $A^1(\cdot, y)$ are monotone (or, equivalently, if and only if for every $x, y \in L$, $A^2(x, \cdot)$ and $A^2(\cdot, y)$ are monotone).*

Proposition 7.7 *An operator $A : L^2 \to L^2$ is symmetric and monotone with respect to both $\leq_i$ and $\leq$ if and only if there is a monotone operator $O : L \to L$ such that for every $x, y \in L$, $A(x, y) = (O(x), O(y))$.*

Proposition 7.8 *An operator $A : L^2 \to L^2$ is symmetric, $\leq_i$-monotone and $\leq$-antimonotone if and only if there is an antimonotone operator $O : L \to L$ such that for every $x, y \in L$, $A(x, y) = (O(y), O(x))$.*

By Propositions 7.7 and 7.8 there is a one-to-one correspondence between monotone (antimonotone, respectively) operators on L and $\leq_i$-monotone and $\leq$-monotone ($\leq_i$-monotone and $\leq$-antimonotone, respectively) operators on L^2.

When L is a complete lattice, Knaster-Tarski Theorem Theorem 7.4 imply that an $\leq_i$-monotone and $\leq$-antimonotone operator $A : L^2 \to L^2$ has $\leq_i$-least and $\leq_i$-greatest fixpoints and a $\leq$-extreme oscillating pair. Let us denote the $\leq_i$-least fixpoint of A by q_A, and the $\leq_i$-greatest fixpoint of A by Q_A. Similarly, let us denote the $\leq$-extreme oscillating pair for A by (e_A, E_A).

If $A : L^2 \to L^2$ is, in addition, symmetric, by Proposition 7.8, there is an antimonotone operator $O : L \to L$ such that $A(x, y) = (O(y), O(x))$. Let us denote by q the least fixpoint of O^2 and by Q the greatest fixpoint of O^2 (Tarski-Knaster Theorem applies as O^2 is monotone). The following theorem, due essentially to Fitting, summarizes the relations between the fixpoints and extreme pairs defined above.

Theorem 7.9 *Let L be a complete lattice. Let $A : L^2 \to L^2$ be a symmetric $\leq_i$-monotone and $\leq$-antimonotone operator on L^2. Then:*

1. $q_A = (q, Q)$, $Q_A = (Q, q)$, $e_A = (q, q)$, $E_A = (Q, Q)$
2. $q_A = glb_{\leq_i}(e_A, E_A)$ *and* $Q_A = lub_{\leq_i}(e_A, E_A)$
3. $e_A = glb_{\leq}(q_A, Q_A)$ *and* $E_A = lub_{\leq}(q_A, Q_A)$.

7.2 Approximating operators

Definition 7.1 *An operator $A : L^2 \to L^2$* extends *an operator $O : L \to L$ if for every $x \in L$, $A(x, x) = (O(x), O(x))$. An operator $A : L^2 \to L^2$ is* extending *if for every $x \in L$, there is $y \in L$ such that $A(x, x) = (y, y)$.*

The *diagonal* of L^2 is the set $\{(x, x) : x \in L\}$ If an operator $A : L^2 \to L^2$ extends $O : L \to L$ then the behavior of A on the diagonal determines the behavior of O.

Proposition 7.10 *Let O be an operator on a lattice L and let A be an operator on L^2 extending O. Then, x is a fixpoint of O if and only if (x, x) is a fixpoint of A.*

If A is symmetric then for each lattice element x, $A^1(x, x) = A^2(x, x)$. Hence $A(x, x)$ is exact and, consequently, A is extending.

Proposition 7.11 *If an operator $A : L^2 \to L^2$ is symmetric then A is extending.*

To study fixpoints of an operator O one might construct an appropriate extending operator A and study its fixpoints instead. Exact fixpoints of the operator A provide a description of the fixpoints of O. A situation is especially interesting if A is symmetric and $\leq_i$-monotone.

Definition 7.2 *An operator $A : L^2 \to L^2$* approximates *an operator $O : L \to L$ if A is symmetric, extends O and is $\leq_i$-monotone. An operator $A : L^2 \to L^2$ is* approximating *if it is symmetric and $\leq_i$-monotone.*

We say that an operator $A : L^2 \to L^2$ is *consistent* if it maps consistent pairs to consistent pairs.

Proposition 7.12 *If $A : L^2 \to L^2$ is an approximating operator, then A is consistent.*

Corollary 7.13 *Let $A : L^2 \to L^2$ be an approximating operator for an operator $O : L \to L$. Then, A has a $\leq_i$-least fixpoint. This fixpoint is consistent and approximates every fixpoint of O.*

If the $\leq_i$-least fixpoint of an approximating operator A for an operator O is exact, say of the form (x, x), then x is the only fixpoint of O. Since in the case of logic programming, the concept of the $\leq_i$-least fixpoint of an approximating operator to the operator T_P gives Kripke-Kleene semantics, we refer to the $\leq_i$-least fixpoint of an approximating operator A as the *Kripke-Kleene fixpoint* of A. We denote this fixpoint by α_A.

Let O be a monotone operator on L. By Proposition 7.7, the operator $A_O(x, y) = (O(x), O(y))$ is $\leq_i$-monotone. It is also symmetric, consistent and extends the operator O. Hence, A_O is an approximating operator for O. By Proposition 7.7, A_O is $\leq$-monotone. In fact, Proposition 7.7 implies that A_O is a unique approximating operator for O that is $\leq$-monotone. The least $\leq_i$-fixpoint of A_O is $(lfp(O), lfp(O))$. We will call A_O the *canonical* approximating operator for a monotone operator O. This algebraic property of monotone operators explains why all major nonmonotonic semantics coincide on the class of Horn theories (or programs) and are given by the least fixpoint construction.

Similarly, if A is an antimonotone operator on L then, by Proposition 7.8, the operator $A_O(x, y) = (O(y), O(x))$ is $\leq_i$-monotone. In addition, A_O is symmetric, consistent and it extends O. Hence, it is an approximating operator for O. By Proposition 7.8, A_O is $\leq$-antimonotone and, in fact, it is a unique approximating operator for O that is $\leq$-antimonotone. We will call A_O the *canonical* approximating operator for an antimonotone operator O. Theorem 7.9 characterizes the fixpoints and the extreme oscillating pair of the trivial approximating operator for an antimonotone operator O.

7.3 Stable operator and well-founded fixpoint

In this section we describe an algebraic construction that assigns to every $\leq_i$-monotone operator A on a bilattice L^2 its *stable* operator $\mathcal{C}_A$ defined also on L^2.

Definition 7.3 *Let L be a complete lattice. Let an operator $A : L^2 \to L^2$ on a bilattice L^2 be symmetric and $\leq_i$-monotone.*

1. *The* complete stable operator for A, $C_A : L \to L$, *is defined by $C_A(y) = lfp(A^1(\cdot, y))$ (or, equivalently, by, $C_A(y) = lfp(A^2(y, \cdot))$).*
2. *The* stable operator for A, $\mathcal{C}_A : L^2 \to L^2$ *is defined by $\mathcal{C}_A(x, y) = (C_A(y), C_A(x))$.*

Since for every $y \in L$ the operators $A^1(\cdot, y)$ and $A^2(y, \cdot)$ are monotone (Proposition 7.5), the operators C_A and $\mathcal{C}_A$ are well-defined.

Let us consider an operator A that is both $\leq_i$- and $\leq$-monotone. Such operators are described in Proposition 7.7. They are of the form $A(x, y) = (O(x), O(y))$, where O is monotone. It follows that $C_A(y) = lfp(O)$ and does not depend on y. Thus, we get the following result.

Proposition 7.14 *Let L be a complete lattice. Let $A : L^2 \to L^2$ be an operator monotone with respect to $\leq_i$ and $\leq$. Then $\mathcal{C}_A$ is constant.*

If an operator A is $\leq_i$-monotone and $\leq$-antimonotone then, by Proposition 7.8, there is an antimonotone operator O such that $A(x, y) = (O(y), O(x))$. Consequently, $A(\cdot, y) = O(y)$. It follows that $C_A(y) = O(y)$, that is, the stable operator for the operator A is A itself.

Proposition 7.15 *Let L be a complete lattice. Let $A : L^2 \to L^2$ be an operator monotone with respect to $\leq_i$ and antimonotone with respect to $\leq$. Then $\mathcal{C}_A = A$.*

The next several results establish properties of the stable operator $\mathcal{C}_A$ and its fixpoints. Our first result shows that fixpoints of $\mathcal{C}_A$ are $\leq$-minimal fixpoints of A.

Theorem 7.16 *Let L be a complete lattice. Let an operator $A : L^2 \to L^2$ on a bilattice L^2 be $\leq_i$-monotone. Every fixpoint of the stable operator $\mathcal{C}_A$ is a $\leq$-minimal fixpoint of A.*

Theorem 7.16 shows, in particular, that if A is $\leq_i$-monotone, a fixpoint of $\mathcal{C}_A$ is also a fixpoint of A. We will call every fixpoint of the stable operator $\mathcal{C}_A$ a *stable* fixpoint of A.

Directly from the definition of the operators C_A and from Proposition 7.5 it follows that C_A is antimonotone. Consequently, by Proposition 7.8, $\mathcal{C}_A$ is $\leq_i$-monotone and $\leq$-antimonotone.

Proposition 7.17 *Let L be a complete lattice. Let A be a symmetric $\leq_i$-monotone operator on L^2. Then, C_A is an antimonotone operator on L and $\mathcal{C}_A$ is a $\leq_i$-monotone and $\leq$-antimonotone operator on L^2.*

Propositions 7.15 and 7.17 imply the following corollary that states that applying the stability construction to a stable operator does not lead to a new operator anymore.

Corollary 7.18 *Let L be a complete lattice. Let A be a symmetric $\leq_i$-monotone operator on L^2. Then $\mathcal{C}_{\mathcal{C}_A} = \mathcal{C}_A$.*

It is also easy to see that $\mathcal{C}_A$ is symmetric and extends the operator C_A. Thus, we obtain the following corollary to Proposition 7.17.

Corollary 7.19 *Let L be a complete lattice. Let A be a $\leq_i$-monotone operator on L^2. Then, the stable operator $\mathcal{C}_A$ is a trivial approximation of the complete stable operator C_A.*

The $\leq_i$-least fixpoint of $\mathcal{C}_A$ is of particular interest as it provides an approximation to every stable fixpoint of A. We call the $\leq_i$-least fixpoint of $\mathcal{C}_A$ the *well-founded fixpoint* of a $\leq_i$-monotone operator A and denote it by β_A. The choice of the term is dictated by the fact that in the case of logic programming, the least fixpoint of the stable operator for the 4-valued van Emden-Kowalski operator $\mathcal{T}_P$ yields the well-founded semantics.

The following result gathers several properties of the well-founded fixpoint of an operator that generalize properties of the well-founded model of a logic program.

Theorem 7.20 *Let L be a complete lattice. Let $A : L^2 \to L^2$ be a $\leq_i$-monotone symmetric operator.*

1. *The Kripke-Kleene fixpoint α_A and the well-founded fixpoint β_A satisfy $\alpha_A \leq_i \beta_A$*
2. *For every stable fixpoint x of A, $\beta_A \leq_i x$*
3. *If β_A is exact then it is the only consistent stable fixpoint of A.*
4. *The operator $\mathcal{C}_A$ is consistent and, consequently, β_A is consistent, too.*

We will now assume that A is an approximating operator for an operator $O : L \to L$ and discuss the relationship between the fixpoints of $\mathcal{C}_A$ and fixpoints of O.

Proposition 7.21 *Let L be a complete lattice. Let $A\colon L^2 \to L^2$ be an approximating operator for an operator $O\colon L \to L$. If (x, x) is a fixpoint of $\mathcal{C}_A$ then x is a $\leq$-minimal fixpoint of O.*

It follows from Proposition 7.21 that if A is an approximating operator for an operator O then fixpoints of O corresponding to exact fixpoints of the stable operator $\mathcal{C}_A$ form an antichain.

We will next consider the case when O is monotone and use the canonical approximation of O, A_O.

Proposition 7.22 *Let L be a complete lattice. If $O : L \to L$ is a monotone operator, then for every $x \in L$, $\mathcal{C}_{A_O}(x, y) = (lfp(O), lfp(O))$ (that is, C_{A_O} is constant).*

If O is monotone, its canonical approximation A_O may have many fixpoints in general and many exact fixpoints, in particular. However, by Proposition 7.22, the stable operator for A_O has only one fixpoint and it corresponds precisely to the least fixpoint of O. In the context of logic programming, this result says that a Horn logic program P has a unique stable model and that it coincides with the least Herbrand model of P.

7.4 Applications in knowledge representation

The results presented here provide us with a uniform framework for semantic studies of major knowledge representation formalisms: logic programming, autoepistemic logic and default logic. Namely, all major semantics for each of these formalisms can be derived from a single operator.

In the case of logic programming, our results extend an algebraic approach proposed in [Fit02]. The lattice of interest here is that of 2-valued interpretations of the Herbrand base of a given program P. We will denote it by $\mathcal{A}_2$. The corresponding bilattice $\mathcal{A}_2 \times \mathcal{A}_2$ is isomorphic with the bilattice $\mathcal{A}_4$ of 4-valued interpretations (in 4-valued Belnap logic). Our results imply that the central role in logic programming is played by the 4-valued van Emden-Kowalski operator $\mathcal{T}_P$ defined on the bilattice $\mathcal{A}_2 \times \mathcal{A}_2$ (or, equivalently, on bilattice $\mathcal{A}_4$). First, the operator $\mathcal{T}_P$ approximates the 2-valued van Emden-Kowalski operator T_P. Second, fixpoints of $\mathcal{T}_P$ represent 4-valued supported models, consistent fixpoints of $\mathcal{T}_P$ represent partial (3-valued) supported models and exact fixpoints of $\mathcal{T}_P$ describe supported models of P. The $\leq_i$-least fixpoint of $\mathcal{T}_P$ (it exists as $\mathcal{T}_P$ is approximating) defines the Kripke-Kleene semantics of P.

Perhaps most importantly, it turns out that our general construction assigning the stable operator to every approximating operator when applied to $\mathcal{T}_P$ yields the 4-valued Przymusinski operator Ψ'_P and the 2-valued Gelfond-Lifschitz operator GL_P. That is, the stable operator for $\mathcal{T}_P$ coincides with Ψ'_P and the complete stable operator for $\mathcal{T}_P$ coincides with GL_P. Thus, the semantics of 4-valued, partial (3-valued) and 2-valued stable models can also be derived from the operator $\mathcal{T}_P$. The same is true for the well-founded semantics since it is determined by the $\leq_i$-least fixpoint of the stable operator of $\mathcal{T}_P$. The structure of the family of operators and semantics for logic programming that can be derived from the operator $\mathcal{T}_P$ is presented in Figure 1.

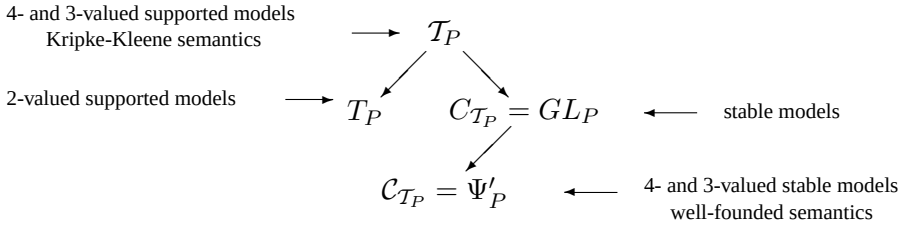

Figure 1: Operators and semantics associated with logic programming

In [DMT98, DMT00b] we developed an algebraic approach to semantics for autoepistemic and default logics. In both cases, our approach can be regarded as a special case of the general approach presented here. In the investigations of autoepistemic and default logics we consider the lattice $\mathcal{W}$ of possible-world structures (sets of 2-valued interpretations) and the corresponding bilattice $\mathcal{B}$ of belief pairs [DMT98]. In the case of autoepistemic logic, the central place is occupied by the operator $\mathcal{D}_T$ (T is a given modal theory) defined on the bilattice of belief pairs and introduced in [DMT98]. It turns out to be an approximating operator for the operator D_T used by Moore to define the notion of an expansion [Moo84]. Thus, the concepts of partial expansions and expansions can be derived from $\mathcal{D}_T$. Similarly, the Kripke-Kleene semantics can be obtained from $\mathcal{D}_T$ as its least fixpoint. The stable operator for $\mathcal{D}_T$ and its complete counterpart lead to semantics for autoepistemic logic that to the best of our knowledge have not been studied in the literature: the semantics of extensions, partial extensions and the well-founded semantics, that are closely related to the corresponding semantics for default logic [DMT00b]. The emerging structure of operators and semantics for autoepistemic logic is depicted in

Figure 2.

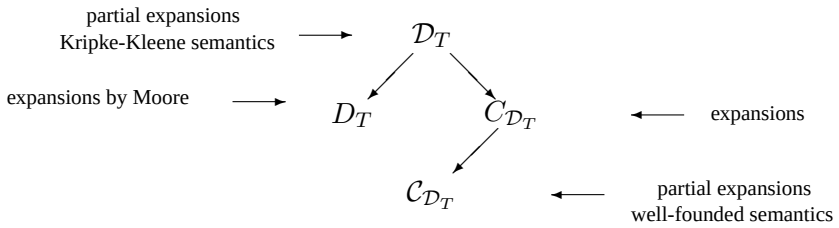

Figure 2: Operators and semantics associated with autoepistemic logic

A very similar picture emerges in the case of default logic, too. In [DMT00b] we described an operator $\mathcal{E}_\Delta$ on the bilattice of belief pairs and argued that all major semantics for default logic can be derived from it. Among them are the semantics of weak extensions [MT89a], partial weak extensions and the corresponding Kripke-Kleene semantics for default logic. In addition, the complete stable operator for $\mathcal{E}_\Delta$ coincides with the Guerreiro-Casanova operator characterizing extensions [GC90] and the $\leq_i$-least fixpoint of the stable operator $\mathcal{C}_{\mathcal{E}_\Delta}$ for $\mathcal{E}_\Delta$ yields the well-founded semantics for default logic described by Baral and Subrahmanian in [BS91]. The semantics landscape of default logic is depicted in Figure 3.

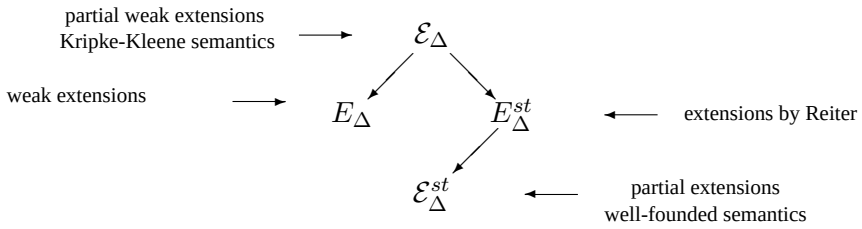

Figure 3: Operators and semantics associated with default logic

The similarity between the families of the semantics for logic programming, default logic and autoepistemic logic is striking. It has been long known that logic program rules can be interpreted as default rules [MT89b, BF91]. Namely, a logic program rule

$$a \leftarrow b_1, \ldots, b_m, \mathit{not}\ c_1, \ldots, \mathit{not}\ c_n$$

can be interpreted as a default

$$\frac{b_1 \wedge \ldots \wedge b_m : \neg c_1, \ldots, \neg c_n}{a}$$

It turns out that under this translation the operators $\mathcal{T}_P$ and $\mathcal{E}_{\Delta(P)}$ are very closely related ($\Delta(P)$ stands for the default theory obtained from the logic program P by means of the translation given above). Namely, let us observe that each interpretation I can be associated with the possible-world structure consisting of all interpretations J such that $I(p) = \mathbf{t}$ implies $J(p) = \mathbf{t}$. Thus, the lattice $\mathcal{A}_2$ can be viewed as a sublattice of $\mathcal{W}$ and the restriction of the operator $\mathcal{E}_{\Delta(P)}$ to this sublattice essentially coincides with $\mathcal{T}_P$. It follows that all the derived operators are similarly related, and we obtain a perfect match between the semantics for logic programming and the semantics for default logic.

Similarly, in [Kon88] it was proposed to interpret a default

$$\frac{\beta_1 \wedge \ldots \wedge \beta_m : \neg\gamma_1, \ldots, \neg\gamma_n}{\alpha}$$

as a modal formula

$$K\beta_1 \wedge \ldots K\beta_m \wedge \neg K\neg\gamma_1 \wedge \ldots \wedge \neg K\neg\gamma_n \supset \alpha.$$

It turns out that under this translations the operators $\mathcal{E}_\Delta$ and $\mathcal{D}_{T(\Delta)}$ coincide (here $T(\Delta)$ is the modal image of a default theory Δ under Konolige's translation). As before, all corresponding pairs of derived operators also coincide. Thus, we obtain a perfect match between the semantics for default and autoepistemic theories[2].

8 Bibliographical and Historical Comments

The area of nonmonotonic reasoning and logic programming was established about 20 years ago. Several research monographs devoted to the subject are now available [Eth88, Bes89, Bre91b, MT93a, Ant97, BDK97, Boc01, Boc05, Mak05]. The area has also its two conference series: Nonmonotonic Reasoning Workshops (NMR workshops) (`http://www.kr.org/NMR/`), and *International Conference on Logic Programming and Nonmonotonic Reasoning*, or LPNMR conferences. NMR meetings are always collocated with Knowledge Representation and Reasoning (KR) conferences (`http://www.kr.org`). It does not have formal proceedings but all papers since NMR-2000 are available in electronic form. LPNMR is a biannual event started in 1991, with proceedings now published by Springer.

Our presentation of default logic follows that in [MT93a]. It differs only slightly from the original approach by Reiter [Rei80]. Perhaps the most significant difference is the emphasis we put on proof-theoretic techniques in our study of default logic. Most of the proofs of the results we presented here can be found in [MT93a].

Basic properties of default logic have been established by the founder of default logic, Raymond Reiter, in [Rei80]. In particular the operator Γ is introduced there. A characterization of default extensions in terms of sets of possible worlds is due to Guereiro and Casanova [GC90]. Characterizations of default extensions in terms of modal nonmonotonic systems are also known. We refer to [Tru91b, Tru91a, DMT00a] for more details. Extensions and variants of default logic are discussed in [GLPT91, Bre91a, Sch92a, MT93b, BDK97].

The complexity results for default logic are due to Gottlob [Got92] and, independently, Stillman [Sti92]. Basic algorithms for computation of extensions were proposed in [MT93a]. A detailed treatment of algorithmic issues related to default logic can be found in [Cho96a, Cho95b, Cho95a]. An automated reasoning system based on default logic, DeReS, is described in [CMT96, CMMT99].

The presentation of basics of logic programming follows [MT93a] and [Apt90]. The basic characterization of the least model of a Horn program comes from [vEK76]. The algorithm to compute the least model of a finite propositional logic program has was first given in [DG84].

Stable models of logic programs were introduced by Gelfond and Lifschitz in [GL88]. Basic properties of stable models come from that paper and from [MT93a, BTK93]. The translation of stable semantics of logic programs into default logic, is due to [BF91] and was discovered independently in [MT89b].

Well-founded semantics for logic programs have been introduced by Van Gelder, Ross and Schlipf

[2]However, this correspondence does not align expansions by Moore and extensions by Reiter. These two semantics occupy different locations in the corresponding hierarchies. A more detailed discussion of this issue can be found in [DMT00b].

in [VRS91]. The alternating (oscillating) fixpoint characterization, that yields a fast algorithm to compute well-founded semantics, was stated and proved in [Van89]. Generalization of well-founded semantics to default logic based on the idea of the alternating fixpoint construction is described in [BS91].

The technique of stratification, that is, a syntactic restriction guaranteeing existence and uniqueness of a stable model, is due to [CH85] and was fully developed and studied in [ABW88]. The relaxed stratification generalization used in these notes, that guarantees neither existence nor uniqueness of stable models but can be viewed as a divide-and-conquer approach to stable model computation, was developed in [Cho96b, Cho95a]. It was discovered in the setting of logic programs in [LT94] and has been since known as *splitting*.

First algorithms and implementations of algorithms for computing stable models go back to mid 1990s [NS96]. Since then, a new computational paradigm of *answer-set programming* was proposed in [MT99, Nie99], which brought a tremendous advances in the performance of stable-model computing software [SNS02, LZ02, BL02, LPF$^+$06, GKNS07]. For the discussion of the performance of these and other solvers, we refer to [GLN$^+$07], which discusses the results of the first answer-set programming competition. The concept of a loop and the result by Lin and Zhao [LZ02] was fundamental to some of the most efficient solvers and established a critical link to SAT solvers.

The connection between stable-model semantics and the logic *here-and-there* [Hey30] is due to Pearce [Pea97]. It lead to further studies of this connection, which culminated in an elegant presentation of general logic programming in [FL05]. We note though that independent precursors of general logic programming are Lifschitz-Woo programs [LW92] and its later generalization, programs with nested expressions [LTT99]. An alternative treatment of general logic programming in the modal logic S4F appeared in [Tru07]. An extension of the connection to generalized default logic is presented there, too.

The research originated by Pearce [Pea97] also resulted in fundamental notions of strong and uniform program equivalence [LPV01]. It resulted in one of the most vibrant topic in logic programing research with a great number of papers [Lin02, Tur03, IS04, ETW05, EFW07, OJ06, OTW07, Wol07]. An algebraic account of strong and uniform equivalence can be found in [Tru06]. An account of strong and uniform equivalence in a modal logic S4F is given in [Tru07]. Extensions of these concepts of equivalence to the semantics of supported and supported minimal models appear in [TW08].

References

[ABW88] K. Apt, H.A. Blair, and A. Walker. Towards a theory of declarative knowledge. In J. Minker, editor, *Foundations of deductive databases and logic programming*, pages 89–142. Morgan Kaufmann, 1988.

[Ant97] G. Antoniou. *Nonmonotonic Reasoning*. MIT Press, 1997.

[Apt90] K. Apt. Logic programming. In J. van Leeuven, editor, *Handbook of theoretical computer science*, pages 493–574. Elsevier, Amsterdam, 1990.

[BDK97] G. Brewka, J. Dix, and K. Konolige. *Nonmonotonic Reasoning, An Overview*. CSLI Publications, 1997.

[Bes89] P. Besnard. *An Introduction to Default Logic*. Springer, Berlin, 1989.

[BF87] N. Bidoit and C. Froidevaux. Minimalism subsumes default logic and circumscription. In *Proceedings of IEEE Symposium on Logic in Computer Science, LICS-87*, pages 89–97. IEEE Press, 1987.

[BF91] N. Bidoit and C. Froidevaux. Negation by default and unstratifiable logic programs. *Theoretical Computer Science*, 78(1, (Part B)):85–112, 1991.

[BL02] Y. Babovich and V. Lifschitz. *Cmodels package*, 2002. `http://www.cs.utexas.edu/users/tag/cmodels.html`.

[BNT08] G. Brewka, I. Niemelä, and M. Truszczyński. Nonmonotonic reasoning. In V. Lifschitz, B. Porter, and F. van Harmelen, editors, *Handbook of Knowledge Representation*, pages 239–284. Elsevier, 2008.

[Boc01] Alexander Bochman. *A Logical Theory of Nonmonotonic Inference and Belief Change*. Springer, Berlin, 2001.

[Boc05] Alexander Bochman. *Explanatory Nonmonotonic Reasoning*, volume 4 of *Advances in Logic*. World Scientific, 2005.

[Bre91a] G. Brewka. Cumulative default logic: in defense of nonmonotonic inference rules. *Artificial Intelligence*, 50(2):183–205, 1991.

[Bre91b] G. Brewka. *Nonmonotonic Reasoning: Logical Foundations of Commonsense*, volume 12 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, UK, 1991.

[BS91] C. Baral and V.S. Subrahmanian. Dualities between alternative semantics for logic programming and nonmonotonic reasoning (extended abstract). In A. Nerode, W. Marek, and V.S. Subrahmanian, editors, *Logic programming and non-monotonic reasoning (Washington, DC, 1991)*, pages 69–86, Cambridge, MA, 1991. MIT Press.

[BSJ95] K. Berman, J. Schlipf, and J.Franco. Computing the well-founded semantics faster. In *Logic Programming and Nonmonotonic Reasoning (Lexington, KY, 1995)*, volume 928 of *Lecture Notes in Artificial Intelligence*, pages 113–125. Springer, 1995.

[BTK93] A. Bondarenko, F. Toni, and R.A. Kowalski. An assumption-based framework for non-monotonic reasoning. In A. Nerode and L. Pereira, editors, *Logic programming and non-monotonic reasoning (Lisbon, 1993)*, pages 171–189, Cambridge, MA, 1993. MIT Press.

[CDS94] M. Cadoli, F. M. Donini, and M. Schaerf. Is intractability of non-monotonic reasoning a real drawback? In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-1994)*, pages 946–951. AAAI Press, 1994.

[CH85] A. K. Chandra and D. Harel. Horn clause queries and generalizations. *Journal of Logic Programming*, 2(1):1–15, 1985.

[Che80] B.F. Chellas. *Modal logic. An introduction*. Cambridge University Press, Cambridge-New York, 1980.

[Cho95a] P. Cholewiński. Reasoning with stratified default theories. In *Logic programming and nonmonotonic reasoning (Lexington, KY, 1995)*, volume 928 of *Lecture Notes in Computer Science*, pages 273–286, Berlin, 1995. Springer.

[Cho95b] P. Cholewiński. Stratified default theories. In *Computer science logic (Kazimierz, 1994)*, volume 933 of *Lecture Notes in Computer Science*, pages 456–470, Berlin, 1995. Springer.

[Cho96a] P. Cholewinski. *Automated Reasoning with Default Logic*. PhD thesis, University of Kentucky, 1996.

[Cho96b] P. Cholewiński. Seminormal stratified default theories. *Annals of Mathematics and Artificial Intelligence*, 17(3-4):213–234, 1996.

[Cla78] K.L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and data bases*, pages 293–322. Plenum Press, New York-London, 1978.

[CMMT99] P. Cholewiński, W. Marek, A. Mikitiuk, and M. Truszczyński. Computing with default logic. *Artificial Intelligence*, 112:105–146, 1999.

[CMT96] P. Cholewiński, W. Marek, and M. Truszczyński. Default reasoning system deres. In *Proceedings of KR-96*, pages 518–528. Morgan Kaufmann, 1996.

[DG84] W.F. Dowling and J.H. Gallier. Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *Journal of Logic Programming*, 1(3):267–284, 1984.

[DMT98] M. Denecker, V. Marek, and M. Truszczyński. Fixpoint 3-valued semantics for autoepistemic logic. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-1998)*, pages 840 – 845. AAAI Press, 1998.

[DMT00a] M. Denecker, V. Marek, and M. Truszczyński. Approximations, stable operators, well-founded fixpoints and applications in nonmonotonic reasoning. In J. Minker, editor, *Logic-Based Artificial Intelligence*, pages 127–144. Kluwer Academic Publishers, 2000.

[DMT00b] M. Denecker, V. Marek, and M. Truszczyński. Unified semantic treatment of default and autoepistemic logics. In *Principles of Knowledge Representation and Reasoning, Proceedings of the 7th International Conference (KR2000)*, pages 74 – 84. Morgan Kaufmann Publishers, 2000.

[DMT03] M. Denecker, V. Marek, and M. Truszczyński. Uniform semantic treatment of default and autoepistemic logics. *Artificial Intelligence Journal*, 143:79–122, 2003.

[Doe94] K. Doets. *From Logic to Logic Programming*. Foundations of Computing Series. MIT Press, Cambridge, MA, 1994.

[EF03] T. Eiter and M. Fink. Uniform equivalence of logic programs under the stable model semantics. In *Proceedings of the 19th International Conference on Logic Programming (ICLP 2003)*, volume 2916 of *LNCS*, pages 224–238. Springer, 2003.

[EFW07] T. Eiter, M. Fink, and S. Woltran. Semantical characterizations and complexity of equivalences in answer set programming. *ACM Transactions on Computational Logic*, 8(3), July 2007. 53 pages.

[EL03] E. Erdem and V. Lifschitz. Tight logic programs. *Theory and Practice of Logic Programming*, 3(4-5):499–518, 2003.

[Eth88] D. W. Etherington. *Reasoning with incomplete information*. Research Notes in Artificial Intelligence. Pitman Publishing, Ltd., London-Boston, MA, 1988.

[ETW05] T. Eiter, H. Tompits, and S. Woltran. On solution correspondences in answer-set programming. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005)*, pages 97–102. Morgan Kaufmann, 2005.

[Fag94] F. Fages. Consistency of Clark's completion and existence of stable models. *Journal of Methods of Logic in Computer Science*, 1:51–60, 1994.

[Fit02] M. C. Fitting. Fixpoint semantics for logic programming – a survey. *Theoretical Computer Science*, 278:25–51, 2002.

[FL05] P. Ferraris and V. Lifschitz. Mathematical foundations of answer set programming. In S.N. Artëmov, H. Barringer, A.S. d'Avila Garcez, L.ís C. Lamb, and J. Woods, editors, *We Will Show Them! Essays in Honour of Dov Gabbay*, pages 615–664. College Publications, 2005.

[FLL06] P. Ferraris, J. Lee, and V. Lifschitz. A generalization of the lin-zhao theorem. *Ann. Math. Artif. Intell.*, 47(1-2):79–101, 2006.

[GC90] R. Guerreiro and M. Casanova. An alternative semantics for default logic. Preprint. The 3rd International Workshop on Nonmonotonic Reasoning, South Lake Tahoe, 1990.

[Gel87] M. Gelfond. On stratified autoepistemic theories. In *Proceedings of AAAI-87*, pages 207–211. Morgan Kaufmann, 1987.

[Gel89] M. Gelfond. Autoepistemic logic and formalization of commonsense reasoning: preliminary report. In *Nonmonotonic reasoning (Grassau, 1988)*, volume 346 of *Lecture Notes in Computer Science*, pages 176–186, Berlin-New York, 1989. Springer.

[GKNS07] M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub. *Clasp* : A conflict-driven answer set solver. In C. Baral, G. Brewka, and J.S. Schlipf, editors, *Logic Programming and Nonmonotonic Reasoning, 9th International Conference, LPNMR 2007, Proceedings*, volume 4483 of *LNCS*, pages 260–265. Springer, 2007.

[GKPS95] G. Gogic, H. Kautz, Ch. Papadimitriou, and B. Selman. The comparative linguistics of knowledge representation. In *IJCAI-95, Vol. 1, 2 (Montreal, PQ, 1995)*, pages 862–869, San Francisco, CA, 1995. Morgan Kaufmann.

[GL88] M. Gelfond and V. Lifschitz. The stable semantics for logic programs. In *Proceedings of the 5th International Conference on Logic Programming (ICLP 1988)*, pages 1070–1080. MIT Press, 1988.

[GL91] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.

[GLN+07] M. Gebser, L. Liu, G. Namasivayam, A. Neumann, T. Schaub, and M. Truszczyński. The first answer set programming system competition. In C. Baral, G. Brewka, and J. Schlipf, editors, *Proceedings of the 9th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2007)*, volume 4483 of *LNCS*, pages 3–17. Springer, 2007.

[GLPT91] M. Gelfond, V. Lifschitz, H. Przymusińska, and M. Truszczyński. Disjunctive defaults. In *Principles of knowledge representation and reasoning (Cambridge, MA, 1991)*, Morgan Kaufmann Series in Representation and Reasoning, pages 230–237, San Mateo, CA, 1991. Morgan Kaufmann.

[Got92] G. Gottlob. Complexity results for nonmonotonic logics. *Journal of Logic and Computation*, 2(3):397–425, 1992.

[HC84] G.E. Hughes and M.J. Cresswell. *A companion to modal logic*. Methuen and Co., Ltd., London, 1984.

[Hey30] A. Heyting. Die formalen Regeln der intuitionistischen Logik. *Sitzungsberichte der Preussischen Akademie von Wissenschaften. Physikalisch-mathematische Klasse*, pages 42–56, 1930.

[HM85] J.Y. Halpern and Y. Moses. Towards a theory of knowledge and ignorance (preliminary report). In K. Apt, editor, *Logics and models of concurrent systems (La Colle-sur-Loup, 1984)*, volume 13 of *NATO ASI Series F: Computer and Systems Sciences*, pages 459–476, Berlin, 1985. Springer.

[IS04] Katsumi Inoue and Chiaki Sakama. Equivalence of logic programs under updates. In *Proceedings of the 9th European Conference on Logics in Artificial Intelligence (JELIA-04)*, volume 3229 of *Lecture Notes in Computer Science*, pages 174–186. Springer, 2004.

[KLM90] S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence Journal*, 44:167–207, 1990.

[Kon88] K. Konolige. On the relation between default and autoepistemic logic. *Artificial Intelligence*, 35(3):343–382, 1988.

[Kow74] R. Kowalski. Predicate logic as a programming language. In *Proceedings of the Congress of the International Federation for Information Processing (IFIP-1974)*, pages 569–574, Amsterdam, 1974. North Holland.

[Kow79] R. Kowalski. *Logic for Problem Solving*. North Holland, Amsterdam, 1979.

[Leh89] D.J. Lehmann. What does a conditional knowledge base entail? In *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning, KR-89*, pages 212–222. Morgan Kaufmann, 1989.

[Lev90] N.G. Leveson. Formal methods in software engineering, special issue on. *IEEE Transaction on Software Engineering*, 16:929–1103, 1990.

[Lif90] V. Lifschitz. On open defaults. In J. Lloyd, editor, *Proceedings of the Symposium on Computational Logic*, pages 80–95. Springer, 1990.

[Lif94] V. Lifschitz. Minimal belief and negation as failure. *Artificial Intelligence*, 70:53–72, 1994.

[Lin02] F. Lin. Reducing strong equivalence of logic programs to entailment in classical propositional logic. In *Proceedings of the 8th International Conference on Principles of Knowledge Representation and Reasoning (KR 2002)*. Morgan Kaufmann, 2002.

[Llo84] J. W. Lloyd. *Foundations of logic programming*. Symbolic Computation. Artificial Intelligence. Springer, Berlin-New York, 1984.

[LM92] D. Lehmann and M. Magidor. What does a conditional knowledge base entail? *Artificial Intelligence Journal*, 55:1–60, 1992.

[LPF+06] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello. The dlv system for knowledge representation and reasoning. *ACM Transactions on Computational Logic*, 7(3):499–562, 2006.

[LPV01] V. Lifschitz, D. Pearce, and A. Valverde. Strongly equivalent logic programs. *ACM Transactions on Computational Logic*, 2(4):526–541, 2001.

[LS90] F. Lin and Y. Shoham. Epistemic semantics for fixed-points nonmonotonic logics. In *Theoretical aspects of reasoning about knowledge (Pacific Grove, CA, 1990)*, Morgan Kaufmann Series in Representation and Reasoning, pages 111–120, San Mateo, CA, 1990. Morgan Kaufmann.

[LT94] V. Lifschitz and H. Turner. Splitting a logic program. In P. Van Hentenryck, editor, *Proceedings of the 11th Internationall Conference on Logic Programming (ICLP 1994)*, pages 23–37, 1994.

[LT00] Z. Lonc and M. Truszczyński. On the problem of computing the well-founded semantics. In *Proceedings of the 1st International Conference on Computational Logic, CL-2000*, pages 673–687. Springer, 2000. Lecture Notes in Artificial Intelligence, Vol. 1861.

[LTT99] V. Lifschitz, L. R. Tang, and H. Turner. Nested expressions in logic programs. *Annals of Mathematics and Artificial Intelligence*, pages 369–389, 1999.

[LW92] V. Lifschitz and T.Y.C. Woo. Answer sets in general nonmonotonic reasoning. In *Proceedings of the 3rd international conference on principles of knowledge representation and reasoning, KR '92*, pages 603–614, San Mateo, CA, 1992. Morgan Kaufmann.

[LZ02] F. Lin and Y. Zhao. ASSAT: Computing answer sets of a logic program by SAT solvers. In *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI 2002)*, pages 112–117. AAAI Press, 2002.

[Mak05] D. Makinson. *Bridges from Classical to Nonmonotonic Logic*, volume 5 of *Texts in Computing*. King's College Publications, 2005.

[McC77] J. McCarthy. Epistemological problems of Artificial Intelligence. In *Proceedings of the 5th Interational Joint Conference on Artificial Intelligence*, pages 1038–1044, 1977.

[McC80] J. McCarthy. Circumscription — a form of non-monotonic reasoning. *Artificial Intelligence*, 13(1-2):27–39, 1980.

[McD82] D. McDermott. Nonmonotonic logic II: nonmonotonic modal theories. *Journal of the ACM*, 29(1):33–57, 1982.

[MD80] D. McDermott and J. Doyle. Nonmonotonic logic I. *Artificial Intelligence*, 13(1-2):41–72, 1980.

[Moo84] R.C. Moore. Possible-world semantics for autoepistemic logic. In *Proceedings of the Workshop on Non-Monotonic Reasoning*, pages 344–354, 1984. Reprinted in: M. Ginsberg, ed., *Readings on Nonmonotonic Reasoning*, pages 137–142, Morgan Kaufmann, 1990.

[Moo85] R.C. Moore. Semantical considerations on nonmonotonic logic. *Artificial Intelligence*, 25(1):75–94, 1985.

[MS92] W. Marek and V.S. Subrahmanian. The relationship between stable, supported, default and autoepistemic semantics for general logic programs. *Theoretical Computer Science*, 103(2):365–386, 1992.

[MT89a] W. Marek and M. Truszczyński. Relating autoepistemic and default logics. In *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning (Toronto, ON, 1989)*, pages 276–288, San Mateo, CA, 1989. Morgan Kaufmann.

[MT89b] W. Marek and M. Truszczyński. Stable semantics for logic programs and default theories. In E.Lusk and R. Overbeek, editors, *Proceedings of the North American Conference on Logic Programming*, pages 243–256. MIT Press, 1989.

[MT91] W. Marek and M. Truszczyński. Autoepistemic logic. *Journal of the ACM*, 38(3):588–619, 1991.

[MT93a] W. Marek and M. Truszczyński. *Nonmonotonic Logic; Context-Dependent Reasoning*. Springer, Berlin, 1993.

[MT93b] A. Mikitiuk and M. Truszczyński. Rational default logic and disjunctive logic programming. In A. Nerode and L. Pereira, editors, *Logic programming and non-monotonic reasoning (Lisbon, 1993)*, pages 283–299, Cambridge, MA, 1993. MIT Press.

[MT99] V.W. Marek and M. Truszczyński. Stable models and an alternative logic programming paradigm. In K.R. Apt, W. Marek, M. Truszczyński, and D.S. Warren, editors, *The Logic Programming Paradigm: a 25-Year Perspective*, pages 375–398. Springer, Berlin, 1999.

[Nie99] I. Niemelä. Logic programming with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence*, 25(3-4):241–273, 1999.

[NS96] I. Niemelä and P. Simons. Efficient implementation of the well-founded and stable model semantics. In *Proceedings of JICSLP-96*. MIT Press, 1996.

[OJ06] E. Oikarinen and T. Janhunen. Modular Equivalence for Normal Logic Programs. In *Proceedings of the 17th European Conference on Artificial Intelligence* (*ECAI 2006*), pages 412–416. IOS Press, 2006.

[OTW07] J. Oetsch, H. Tompits, and S. Woltran. Facts do not Cease to Exist Because They are Ignored: Relativised Uniform Equivalence with Answer-Set Projection. In *Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI-2007)*, pages 458–464. AAAI Press, 2007.

[Pea90] Judea Pearl. System Z: A natural ordering of defaults with tractable applications to nonmonotonic reasoning. In *Proceedings of the 3rd Conference on Theoretical Aspects of Reasoning about Knowledge, TARK-90*, pages 121–135. Morgan Kaufmann, 1990.

[Pea97] D. Pearce. A new logical characterisation of stable models and answer sets. In Jürgen Dix, Luís Moniz Pereira, and Teodor C. Przymusinski, editors, *Non-Monotonic Extensions of Logic Programming, NMELP '96*, volume 1216 of *Lecture Notes in Computer Science*, pages 57–70. Springer, 1997.

[Poo88] D. Poole. A logical framework for default reasoning. *Artificial Intelligence*, 36(1):27–47, 1988.

[Rei78] R. Reiter. On closed world data bases. In H. Gallaire and J. Minker, editors, *Logic and data bases*, pages 55–76. Plenum Press, 1978.

[Rei80] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1-2):81–132, 1980.

[Sch92a] T. Schaub. On constrained default theories. In *Proceedings of the 11th European Conference on Artificial Intelligence, ECAI'92*, pages 304–308. Wiley and Sons, 1992.

[Sch92b] G.F. Schwarz. Minimal model semantics for nonmonotonic modal logics. In *Proceedings of LICS-92*, pages 34–43, 1992.

[Seg71] K. Segerberg. *An essay in classical modal logic*. Number 13 in Filosofiska Studier. Filosofiska Föreningen och Filosofiska Institutionen vid Uppsala Universitet, Uppsala, 1971.

[SNS02] P. Simons, I. Niemelä, and T. Soininen. Extending and implementing the stable model semantics. *Artificial Intelligence*, 138:181–234, 2002.

[ST94] G.F. Schwarz and M. Truszczyński. Minimal knowledge problem: a new approach. *Artificial Intelligence*, 67(1):113–141, 1994.

[ST96] G.F. Schwarz and M. Truszczyński. Nonmonotonic reasoning is sometimes simpler! *Journal of Logic and Computation*, 6(2):295–308, 1996.

[Sta80] R.C. Stalnaker. A note on nonmonotonic modal logic. Unpublished manuscript, 1980.

[Sti92] J. Stillman. The complexity of propositional default logics. In *AAAI-92. Proceedings, 10th National Conference on Artificial Intelligence (San Jose, CA, 1992)*, pages 794–799, Menlo Park, CA, 1992. American Association for Artificial Intelligence, Morgan Kaufmann.

[Tar55] A. Tarski. Lattice-theoretic fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.

[Tru91a] M. Truszczyński. Modal interpretations of default logic. In *Proceedings of IJCAI-91*, pages 393–398. Morgan Kaufmann, 1991.

[Tru91b] M. Truszczyński. Modal nonmonotonic logic with restricted application of the negation as failure to prove rule. *Fundamenta Informaticae*, 14(3):355–366, 1991.

[Tru06] M. Truszczyński. Strong and uniform equivalence of nonmonotonic theories — an algebraic approach. In P. Doherty, J. Mylopoulos, and C.A. Welty, editors, *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 389–399. AAAI Press, 2006.

[Tru07] M. Truszczyński. The modal logic S4F, the default logic, and the logic here-and-there. In *Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI 2007)*. AAAI Press, 2007.

[Tur01] H. Turner. Strong equivalence for logic programs and default theories (made easy). In *Proceedings of Logic Programming and Nonmonotonic Reasoning Conference, LPNMR 2001*, volume 2173, pages 81–92. Lecture Notes in Artificial Intelligence, Springer, 2001.

[Tur03] H. Turner. Strong equivalence made easy: nested expressions and weight constraints. *Theory and Practice of Logic Programming*, 3:609–622, 2003.

[TW08] M. Truszczyński and S. Woltran. Hyperequivalence of logic programs with respect to supported models. In *Proceedings of the 23rd National Conference on Artificial Intelligence (AAAI 2008)*. AAAI Press, 2008.

[Van89] A. Van Gelder. The alternating fixpoints of logic programs with negation. In *ACM Symposium on Principles of Database Systems*, pages 1–10, 1989.

[vEK76] M.H. van Emden and R.A. Kowalski. The semantics of predicate logic as a programming language. *Journal of the ACM*, 23(4):733–742, 1976.

[Voo91] F. Voorbraak. The logic of objective knowledge and rational belief. In *Proceedings of the European Workshop on Logics in AI (JELIA 1990)*, volume LNCS 478, pages 499–515. Springer, 1991.

[VRS88] A. Van Gelder, K.A. Ross, and J.S. Schlipf. Unfounded sets and well-founded semantics for general logic programs. In *ACM Symposium on Principles of Database Systems*, pages 221–230, 1988.

[VRS91] A. Van Gelder, K.A. Ross, and J.S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991.

[Wol07] S. Woltran. A Common View on Strong, Uniform, and Other Notions of Equivalence in Answer-Set Programming. In D. Pearce, A. Polleres, A. Valverde, and S. Woltran, editors, *Proceedings of the 1st Workshop Correspondence and Equivalence for Nonmonotonic Theories (CENT'07)*, volume 265 of *CEUR Workshop Proceedings*, pages 13–24. CEUR-WS.org, 2007.