An application of proof-theory in ASP

V. W. Marek[1] and J.B. Remmel[2]

[1] Department of Computer Science
University of Kentucky
Lexington, KY 40506-0046, USA
[2] Department of Mathematics
University of California
La Jolla, CA 92093

Abstract. Using a characterization of stable models of logic programs P as satisfying valuations of a suitably chosen propositional theory, called the set of *reduced defining equations* rEq_P, we show that the finitary character of that theory rEq_P is equivalent to a certain continuity property of the Gelfond-Lifschitz operator GL_P associated with the program P.

1 Introduction

The use of proof theory in logic based formalisms for constraint solving is pervasive. For example, in Satisfiability (SAT), proof theoretic methods are used to find lower bounds on complexity of various SAT algorithms. Proof theoretic methods has not played as prominent role in Answer Set Programming (ASP) formalisms. This is not to say that there were no attempts to apply proof-theoretic methods in ASP. To give a few examples, Marek and Truszczynski in [MT93] used the proof-theoretic methods to characterize Reiter's extensions in Default Logic (and thus stable semantics of logic programs). Bonatti [Bo04] and separately Milnikel [Mi05] devised non-monotonic proof systems to study skeptical consequences of programs and default theories. Lifschitz [Li96] used proof-theoretic methods to approximate well-founded semantics of logic programs. Bondarenko et.al. [BTK93] studied an approach to stable semantics using methods with a clear proof-theoretic flavor. Marek, Nerode, and Remmel in a series of papers, [MNR90a,MNR90b,MNR91,MNR92,MNR94a,MNR94b], developed proof theoretic methods to study what they termed *non-monotonic rule systems* which have as special cases almost all ASP formalisms that have been seriously studied in the literature. Nevertheless, there is no clear classification of proof systems for nonmonotonic reasoning analogous to that present in classical logic, and SAT in particular.

In this paper, we define a notion of P-proof schemes, which is a kind of a proof system that was previously used by Nerode and the authors to study complexity issues for stable semantics of logic programs [MNR94a]. This proof system abstracts of M-proofs of [MT93] and produces Hilbert-style proofs. The nonmonotonic character of our P-proofs is provided by the presence of guards, called the *support* of the proof scheme, to insure context-dependence. A different but equivalent, presentation of proof schemes, using a guarded resolution is also possible.

We shall show that we can use P-proof schemes to find a characterization of stable models via *reduced defining equations*. While in general these defining equations may be infinitary, we study the case of programs for which all these equations are finitary. This resulting class of programs, called FSP-programs, turn out to be characterized by a form of continuity of the Gelfond-Lifschitz operator.

The outline of this paper is as follows. In section 2, we provide the necessary background on logic programs and stable model to present our results. In section 3, we introduce P-proof schemes and the reduced defining equations for a logic program P as well as the associated equivalence theorems. In section 4, we discuss the continuity properties of operators. Finally in section 5, we suggest some possible extensions of our techniques to the context of certain programs with cardinality constraints.

2 Preliminaries

Let At be a countably infinite set of atoms. We will study programs consisting of clauses built of the atoms from At. A *program clause* C is a string of the form

$$p \leftarrow q_1, \ldots, q_m, \neg r_1, \ldots, \neg r_n \tag{1}$$

m or n or both can be 0. The atom p will be called the head of C and denoted $head(C)$. The set $\{q_1, \ldots, q_m\}$ will be denoted $PosBody(C)$. The set $\{r_1, \ldots, r_n\}$ will be denoted by $NegBody(C)$. Let us stress that this last set is a set of atoms, not a set of negated atoms as sometimes is used in the literature. A normal propositional program is a set P of such clauses. For any $M \subseteq At$, we say that M is model of C if whenever $q_1, \ldots, q_n \in M$ and $\{r_1, \ldots, r_m\} \cap M = \emptyset$, then $p \in M$. We say that M is a model of a program P if M is a model of each clause $C \in P$. Horn clauses are clauses with no negated literals, i.e. clauses of the form (1) where $m = 0$. We will denote by $Horn(P)$ the part of the program P consisting of its Horn clauses. Horn programs are logic programs P consisting entirely of Horn clauses. Thus for a Horn program P, $P = Horn(P)$.

Each Horn program P has a least model in the Herbrand base and the least model of P is the least fixed point of a continuous operator T_P representing 1-step Horn clause logic deduction ([L89]). That is, for any set $I \subseteq At$, we let $T_P(I)$ equal the set of all $p \in At$ such that there is a clause $C = p \leftarrow q_1, \ldots, q_n$ in P such that $q_1, \ldots, q_n \in I$. Then T_P has a least fixed point F_P which is obtained by iterating T_P starting at the empty set for ω steps, i.e., $F_P = \bigcup_{n \in \omega} T_P^n(\emptyset)$ where for any $I \subseteq At$, $T_P^0(I) = I$ and $T_P^{n+1}(I) = T_P(T_P^n(I))$. Then F_P is the least model of P.

The semantics of interest for us is the *stable semantics* of normal programs, although we will discuss some extensions in Section 5. The stable models of a program P are defined as fixed points of the operator $T_{P,M}$. If P is a program and $M \subseteq At$ is a subset of the Herbrand base, define operator $T_{P,M} : \mathcal{P}(At) \rightarrow \mathcal{P}(At)$ as follows:

$$\begin{aligned} T_{P,M}(I) = \{p\colon & \text{ there exist a clause } C = p \leftarrow q_1, \ldots, q_n, \neg r_1, \ldots, \neg r_m \\ & \text{in } P \text{ such that } q_1 \in I, \ldots, q_n \in I, r_1 \notin M, \ldots, r_m \notin M\}\} \end{aligned}$$

The following is immediate, see [Ap90] for unexplained notions.

Proposition 1. *For every program P and every M of atoms the operator $T_{P,M}$ is monotone and continuous.*

Thus the operator $T_{P,M}$ possesses a least fixed point $F_{P,M}$.

Given program P and $M \subseteq At$, we define the *Gelfond-Lifschitz reduct* of P, P_M, as follows. For every clause $C = p \leftarrow q_1, \ldots, q_n, \neg r_1, \ldots, \neg r_m$ of P, execute the following operations.
(1) If some atom r_i belongs to M, then eliminate C altogether.
(2) In the remaining clauses that have not been eliminated by operation (1), eliminate all the negated atoms.
The resulting program P_M is a Horn propositional program. The program P_M possesses a least Herbrand model. If that least model of P_M coincides with M, then M is called a *stable model* for P. This gives rise to an operator GL_P which associates to each $M \subseteq At$, the least fixed point of $T_{P,M}$. We will discuss the operator GL_P and its proof-theoretic connections in section 4.2.

3 Proof schemes and reduced defining equations

In this section we recall the notion of a *proof scheme* and introduce a related notion of *defining equations*.

Given a propositional logic program P, a proof scheme is defined by induction on its length. Specifically, a proof scheme w.r.t. P (in short P-proof scheme) is a sequence $S = \langle\langle C_1, p_1\rangle, \ldots, \langle C_n, p_n\rangle, U\rangle$ subject to this conditions:
(I) when $n = 1$, $\langle\langle C_1, p_1\rangle, U\rangle$ is a P-proof scheme if $C_1 \in P$, $p = head(C_1)$, $PosBody(C_1) = \emptyset$, and $U = NegBody(C_1)$ and
(II) when $\langle\langle C_1, p_1\rangle, \ldots, \langle C_n, p_n\rangle, U\rangle$ is a P-proof scheme,
$C = p \leftarrow PosBody(C), \neg NegBody(C)$ is a clause in P, and $PosBody(C) \subseteq \{p_1, \ldots, p_n\}$, then

$$\langle\langle C_1, p_1\rangle, \ldots, \langle C_n, p_n\rangle, \langle C, p\rangle, U \cup NegBody(C)\rangle$$

is a P-proof scheme.
If $S = \langle\langle C_1, p_1\rangle, \ldots, \langle C_n, p_n\rangle, U\rangle$ is a P-proof scheme, then we call (i) the integer n – the *length* of S, (ii) the set U – the *support* of S, and (iii) the atom p_n – the *conclusion* of S. We denote U by $supp(S)$.

Example 1. Let P be a program consisting of four clauses: $C_1 = p \leftarrow$, $C_2 = q \leftarrow p, \neg r$, $C_3 = r \leftarrow \neg q$, and $C_4 = s \leftarrow \neg t$. Then we have the following useful examples of P-proof schemes:

(a) $\langle\langle C_1, p\rangle, \emptyset\rangle$ is a P-proof scheme of length 1 with conclusion p and empty support.
(b) $\langle\langle C_1, p\rangle, \langle C_2, q\rangle, \{r\}\rangle$ is a P-proof scheme of length 2 with conclusion q and support $\{r\}$.
(c) $\langle\langle C_1, p\rangle, \langle C_3, r\rangle, \{q\}\rangle$ is a P-proof scheme of length 2 with conclusion r and support $\{q\}$.
(d) $\langle\langle C_1, p\rangle, \langle C_2, q\rangle, \langle C_3, r\rangle, \{q, r\}\rangle$ is a P-proof scheme of length 3 with conclusion r and support $\{q, r\}$.

In this example we see that the proof scheme in (c) had unnecessary items (the first term), while in (d) the proof scheme was supported by a set containing q, one of atoms that were proved on the way to r. $\Box$

A P-proof scheme carries within itself its own applicability condition. In effect, a P-proof scheme is a *conditional* proof of its conclusion. It becomes applicable when all the constraints collected in the support are satisfied. Formally we say, for a set M of atoms, that a P-proof scheme S is M-*applicable* if $M \cap supp(S) = \emptyset$. We also say that M *admits* S if S is M-applicable.

The fundamental connection between proof schemes and stable models is this:

Proposition 2. *For every normal propositional program P and every set M of atoms, M is a stable model of P if and only if:*

(i) For every $p \in M$, there is a P-proof scheme S such that M admits S
(ii) For every $p \notin M$, there is no P-proof scheme S such that M admits S

Proposition 2 says that the presence and absence of the atom p in a stable model depends *only* on the supports of proof schemes. This fact naturally leads to a characterization of stable models in terms of propositional satisfiability. Given $p \in At$, the *defining equation* for p w.r.t. P is the following propositional formula:

$$p \Leftrightarrow (\neg U_1 \vee \neg U_2 \vee \ldots) \tag{2}$$

where $\langle U_1, U_2, \ldots \rangle$ is the list of all supports of P-proof schemes for P. Here for any finite set $S = \{s_1, \ldots, s_n\}$ of atoms, $\neg S = \neg s_1 \wedge \cdots \wedge \neg s_n$. Up to a total ordering of the finite sets of atoms such formula is unique. For example, suppose we fix a total order on At, $p_1 < p_2 < \cdots$. Then given two sets of atoms, $U = \{u_1 < \cdots < u_m\}$ and $V = \{v_1 < \cdots < v_n\}$, we say that $U \prec V$, if either (i) $u_m < v_n$, (ii) $u_m = v_n$ and $m < n$, or (iii) $u_m = v_n$, $n = m$, and $(u_1, \ldots, u_n)$ is lexicographically less than $(v_1, \ldots, v_n)$. We say that (2) is the *defining equation* for p relative to P if $U_1 \prec U_2 \prec \cdots$. We will denote the defining equation for p with respect to P by Eq_p^P.

Let Φ_P be the set $\{Eq_p^P : p \in At\}$. We then have the following consequence of Proposition 2.

Proposition 3. *Let P be a normal propositional program. Then stable models of P are precisely the propositional models of the theory Φ_P.*

When P is *purely negative*, i.e. all clauses of P have $PosBody = \emptyset$, the stable and supported models of P coincide [DK89] and the defining equations reduce to Clark's completion [Cl78] of P.

Let us observe that in general the propositional formulas on the right-hand-side of the defining equations may be infinitary.

Example 2. Let P be an infinite program consisting of clauses $p \leftarrow \neg p_i$, for all $i \in n$. The defining equation for p in P is infinitary; it is

$$p \Leftrightarrow (\neg p_1 \vee \neg p_2 \vee \neg p_3 \ldots)$$

The following observation is quite useful. If U_1, U_2 are two finite sets of propositional atoms then

$$U_1 \subseteq U_2 \text{ if and only if } \neg U_2 \models \neg U_1$$

Here $\models$ is the propositional consequence relation. The effect of this observation is that not all the supports of proof schemes are important; only the inclusion-minimal ones.

Example 3. Let P be an infinite program consisting of clauses $p \leftarrow \neg p_1, \ldots, \neg p_i$, for all $i \in N$. The defining equation for p in P is infinitary;

$$p \Leftrightarrow [\neg p_1 \vee (\neg p_1 \wedge \neg p_2) \vee (\neg p_1 \wedge \neg p_2 \wedge \neg p_3) \ldots]$$

But our observation above implies that this formula is *equivalent* to the formula

$$p \Leftrightarrow \neg p_1$$

Motivated by the Example 3, we define the *reduced defining equation* for p relative to P to be the formula

$$p \Leftrightarrow (\neg U_1 \vee \neg U_2 \vee \ldots) \tag{3}$$

where U_i range over *inclusion-minimal* supports of P-proof schemes for the atom p and $U_1 \prec U_2 \prec \cdots$. We denote this formula as rEq_p^P, and define $r\Phi_P$ to be the theory consisting of rEq_p^P for all $p \in At$. We then have the following strengthening of Proposition 3.

Proposition 4. *Let P be a normal propositional program. Then stable models of P are precisely the propositional models of the theory $r\Phi_P$.*

In our example 3, the theory Φ_P was infinitary, but the theory $r\Phi_P$ was finitary.

Given a normal propositional program P, we say that P is a *finitary support program* (FSP-program) if all the reduced defining equations for atoms with respect to P are finitary propositional formulas. Equivalently, a program P is an *FSP*-program if for every atom p there is only finitely many inclusion-minimal supports of P-proof schemes for p.

4 Continuity properties of operators and proof schemes

In this section we investigate continuity properties of operators and we will see that one of those properties characterizes the class of FSP programs.

4.1 Continuity properties of monotone and antimonotone operators

Let $\mathcal{P}(At)$ denote the set of all subsets of At. By an operator on the set At of propositional atoms, we mean any function $O : \mathcal{P}(At) \rightarrow \mathcal{P}(At)$. An operator O is *monotone* if for all sets $X \subseteq Y \subseteq At$, $X \subseteq Y$ implies $O(X) \subseteq O(Y)$. Likewise an operator O is *antimonotone* if for all sets $X \subseteq Y \subseteq At$, $X \subseteq Y$ implies $O(Y) \subseteq O(X)$. For a sequence $\langle X_n \rangle_{n \in N}$ of sets of atoms, we say that $\langle X_n \rangle_{n \in N}$ is *monotonically increasing* if for all $i, j \in N$, $i \leq j$ implies $X_i \subseteq X_j$. Likewise for a sequence $\langle X_n \rangle_{n \in N}$ of sets of atoms, we say that $\langle X_n \rangle_{n \in N}$ is *monotonically decreasing* if for all $i, j \in N$, $i \leq j$ implies $X_j \subseteq X_i$.

There are four distinct classes of operators that will concern us in our investigations. First, we consider monotone operators. A monotone operator O is *upper-half* continuous if for every monotonically increasing sequence $\langle X_n \rangle_{n \in N}$, $O(\bigcup_{n \in N} X_n) = \bigcup_{n \in N} O(X_n)$.
A monotone operator O is *lower-half* continuous if for every monotonically decreasing sequence $\langle X_n \rangle_{n \in N}$, $O(\bigcap_{n \in N} X_n) = \bigcap_{n \in N} O(X_n)$.

In the Logic Programming literature the first of these properties is called *continuity*. The classic result due to van Emden and Kowalski is the following.

Proposition 5. *For every Horn program P, the operator T_P is upper-half continuous.*

Generally, the operator T_P for Horn programs is *not* lower-half continuous. Specifically, for the program P consisting of clauses $p \leftarrow p_i$, for $i \in N$, the operator T_P is not lower-half continuous.

The lower-half continuous monotone operators have appeared in Logic Programming literature [Do94]. Even more generally, for a monotone operator O, let us define its *dual* operator O^d as follows:

$$O^d(X) = At \setminus O(At \setminus X).$$

Then an operator O is upper-half continuous if and only if O^d is lower-half continuous [JT51]. Therefore, for any Horn program P, the operator T_P^d is lower-half continuous.

In case of antimonotone operators, we have two additional notions of continuity. An antimonotone operator O is *upper-half* continuous if for every monotonically increasing sequence $\langle X_n \rangle_{n \in N}$, $O(\bigcup_{n \in N} X_n) = \bigcap_{n \in N} O(X_n)$.
An antimonotone operator O is *lower-half* continuous if for every monotonically decreasing sequence $\langle X_n \rangle_{n \in N}$, $O(\bigcap_{n \in N} X_n) = \bigcup_{n \in N} O(X_n)$.

4.2 Gelfond-Lifschitz operator GL_P and proof-schemes

For the completeness sake, let us recall that the Gelfond-Lifschitz operator for a program P which we denote GL_P which assigns to a set of atoms M the least fixed point of $T_{P,M}$ or, equivalently, the least model N_M of the reduced program P_M [GL88]. Here, the program P_M is the Gelfond-Lifschitz reduct of P via M. The following fact is crucial.

Proposition 6 ([GL88]). *The operator GL is antimonotone.*

Here is a useful proof-theoretic characterization of the operator GL.

Proposition 7. *Let P be a normal propositional program and M be a set of atoms. Then*

$$GL_P(M) = \{p : \text{ there exists a } P\text{-proof scheme } S \text{ for } p \text{ such that } M \text{ admits } S, \text{ and } p \text{ is the conclusion of } S\}$$

Proof: Let us assume that $p \in GL_P(M)$ that is $p \in N_M$. As N_M is the least model of the Horn program P_M, $N_M = \bigcup_{n \in N} T^n_{P_M}(\emptyset)$. Then by an easy induction on the n such that $p \in T^n_{P_M}(\emptyset)$ we find a P-proof scheme S_p so that p is the conclusion of S_p, and S_p is admitted by M.
Conversely, we can show, by induction on the length of the P-proof schemes, that whenever such P-proof scheme S is admitted by M, then p belongs to $GL_P(M)$. □

4.3 Continuity properties of the operator GL_P

This section will be devoted to proving two results. First, we prove that for every program P, the operator GL_P is lower-half continuous. Second, we show that the operator GL_P is upper-half continuous if and only if P is an *FSP*-program. That is, GL_P is upper-half continuous if for all atoms p the reduced defining equation for any p (w.r.t P) is finite.

Proposition 8. *For every normal program P, the operator GL_P is lower-half continuous.*

Proof: We need to prove that for every program P and every monotonically decreasing sequence $\langle X_n \rangle_{n \in N}$

$$GL_P(\bigcap_{n \in N} X_n) = \bigcup_{n \in N} GL_P(X_n).$$

We need to prove two inclusions: $\subseteq$, and $\supseteq$.
We first show $\supseteq$. Since

$$\bigcap_{j \in N} X_j \subseteq X_n$$

for every $n \in N$, by antimonotonicity of GL_P we have

$$GL_P(X_n) \subseteq GL_P(\bigcap_{j \in N} X_j).$$

As n is arbitrary,

$$\bigcup_{n \in N} GL_P(X_n) \subseteq GL_P(\bigcap_{j \in N} X_j).$$

Thus the inclusion $\supseteq$ holds.
Conversely, let $p \in GL_P(\bigcap_{n \in N} X_n)$. Then, by Proposition 7, there must be a proof scheme S with support support U and conclusion p such that

$$U \cap \bigcap_{n \in N} X_n = \emptyset.$$

But the family $\langle X_n \rangle_{n \in n}$ is monotonically descending and Y is finite. Thus there is an integer n_0 so that

$$U \cap X_{n_0} = \emptyset.$$

This, however, implies that $p \in GL_P(X_{n_0})$, and thus

$$p \in \bigcup_{n \in N} GL_P(X_n).$$

As p is arbitrary, the inclusion $\subseteq$ holds. Thus $GL_P(\bigcap_{n \in N} X_n) = \bigcup_{n \in N} GL_P(X_n)$. $\Box$

We are now ready to prove the next result of this paper.

Proposition 9. *Let P be a normal propositional program. The following are equivalent:*

(a) P is an FSP*-program*
(b) The operator GL_P is upper-half continuous, i.e.

$$GL_P(\bigcup_{n \in N} X_n) = \bigcap_{n \in N} GL_P(X_n)$$

for every monotonically increasing sequence $\langle X_n \rangle_{n \in N}$.

Proof: Two implications need to be proved: $(a) \Rightarrow (b)$, and $(b) \Rightarrow (a)$.
Proof of the implication $(a) \Rightarrow (b)$. Here, assuming (a) we need to prove two inclusions:
(i) $GL_P(\bigcup_{n \in N} X_n) \subseteq \bigcap_{n \in N} GL_P(X_n)$, and
(ii) $\bigcap_{n \in N} GL_P(X_n) \subseteq GL_P(\bigcup_{n \in N} X_n)$.
To prove (i), note that since $X_n \subseteq \bigcup_{j \in N} X_j$, we have

$$GL_P(\bigcup_{j \in N} X_j) \subseteq GL_P(X_n).$$

As n is arbitrary,

$$GL_P(\bigcup_{j \in N} X_j) \subseteq \bigcup_{n \in N} GL_P(X_n).$$

This proves (i).
To prove (ii), let $p \in \bigcap_{n \in N} GL_P(X_n)$. Then, for every $n \in N$ $p \in GL_P(X_n)$ and so for every $n \in n$ there is an inclusion-minimal support for p, U, such that

$$U \cap X_n = \emptyset.$$

But by (a) there is only finitely many inclusion-minimal supports for P-proof schemes for p. Therefore there is a U_0 such that for infinitely many n's

$$U_0 \cap X_n = \emptyset.$$

But the sequence $\langle X_n \rangle_{n \in N}$ is monotonically increasing. Therefore for *all* $n \in N$, $U_0 \cap X_n = \emptyset$. But then

$$U_0 \cap \bigcup_{n \in N} X_n = \emptyset,$$

so that $p \in GL_P(\bigcup_{n \in N} X_n$. Thus (ii) holds and the implication $(a) \Rightarrow (b)$ follows.
Proof of implication $(b) \Rightarrow (a)$.

Assume that the operator GL_P is upper-half continuous. We need to show that for every p, the reduced defining equation for p is finitary. So let us assume that $rEq)_p^P$ is not finitary. This means that there is an infinite set $\mathcal{X} = \{U_1, U_2, \ldots\}$, where $U_1 \prec U_2 \prec \cdots$, such that

1. each U_i is finite,
2. the elements of $\mathcal{X}$ are pairwise inclusion-incompatible, and
3. for every set of atoms M, $p \in GL_P(M)$ if and only if for some $U_i \in \mathcal{X}$, $U_i \cap M = \emptyset$.

We will now define two sequences:

1. a sequence $\langle K_n \rangle_{n \in N}$ of infinite sets of integers and
2. a sequence $\langle p_n \rangle_{n \in N \setminus \{0\}}$ of atoms

We define $K_0 = N$, and we define p_1 as the first element of U_1 such that

$$\{j : p \notin U_j\}$$

is infinite. Clearly, K_0 is well-defined. We need to show that p_1 is well-defined. If p_1 is not well-defined, then for every $p \in U_1$ there is an integer i_p so that for all $m > i_p$, $p \in U_m$. But U_1 is finite so that taking $n = \max_{p \in U_1} i_p$, we find that for *all* $m > n$, $U_1 \subseteq U_m$ - which contradicts the fact that the sets in $\mathcal{X}$ are pairwise inclusion-incompatible. Thus p_1 is well-defined. We now set

$$K_1 = \{n \in K_0 : p_1 \notin U_n\}.$$

Let us rewrite K_1 as $\{n \in K_0 : \{p_1\} \cap U_n = \emptyset\}$. Clearly. K_1 is infinite.

Now, let us assume that we already defined p_l and K_l so that $K_l = \{n : U_n \cap \{p_1, \ldots, p_l\} = \emptyset\}$ is an infinite subsets of N. We select p_{l+1} as the first element $p \in U_{l+1}$ so that

$$\{j : j \in K_l \text{ and } p \notin U_j\}$$

is infinite. Clearly, by an argument as above, there is such p, and so p_{l+1} is well-defined. We then set

$$K_{l+1} = \{j \in K_l : p_{l+1} \notin U_j\}.$$

Since $\{p_1, \ldots, p_l\} \cap U_j = \emptyset$ for all $j \in K_l$, $\{p_1, \ldots, p_{l+1}\} \cap U_j = \emptyset$ for all $j \in K_{l+1}$. By construction, the set K_{l+1} is infinite.

Now, we complete the argument as follows. We set $X_n = \{p_1, \ldots, p_n\}$. The sequence $\langle X_n \rangle_{n \in N}$ is monotonically increasing. For each n there is j (in fact infinitely many j's) so that $X_n \cap U_j = \emptyset$. Therefore, for each n, $p \in GL_P(X_n)$. Hence $p \in \bigcap_{n \in N} GL_P(X_n)$.

On the other hand, let $X = \bigcup_{n \in N} X_n$. Then

$$X = \{p_1, p_2, ...\}.$$

By our construction, $p_n \in U_n$, and so $U_n \cap X \neq \emptyset$. Therefore X does not admit *any* P-proof scheme for p. Thus $p \notin GL_P(X)$, i.e. $p \notin GL_P(\bigcup_{n \in N} X_n)$, as desired. Thus, if P is not an *FSP*-program (i.e. $\neg(a)$), then the operator GL_P is not upper-half continuous ($\neg(b)$). Thus $(b) \Rightarrow (a)$ holds, and the argument is complete. □

5 Further work, discussion, and conclusions

In [SNS02] Niemelä and coauthors defined a significant extension of logic programming with stable semantics which allows for a programming with cardinality constraints, and, more generally, with weight constraints. This extension have been further studied in [MR04,MNT07]. We will limit our discussion to cardinality constraints only, although it is possible to extend our arguments to any class of convex constraints [LT05]. *Cardinality constraints* are expressions of the form lXu, where $l, u \in N$, $l \leq u$ and X is a finite set of atoms. The semantics of such new atom lXu is that the putative model M contains at least l but not more than u atoms from X. When $l = 0$ we do not write it, and likely, when $u \geq |X|$ we omit it, too. Thus a atom p has the same meaning as $1\{p\}$, while $\neg a$ is $\{p\}0$.

As in the original Gelfond-Lifschitz proposal the semantics of programs admitting cardinality constraints is defined via fixpoints of operators (see the details in [SNS02] and [MR04]). The operator in question is neither monotone nor antimonotone. But when we limit our attention to the programs P where clauses have the property that the head consists of a single atom (i.e. are of the form $1\{p\}$), then one can define an operator O_P which is antimonotone and whose fixpoints are stable models of P. This is done as follows: Given a clause C

$$p \leftarrow l_1 X_1 u_1, \ldots, l_m X_m u_m$$

we transform it into the clause

$$p \leftarrow l_1 X_1, \ldots, l_m X_m, X_1 u_1, \ldots, X_m u_m \tag{4}$$

Given a set of atoms M, we eliminate from P (to be precise the set of transformed clauses from P) those clauses where the upper-constraints ($X_i u_i$) are not satisfied by M. In the remaining clauses the upper constraints are eliminated altogether. The transformed program defines a monotone operator (as pointed in [SNS02]) and if the least fixpoint of that operator coincides with M, M is a stable models of P. On analogy with ordinary programs, we will call clauses as in (4) *normal* CC-clauses. The equivalence of this construction and the original construction in [SNS02] for normal CC-programs is shown in [MNT07].

The construction outlined above allows for application of proof-theoretic methods very similar to those defined above. Namely, on analogy to the construction of P-proof schemes defined above we can define proof schemes for normal CC-programs. This is done by induction as follows. When

$$C = p \leftarrow X_1 u_1, \ldots, X_k u_k$$

is a normal CC-clause without the cardinality-constraints of the form $l_i X_i$ then

$$\langle \langle C, p \rangle, \{X_1 u_1, \ldots, X_k u_k\} \rangle$$

is a proof scheme. Likewise when

$$S = \langle \langle C_1, p_1 \rangle, \ldots, \langle C_n, p_n \rangle, U \rangle$$

is a P-proof scheme and

$$p \leftarrow l_1 X_1, \ldots, l_m X_m, X_1 u_1, \ldots, X_m u_m$$

is a clause in P, and $|X_1 \cap \{p_1, \ldots, p_n\}| \geq l_1$, …, $|X_m \cap \{p_1, \ldots, p_n\}| \geq l_m$, then

$$\langle \langle C_1, p_1 \rangle, \ldots, \langle C_n, p_n \rangle, \langle C, p \rangle, U \cup \{X_1 u_1, \ldots X_m u_m\} \rangle$$

is a P-proof scheme.

The notion of admittance of a P-proof scheme is similar. The support of a proof scheme S consists of cardinality constraints of the form Xu. The proof scheme S is admitted by M if for all such constraint Xu, $M \models Xu$, i.e. $|M \cap X| \leq u$.

Not surprisingly, the theory discussed above lifts almost verbatim. There are, however, some new issues with the definition of FSP-CC-programs. The main reason is that the key simplification fact

$$U_1 \subseteq U_2 \text{ if and only if } \neg U_2 \models \neg U_1$$

has no known analogy in the setting of cardinality constraints. The defining equations and reduced defining equations can be defined but at a cost of exponential blow up of the size of equations resulting from the process of translating disjunctions of (finite number) of cardinality constraints into disjunctive normal form. We note that, alternatively, one can easily give a direct reduction of our CC-programs to normal logic programs using the methods of [FL05] and the distributivity result of [LTT99].

We note that investigations of proof systems in a related area - SAT - play a key role in establishing lower bounds on the complexity of algorithms for finding the models. We wonder if there are analogous results in ASP. For achieving such a goal, we need to find and investigate proof systems for ASP. One candidate for such a proof system is provided in this paper by using P-proof schemes. We wonder if such a proof system can be used to develop a deeper understanding of the complexity issues related to finding stable models.

Acknowledgments

This research of the first author was supported by the National Science Foundation under Grant IIS-0325063. This research of the second author was supported by the National Science Foundation under Grant DMS 0400507.

References

[Ap90] Apt, K.. Logic programming, In: J. van Leeuven, ed, *Handbook of Theoretical Computer Science*, pages 493–574, MIT Press, 1990.

[Bo04] Bonatti, P.A. Reasoning with infinite stable models. *Artificial Intelligence 156*, 75–111.

[BTK93] Bondarenko, A., Toni, F. and Kowalski, R.A., An Assumption-Based Framework for Non-Monotonic Reasoning. *Proceedings of LPNMR-93*, MIT Press, pages 171–189, 1993. j

[Cl78] Clark, K. Negation as failure. In *Logic and data bases*, H. Gallaire and J. Minker, Eds. Plenum Press, 293–322.

[DP92] Davey, B.A., and Priestley, H.A., *Introduction to Lattices and Order*, Cambridge University Press, 1992.

[DK89] Dung, P.M. and Kanchanasut, K., On the generalized predicate completion of non-Horn programs, Logic programming. Proceedings of the North American Conference, 1989.

[Do94] Doets, K., *From Logic to Logic Programming*, MIT Press, 1994.

[FL05] Ferraris, P., and Lifschitz, V., Weight constraints as nested expressions, *Theory and Practice of Logic Programming*, 5, pages 45-74, 2005.

[GL88] Gelfond, M. and Lifschitz, V. The stable model semantics for logic programming. In *Proceedings. of the International Joint Conference and Symposium on Logic Programming*. MIT Press, 1070–1080, 1988.

[JT51] Jonsson, B. and Tarski, A. Boolean Algebras with Operators. *American Journal of Mathematics* 73:891–939, 1951.

[LT05] Liu, L. and Truszczyński, M., Properties of programs with monotone and convex constraints, *Proceedings of the 20th National Conference on Artificial Intelligence*, pages 701-706, 2005.

[Li96] Lifschitz, V., Foundations of logic programming, in *Principles of Knowledge Representation* , CSLI Publications, pages 69-127, 1996.

[LTT99] V. Lifschitz, L. R. Tang and H. Turner, Nested expressions in logic programs, *Annals of Mathematics and Artificial Intelligence*, 25, pages 369-389, 1999.

[L89] J. Lloyd, *Foundations of Logic Programming*, Springer-Verlag, 1989.

[MNR90a] Marek, W., Nerode, A., and Remmel, J.B., Nonmonotonic Rule Systems I. *Annals of Mathematics and Artificial Intelligence*, 1:241–273, 1990.

[MNR90b] Marek, W., Nerode, A., and Remmel, J.B., Nonmonotonic Rule Systems II. *Annals of Mathematics and Artificial Intelligence*, 5:229-264, 1992.

[MNR91] Marek, W., Nerode, A., and Remmel, J.B., A Context for Belief Revision: Normal Logic Programs (Extended Abstract) *Proceedings, Workshop on Defeasible Reasoning and Constraint Solving*, International Logic Programming Symposium, San Diego, CA., 1991.

[MNR92] Marek, W., Nerode, A., and Remmel, J.B., How Complicated is the Set of Stable Models of a Logic Program? *Annals of Pure and Applied Logic*, 56:119-136, 1992.

[MNR94a] Marek, W., Nerode, A., and Remmel, J.B., The stable models of predicate logic programs. *Journal of Logic Programming* 21:129-154, 1994.

[MNR94b] Marek, W., Nerode, A., and Remmel, J.B., Context for belief revision: Forward chaining-normal nonmonotonic rule systems, *Annals of Pure and Applied Logic* 67:269-324, 1994.

[MNR94] Marek, W., Nerode, A., and Remmel, J.B., The stable models of predicate logic programs. *Journal of Logic Programming* 21, pages 129-154, 1994.

[MNT07] Marek, V.W., Niemelä, I. and Truszczynski, M. Logic programs with monotone abstract constraint atoms, *Theory and Practice of Logic Programming*, in print.

[MR04] Marek, V.W. and Remmel, J.B. 2004. Set Constraints in Logic Programming. In *Logic Programming and Nonmonotonic Reasoning, Proceedings of the 7th International Conference (LPNMR-04)*. *LNAI 2923*. Springer, 154–167.

[MT93] Marek, W. and Truszczyński, M. *Nonmonotonic Logic*, Springer Verlag, Berlin, 1993.

[Mi05] Milnikel, R.S., Sequent Calculi for Skeptical Reasoning in Predicate Default Logic and Other Nonmonotonic Systems, *Annals of Mathematics and Artificial Intelligence* 44:1, 1-34, 2005.

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