Cautious Models for General Logic Programs

Tomi Janhunen*,
Digital Systems Laboratory,
Helsinki University of Technology,
Otakaari 1, FIN-02150 Espoo, FINLAND.
Email: Tomi.Janhunen@hut.fi

May 24, 1995

Abstract

In this paper, *cautious models* of general logic programs are investigated. Such models are constructed iteratively using a monotonic operator which performs case analysis on total interpretations generated by enumerations of atoms. Consequently, every general logic program has a unique cautious model. The new class of partial models is compared with well-founded and stable models of general logic programs. Various extensions of these models are are also addressed. The time complexity of constructing cautious models is analyzed. Results indicate that the major reasoning task is co-NP-complete. Finally, the connection to cautious autoepistemic logic is explained.

1 Introduction

A *definite logic program* consists of a set of facts and a set of unidirectional inference rules by means of which new facts can be derived. Such logic programs have a clear declarative meaning determined by their minimal Herbrand model [11]. The expressive power of definite logic programs can be increased by introducing a form of *negation* based on *failure* to derive [4]. Such logic programs are known as *general logic programs* (or *normal logic programs*). The idea behind negation as failure is that if a fact p is not derivable using the rules and facts of a general logic program, the negation of the fact ($\sim p$) is then considered to hold. Despite of the simplicity of this principle, complications arise on the semantical aspects. A reason for this is that negation as failure makes reasoning in logic programming *nonmonotonic*: conclusions may be retracted when programs are extended by adding new facts or rules. Consequently, it is relatively difficult to define the declarative meaning of general logic programs, i.e. the correct set of conclusions that a given program induces.

A variety of formal semantics have been proposed to provide a declarative meaning for general logic programs. Among these is *stable model semantics* proposed by Gelfond and Lifschitz

*This work has been financed by Academy of Finland and Helsinki Graduate School in Computer Science and Engineering. The support from Alfred Kordelin Foundation, Emil Aaltonen Foundation and Foundation of Technology is also acknowledged with gratitude.

[8]. Stable model semantics coincides with the semantics based on stable expansions of *autoepistemic logic* [13]. This demonstrates how the declarative meaning of general logic programs can be defined using methods of *nonmonotonic reasoning*. Stable models are total in the sense that, given some fact p, p is either considered to hold (it is derivable by the rules of the program) or its negation $\sim p$ is considered to hold (it cannot be derived). However, this requirement is rather strong: there are general logic programs without stable models. It is also possible that a general logic program has many stable models. The well-founded semantics proposed by Van Gelder et al. [17] approaches these problems of stable model semantics. A *partial model* is constructed for a general logic program: some facts may remain undefined in the model, i.e. they are neither true or false. Resorting to partial models is in favor of the existence of a model: every general logic program has a unique well-founded model. Moreover, the stable models of a general logic program (given such models exist) extend the well-founded model of the program. In this way, the well-founded model of a general logic program P approximates the intersection of the stable models of P.

An alternative semantics for general logic programs is proposed in this paper. This semantics is based on *cautious models* of general logic programs and it fulfills two important design criteria: (i) it extends the well-founded semantics and (ii) it has a natural definition as the least fixed point of a monotonic operator. The structure of the paper is as follows. General logic programs and their partial models are introduced in Section 2. A notion of derivability with general logic programs is also provided. In Section 3 we present the major definitions and show that every general logic program has a unique cautious model. A comparison with the well-founded semantics and its extensions is made in Section 4. Section 5 concentrates on the connection to the stable model semantics and its generalizations. Some abstract properties of cautious model semantics are addressed in Section 6. The computational complexity of the major decision problem is examined in Section 7. Finally in Section 8, we describe the connection to cautious autoepistemic logic introduced by Janhunen and Niemelä [9].

2 Preliminaries

In this section, we define a notion of *derivability* for general logic programs and introduce *partial models* following [17]. For brevity, general logic programs are called logic programs in the sequel. Formally, rules of a logic program are of the form

$$p \leftarrow p_1, ..., p_n, \sim q_1, ..., \sim q_m$$

where p, $p_1, ..., p_n$ and $q_1, ..., q_n$ are atoms and $\sim$ presents negation as failure. The intuition behind the rule is that if the *positive subgoals* $p_1, ..., p_n$ of the rule are derivable, but not the atoms $q_1, ..., q_n$ in the *negative subgoals* of the rule, then p is derivable. Facts are represented as rules without subgoals. For the sake of simplicity, we restrict ourselves to logic programs which are *finite* and *fully instantiated*. The latter implies that the logic programs under consideration are effectively propositional ones. The Herbrand base $\mathrm{Hb}(P)$ of a logic program is the set of atoms that appear in P. For logic programs under our consideration $\mathrm{Hb}(P)$ is always finite. We define *literals* in the usual manner using the symbol $\sim$ for negation. For a set of literals L, the sets $\{p \mid p \in L\}$ and $\{\sim p \mid \sim p \in L\}$ are denoted by $[L]^+$ and $[L]^-$, respectively. Moreover, let $\mathrm{atoms}(L)$ denote the set of atoms that appear in L. Rules of logic programs are often represented in the form $p \leftarrow B$ where p is an atom (the *head* of the rule) and B is a set of literals (the *body* of the rule). This representation stresses that the order of the subgoals in the body of a rule is irrelevant.

We will often need *operators*, i.e. functions from sets of literals (atoms) to sets of literals (atoms), to construct sets of literals (atoms) through *iteration*. These sets of literals and atoms are formed from atoms of $\mathrm{Hb}(P)$ for some logic program P. We use $F \uparrow^i (I)$ to denote the result of iterating an operator F i times on a set of literals (atoms) I. By the famous lemma of Knaster and Tarski, such an iteration has a *least fixed point* $\mathrm{lfp}(F, I) = F(\mathrm{lfp}(F, I))$ provided the sequence $F \uparrow^i (I)$ $(i \geq 0)$ is nondecreasing, i.e. $F \uparrow^i (I) \subseteq F \uparrow^{i+1} (I)$ for all $i \geq 0$. Notice that $\mathrm{lfp}(F, I) = F \uparrow^\omega (I)$ when $\mathrm{Hb}(P)$ is finite. An operator F is *monotonic* if $I \subseteq J$ implies $F(I) \subseteq F(J)$.

In Definition 2.1, we formalize a notion of derivability for logic programs given a set of literals I as initial assumptions. The purpose of negative literals $[I]^-$ is to select the rules of P that are used in the derivation. A rule can be used only if its all negative subgoals are given as initial assumptions. When applied to a set of atoms A, the operator $\mathbf{A}_{P,I}$ performs one step of derivation: it gives the heads of the rules whose all subgoals have been either derived or given as assumptions. The operator $\mathbf{A}_{P,I}$ is clearly monotonic on sets of atoms A. This justifies the definition of the operator $\mathbf{A}_P$ as a least fixed point of $\mathbf{A}_{P,I}$. Intuitively, $\mathbf{A}_P(I)$ gives the atoms that are derivable using the rules of P from the initial assumptions I (see Example 2.2). The operator $\mathbf{A}_P$ is also monotonic: given sets of literals $I_1 \subseteq I_2$, it can be proved by induction that $\mathbf{A}_{P,I_1} \uparrow^i ([I_1]^+)$ is a subset of $\mathbf{A}_{P,I_2} \uparrow^i ([I_2]^+)$ for all $i \geq 0$.

Definition 2.1 *Let P be a logic program, I a set of literals and A a set of atoms. Define operators* $\mathbf{A}_{P,I}(A) = A \cup \{p \mid p \leftarrow B$ *is a rule of* $P, [B]^- \subseteq I$ *and* $[B]^+ \subseteq A\}$ *and* $\mathbf{A}_P(I) = \mathrm{lfp}(\mathbf{A}_{P,I}, [I]^+)$.

Example 2.2 Let $P = \{r \leftarrow \sim p; s \leftarrow q, r\}$[1] and $I = \{\sim p, q\}$. Then $\mathbf{A}_{P,I}(\{q\}) = \{q, r\}$ and $\mathbf{A}_{P,I}(\{q, r\}) = \{q, r, s\}$. Thus $\mathbf{A}_P(I) = \{q, r, s\}$.

We shall use the same notions of *partial* interpretations and models of logic programs as suggested by Van Gelder et al. [17]. A *partial interpretation* I of a logic program P is a consistent set of literals over $\mathrm{Hb}(P)$. Let $\mathrm{Int}(P)$ denote the set of all partial interpretations of P. Given $I \in \mathrm{Int}(P)$, an atom p is *true* (*false*) in I if $p \in I$ $(\sim p \in I)$. Otherwise p is *undefined* in I. If no $p \in \mathrm{Hb}(P)$ is undefined in a partial interpretation I, I is a *total interpretation* of P. A negative literal $\sim p$ is true (false) in I if p is false (true) in I. Otherwise $\sim p$ is undefined in I. A body B of a rule which is seen as conjunction of literals is true (false) in I if its all subgoals are true in I (some of its subgoals is false in I). Otherwise, B is undefined in I. A rule $p \leftarrow B$ is true (false) in I if the head p is true in I or the body B is false in I (the head p is false in I and B is true in I). Thus Van Gelder et al. interpret the rules of a logic program as Kleene's three-valued strong implications. A total interpretation I is a *total model* of P if every rule of P is true in I. A partial interpretation I of P is a *partial model* of P if I can be extended to a total model of P. These concepts are clarified in the following example.

Example 2.3 Let P be the logic program $\{p \leftarrow \sim q; q \leftarrow \sim p; r \leftarrow \sim p, \sim q\}$. Now $I = \{\sim r\}$ is a partial model of P as I can be extended to a total interpretation $I' = \{\sim p, q, \sim r\}$ in which all rules of P are true.

[1] We use semicolons to separate program rules.

3 Cautious Models of Logic Programs

Let us now propose an alternative semantics for logic programs. This semantics is based on *cautious models* of logic programs that are defined as a least fixed point of an operator on partial models of logic programs. Given a partial model I of a logic program P, this operator checks the possible extensions of I to total interpretations (not necessarily total models) of P and extends I to a larger partial model on the basis of these extensions. However, all extensions of I are not equally interesting, so the key question is how to select among all the alternatives. We shall use an enumeration-based method in analogy to cautious autoepistemic logic [9]. From now on we use a restriction of operator $\mathbf{A}_P$ as derivability with logic programs: $\mathbf{B}_P(I) = \mathbf{A}_P([I]^-)$. This ensures that atoms derived by the logic program depend solely on the negative assumptions $[I]^-$. The monotonicity of $\mathbf{B}_P$ is implied by the monotonicity of $\mathbf{A}_P$.

In Definition 3.1 we construct extensions $\mathrm{E}_P^\varepsilon(I)$ of a partial interpretation I using an enumeration ε of atoms from $\mathrm{Hb}(P) - \mathrm{atoms}(I)$. At i^{th} step of the construction the truth of p_i is considered. If $p_i \in \mathbf{B}_P(I \cup \mathrm{E}_P^{\varepsilon,i-1}(I))$, p_i is set true in $\mathrm{E}_P^{\varepsilon,i}(I)$ and false otherwise. As the operator $\mathbf{B}_P$ coincides with the derivability by the rules of P, a closed world assumption is made whenever $\sim p_i$ is included in $\mathrm{E}_P^{\varepsilon,i}(I)$. Notice that $I \cup \mathrm{E}_P^\varepsilon(I)$ is always a partial interpretation of P as no member of $\mathrm{atoms}(I)$ is included in ε. Moreover, if ε enumerates $\mathrm{Hb}(P) - \mathrm{atoms}(I)$ in its entirety, then $I \cup \mathrm{E}_P^\varepsilon(I)$ is a total interpretation of P. If there is an enumeration ε of $\mathrm{atoms}(J) - \mathrm{atoms}(I)$ such that $J = I \cup \mathrm{E}_P^\varepsilon(I)$, we say that the enumeration ε *generates* the partial interpretation J from I. In the case $I = \emptyset$, J simply *has a generating enumeration* ε.

Definition 3.1 *Let P be a logic program and I its partial interpretation. Let $\varepsilon = p_1, p_2, ..., p_n$ be an enumeration of a subset of* $\mathrm{Hb}(P) - \mathrm{atoms}(I)$. *Define* $\mathrm{E}_P^{\varepsilon,0}(I) = \emptyset$ *and for all* $0 < i \leq n$

$$\mathrm{E}_P^{\varepsilon,i}(I) = \begin{cases} \mathrm{E}_P^{\varepsilon,i-1}(I) \cup \{p_i\}, & \textit{if } p_i \in \mathbf{B}_P(I \cup \mathrm{E}_P^{\varepsilon,i-1}(I)) \textit{ and} \\ \mathrm{E}_P^{\varepsilon,i-1}(I) \cup \{\sim p_i\}, & \textit{otherwise.} \end{cases}$$

Finally define the enumeration-based extension $\mathrm{E}_P^\varepsilon(I)$ *of* I *as* $\mathrm{E}_P^{\varepsilon,n}(I)$.

Example 3.2 Let P be the program $\{p \leftarrow\sim q; q \leftarrow\sim p; r \leftarrow\sim p, \sim q\}$ and $\varepsilon = p, q, r$ an enumeration. Now $p \notin \mathbf{B}_P(\emptyset)$ implies that $\mathrm{E}_P^{\varepsilon,1}(\emptyset) = \{\sim p\}$ and $q \in \mathbf{B}_P(\{\sim p\})$ implies $\mathrm{E}_P^{\varepsilon,2}(\emptyset) = \{\sim p, q\}$. Finally $\mathrm{E}_P^{\varepsilon,3}(\emptyset) = \mathrm{E}_P^\varepsilon(\emptyset) = \{\sim p, q, \sim r\}$, as $r \notin \mathbf{B}_P(\{\sim p, q\})$.

The false atoms of cautious models are determined using the the enumeration-based extensions. The following case analysis is performed: if an atom p is not derivable in any of the enumeration-based extensions, it is safe to include $\sim p$ in the cautious model. This idea of cautiousness is embodied in the operator $\mathbf{D}_P$ given in Definition 3.3 (the partial model I represents the cautious model under construction). Using the monotonicity of $\mathbf{B}_P$, it is straightforward to show that $\mathbf{B}_P(I)$ and $\mathbf{D}_P(I)$ are mutually disjoint for any partial interpretation I. Thus $\mathbf{C}_P(I)$ is always a partial interpretation.

Definition 3.3 *Let P be a logic program and I its partial interpretation. Define $\mathbf{D}_P(I)$ as the set of atoms p of* $\mathrm{Hb}(P)$ *for which there is no enumeration ε of* $\mathrm{Hb}(P) - \mathrm{atoms}(I)$ *such that* $p \in \mathbf{B}_P(I \cup \mathrm{E}_P^\varepsilon(I))$. *Finally, define* $\mathbf{C}_P(I) = \mathbf{B}_P(I) \cup \sim\mathbf{D}_P(I)$.

Given partial interpretations I_1 and I_2 such that $I_1 \subseteq I_2$ and I_2 has a generating enumeration from I_1, it can be shown that $\mathbf{D}_P(I_1) \subseteq \mathbf{D}_P(I_2)$. This property is needed in the proof of Theorem 3.4 in addition to the monotonicity of $\mathbf{B}_P$. The key idea in the inductive proof is that in fact any enumeration of $\mathrm{atoms}(I_i - I_{i-1})$ generates I_i from I_{i-1} so that $p \in \mathbf{D}_P(I_{i-1})$ implies $p \in \mathbf{D}_P(I_i)$. The partial interpretation I_i can be extended a total model I' of P by taking any enumeration ε of $\mathrm{Hb}(P) - \mathrm{atoms}(I_i)$, setting the atoms p of $\mathrm{Hb}(P) - \mathrm{atoms}(I_i)$ such that $p \in \mathbf{B}_P(I_i \cup \mathrm{E}^{\varepsilon}_P(I_i))$ true in I' and setting the remaining atoms of $\mathrm{Hb}(P) - \mathrm{atoms}(I_i)$ false in I'. In particular, the properties of I_i that $q \in I_i$ implies $q \in \mathbf{B}_P(I_i)$ and $\sim q \in I_i$ implies $q \in \mathbf{D}_P(I_i)$ are required to show that all rules of P are true in I'.

Theorem 3.4 *Let P be a logic program. Define the sequence $I_0, I_1, \ldots$ of partial interpretations of P by setting $I_i = \mathbf{C}_P \uparrow^i (\emptyset)$ for all $i \geq 0$. Then for all $i \geq 0$ it holds that (i) I_i has a generating enumeration, (ii) $I_i \subseteq I_{i+1}$ and (iii) I_i is a partial model of P.*

Definition 3.5 *For a logic program P, define* $\mathrm{CM}(P) = \mathrm{lfp}(\mathbf{C}_P, \emptyset)$.

The *cautious model* $\mathrm{CM}(P)$ of P is defined as the least fixed point of $\mathbf{C}_P$. By Theorem 3.4, $\mathrm{CM}(P)$ is a partial model of P and it is defined for every logic program P. In Example 3.6, a cautious model $\mathrm{CM}(P) = \{\sim r\}$ is constructed. Recall that we demonstrated earlier in Example 2.3 that $\{\sim r\}$ is a partial model of P. It should yet be emphasized, that it is not necessary to go trough all enumerations of $\mathrm{Hb}(P) - \mathrm{atoms}(I)$ in order to compute $\mathbf{D}_P(I)$. It is the extensions generated by enumerations that matter. Notice that there are only two such extensions in Example 3.6. This issue is further discussed when the complexity of the major decision problem is analyzed in Section 7. The *cautious model semantics* of P is determined by $\mathrm{CM}(P)$. Notice that it is also possible to define the other fixed points of $\mathbf{C}_P$ as cautious models of P. In this setting, $\mathrm{CM}(P)$ is the least one of such models and $\mathrm{CM}(P)$ is contained in every other cautious model. In addition, it would be natural to define the cautious model semantics as intersection of all cautious models of P. This leads to an equivalent definition because $\mathrm{CM}(P)$ is the intersection of all fixed points of $\mathbf{C}_P$.

Example 3.6 Let P be the logic program $\{p \leftarrow\sim q; q \leftarrow\sim p; r \leftarrow\sim p, \sim q\}$. Let $I_0 = \emptyset$. Now $\mathbf{B}_P(I_0)$ is empty. The enumerations of $\mathrm{Hb}(P) - \mathrm{atoms}(I_0) = \{p, q, r\}$ are used for constructing extensions of I_0. Only two extensions appear: $\{\sim p, q, \sim r\}$ and $\{p, \sim q, \sim r\}$. These are generated e.g. by enumerations $\varepsilon_1 = p, q, r$ and $\varepsilon_2 = q, p, r$, respectively. Thus r is the only atom that belongs to $\mathbf{D}_P(I_0)$. So $I_1 = \{\sim r\}$. Again $\mathbf{B}_P(I_1)$ is empty. The two enumerations p, q and q, p of $\mathrm{Hb}(P) - \mathrm{atoms}(I_1) = \{p, q\}$ regenerate the extensions $\{\sim p, q, \sim r\}$ and $\{p, \sim q, \sim r\}$, respectively. Thus $\mathbf{D}_P(I_1) = \{r\}$, $I_2 = I_1$ and $\mathrm{CM}(P) = \{\sim r\}$. This partial model is reasonable as the first two rules of P effectively express that either p or q holds implying that r is not derivable.

4 Relationship to Well-founded Models

Let us now turn our attention to the relationship between cautious models and other models of logic programs. We begin by well-founded models proposed by Van Gelder et al. [17]. In the definition, $\sim L$ denotes $\{\sim p \mid p \in L\} \cup \{p \mid \sim p \in L\}$ for a set of literals L. Thus $\sim B \cap I = \emptyset$ holds if and only if B is consistent with I. Given a set of atoms A, $\overline{A}$ denotes the relative complement $\mathrm{Hb}(P) - A$ (the program P is clear from the context). The operator $\mathbf{T}_P$ sets true the head of every rule $p \leftarrow B$ whose body B is true in I, i.e. $B \subseteq I$. The purpose of the operator $\mathbf{R}_P$ is to analyze which atoms could *possibly* be derivable by the rules of P (under any negative assumptions extending I). The atoms in the complement $\overline{\mathbf{R}_P(I)}$ are therefore guaranteed to be non-derivable and these atoms can be safely set false in the partial interpretation under construction. Van Gelder et al. have shown that the sequence $I_i = \mathbf{W}_P \uparrow^i (\emptyset)$ where $i \geq 0$ is non-decreasing and consists of partial models of P. Thus $\mathrm{WFM}(P)$ is a partial model of P as well.

Definition 4.1 *Let P be a logic program and I its partial interpretation. Define operators*

$$\begin{array}{ll} \mathbf{T}_P(I) = & \{h \mid h \leftarrow B \text{ is a rule of } P \text{ and } B \subseteq I\}, \\ \mathbf{R}_{P,I}(J) = & \{h \mid h \leftarrow B \text{ is a rule of } P,\ {\sim}B \cap I = \emptyset \text{ and } [B]^+ \subseteq J\} \text{ and} \\ \mathbf{R}_P(I) = & \mathrm{lfp}(\mathbf{R}_{P,I}, \emptyset). \end{array}$$

Finally define the operator $\mathbf{W}_P(I) = \mathbf{T}_P(I) \cup \sim \overline{\mathbf{R}_P(I)}$ and the well-founded model of P as $\mathrm{WFM}(P) = \mathrm{lfp}(\mathbf{W}_P, \emptyset)$.

It can be proved that the well-founded model of a logic program P is subsumed by the cautious model of P. The reason for this is that no member p of $\overline{\mathbf{R}_P(I)}$ is either derivable by the enumeration-based method, i.e. $p \in \mathbf{D}_P(I)$. Consequently, the cautious model provides at least as accurate set of conclusions as the well-founded model does. The general logic program $\{p \leftarrow \sim p\}$ is an example for which the two semantics coincide: $\mathrm{WFM}(\{p \leftarrow \sim p\}) = \mathrm{CM}(\{p \leftarrow \sim p\}) = \emptyset$. However, there are logic programs whose cautious model is more accurate than their well-founded model (see Example 4.3). As explained earlier in Section 3, the difference is due to the case analysis performed by the operator $\mathbf{D}_P$.

Theorem 4.2 *Let P be a logic program. Then* $\mathrm{WFM}(P) \subseteq \mathrm{CM}(P)$.

There are other semantics of logic programs that extend the well-founded semantics. One possibility is to use minimal model reasoning. Generalized well-founded semantics $\mathrm{GWFS}(P)$ proposed by Baral et al. [1] and two other semantics $\mathrm{EWFS}(P)$ and $\mathrm{WFS}^+(P)$ proposed by Dix [5] follow this approach. These semantics consider the minimal Herbrand models of P that extend $\mathrm{WFM}(P)$. Depending on the semantics, atoms that are true (false) in all minimal Herbrand models that extend the current model ($\mathrm{WFM}(P)$ or its extension) are set true (false) in the model. For the program in Example 4.3, both $\mathrm{GWFS}(P)$ and $\mathrm{EWFS}(P)$ coincide with $\mathrm{CM}(P)$, but $\mathrm{WFS}^+(P) = \emptyset$ differs. Dix [7] has recently shown that $\mathrm{WFS}^+(P)$ coincides with Schlipf's well-founded-by-case semantics $\mathrm{WFS}_{\mathrm{C}}(P)$. Thus $\mathrm{WFS}_{\mathrm{C}}(P) = \emptyset$ for the program P in Example 4.3.

Example 4.3 (Negative case analysis) Let $P = \{p \leftarrow \sim q; q \leftarrow \sim p; r \leftarrow \sim p, \sim q\}$ be a logic program. Now $\mathrm{WFM}(P) = \emptyset$, because $\mathbf{T}_P(\emptyset) = \emptyset$ and $\mathbf{R}_P(\emptyset) = \{p, q, r\}$. However, $\mathrm{CM}(P) = \{\sim r\}$ as demonstrated in Example 3.6.

Given a logic program $P = \{p \leftarrow \sim p\}$, the semantics $\mathrm{GWFS}(P)$, $\mathrm{EWFS}(P)$ and $\mathrm{WFS}^+(P)$ consider p true because p is true in the only minimal Herbrand model $\{p\}$ of P. This is somewhat unjustified because if $p \leftarrow \sim p$ interpreted as an inference rule but not a classical implication $\neg p \rightarrow p$, such an inference is impossible given p is undefined. Recall that $\mathrm{CM}(P) = \emptyset$, i.e. p remains undetermined in the cautious model of P.

When the atoms true in $\mathrm{CM}(P)$ are determined no case analysis is performed (using the enumeration-based method). In Example 4.4 this weakness of $\mathrm{CM}(P)$ is demonstrated. Even though r is included in every enumeration-based extension, $\mathrm{CM}(P)$ remains empty. The semantics $\mathrm{GWFS}(P)$, $\mathrm{EWFS}(P)$ and $\mathrm{WFS}^{+}(P)$ end up in the more intuitive set of conclusions $\{r\}$. Despite $\mathrm{CM}(P)$ remains empty in Example 4.4, this does not mean that such inferences are impossible under the cautious model semantics. If we replace the last rule of Example 4.3 by the rules $r' \leftarrow \sim p, \sim q$ and $r \leftarrow \sim r'$ the respective cautious model is $\{\sim r', r\}$. This is how reasoning expected in the floating conclusions problem, is carried out by the negative case analysis of the operator $\mathbf{D}_P$. On the other hand, it seems possible to modify cautious models such that the floating conclusions problem is solved directly. The idea is to replace $\mathbf{B}_P$ (from the definition of $\mathbf{C}_P$) by an operator that – in analogy to $\mathbf{D}_P$ – performs case analysis on enumeration-based extensions. In order to set an atom p true, p has to belong to every enumeration-based extension as suggested by Example 4.4. However, extra care is required to ensure the monotonicity the corresponding operator. For example, given the logic program $P = \{p \leftarrow \sim p\}$, p belongs to every enumeration based extension of $\emptyset$. But we argued earlier that p should not be inferred from P. Nevertheless, the sketched modification probably affects the properties of cautious models and we leave the further study of this variant to be carried out elsewhere.

Example 4.4 (Floating conclusions) Let $P = \{p \leftarrow \sim q; q \leftarrow \sim p; r \leftarrow p; r \leftarrow q\}$ be a logic program. Now p and q remain undetermined in $\mathrm{CM}(P)$. Consequently, this happens also to r as r does not belong to $\mathbf{B}_P(\emptyset)$.

Pereira et al. have proposed O-semantics [14]. Their idea is to perform closed world reasoning on $\mathrm{WFM}(P)$. A difference with cautious models becomes apparent with a program $P = \{p \leftarrow \sim p; q \leftarrow p\}$. O-semantics and $\mathrm{CM}(P)$ agree on p by leaving it undefined, but $\sim q$ can be inferred according to O-semantics. This is reasonable in the sense that given that p remains undefined in the model, q cannot be inferred by the rule $q \leftarrow p$. Notice that the interpretation $\{\sim q\}$ provided by O-semantics is not a partial model of P (none of the total interpretations $\{p, \sim q\}$ and $\{\sim p, \sim q\}$ is a model of P). O-semantics captures $\sim r$ in Example 4.3 but not r in Example 4.4. Thus O-semantics has a limited case analysis capability. However, generally speaking, the O-semantics seems to provide a larger set of conclusions than the cautious model semantics.

5 Relationship to Stable Models

Our study goes on with *stable models* of logic programs proposed by Gelfond and Lifschitz [8]. Stable models are defined using a transformation on logic programs. Given any set of atoms $A \subseteq \mathrm{Hb}(P)$, the transformed logic program

$$\Pi_A(P) = \{h \leftarrow [B]^+ \mid h \leftarrow B \text{ is a rule of } P \text{ and } [B]^- \subseteq \sim \overline{A}\}$$

is a definite logic program, i.e. has no negative subgoals. Such a logic program has a unique (minimal) Herbrand model that can be constructed as the least fixed point of the operator

$\mathbf{T}_{\Pi_A(P)}$ [11]. Stable models are Herbrand models that are reconstructed by this reduction and minimal model construction. The corresponding fixed point condition is provided in terms of the operator Γ_P defined below:

Definition 5.1 *Let P be a logic program and $\Gamma_P(I) = \mathrm{lfp}(\mathbf{T}_{\Pi_I(P)}, \emptyset)$ an operator on Herbrand interpretations I of P. A Herbrand interpretation S of P is a stable model of P if and only if $S = \Gamma_P(S)$.*

There is a close connection between the operators Γ_P and $\mathbf{B}_P$: stable models could be defined in terms of the latter as well (see Theorem 5.2). Thus every stable model S satisfies a *groundedness* condition: $p \in S$ if and only if p is derivable by the rules of P given $\sim\overline{S}$ as assumptions. In addition, partial models corresponding to stable models have generating enumerations. This justifies the use of enumerations in constructing the cautious model of a logic program. It follows that stable models are included in the partial interpretations generated by enumerations. By Theorem 5.2 the conclusions of $\mathrm{CM}(P)$ are included in every stable model of P. This supports the view of $\mathrm{CM}(P)$ as approximating the intersection of stable models of P *given such models exist.* Moreover, if $\mathrm{CM}(P)$ is a total model, $\mathrm{CM}(P)$ is the unique stable model of P.

Theorem 5.2 *Let P be a logic program. Then (i) S is a stable model of P if and only if $S = \mathbf{B}_P(\sim\overline{S})$ and (ii) for every stable model S of P, $S\cup\sim\overline{S}$ has a generating enumeration and $\mathrm{CM}(P) \subseteq S\cup\sim\overline{S}$.*

Przymusinski [15] has proposed *stationary models* (also called *partial stable models*) as a generalization of stable models. A Herbrand interpretation S of a logic program P is a stationary model of P if and only if $S = \Gamma_P(\Gamma_P(S))$ and $S \subseteq \Gamma_P(S)$. The operator corresponding to two applications of Γ_P is monotonic, implying that every logic program has at least one stationary model. Given a stationary model S, the corresponding partial model is $S\cup\sim\overline{\Gamma_P(S)}$. Thus the atoms in $\Gamma_P(S) - S$ are considered undefined. By the results of Przymusinski, the least stationary model of P coincides with the well-founded model of P. Consequently, the stationary model semantics, i.e. the intersection of stationary models, coincides with the well-founded semantics. This differentiates the stationary model semantics from the cautious model semantics. In Example 5.3, the stationary models of P correspond to the partial models $\emptyset$, $\{p, \sim q, \sim r\}$ and $\{\sim p, q, \sim r\}$, respectively. In particular, the program P does not have a stationary model corresponding to its cautious model $\{\sim r\}$.

Example 5.3 Let P be the logic program $\{p \leftarrow\sim q; q \leftarrow\sim p; r \leftarrow\sim p, \sim q\}$. The stationary models of P are $\emptyset$, $\{p\}$ and $\{q\}$, because $\Gamma_P(\{\emptyset\}) = \{p, q, r\}$, $\Gamma_P(\{p\}) = \{p\}$, $\Gamma_P(\{q\}) = \{q\}$ and $\Gamma_P(\{p, q, r\}) = \emptyset$. Notice that $\{p, q, r\}$ is not a stationary model of P as $\{p, q, r\}$ is not a subset of $\Gamma_P(\{p, q, r\})$.

Regular model semantics[2] suggested by You and Yuan [19] has a close connection to the stationary model semantics. In fact, the regular models of P correspond exactly to the stationary models of P that are maximal with respect to subset inclusion (in their representations as partial models) [7]. Therefore, in Example 5.3, the regular models of P are $\{p, \sim q, \sim r\}$ and

[2] By [18], the class of regular models coincides with those of *partial stable models* of Saccà and Zaniolo as well as Dung's *preferred extensions.*

$\{\sim p, q, \sim r\}$. Notice that $\sim r$ is included in both of these, i.e. the regular model semantics agrees with $\mathrm{CM}(P)$. The two semantics are mutually incomparable: in Example 4.4 r belongs to the intersection of the regular models of P while $\mathrm{CM}(P) = \emptyset$ and in Example 5.4 the only regular model is $\emptyset$ while $\mathrm{CM}(P) = \{\sim r\}$. The latter example illustrates a subtle difference in the way the derivability of atoms is tested in the two approaches. Given a stationary model S, the atoms in $\Gamma_P(S)$ are under suspicion of being derivable. In Example 5.4, these atoms are $\Gamma_P(S) = \mathbf{B}_P(\sim\overline{S}) = \{p, q, r\}$ for $S = \emptyset$. Thus the negative literals $\sim\overline{S} = \{\sim p, \sim q, \sim r\}$ are used as assumptions. However, $\mathbf{D}_P$ considers only the two enumeration-based extensions which realistically take the mutual exclusion of p and q into account.

Example 5.4 Let $P = \{p \leftarrow\sim q; q \leftarrow\sim p; p \leftarrow\sim p; q \leftarrow\sim q; r \leftarrow\sim p, \sim q\}$ be a logic program. The only stationary model of P as well as the only regular model of P is $\emptyset$, but $\mathrm{CM}(P) = \{\sim r\}$ because r is not derivable in the two enumeration-based extensions $\{p, \sim q, \sim r\}$ and $\{\sim p, q, \sim r\}$.

Baral and Subrahmanian have proposed *stable class semantics* for logic programs [2]. A *stable class* $\mathcal{S}$ is a class of Herbrand interpretations of a logic program P such that $\mathcal{S}$ is stable under Γ_P, i.e. $\mathcal{S} = \{\Gamma_P(I) \mid I \in \mathcal{S}\}$. A stable class $\mathcal{S}$ is called *strict* if no proper subset of $\mathcal{S}$ is a stable class. Stable classes are ordered pointwise so that $\mathcal{S}_1 \preceq \mathcal{S}_2$ if and only if for every $I \in \mathcal{S}_1$ there is $I' \in \mathcal{S}_2$ such that $I \subseteq I'$. The $\preceq$-minimal strict stable classes of P are considered to determine its meaning. In Example 4.3, P has three strict stable classes $\{\{p\}\}$, $\{\{q\}\}$ and $\{\emptyset, \{p, q, r\}\}$. The first two are $\preceq$-minimal. As r is false in every interpretation of these, P entails $\sim r$ according to stable class semantics. A difference with $\mathrm{CM}(P)$ is encountered in Example 4.4 where $\{\{p, r\}\}$ and $\{\{q, r\}\}$ are $\preceq$-minimal strict stable classes. Thus r follows from P by the stable class semantics. Finally, in Example 5.4 the only strict stable class is $\{\emptyset, \{p, q, r\}\}$. Thus the semantics based on stable classes and cautious models are mutually incomparable in accuracy.

6 Properties of Cautious Model Semantics

Dix [6] has developed an extensive framework for classifying semantics of logic programs by their abstract properties. Especially, structural rules such as *cut* and *cautious monotonicity* suggested by Kraus et al. [10] have been in central role. Following Dix's interpretation of the rules, we can formalize these rules for a *skeptical semantics* $\mathrm{SEM}(P)$ where intersection of the models determined by the semantics is considered. For a logic program P and an atom $a \in \mathrm{SEM}(P)$[3] define the program P^a which consists of P with a added as a fact, i.e. a rule with an empty body. The cut rule (the cautious monotonicity rule) is satisfied if $\mathrm{SEM}(P^a) \subseteq \mathrm{SEM}(P)$ (if $\mathrm{SEM}(P) \subseteq \mathrm{SEM}(P^a)$). A *cumulative* semantics satisfies both rules. Within such a semantics, atoms from $\mathrm{SEM}(P)$ can be inserted to P without affecting the semantics. For example, $\mathrm{WFM}(P)$, O-semantics and $\mathrm{WFS}^+(P)$ are cumulative and the semantics $\mathrm{EWFS}(P)$ does not satisfy the cut rule [6]. It can be shown that $\mathrm{CM}(P)$ satisfies the cut rule, i.e. the conclusions based on $\mathrm{CM}(P^a)$ are sound with respect to $\mathrm{CM}(P)$. However, the question whether $\mathrm{CM}(P)$ is cumulative remains open[4].

[3]Dix considers the effect of adding a set of atoms from $\mathrm{SEM}(P)$.

[4]Counterexamples [5, 6] showing that the stable and regular model semantics and $\mathrm{GWFS}(P)$ are not cumulative do not apply to $\mathrm{CM}(P)$. The author has neither succeeded in proving $\mathrm{CM}(P)$ cumulative.

Theorem 6.1 *Let P be a logic program and a an atom in* $\mathrm{CM}(P)$. *Then*

$$\mathrm{CM}(P^a) \subseteq \mathrm{CM}(P).$$

Dix has also criticized the generalized well-founded semantics for not satisfying the principle of *partial evaluation* [7]. Given a logic program P and an atom c that does not appear in the negative subgoals of the rules and the positive subgoals of the rules defining c, define the *partially evaluated* program P_c as follows. Let $c \leftarrow B_1, ..., c \leftarrow B_n$ be the rules of P having c as its head. These rules are removed from P. Moreover, for each rule $p \leftarrow B$ having c as a positive subgoal, i.e. $c \in B$, is replaced by the rules $p \leftarrow (B \cup B_1) - \{c\}, ..., p \leftarrow (B \cup B_n) - \{c\}$. Notice that P_c does not contain c in its rules. Thus P_c is a logic program obtained from P by partially evaluating P with respect to c. Dix [7] argues that any reasonable semantics should assing the same meaning to P and any partial evaluation of it. This is the case with cautious model semantics. This stems essentially from the fact that $\mathbf{B}_P(I) - \{c\} = \mathbf{B}_{P_c}(I - \{c, \sim c\})$.

Theorem 6.2 *Let P be a logic program and P_c its partial evaluation. Then*

$$\mathrm{CM}(P) - \{c, \sim c\} = \mathrm{CM}(P_c).$$

7 Computational Complexity

In this section, we analyze the computational complexity of the obvious decision problem given a logic program P: the membership of a literal in $\mathrm{CM}(P)$. The set of the positive problem instances is given in Definition 7.1. In the following we present algorithms and a nondeterministic Turing-machine to show that **CM** belongs to **co-NP**, i.e. the class of sets whose complements can be accepted by a nondeterministic Turing-machine in polynomial time. We assume that there is an algorithm Cons available such that given a logic program P, a set of literals I and an atom p, $\mathsf{Cons}(P, I, p)$ returns True if and only if $p \in \mathbf{B}_P(I)$. Cons can be implemented such that its time requirement is quadratic in $|\mathrm{Hb}(P)|$.

Definition 7.1 *Let* **CM** *be the set of pairs $\langle P, l \rangle$ such that P is a logic program and l a literal that belongs to* $\mathrm{CM}(P)$.

The operator $\mathbf{D}_P$ is defined in terms of enumerations of atoms and total interpretations generated by them. From the computational point of view, it is important to notice that the number of enumerations grows faster than the number of extensions. Given $|\mathrm{Hb}(P)| = n$ there are $n!$ enumerations but 2^n total interpretations. Thus our algorithms concentrate on finding out total interpretations that have generating enumerations[5]. Given a partial interpretation I, the set of atoms undefined in I is denoted by $\mathrm{Undef}_P(I)$.

Functions NegGen and Gen (see below) check whether a set of literals has a generating enumeration. For a partial interpretation and J a *set of negative literals* such that $\mathrm{atoms}(J) \subseteq \mathrm{Undef}_P(I)$, it can be shown that $\mathsf{NegGen}(P, I, J)$ returns True if and only $I \cup J$ has a generating enumeration from I. Similarly, if J is a *set of literals* such that $\mathrm{atoms}(J) \subseteq \mathrm{Undef}_P(I)$,

[5] This was suggested by Jussi Rintanen (personal communication) in conjunction with cautious autoepistemic logic.

$\mathsf{Gen}(P, I, J)$ returns True if and only if $I \cup J$ has a generating enumeration from I. In addition, the calls $\mathsf{NegGen}(P, I, J)$ and $\mathsf{Gen}(P, I, J)$ involve at most $|J|^2$ subsequent calls to Cons. This implies that testing the generability of a set of literals can be accomplished in time polynomial to $|\mathrm{Hb}(P)|$.

```
Function NegGen(P, I, J): Boolean;
Begin
   If J = ∅ Then
      Return True
   Else Begin
      For ∼p ∈ J Do
         If Cons(P, I ∪ (J − {∼p}), p) Then
            Continue
         Else
            Return NegGen(P, I, J − {∼p});
      Return False
   End
End
```

```
Function Gen(P, I, J): Boolean;
Begin
   For p ∈ [J]⁺ Do
      If Not Cons(P, I ∪ [J]⁻, p) Then
         Return False;
   Return NegGen(P, I, [J]⁻)
End
```

The nondeterministic Turing-machine $\mathsf{Co\text{-}Cm}$ (see below) has been designed to accept the complement of **CM**. Given a partial interpretation I of P, the set of extensions J of I into a total interpretations $I \cup J$ of P is denoted by $\mathrm{Ext}(P, I)$. Note that each $J \in \mathrm{Ext}(P, I)$ satisfies $\mathrm{atoms}(J) = \mathrm{Undef}_P(I)$. We have collected important properties of the machine $\mathsf{Co\text{-}Cm}$ in Proposition 7.2. In particular, $\mathrm{CM}(P)$ is the least partial interpretation that passes the tests imposed by $\mathsf{Co\text{-}Cm}$. Therefore $\mathsf{Co\text{-}Cm}$ accepts exactly the complement of $\overline{\mathbf{CM}}$. Moreover, the computations of $\mathsf{Co\text{-}Cm}$ can be carried out in time polynomial to $|\mathrm{Hb}(P)|$. Thus **CM** is in the class **co-NP**.

```
Machine Co-Cm(P, l);
Begin
   Choose I From Int(P);
   If Not Gen(P, ∅, I) Then Reject;
   For q ∈ Hb(P) Do
      If (q ∈ I and Not Cons(P, I, q)) or
         (Not q ∈ I and Cons(P, I, q)) Then Reject;
   For q ∈ Undef_P(I) Do Begin
      Choose J From Ext(P, I);
      If Not Gen(P, I, J) or Not Cons(P, J, q) Then Reject
   End;
   If Not l ∈ I Then Accept Else Reject
End
```

The hardness of **CM** in **co-NP** can be shown by solving the satisfiability problem of propositional sentences using $\overline{\mathbf{CM}}$. The required polynomial-time transformation maps a propositional sentence ϕ into a logic program P_ϕ as follows. For each atom p in ϕ the rules $t_p \leftarrow \sim f_p$ and $f_p \leftarrow \sim t_p$ are included in P_ϕ. The truth and falsity of p is therefore represented by atoms t_p and f_p in P_ϕ. Moreover, rules describing the truth value of each subsentence of ϕ are added to P_ϕ. For example, the truth and falsity of a conjunction $\phi_1 \wedge \phi_2$ is handled by the rules $t_{\phi_1\wedge\phi_2} \leftarrow t_{\phi_1}, t_{\phi_2}$; $f_{\phi_1\wedge\phi_2} \leftarrow f_{\phi_1}$ and $f_{\phi_1\wedge\phi_2} \leftarrow f_{\phi_2}$. The size of P_ϕ is polynomial in $|\phi|$. In this setting, the enumerations of $\mathrm{Hb}(P)$ generate all valuations to the subsentences of ϕ. Thus $\sim t_\phi$ is included in $\mathrm{CM}(P_\phi)$ if and only if ϕ is unsatisfiable, i.e. $\langle P_\phi, \sim t_\phi\rangle \in \mathbf{CM}$ if and only if $\phi \notin \mathbf{SAT}$. In other words, **SAT** is polynomial time many-to-one reducible to $\overline{\mathbf{CM}}$.

Proposition 7.2 *If the machine* Co-Cm *accepts its input, (i) the chosen partial interpretation I has a generating enumeration, (ii) $p \in I$ if and only if $p \in \mathbf{B}_P(I)$, (iii) $p \in \mathbf{D}_P(I)$ implies $\sim p \in I$, (iv) $\sim p \in I$ implies $p \notin \mathbf{B}_P(I)$ and (v) $\mathrm{CM}(P) \subseteq I$.*

Theorem 7.3 *The set* **CM** *is* **co-NP***-complete.*

Given a logic program P, its well-founded model can be computed in time polynomial to $|P|$. As long as $\mathbf{P} \neq \mathbf{NP}$, Theorem 7.3 implies this is not possible with cautious models: the time requirement of computing such models grows exponentially in $|P|$. Moreover, the complexity results indicate that logic programs under the well-founded semantics and the cautious model semantics are different in expressive power. For example, in contrast to well-founded semantics, the unsatisfiability of propositional sentences can be expressed by a logic program under the cautious model semantics. On the other hand, the problem of determining the membership in the intersection of the stable models of a logic program P is also a **co-NP**-complete. This suggests that the stable model semantics and the cautious model semantics have a similar expressive power.

8 Relationship to Autoepistemic Logic

It is time to explain how cautious models originate from cautious autoepistemic logic [9]. In autoepistemic logic, a modal operator L is introduced to formalize *beliefs* of an agent. The sentences of the form $L\phi$ extend propositional language $\mathcal{L}$ to its autoepistemic counterpart $\mathcal{L}_{\mathrm{ae}}$. In autoepistemic logic, the main interest is what are the possible sets of beliefs $\Delta \subseteq \mathcal{L}_{\mathrm{ae}}$ of the agent given a set of premises as its initial assumptions. The premises of the agent refer to the beliefs of the agent trough the modality L. Moore [13] defines the sets of beliefs of the agent as *stable expansions* of the set of premises Σ, i.e. the fixed points of the equation $\Delta = \mathrm{Cn}(\Sigma \cup L\Delta \cup \neg L\overline{\Delta})$ where $L\Gamma = \{L\phi \mid \phi \in \Gamma\}$, $\neg\Gamma = \{\neg\phi \mid \phi \in \Gamma\}$ and $\overline{\Gamma} = \mathcal{L}_{\mathrm{ae}} - \Gamma$ for a subset Γ of $\mathcal{L}_{\mathrm{ae}}$. The logical consequence relation $\models_{\mathrm{ae}}$ of autoepistemic logic extends the one of propositional logic by treating sentences of the form $L\phi$ as atomic sentences. Stable models of a logic program P can be characterized in terms of stable expansions of a translation of P into a set of autoepistemic sentences:

Definition 8.1 ([8]) *Let P be a logic program. Define the translation $\mathrm{tr}_{\mathrm{LP,AEL}}(P)$ as the set of sentences $\{\neg Lq_1 \wedge ... \wedge \neg Lq_m \wedge p_1 \wedge ... \wedge p_n \rightarrow p \mid p \leftarrow p_1, ..., p_n, \sim q_1, ..., \sim q_m \in P\}$.*

By the results presented in [8], the stable models of a logic program P are in one-to-one correspondence with the stable expansions of $\mathrm{tr}_{\mathrm{LP,AEL}}(P)$. A similar connection can be established for cautious models and cautious autoepistemic logic. This logic is defined using enumeration-based introspection which is a construction analogous to that of Definition 3.1. Given a set of sentences $\Sigma \subseteq \mathcal{L}_{\mathrm{ae}}$ and an enumeration $\varepsilon = \psi_1, \psi_2, ...$ of autoepistemic sentences, a set of belief literals is constructed as follows. First of all, $\mathrm{EB}^{\varepsilon,0}(\Sigma)=\emptyset$. For each $i>0$ $\mathrm{EB}^{\varepsilon,i}(\Sigma)$ is formed by adding either the sentence $L\psi_i$ or the sentence $\neg L\psi_i$ to $\mathrm{EB}^{\varepsilon,i-1}(\Sigma)$ as an indication of belief or disbelief in ψ_i, respectively. The decision depends on whether ψ_i is a logical consequence of $\Sigma\cup\mathrm{EB}^{\varepsilon,i-1}(\Sigma)$ or not. The set of beliefs induced by ε is therefore $\mathrm{EB}^{\varepsilon,|\varepsilon|}(\Sigma)$ if ε is finite and $\bigcup_{i=0}^{\infty}\mathrm{EB}^{\varepsilon,i}(\Sigma)$ otherwise. Finally, define $\mathrm{EE}^{\varepsilon}(\Sigma)$ as $\{\phi\,|\,\Sigma\cup\mathrm{EB}^{\varepsilon}(\Sigma)\models_{\mathrm{ae}}\phi\}$. We may now define the notion of expansions used in cautious autoepistemic logic.

Definition 8.2 *Let Σ be a set of premises. A pair $\langle\Delta^+,\Delta^-\rangle$ is a cautious expansion of Σ if and only if $\Delta^+=\mathrm{Cn}(\Sigma\cup L\Delta^+\cup\neg L\Delta^-)$ and $\Delta^-=\{\phi\in\mathcal{L}_{\mathrm{ae}}|$ there is no enumeration ε of $\mathcal{L}_{\mathrm{ae}}-(\Delta^+\cup\Delta^-)$ such that $\phi\in\mathrm{EE}^{\varepsilon}(\Sigma\cup L\Delta^+\cup\neg L\Delta^-)\}$.*

Cautious expansions that satisfy the **N**-groundedness condition of Marek and Truszczyński [12] are of special interest because their intersection, i.e. the least **N**-grounded cautious expansion, can be iteratively constructed like $\mathrm{CM}(P)$. This unique cautious expansion is the one that coincides with $\mathrm{CM}(P)$ in the following way:

Theorem 8.3 *Let $\langle\Delta^+,\Delta^-\rangle$ be the least **N**-grounded cautious expansion of $\mathrm{tr}_{\mathrm{LP,AEL}}(P)$. Then $\mathrm{CM}(P)=\{p\in\mathrm{Hb}(P)\,|\,p\in\Delta^+\}\cup\{\sim p\in\sim\mathrm{Hb}(P)\,|\,p\in\Delta^-\}$.*

9 Conclusions

In this paper, a novel class of partial models for general logic programs is proposed. The *cautious model* $\mathrm{CM}(P)$ of a general logic program P can be constructed iteratively by a monotonic operator. In general, the cautious model $\mathrm{CM}(P)$ extends the well-founded model $\mathrm{WFM}(P)$. This is because a case analysis is performed when the false atoms of $\mathrm{CM}(P)$ are determined. Cautious model semantics differs from other extensions of the well-founded semantics such as extended well-founded semantics $\mathrm{EWFS}(P)$, generalized well-founded semantics $\mathrm{GWFS}(P)$, the semantics $\mathrm{WFS}^+(P)$, well-founded-by-case semantics $\mathrm{WFS}_{\mathrm{C}}(P)$ and O-semantics.

The cautious model $\mathrm{CM}(P)$ can be seen as an approximation of the intersection of stable models of a general logic program P – given such models exist. It seems that the cautious model provides a reasonable set of conclusions even if this is not the case. This is because $\mathrm{CM}(P)$ is always a partial model of P, i.e. it can be extended to a total model of P. In light of the complexity results, cautious models and stable models have a similar time complexity when the membership in the intersection of models is considered. Presented examples distinguish cautious model semantics from the various generalizations of the stable model semantics such as stationary model semantics, regular model semantics and stable class semantics. Cautious model semantics fails to solve the floating conclusions problem. On the other hand, it provides a more accurate partial model on other examples, due to its case analysis capability.

Cautious models can be characterized in terms of cautious autoepistemic logic. Other analogous approaches are the variants of autoepistemic logic presented by Bonatti [3] and Przymusinski [16]. However, they lead to partial models of general logic programs that coincide with the well-founded model. In contrast to these, the results of this paper indicate that even more accurate partial models can be devised by means of the autoepistemic framework.

An interesting subject for further research is improving the accuracy of cautious models. The major possibilities are (i) to carry out case analysis when the true atoms of the cautious model are determined and (ii) to use slightly different notions of derivability. It seems that the enumeration-based method can also be applied to larger classes of logic programs such as logic programs with classical negation and disjunctive logic programs.

Acknowledgements

The author thanks Dr Ilkka Niemelä and anonymous referees for comments and suggestions to improve the paper.

References

[1] C. Baral, J. Lobo, and J. Minker. Generalized well-founded semantics for logic programs. In M. E. Stickel, editor, *10th International Conference on Automated Deduction, LNAI 449, subseries LNCS*, pages 102–116. Springer, July 1990.

[2] C. R. Baral and V.S. Subrahmanian. Stable and extension class theory for logic programs and default logics. *Journal of Automated Reasoning*, 8:345–366, 1992.

[3] P.A. Bonatti. Autoepistemic logics as a unifying framework for the semantics of logic programs. In *Proceedings of the 1992 Joint International Conference and Symposium on Logic Programming*, Washington, D.C., USA, November 1992.

[4] K.L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, New York, 1978.

[5] J. Dix. A framework for representing and characterizing semantics of logic programs. In *Principles of Knowledge Representation and Reasoning, Proceedings of the 3rd International Conference*, pages 591–602. Morgan Kaufmann Publishers, October 1992.

[6] J. Dix. A classification theory of semantics of normal logic programs: I strong properties. *Fundamenta Informaticae*, 22:227–255, 1995.

[7] J. Dix. A classification theory of semantics of normal logic programs: II weak properties. *Fundamenta Informaticae*, 22:257–288, 1995.

[8] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proceedings of the 5th International Conference on Logic Programming*, pages 1070–1080, Seattle, USA, August 1988. The MIT Press.

[9] T. Janhunen and I. Niemelä. A scheme for weakened negative introspection in autoepistemic reasoning. In *Proceedings of the 3rd Kurt Gödel Colloquium on Computational Logic and Proof Theory*, pages 211–222, Brno, Czech Republic, August 1993. Springer-Verlag.

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

[11] J.W. Lloyd. *Foundations of Logic Programming.* Springer-Verlag, Berlin, 1984.

[12] W. Marek and M. Truszczyński. Modal logic for default reasoning. *Annals of Mathematics and Artificial Intelligence*, 1:275–302, 1990.

[13] R.C. Moore. Semantical considerations on nonmonotonic logic. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 272–279, Karlsruhe, FRG, August 1983. Morgan Kaufmann Publishers.

[14] L. M. Pereira, J. J. Alferes, and J. N. Aparício. Adding closed world assumptions to well-founded semantics. *Theoretical Computer Science*, 122:49–68, 1994.

[15] H. Przymusinska and T.C. Przymusinski. Stationary default extensions. In *Working Notes of the 4th International Workshop on on Nonmonotonic Reasoning*, pages 179–193, Plymouth, Vermont, USA, May 1992.

[16] T.C. Przymusinski. Autoepistemic logics of closed beliefs and logic programming. In *Proceedings of the 1st International Workshop on Logic Programming and Non-monotonic Reasoning*, pages 3–20, Washington, D.C., USA, July 1991. The MIT Press.

[17] 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, July 1991.

[18] J. You and L. Yuan. On the equivalence of semantics for normal logic programs. In *Proceedings of Workshop on Logic Programming with Incomplete Information*, pages 161–176, October 1993.

[19] J. You and L. Yuan. A three-valued semantics for deductive databases and logic programs. *Journal of Computer and System Sciences*, 49:334–361, 1994.