Complexity of computing with extended propositional logic programs

V. Wiktor Marek[1]
Arcot Rajasekar[2]
Mirosław Truszczyński[1]
Department of Computer Science
University of Kentucky
Lexington, KY 40506–0027
{marek, sekar, mirek}@ms.uky.edu

1 Introduction

In this paper we introduce the notion of an $\mathcal{F}$-program, where $\mathcal{F}$ is a collection of formulas, and study the complexity of computing with $\mathcal{F}$-programs. $\mathcal{F}$-programs can be regarded as a generalization of standard logic programs. Clauses of $\mathcal{F}$-programs are built of formulas from $\mathcal{F}$. In particular, formulas from $\mathcal{F}$ other than atoms are also allowed. Typical examples of $\mathcal{F}$ are the set of all atoms, the set of all literals, the set of all Horn clauses, the set of all clauses, the set of all clauses with at most two literals, the set of all clauses with at least three literals, etc. The notions of minimal and stable models [ABW88, GL88] of a logic program have natural generalizations to the case of $\mathcal{F}$-programs. The resulting notions are referred to in the paper as *minimal* and *stable* answer sets. We study the complexity of reasoning involving these notions. In particular, we establish the complexity of determining the existence of stable and consistent minimal answer sets, and determining the membership in some (or all) stable (or minimal) answer sets.

When $\mathcal{F}$ consists of all atoms, $\mathcal{F}$-programs coincide with standard logic programs. On the other end of the spectrum, when $\mathcal{F}$ is the class of all formulas, the formalism of $\mathcal{F}$-programs is equivalent to default logic [Rei80] in the sense that under a natural, *one-to-one* and *onto*, interpretation of $\mathcal{F}$-program clauses by defaults there is a correspondence between stable answer sets and extensions, and between minimal answer sets and minimal sets closed under default theories. The notion of an $\mathcal{F}$-program allows us to study how the complexity of the class $\mathcal{F}$ affects the complexity of reasoning with stable answer sets as we move from standard logic programs ($\mathcal{F}$ consists of atoms) to default logic ($\mathcal{F}$ consists of all formulas).

[1]This work was partially supported by by National Science Foundation under grant IRI-9012902, National Science Foundation and the Commonwealth of Kentucky EPSCoR program under grant RII 8610671
[2]This work was partially supported by National Science Foundation under grant CCR-9110721.

The complexity of problems involving stable models of standard logic programs is well understood [MT91, BF91]. Most of the problems of interest are located on the first level of the polynomial hierarchy (they are NP- or co-NP-complete). Analogous problems for extensions in full default logic are on the second level of the hierarchy (they are Σ_2^P- or Π_2^P-complete). We show here that, not surprisingly, if there is a polynomial algorithm which, given a finite set $X \subseteq \mathcal{F}$ and a formula $\varphi \in \mathcal{F}$ determines whether $X \vdash \varphi$, then the complexity of decision problems involving stable answer sets remains on the first level of the polynomial hierarchy. On the other hand, for several classes $\mathcal{F}$ of formulas for which the satisfiability problem is NP-complete, we show that the complexity of problems involving stable answer sets goes up and they become Σ_2^P- or Π_2^P-complete. These results strengthen the results of Gottlob [Got92] who investigated the complexity of full default logic without any syntactic restrictions on the class of formulas defaults are built of.

While the complexity of problems involving minimal models of logic programs is well understood [Got91, CS93] the complexity of reasoning with theories closed under default theories has not been investigated. In this paper we show that for every nontrivial class $\mathcal{F}$, that is, containing at least all atoms of the language, the problem to determine if a formula follows from some minimal answer set is NP-complete and the problem to determine if a formula follows from every minimal answer set is co-NP-complete. That is, even in the most complex case of default logic (when $\mathcal{F}$ consists of all formulas) they are not harder than the corresponding problems in the simplest case of standard logic programs. Not surprisingly, then, the main difficulty in proving these results lies in showing that they are in NP (co-NP). All proofs known to us so far rely on the relationship between minimal answer sets and modal logic S5. Reasoning with minimal answer sets is related to Generalized Closed World Assumption of Minker [Min82]. We will discuss the details of the relationship later in the paper.

Recently, both logic programming and default logic have been extended to facilitate reasoning with nonstandard disjunction [GL90, GLPT91]. We introduce here the class of disjunctive $\mathcal{F}$-programs and define for them the notions of minimal and stable answer sets. Under our definitions, $\mathcal{F}$-programs are just special disjunctive $\mathcal{F}$-programs. In addition, if $\mathcal{F}$ consists of atoms, disjunctive $\mathcal{F}$-programs coincide with disjunctive databases [GL90] and when $\mathcal{F}$ is the class of all formulas, disjunctive $\mathcal{F}$-programs are equivalent to disjunctive default theories [GLPT91]. In this paper we provide several results on the complexity of reasoning with disjunctive $\mathcal{F}$-programs. A straightforward way of verifying that a subset S of $\mathcal{F}$ is a stable answer set for a disjunctive $\mathcal{F}$-program P requires us to check that S is a minimal answer set for some positive $\mathcal{F}$-program P'. So far, no algorithm to accomplish that task and requiring only polynomially many calls to a propositional satisfiability checking procedure is known. Despite this, we are able to show that problems involving stable answer sets for disjunctive $\mathcal{F}$-programs reside on the second level of the polynomial hierarchy. Again, the technique is to use a relationship between stable answer sets and some modal logic, this time the nonmonotonic logic S4F [Tru91]. We also show that allowing disjunctions in the heads of $\mathcal{F}$-program clauses has essentially no effect on the complexity of reasoning with

minimal answer sets.

The paper contains several results on the complexity of reasoning with various classes of programs. But perhaps the most significant result of the work presented here is the apparent importance of the "modal logic connection" in establishing, as it turns out, nontrivial upper bounds on the complexity of some of the problems considered.

There are several problems involving stable answer sets whose complexity remains unresolved. We list some of them throughout the paper as open problems or conjectures.

2 Preliminaries

By $\mathcal{L}$ we denote a fixed language of propositional calculus with infinitely many atoms, and $\mathcal{F}$ always stands for a subset of $\mathcal{L}$ such that **every atom of $\mathcal{L}$ is in $\mathcal{F}$**.

In this section we provide definitions of $\mathcal{F}$-programs, disjunctive $\mathcal{F}$-programs, and stable and minimal answer sets. We leave to the reader a rather straightforward verification that if $\mathcal{F}$ is the class of all atoms in $\mathcal{L}$ then $\mathcal{F}$-programs are simply logic programs, and stable (minimal) answer sets coincide with stable (minimal) models of such programs. Similarly, disjunctive $\mathcal{F}$-programs reduce to disjunctive databases as defined in [GL90] and stable answer sets coincide with the notion of answer sets defined there. On the other hand, for every class $\mathcal{F}$, $\mathcal{F}$-programs (disjunctive $\mathcal{F}$-programs) can be interpreted as default theories (disjunctive default theories) in such a way that there is a perfect correspondence between stable (minimal) answer sets and extensions (minimal theories closed under defaults) [MT89, GL90, GLPT91].

Definition 2.1 An *$\mathcal{F}$-rule* is any expression of the form

$$\alpha \leftarrow \beta_1, \ldots, \beta_m, \mathbf{not}(\gamma_1), \ldots, \mathbf{not}(\gamma_n), \tag{1}$$

where α, β_i $1 \leq i \leq m$, γ_i, $1 \leq i \leq n$, are all in $\mathcal{F}$. An *$\mathcal{F}$-program* is any collection of $\mathcal{F}$-rules. □

With an $\mathcal{F}$-program we associate collections of theories closed under propositional consequence. These collections of theories specify the meaning of a program. Suppose that $\mathcal{C}$ is a class of theories closed under propositional consequence assigned to an $\mathcal{F}$-program P. If a formula belongs to all such theories it could be regarded as a "consequence" of the program. There is another possibility. Select one theory, say C, from $\mathcal{C}$ and regard φ as a "consequence" of P if φ belongs to C. For any of these two methods of reasoning on the basis of $\mathcal{F}$-programs to be of use, one must be able to characterize theories that should be assigned to a program and, even more importantly, one must have means to compute them. Theories closed under propositional consequence are infinite. Hence, they cannot be

represented explicitly in a computer. Instead, finite representations of these theories have to be used. A representation of a theory is not unique. Sometimes a specific representation can be selected. This is for instance the case when $\mathcal{F}$ consists of atoms and, as we will shortly see, for any $\mathcal{F}$ in the case of **not**-free *programs*.

With this in mind, throughout the paper instead of considering theories closed under provability we consider subsets of $\mathcal{F}$ that generate such theories. We specify these subsets using a procedural interpretation of $\mathcal{F}$-rules as "nonstandard" inference rules. This approach is already present in the next definition. Instead of defining the closure under an $\mathcal{F}$-rule for theories closed under propositional provability, we define this concept for subsets of $\mathcal{F}$.

Definition 2.2 A set $S \subseteq \mathcal{F}$ is *closed under* an $\mathcal{F}$-rule (1) if the following condition is satisfied:

$$\textbf{whenever } S \vdash \beta_i,\ 1 \leq i \leq m, \text{ and } S \nvdash \gamma_i,\ 1 \leq i \leq n, \textbf{ then } \alpha \in S. \tag{2}$$

A set S is *closed under* an $\mathcal{F}$-program P if S is closed under every $\mathcal{F}$-rule in P. □

There is a natural preorder relation $\preceq$ on the collection of subsets of $\mathcal{F}$. Namely, for every $S_1, S_2 \subseteq \mathcal{F}$, we define $S_1 \preceq S_2$ if $Cn(S_1) \subseteq Cn(S_2)$. We define $S_1 \prec S_2$ if $S_1 \preceq S_2$ and $Cn(S_1) \neq Cn(S_2)$. Given a collection $\mathcal{G}$ of subsets of $\mathcal{F}$, we say that $S \in \mathcal{G}$ is $\preceq$-*minimal* in $\mathcal{G}$ if there is no $S' \in \mathcal{G}$ such that $S' \prec S$. We say that S is $\preceq$-*least* in $\mathcal{G}$ if for every $S' \in \mathcal{G}$, $S \preceq S'$. Since $\preceq$ is not an order, the $\preceq$-least element may not be unique. However, all such theories have the same closure under propositional provability.

Proposition 2.1 *Let P be an $\mathcal{F}$-program. Then there exists a $\preceq$-minimal set $S \subseteq \mathcal{F}$ closed under P. If P is an $\mathcal{F}$-program without* **not**, *then there exists a $\preceq$-least set $S \subseteq \mathcal{F}$ closed under P.* □

We will prove a more general version of the first assertion of Proposition 2.3 below. The second part of the assertion is proved using the standard argument of [vEK76].

Let P be an $\mathcal{F}$-program without **not**. By $LEAST(P)$ we denote the set of all $\preceq$-least sets $S \subseteq \mathcal{F}$ closed under P, whose existence is guaranteed by Proposition 2.1. The well-known method of computing the least Herbrand model of a program in a "bottom-up" fashion [vEK76] can easily be extended to the case of $\mathcal{F}$-programs. Assume that $SAT_{\mathcal{F}}$ is a procedure which, given a finite set $X \subseteq \mathcal{F}$ and a formula $\varphi \in \mathcal{F}$ decides whether $X \vdash \varphi$. We have the following result.

Proposition 2.2 *There is a "bottom-up" algorithm which, given a finite $\mathcal{F}$-program P without occurrences of* **not**, *computes an element of $LEAST(P)$ and, assuming that each call to the procedure $SAT_{\mathcal{F}}$ is counted as 1, performs polynomially many operations.* □

Again, the argument of Proposition 2.2 modifies the usual algorithm of finding the least set of atoms satisfying a Horn program. An argument of Dowling and Gallier [DG84] can be used here. The number of calls to the procedure $SAT_{\mathcal{F}}$ is, in fact, linear, when this technique is adapted.

We will denote the $\preceq$-least subset of $\mathcal{F}$ constructed by the algorithm of Proposition 2.2 by *least*(P).

Definition 2.3 Let P be an $\mathcal{F}$-program and let $S \subseteq \mathcal{F}$. By the *S-reduct of P* we mean the program P^S obtained by removing from P each rule

$$\alpha \leftarrow \beta_1, \ldots, \beta_m, \mathbf{not}(\gamma_1), \ldots, \mathbf{not}(\gamma_n)$$

such that $S \vdash \gamma_i$ for some i, $1 \leq i \leq n$, and by removing from the body of every other rule all terms $\mathbf{not}(\gamma_i)$. □

Now, we are ready to define the notions of a *stable* and *minimal answer sets* for an $\mathcal{F}$-program P.

Definition 2.4 Let P be an $\mathcal{F}$-program and let $S \subseteq \mathcal{F}$.

1. If $S = \mathit{least}(P^S)$, then S is called a *stable answer set* for P. By $\mathit{STABLE}(P)$ we denote the family of all stable answer sets for P.
2. If S is a $\preceq$-minimal subset of $\mathcal{F}$ closed under P, then S is called a *minimal answer set* for P. By $\mathit{MIN}(P)$ we denote the family of all minimal answer sets for P. □

Our definitions can naturally be extended to the case of disjunctive $\mathcal{F}$-programs.

Definition 2.5 A *disjunctive $\mathcal{F}$-rule* is any expression of the form

$$\alpha_1 | \ldots | \alpha_k \leftarrow \beta_1, \ldots, \beta_m, \mathbf{not}(\gamma_1), \ldots, \mathbf{not}(\gamma_n), \tag{3}$$

where α, β_i $1 \leq i \leq m$, γ_i, $1 \leq i \leq n$, are all in $\mathcal{F}$. A *disjunctive $\mathcal{F}$-program* is any collection of disjunctive $\mathcal{F}$-rules. □

Intuitively, the operator | is interpreted as an "effective" disjunction (see [GL90, GLPT91]): if all the premises in the body of a rule are satisfied, then at least one of the formulas in the head has to be concluded. Two ways of making this intuition precise are given below. One leads to the notion of a minimal answer set, the other one yields the concept of a stable answer set.

Definition 2.6 A set $S \subseteq \mathcal{F}$ is *closed under* a disjunctive $\mathcal{F}$-rule (3) if the following condition is satisfied:

$$\textbf{whenever } S \vdash \beta_i,\ 1 \leq i \leq m, \text{ and } S \nvdash \gamma_i,\ 1 \leq i \leq n, \textbf{ then } \text{at least one } \alpha_i \in S. \quad (4)$$

A set S is *closed under* a disjunctive $\mathcal{F}$-program P if S is closed under every $\mathcal{F}$-rule in P. □

In the case of disjunctive $\mathcal{F}$-programs without **not** the $\preceq$-least sets closed under P need not exist (consider the program $\{a|b \leftarrow\}$). However, applying again Zorn Lemma to the preorder $\preceq$ we obtain the following result.

Proposition 2.3 *Let P be a disjunctive $\mathcal{F}$-program. Then there exists a $\preceq$-minimal set $S \subseteq \mathcal{F}$ closed under P.* □

Proof: Given $Z \subseteq \mathcal{F}$ we define:

$$Cl(Z) = Cn(Z) \cap \mathcal{F}. \quad (5)$$

Let $\mathcal{H} = \{Z : Cl(Z) = Z\}$. The elements of $\mathcal{H}$ are "saturated" with respect to propositional consequence restricted to elements of $\mathcal{F}$. Moreover

$$Cl(Cl(Z)) = Cl(Z) \text{ and } Cn(Cl(Z)) = Cn(Z).$$

Thus, $Z \preceq Cl(Z)$ and $Cl(Z) \preceq Z$.

Our next observation is that if Z is closed under a disjunctive rule C of the form (3) then so is $Cl(Z)$. Indeed, if $Cl(Z) \vdash \beta_i$ for for all i, $1 \leq i \leq m$, and $Cl(Z) \nvdash \gamma_i$ for all i, $1 \leq i \leq n$ then by (5) $Z \vdash \beta_i$ for all i, $1 \leq i \leq m$, and $Z \nvdash \gamma_i$ for for all i, $1 \leq i \leq n$. Since Z is closed under C, therefore for some i, $1 \leq i \leq k$, $\alpha_i \in Z$. But $Z \subseteq Cl(Z)$, thus $\alpha_i \in Cl(Z)$.

The above observation implies that if Z is closed under all the rules in P, then so is $Cl(Z)$. Let $\mathcal{G}$ be the subset of $\mathcal{H}$ consisting of sets closed under all clauses of the disjunctive program P. Clearly $\mathcal{G}$ is nonempty, as $\mathcal{F}$ itself is closed under all the rules of P.

Next, we will show that there is a minimal element of $\mathcal{G}$. we apply Zorn lemma to the poset $\langle \mathcal{G}, \subseteq \rangle$. To this end, let $\langle Z_\xi \rangle_{\xi<\eta}$ be a descending family of sets from $\mathcal{H}$, all Z_ξ, $\xi < \eta$, closed under all rules from P. That is:

1. For every ξ, $\xi < \eta$ implies $Z_\xi \in \mathcal{H}$;
2. For all ξ_1, ξ_2, $\xi_1 < \xi_2 < \eta$ implies that $Z_{\xi_2} \subseteq Z_{\xi_1}$;
3. For every ξ, $\xi < \eta$ implies that Z_ξ is closed under all rules in P.

We show that the set $W = \bigcap_{\xi<\eta} Z_\xi$ is closed under all the rules in P and W belongs to $\mathcal{H}$.

Let C be a rule of the form (3). Assume that for all i, $1 \leq i \leq m$, $W \vdash \beta_i$ and that for all i, $1 \leq i \leq n$, $W \nvdash \gamma_i$. If the ordinal η is nonlimit, $\eta = \zeta + 1$ then $W = Z_\zeta$ and so W is closed under the rule C. Hence assume that η is a limit ordinal. Clearly $W \subseteq Z_\xi$ for all ξ, $\xi < \eta$. Therefore for every ξ, $\xi < \eta$, $Z_\xi \vdash \beta_i$ for all i, $1 \leq i \leq m$.

Next, if $1 \leq i \leq n$ and $W \nvdash \gamma_i$ then there must be an ordinal ρ_i, $\rho_i < \eta$ such that for all $\xi > \rho_i$, $Z_\xi \nvdash \gamma_i$. Indeed, assume that for arbitrarily large ξ below η, $Z_\xi \vdash \gamma_i$. Since $Cl(Z_\xi) = Z_\xi$, $\gamma_i \in Z_\xi$. Since the family $\langle Z_\xi \rangle_{\xi<\eta}$ is descending, γ_i belongs to Z_ξ for all $\xi < \eta$. Thus $\gamma_i \in W$ and so $W \vdash \gamma_i$ – which is not the case.

Now, let $\rho = \max\{\rho_i : 1 \leq i \leq n\}$. Then the ordinal ρ has the property that for all ξ, $\rho < \xi < \eta$ there must be a natural number i_ξ, $1 \leq i_\xi \leq k$ such that $\alpha_{i_\xi} \in Z_\xi$. Since k is finite, there must be j_0, $1 \leq j_0 \leq k$ such that for every ξ, $\xi < \eta$, there must exist ζ, $\xi < \zeta < \eta$ such that $\alpha_{j_0} \in Z_\zeta$. In other words, there must be arbitrarily large ζ below η such that $\alpha_{j_0} \in Z_\zeta$.

Recall now that the family $\langle Z_\xi \rangle_{\xi<\eta}$ is descending. Therefore α_{j_0} belongs to all Z_ξ for $\xi < \eta$. Hence α_{j_0} belongs to W. Thus W is closed under C. Since C was an arbitrary rule in P, W is closed under P.

Moreover W itself belong to $\mathcal{H}$. Indeed, consider $Cl(W) = Cn(W) \cap \mathcal{F}$. We show that $W = Cl(W)$. Clearly, $W \subseteq Cl(W)$. Next, since $W \subseteq Z_\xi$ for all ξ, $\xi < \eta$, we have

$$Cl(W) \subseteq Cl(Z_\xi) = Z_\xi$$

Thus $Cl(W)$ is included in all Z_ξ, $\xi < \eta$. Hence $Cl(W) \subseteq \bigcap_{\xi<\eta} Z_\xi = W$. This shows that $W \in \mathcal{H}$, and so $W \in \mathcal{G}$.

All the above show that every descending well-ordered chain of elements of $\mathcal{G}$ can be bound from below by an element of $\mathcal{G}$. Thus $\mathcal{G}$ possesses a minimal element.

Now, let Z be a minimal element of $\mathcal{G}$. We claim that Z is a $\preceq$-minimal subset of $\mathcal{F}$ closed under all the rules form P. Indeed, assume that S is a subset of $\mathcal{F}$ such that $S \preceq Z$, S closed under all the rules in P. Then, as shown above, $Cl(S)$ is closed under all the rules from P. Now, since $S \preceq Z$, $Cn(S) \cap \mathcal{F} \subseteq Cn(Z) \cap \mathcal{F} = Z$. Thus $Cl(S) \subseteq Z$ and since $Cl(S) \in \mathcal{G}$, $Cl(S) = Z$. Therefore $Cn(S) = Cn(Z)$ and, in particular $Z \preceq S$. □

Let P be a disjunctive $\mathcal{F}$-program. A set $S \subseteq \mathcal{F}$ is called a *minimal answer set* for P if S is $\preceq$-minimal among sets closed under P. By $MIN(P)$ we denote the family of all minimal answer sets for P.

Definition 2.7 Let P be an $\mathcal{F}$-program and let $S \subseteq \mathcal{F}$. By the *S-reduct of P* we mean the program P^S obtained by removing from P each rule (3) such that $S \vdash \gamma_i$ for some i, $1 \leq i \leq n$, and by removing from the body of every other rule all conjuncts $\mathbf{not}(\gamma_i)$. □

Finally, we define the notion of a *stable answer set* for a disjunctive $\mathcal{F}$-program P.

Definition 2.8 Let P be a disjunctive $\mathcal{F}$-program and let $S \subseteq \mathcal{F}$. If $S \in MIN(P^S)$, then S is called a *stable answer set* for P. By $STABLE(P)$ we denote the family of all stable answer sets for the program P. □

3 Complexity of reasoning with stable answer sets

Several problems considered in this paper are shown to be Σ_2^P- or Π_2^P-complete. The proofs of these results use some well-known results providing examples of problems which are complete in these classes. In particular, let $\mathrm{QBF}_{2,\exists}$ denote the class of valid boolean formulas of the form

$$\exists p_1, \ldots, p_m \forall q_1, \ldots, q_n E, \tag{6}$$

where $\{p_1, \ldots, p_m\}$ and $\{q_1, \ldots, q_n\}$ are disjoint sets of propositional variables and E is a boolean formula built of $p_1, \ldots, p_m$ and $q_1, \ldots, q_n$. It is well known (see [GJ79]) that given a formula Φ of the form (6), the problem to decide whether $\Phi \in \mathrm{QBF}_{2,\exists}$ is Σ_2^P-complete.

We will now deal with the following problems:

S1($\mathcal{F}$) given a finite $\mathcal{F}$-program P, is there a stable answer set for P?

S2($\mathcal{F}$) given a finite $\mathcal{F}$-program P and a formula $\varphi \in \mathcal{F}$, is φ a logical consequence of some stable answer set of P?

S2′($\mathcal{F}$) given a finite $\mathcal{F}$-program P and a formula $\varphi \in \mathcal{F}$, is there a stable answer set for P which does not imply φ?

S3($\mathcal{F}$) given an $\mathcal{F}$-program P and a formula $\varphi \in \mathcal{F}$, is φ a logical consequence of every stable answer set of P?

Theorem 3.1 *Assume that $\mathcal{F}$ contains all atoms of the language $\mathcal{L}$. If there is a polynomial time algorithm which, given $\varphi \in \mathcal{F}$ and a finite set $X \subseteq \mathcal{F}$, decides whether $X \vdash \varphi$, then each of the problems S1($\mathcal{F}$), S2($\mathcal{F}$) and S2′($\mathcal{F}$) is NP-complete and the problem S3($\mathcal{F}$) is co-NP-complete.*

Proof: Consider the problem S1($\mathcal{F}$). One can non-deterministically guess a stable answer set S, compute the reduct P^S and use the algorithm in Proposition 2.2 to verify whether S is a stable answer set for P. Since *least*(P) for a **not**-free $\mathcal{F}$-program can be computed with polynomial number of calls to the procedure $SAT_{\mathcal{F}}$ (Proposition 2.2), it follows that the problem S1($\mathcal{F}$) is in NP. Similar reasoning can be used to show that S2($\mathcal{F}$) and S2′($\mathcal{F}$) are in NP. Since S3($\mathcal{F}$) is the complement of S2′($\mathcal{F}$) it follows that S3($\mathcal{F}$) is co-NP.

The hardness of the result can be seen as follows. Marek and Truszczynski [MT91] have shown that the completeness of S1($\mathcal{A}$) is NP-complete when $\mathcal{A}$ is the set of atoms in

$\mathcal{L}$. Since $\mathcal{F}$ always contains the set of all atoms of the propositional language $\mathcal{L}$, the NP-completeness of the general version of S1($\mathcal{F}$) follows. Completeness of S2($\mathcal{F}$) follows from the completeness of S1($\mathcal{F}$). Assuming otherwise, if S2($\mathcal{F}$) is polynomial then a polynomial-time algorithm can be defined for solving S1($\mathcal{F}$) based on the algorithm for S2($\mathcal{F}$), implying that S1($\mathcal{F}$) cannot be NP-complete. (Such an algorithm would start by checking for the existance of an inconsistent stable answer set for P and if no such answer set exists, it would try to solve the problem of S2($\mathcal{F}$) for every formula from $\mathcal{F}$ occurring in P.) (Such an algorithm would start by checking for the existance of an inconsistent stable answer set for P and if no such answer set exists, it would check whether a is a logical consequence of some stable answer set of $P \cup \{a\}$ where a is an atom in $\mathcal{F}$ which does not occur in P.) Hence, S2($\mathcal{F}$) is NP-complete. Similar argument can be used to show the NP-completeness of S2$'$($\mathcal{F}$). Since, S3($\mathcal{F}$) is the complement of S2$'$($\mathcal{F}$), its co-NP-completeness follows. □

If $\mathcal{F}$ is the class of all formulas, then the formalism of $\mathcal{F}$-programs with stable answer sets is isomorphic to default logic of Reiter. The complexity of problems involving extensions of default theories was studied by Gottlob [Got92]. In our terminology, his results state that the problems S1($\mathcal{F}$), S2($\mathcal{F}$), S2$'$($\mathcal{F}$) are Σ_2^P-complete and the problem S3($\mathcal{F}$) is Π_2^P-complete. It turns out that the results of Gottlob can be strengthened. They remain true even if we restrict $\mathcal{F}$ to consist of clauses with at most three elements. The proof by Gottlob relies on a polynomial reduction of the problem QBF$_{1,\exists}$ to S1($\mathcal{F}$). The default theory constructed by Gottlob is not built of clauses. Hence, his argument cannot directly be applied to our problem for $\mathcal{F}$ consisting of clauses and additional technical means are needed.

Let φ be a formula and let P be a consistent set of literals. Define $\varphi(P)$ recursively, as follows:

1. If φ is a literal then put
$$\varphi(P) = \begin{cases} \top & \text{if } \varphi \in P \\ \varphi & \text{if } \varphi \notin P; \end{cases}$$
2. If $\varphi = \neg\psi$, then define $\varphi(P) = \neg\psi(P)$.
3. If $\varphi = \psi_1 \circ \psi_2$, where $\circ$ stands for a binary boolean connective, then define $\varphi(P) = \psi_1(P) \circ \psi_2(P)$;

We have the following simple property of the formula $\varphi(P)$:

(P0) Let P be a consistent set of literals. For every valuation v such that for every $p \in P$, $v(p) = 1$, $v(\varphi) = v(\varphi(P))$. In particular, if $\varphi \in Cn(W)$ then $\varphi(P) \in Cn(W \cup P)$, for every theory W.

Let E be any boolean formula built of $p_1, \ldots, p_m$ and $q_1, \ldots, q_n$ (we assume that $p_i \neq q_j$, for $1 \leq i \leq m$ and $1 \leq j \leq n$). Let $E^{\leq}$ be the set of all subformulas of E. For every $\varphi \in E^{\leq}$, let x_φ be a new atom. We will define now a theory $T(E)$ recursively, as follows.

1. If $E = r$, where r is an atom, then define $T(E) = \{x_E \equiv r\}$;
2. If $E = \neg E_1$, then define $T(E) = T(E_1) \cup \{x_E \equiv \neg x_{E_1}\}$;
3. If $E = E_1 \circ E_2$, where $\circ$ stands for a binary boolean connective, then $T(E) = T(E_1) \cup T(E_2) \cup \{x_E \equiv (x_{E_1} \circ x_{E_2})\}$;

The theory $T(E)$ has several useful properties. They are self-evident and we omit their proofs.

(P1) $T(E)$ has size (measured as the total length of all formulas in $T(E)$) which is linear in the size of E;

(P2) $T(E)$ can be represented by a set of clauses, each consisting of at most three literals, with a total length of all clauses linear in the size of E (for example, we can replace formulas of the form $x_E \equiv \neg x_{E_1}$ by a pair of clauses $(\neg x_E \vee \neg x_{E_1})$ and $(x_E \vee x_{E_1})$;

(P3) For every consistent subset P of $\{p_1, \ldots, p_m \neg p_1, \ldots, \neg p_m\}$, $T(E) \cup P$ is consistent;

(P4) $x_E \equiv E \in Cn(T(E))$.

Theorem 3.2 *Let $\mathcal{F}$ be any class of formulas containing all clauses consisting of at most three literals. Then each of the problems S1($\mathcal{F}$), S2($\mathcal{F}$) and S2$'$($\mathcal{F}$) is Σ_2^P-complete and the problem S3($\mathcal{F}$) is Π_2^P-complete.*

Proof: Even if $\mathcal{F}$ is the class of all formulas, the problems $S1(\mathcal{F})$, $S2(\mathcal{F})$ and $S2'(\mathcal{F})$ are in the class Σ_2^P and the problem $S3(\mathcal{F})$ is in the class Π_2^P (see [Got91]). Hence, the upper estimates for the complexities of the problems discussed in the assertion of the theorem hold true.

We will provide arguments for the "hardness" parts now. We will first show that the problem whether a formula Φ of the form (6) is in $\mathrm{QBF}_{2,\exists}$ is polynomially reducible to the problem $S1(\mathcal{F})$, where $\mathcal{F}$ is a class of formulas containing all clauses of at most three elements.

Let Φ be a formula of the form (6). By (P2) we can assume that $T(E)$ is represented by a set of clauses of at most three literals each and with the total size linear in the size of E.

For each clause $\varphi \in T(E)$ define

$$C_\varphi = (\varphi \leftarrow).$$

Next, define an $\mathcal{F}$-program Q by

$$Q = \{C_\varphi : \varphi \in T(E)\} \cup \{p_i \leftarrow \mathbf{not}(\neg p_i),\ \neg p_i \leftarrow \mathbf{not}(p_i) : 1 \leq i \leq m\} \cup \{\bot \leftarrow \mathbf{not}(x_E)\}.$$

Clearly, the program Q can be constructed in polynomial time. We will prove now that $\Phi \in \mathrm{QBF}_{2,\exists}$ if and only if Q has a stable answer set.

First, assume that $\Phi \in \mathrm{QBF}_{2,\exists}$. Let $P \subseteq \{p_1, \ldots, p_m \neg p_1, \ldots, \neg p_m\}$ be consistent and complete and such that $E(P)$ is a tautology.

Consider a set $U = T(E) \cup P$. Since $(x_E \equiv E) \in Cn(T(E))$ (property (P4)), it follows (property (P0)) that

$$(x_E \equiv E)(P) = x_E \equiv E(P) \in Cn(T(E) \cup P).$$

Since $E(P)$ is a tautology,

$$x_E \in Cn(T(E) \cup P).$$

Consequently,

$$Q^U = \{C_\varphi \colon \varphi \in T(E)\} \cup \{p \leftarrow \colon p \in P\}.$$

Hence, U is the least subset of $\mathcal{F}$ closed under the rules from Q^U. Consequently, U is a stable answer set for Q.

Conversely, assume that U is an stable answer set for the program Q. Assume U is inconsistent. Then

$$Q^U = \{C_\varphi \colon \varphi \in T(E)\}.$$

Since $T(E)$ is consistent, the least subset of $\mathcal{F}$ closed under the clauses in Q^U is consistent. Consequently, U is not the least subset of $\mathcal{F}$ closed under Q^U, a contradiction. It follows that U is consistent. Hence, $U \vdash x_E$. In addition, by the definition of Q, we have

$$U = T(E) \cup P.$$

for some complete and consistent set $P \subseteq \{p_1, \ldots, p_m \neg p_1, \ldots, \neg p_m\}$. Hence,

$$T(E) \cup P \vdash x_E.$$

We also have $T(E) \vdash x_E \equiv E$. Consequently, $T(E) \cup P \vdash E(P)$. Assume that $E(P)$ is not a tautology. Then there is a valuation w of $q_1, \ldots, q_n$ such that $w(E(P)) = 0$. Since $P \cap Q = \emptyset$, w can be extended to a valuation W' of $\{p_1, \ldots, p_m\} \cup \{q_1, \ldots, q_n\} \cup \{x_F \colon F \in E^{\leq}$ so that $w'(T(E) \cup P) = 1$. It follows that $T(E) \cup P \nvdash E(P)$, a contradiction. Hence, $E(P)$ is a tautology. □

Conjecture 3.3 *Assume that it is NP-complete to answer whether the theory $\{\neg\varphi\} \cup X$ is satisfiable, where $\varphi \in \mathcal{F}$ and $X \subseteq \mathcal{F}$. Then each of the problems S1($\mathcal{F}$), S2($\mathcal{F}$) and S2$'$($\mathcal{F}$) is Σ_2^P-complete and the problem S3($\mathcal{F}$) is Π_2^P-complete.* □

4 Complexity of reasoning with stable answer sets — the case of disjunctive $\mathcal{F}$-programs

In this section we investigate the complexity of problems S1($\mathcal{F}$), S2($\mathcal{F}$), S2$'$($\mathcal{F}$) and S3($\mathcal{F}$) in the case of disjunctive $\mathcal{F}$-programs. One might expect that extending the class of programs will result in higher complexity than in the case when disjunctions are not allowed (Theorem 3.2). As we have seen (Proposition 2.2), in the case of disjunction-free $\mathcal{F}$-programs without **not** there is an algorithm, using polynomially many calls to the procedure for checking propositional provability, that computes an element of $LEAST(P)$. It immediately follows from this fact that problems S1($\mathcal{F}$), S2($\mathcal{F}$), S2$'$($\mathcal{F}$) and S3($\mathcal{F}$) for disjunction-free programs are on the second level of the polynomial hierarchy. For example, in the case of the problem S1($\mathcal{F}$), given a disjunction-free $\mathcal{F}$-program, one first nondeterministically guesses a stable answer set S, then computes the reduct and finally uses the algorithm of Proposition 2.2 to verify that S is indeed a stable answer set for P. It is easy to see that all these steps can be accomplished by means of polynomially many calls to the procedure for checking propositional provability. In the case of disjunctive programs, however, the situation gets complicated. The problem if $S \in MIN(P^S)$ does not seem to be in the class Δ_2^P and the same argument that worked before cannot be adapted easily to the present case. It turns out, though, that we can prove that the problems S1($\mathcal{F}$), S2($\mathcal{F}$), S2$'$($\mathcal{F}$) and S3($\mathcal{F}$) remain on the second level of the hierarchy if we use the embedding of $\mathcal{F}$-programs in the nonmonotonic modal logic S4F and the results of [Tru91]. The "hardness" part is now trivial, as it follows directly from Theorem 3.2 (recall that $\mathcal{F}$-programs form a subclass of disjunctive $\mathcal{F}$-programs).

The translation mentioned above assigns to a clause C of the form (3) the modal formula

$$emb(C) = L\beta_1 \wedge \ldots \wedge L\beta_m \wedge LM\neg\gamma_1 \wedge \ldots \wedge LM\neg\gamma_n \supset L\alpha_1 \vee \ldots \vee L\alpha_k.$$

Given an $\mathcal{F}$-program P, we define $emb(P) = \{emb(C) : C \in P\}$. Informally, the result of [Tru91] can be stated as follows.

Theorem 4.1 *Let P be a disjunctive $\mathcal{F}$-program. A set $S \subseteq \mathcal{F}$ is a stable answer set for P if and only if the stable theory generated by S is an* S4F*-expansion for $emb(P)$.*

Since it takes polynomial time to construct $emb(P)$, and since the problems involving S4F-expansions can be shown to be on the second level of the polynomial hierarchy, the following result follows.

Theorem 4.2 *Let $\mathcal{F}$ be any class of formulas containing all clauses of at most three literals. Then problems S1($\mathcal{F}$), S2($\mathcal{F}$) and S2$'$($\mathcal{F}$), for the case of disjunctive $\mathcal{F}$-programs, are Σ_2^P-complete and the problem S3($\mathcal{F}$) for the case of disjunctive $\mathcal{F}$-programs is Π_2^P-complete.* □

It seems likely that the complexity of problems discussed in Theorem 4.2 does not go down even if we restrict to standard logic programs. That is, we conjecture that the assertion of Theorem 4.2 remains true even if $\mathcal{F}$ consists of the atoms of the language only.

5 Complexity of problems involving minimal answer sets

In this section we will investigate the complexity of reasoning with minimal answer sets. It turns out that in this case, allowing disjunctions does not have any effect on the complexity of reasoning. We study the following problems:

M1($\mathcal{F}$) given a finite (disjunctive) $\mathcal{F}$-program P, is there a *consistent* minimal answer set for P?

M2($\mathcal{F}$) given a finite (disjunctive) $\mathcal{F}$-program P and a formula $\varphi \in \mathcal{F}$, is there a minimal answer set for P which does not imply φ?

M3($\mathcal{F}$) given a finite (disjunctive) $\mathcal{F}$-program P and a formula $\varphi \in \mathcal{F}$, is φ a logical consequence of every minimal answer set of P?

We have here the same situation as in the previous section. That is, proving upper bounds for the complexity turns out to be more difficult than proving hardness. Again resorting to a modal logic, in this case the modal logic S5, provides a solution. Given a disjunctive $\mathcal{F}$-program clause C

$$\alpha_1|\ldots|\alpha_k \leftarrow \beta_1,\ldots,\beta_m,\mathbf{not}(\gamma_1),\ldots,\mathbf{not}(\gamma_n), \tag{7}$$

define

$$emb'(C) = L\beta_1 \wedge \ldots L\beta_m \wedge \neg L\gamma_1 \wedge \ldots \wedge \neg L\gamma_n \supset L\alpha_1 \vee \ldots L\alpha_k. \tag{8}$$

Given an $\mathcal{F}$-program P, define

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

We have the following lemma.

Lemma 5.1 *Let P be a disjunctive $\mathcal{F}$-program. A set $U \subseteq \mathcal{F}$ is a minimal answer set for P if an only if $ST(U)$ is a $\sqsubseteq$-minimal stable theory containing $emb(P)$.*

Proof: U is closed under clauses in P if and only if $emb(P) \subseteq ST(U)$. Hence, the assertion follows. □

We will also need the following property of stable theories proved by McDermott [McD82].

Lemma 5.2 *Let I be a theory in the modal language and let φ be a modal formula. We have $I \vdash_{\mathrm{S5}} \varphi$ if and only if $\varphi \in T$, for every stable theory T containing I. In particular, if φ is modal-free, $I \vdash_{\mathrm{S5}} \varphi$ if and only if $\varphi \in T$ for every $\sqsubseteq$-minimal stable theory containing I.*

Finally, we will utilize the following complexity result by Ladner [Lad77].

Lemma 5.3 *The problem to decide whether for a finite modal theory I and a modal formula φ, $I \nvdash_{\mathrm{S5}} \varphi$, is NP-complete.*

Theorem 5.4 *For every class $\mathcal{F}$ of formulas from $\mathcal{L}$ problems M2($\mathcal{F}$) and M1($\mathcal{F}$) are in the class* NP *and problem M3($\mathcal{F}$) is in class co-NP.*

Proof: Observe that the formula $\bot$ is not implied by some minimal answer set for an $\mathcal{F}$-program P if and only if P has a consistent answer set (hence, a consistent minimal answer set). Consequently, problem M1($\mathcal{F}$) is polynomially reducible to M2($\mathcal{F}$). Notice also that the set of YES instances of the problem M3($\mathcal{F}$) coincides with the set of NO instances of the problem M2($\mathcal{F}$). Hence, to prove the assertion, it is enough to show that M2($\mathcal{F}$) is in NP.

Let φ be a formula from $\mathcal{F}$. By Lemmas 5.1 and 5.2, it follows that there is a minimal answer set U for P such that $U \nvdash \varphi$ if and only if $emb'(P) \nvdash_{\mathrm{S5}} \varphi$. Since $emb'(P)$ can be constructed in polynomial time, it follows from the resutls of Ladner [Lad77] (Lemma 5.3) that the problem M2($\mathcal{F}$) is in NP. □

Theorem 5.5

1. *If $\mathcal{F}$ is consistent (in particular, if $\mathcal{F}$ is the set of all atoms of the language, the set of all positive clauses, the set of all program Horn clauses) then the problem M1($\mathcal{F}$) has always answer YES.*
2. *If $\mathcal{F}$ contains the class of all literals then the problem M1($\mathcal{F}$) is NP-complete (even for programs without disjunctions).*
3. *If $\mathcal{F}$ contains the class of all atoms of the language (in particular, if $\mathcal{F}$ is the class of all atoms of the language) then the problem M2($\mathcal{F}$) is NP-complete and the problem M3($\mathcal{F}$) is co-NP-complete (even for programs without disjunctions).*

Proof: Assertion (1) follows from Proposition 2.3.

Assertion (2) .....

To prove assertion (3) it suffices to show NP-completeness of M2($\mathcal{F}$) in the case when $\mathcal{F}$ is the set of atoms. The problem is in NP by Theorem 5.4. To prove NP-hardness we will show that the satisfiability problem is polynomially reducible to M2(At). Let $C_1, \ldots, C_k$ be a set of clauses, say

$$C_i = b^i_1 \vee \ldots \vee b^i_{m_i} \vee \neg c^i_1 \vee \ldots \vee \neg c^i_{n_i},$$

where b^i_j, $1 \leq j \leq m_i$, and c^i_j, $1 \leq j \leq n_i$, are atoms.

Let p be a new atom. For $i = 1, \ldots, k$ form an At-clause C'_i

$$p \leftarrow c^i_1, \ldots, c^i_{n_i}, \mathbf{not}(b^i_1), \ldots, \mathbf{not}(b^i_{m_i}).$$

It is easy to check that there exists a minimal answer set for $\{C'_i : i = 1, \ldots, k\}$ not implying p (that is, not containing p) if and only if the set of clauses $\{C_1, \ldots, C_k\}$ is satisfiable. □

Let us compare Theorem 5.5 with the results on the complexity of Generalized Closed World Assumption or GCWA (see [Min82] for the definition). Let us consider the case when $\mathcal{F}$ consists of the atoms of the language $\mathcal{L}$. In such case the notion of a minimal answer set coincides with the notion of a minimal model. In addition, an atom is true in a model M (where a model is represented by a set of atoms) if and only if it is a logical consequence of M. Consequently, as long as we are interested in the status of atoms, there is no difference between GCWA and our approach. However, when we ask about literals, the situation changes. If p is an atom such that $p \notin M$, then it is not the case that the literal $\neg p$ is a logical consequence of M. On the other hand, in GCWA we use M as a representation of a model. Clearly, since $p \notin M$, $\neg p$ is true in the model M. Hence, the two formalisms are different. In fact, as Theorem 5.5 implies, the consequence operator assigning to a program the collection of all formulas that are logical consequences of all minimal answer sets for a program is monotonic, while the consequence operator of GCWA is not. Not surprisingly, the complexity of GCWA is higher. As proved by Gottlob [Got91], the problem whether a literal is true in every minimal model of a propositional theory is Π^P_2-hard.

Finally, let us mention that we were not able to establish the complexity of the following problem. Given a finite (disjunctive) $\mathcal{F}$-program P and a formula $\varphi \in \mathcal{F}$, It is unclear that it belongs to NP (it is quite likely harder) since to verify that the answer is YES for a given input instance it is not enough to guess a subset U of $\mathcal{F}$ but also to verify that it is a minimal answer set. is φ a logical consequence of some minimal answer set of P?

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. Papers from the workshop held in Washington, D.C., August 18–22, 1986*, pages 89–142, Palo Alto, CA, 1988. Morgan Kaufmann.

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

[CS93] M. Cadoli and M. Schaerf. A survey of complexity results for non-monotonic logics. *Journal of Logic Programming*, 17(2-4):127–160, 1993.

[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.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and intractability. A guide to the theory of NP-completeness*. W.H. Freeman and Co., San Francisco, Calif., 1979.

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

[GL90] M. Gelfond and V. Lifschitz. Logic programs with classical negation. In D. Warren and P. Szeredi, editors, *Logic programming (Jerusalem, 1990)*, MIT Press Series in Logic Programming, pages 579–597, Cambridge, MA, 1990. MIT Press.

[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.

[Got91] G. Gottlob. Propositional circumscription and extended closed world reasoning are π_2^p-complete. Technical Report CD-TR91/20, Institut für Informationssysteme, Technische Universität Wien, 1991.

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

[Lad77] R. E. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM Journal on Computing*, 6(3):467–480, 1977.

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

[Min82] J. Minker. On indefinite databases and the closed world assumption. In *6th conference on automated deduction (New York, 1982)*, volume 138 of *Lecture Notes in Computer Science*, pages 292–308, Berlin-New York, 1982. Springer.

[MT89] 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.

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

[Tru91] M. Truszczyński. Modal interpretations of default logic. In *Proceedings of IJCAI-91*, pages 393–398. Morgan Kaufmann, 1991.

[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.