On the continuity of Gelfond-Lifschitz operator and other applications 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* $r\Phi_P$, we show that the finitary character of that theory $r\Phi_P$ is equivalent to a certain continuity property of the Gelfond-Lifschitz operator GL_P associated with the program P. The introduction of the formula $r\Phi_P$ leads to a double-backtracking algorithm for computation of stable models by reduction to satisfiability of suitably chosen propositional theories. This algorithm does not use the reduction via loop-formulas as proposed in [LZ02] or its extension proposed in [FLL06]. Finally, we discuss possible extensions of techniques proposed in this paper to the context of cardinality constraints.

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. However, proof-theoretic methods have 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. Recently the area of proof systems for ASP (and more generally, nonmonotonic logics) received a lot of attention [GS07,JO07]. It is clear that the community feels that an additional attention to these area is necessary. Nevertheless, there is no clear classification of proof systems for nonmonotonic reasoning analogous to those 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 Marek, Nerode, and Remmel 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 infinite, we study the case of programs for which all these equations are finite. This resulting class of programs, called FSP-programs, turns out to be characterized by a form of continuity of the Gelfond-Lifschitz operator.

Contributions of the paper

The contributions of this paper consist, primarily, of investigations that elucidate the proof-theoretical character of the stable semantics for logic programs, an area with 20 years history [GL88]. The two principal results of this paper are the following.

1. We show that the Gelfond-Lifschitz operator GL_P is, in fact, a proof-theoretical construct (Proposition 7).
2. Given (1), we show that the upper-half continuity of that operator is equivalent to finiteness of (propositional) formulas in a certain class associated with the program P (Proposition 10).

These two results hold for arbitrary programs. A third contribution of this paper which is in a somewhat different direction from our, first two results, is to show that in case of the finite programs P, we can use our proof theory techniques to construct a class of theories C_P, which we call the set of candidate theories associated with P, with the following properties: (i) the theories in C_P are of size linear in P, (ii) the propositional models of any $T \in C_P$ are stable models of P, and (iii) for every stable model M of P, there is $T \in C_P$ such that M is a model of T. Thus we can find stable models of P by using SAT solvers to find models of $T \in C_P$. This result shows how the exponential size of completion of P with loop formulas [LZ02] can be traded for exponential number of linear-size propositional theories.
The outline of this paper is as follows. In section 2, we provide the necessary background on logic programs and stable models 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 certain associated equivalence theorems. In section 4, we discuss the continuity properties of operators. In section 5, we introduce an algorithm (and establish its correctness) for stable model computation that follows from the techniques outlined

in earlier sections. In Section 6 we extend our techniques to the context of programs with cardinality constraints. Finally in Section 7, we have provide some conclusions and directions for future work.

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

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

The integers m or n or both can be 0. The atom p will be called the head of C and denoted $head(C)$. We let $PosBody(C)$ denote the set $\{q_1, \ldots, q_m\}$ and $NegBody(C)$ denote the set $\{r_1, \ldots, r_n\}$. For any set of atoms X, we let $\neg X$ denote the conjunction of negations of atoms from X. Thus, we can write clause (1) as

$$head(C) \leftarrow Posbody(C), \neg negBody(C).$$

Let us stress that the set $NegBody(C)$ is a set of atoms, not a set of negated atoms as is sometimes 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 $q_1, \ldots, q_m \in M$ and $\{r_1, \ldots, r_n\} \cap M = \emptyset$, 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 $n = 0$. 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 $C = p \leftarrow q_1, \ldots, q_m$ in P and $q_1, \ldots, q_m \in I$. 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 $T_{P,M}$. This operator is defined on the set of all subsets of At, $\mathcal{P}(At)$. If P is a program and $M \subseteq At$ is a subset of the Herbrand base, define operator $T_{P,M} : \mathcal{P}(At) \to \mathcal{P}(At)$ as follows:

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

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

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

Thus the operator $T_{P,M}$ like all monotonic continuous operators, 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 $C = p \leftarrow q_1, \ldots, q_m, \neg r_1, \ldots, \neg r_n$ of P, execute the following operations.
(1) If some atom r_i, $1 \leq i \leq n$, 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* as defined in [MNR90a,MT93] 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 the following conditions:
(I) when $n = 1$, $\langle\langle C_1, p_1\rangle, U\rangle$ is a P-proof scheme if $C_1 \in P$, $p_1 = \mathit{head}(C_1)$, $\mathit{PosBody}(C_1) = \emptyset$, and $U = \mathit{NegBody}(C_1)$ and
(II) when $\langle\langle C_1, p_1\rangle, \ldots, \langle C_n, p_n\rangle, U\rangle$ is a P-proof scheme,
$C = p \leftarrow \mathit{PosBody}(C), \neg \mathit{NegBody}(C)$ is a clause in the program P, and $\mathit{PosBody}(C) \subseteq \{p_1, \ldots, p_n\}$, then

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

is a P-proof scheme.
When $S = \langle\langle C_1, p_1\rangle, \ldots, \langle C_n, p_n\rangle, U\rangle$ 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 $\mathit{supp}(S)$.

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 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 $\{q, r\}$.

Proof scheme in (c) is an example of a proof scheme with unnecessary items (the first term). Proof scheme (d) is an example of a proof scheme which is not internally consistent in that r is in the support of its proof scheme and is also its conclusion. $\square$

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, for any set of atoms M, we say that a P-proof scheme S is *M-applicable* if $M \cap \mathit{supp}(S) = \emptyset$. We also say that M *admits* S if S is M-applicable.
The fundamental connection between proof schemes and stable models [MNR90a,MT93] is given by the following proposition.

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 the following conditions hold.*

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

$$p \Leftrightarrow (\neg U_1 \vee \neg U_2 \vee \ldots) \qquad (2)$$

where $\langle U_1, U_2, \ldots \rangle$ is the list of all supports of P-proof schemes with conclusion p. Here for any finite set $S = \{s_1, \ldots, s_n\}$ of atoms, $\neg S = \neg s_1 \wedge \cdots \wedge \neg s_n$. If p is not the conclusion of any proof scheme, then we set the defining equation of p to be $p \Leftrightarrow \bot$. Also, in the case where all the supports of proof schemes of p are empty, we set the defining equation of p to be $p \Leftrightarrow \top$. Up to a total ordering of the finite sets of atoms such a 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.
For example, if P is a Horn program, then for every atom p, either the support of all its proof schemes are empty or p is not the conclusion of any proof scheme. The first of these alternatives occurs when p belongs to the least model of P, $\mathit{lm}(P)$. The second alternative occurs when $p \notin \mathit{lm}(P)$. The defining equations are $p \Leftrightarrow \top$ (that is p) when $p \in \mathit{lm}(P)$ and $p \Leftrightarrow \bot$ (that is $\neg p$) when $p \notin \mathit{lm}(P)$. When P is a stratified program the defining equations are more complex, but the resulting theory is logically equivalent to

$$\{p : p \in \mathit{Perf}_P\} \cup \{\neg p : p \notin \mathit{Perf}_P\}$$

where Perf_P is the unique stable model of 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 C of P have $PosBody(C) = \emptyset$, 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 infinite.

Example 2. Let P be an infinite program consisting of clauses $p \leftarrow \neg p_i$, for all $i \in n$. In this case, the defining equation for p in P is infinite. That is, it is

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

$\square$

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

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

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

$$p \Leftrightarrow \neg p_1$$

$\square$

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 $U_1 \prec U_2 \prec \cdots$. Again, if p is not the conclusion of any proof scheme, then we set the defining equation of p to be $p \Leftrightarrow \bot$. In the case, where there is a proof scheme of p with empty support, then we set the defining equation of p to be $p \Leftrightarrow \top$. 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 involved formulas with infinite disjunctions, but the theory $r\Phi_P$ contains only usual finite propositions.
Given a normal propositional program P, we say that P is a *finite support program* (FSP-program) if all the reduced defining equations for atoms with respect to P are finite propositional formulas. Equivalently, a program P is an *FSP*-program if for every atom p, there are 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 us recall that $\mathcal{P}(At)$ denotes the set of all subsets of At. We say that any function $O : \mathcal{P}(At) \rightarrow \mathcal{P}(At)$ is an operator on the set At of propositional atoms. An operator O is *monotone* if for all sets $X, Y \subseteq At$, $X \subseteq Y$ implies $O(X) \subseteq O(Y)$. Likewise an operator O is *antimonotone* if for all sets $X, Y \subseteq At$, $X \subseteq Y$ 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 $X_i \subseteq X_j$ and 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 we shall consider in this paper. First, we shall consider two types of monotone operators, upper-half continuous monotone operators and lower-half continuous monotone operators. That is, we say that a monotone operator O is *upper-half continuous* if for every monotonically increasing sequence $\langle X_n \rangle_{n \in N}$, $O(\bigcup_{n \in N} X_n) = \bigcup_{n \in N} O(X_n)$. We say that 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.*

In general, the operator T_P for Horn programs is *not* lower-half continuous. For example, let P be the program consisting of the clauses $p \leftarrow p_i$ for $i \in N$. Then the operator T_P is not lower-half continuous. That is, if $X_i = \{p_i, p_{i+1}, \ldots\}$, then clearly $p \in T_P(X_i)$ for all i. However, $\bigcap_i X_i = \emptyset$ and $p \notin T_P(\emptyset)$.
Lower-half continuous monotone operators have appeared in the 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.
For antimonotone operators, we have two additional notions of continuity. We say that an antimonotone operator O is *upper-half* continuous if for every monotonically increasing sequence $\langle X_n \rangle_{n \in N}$, $O(\bigcup_{n \in N} X_n) = \bigcap_{n \in N} O(X_n)$. Similarly, we say 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, assigns to a set of atoms M the least fixpoint of the operator $T_{P,M}$ or, equivalently, the least model N_M of the program P_M which is the Gelfond-Lifschitz reduct of P via M [GL88]. The following fact is crucial.

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

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

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

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

4.3 Continuity properties of the operator GL_P

In this subsection, we state our results on the continuity properties of the operator GL_P. First, it is easy to prove that for every program P, the operator GL_P is lower-half continuous. Moreover, we can prove that if f is a lower-half continuous antimonotone operator, then $f = GL_P$ for a suitably chosen program P. Finally, we can prove 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. Thus we have the following results.

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

The lower-half continuity of antimonotone operators is closely related to programs, as shown in the following result.

Proposition 9. *Let At be a denumerable set of atoms. Let f be an antimonotone and lower-half continuous operator on $\mathcal{P}(At)$. Then there exists a normal logic program P such that $f = GL_P$.*

We are now ready to state one of the main result of this paper.

Proposition 10. *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 $\langle X_n \rangle_{n\in N}$.

5 Computing stable models via satisfiability, but without loop formulas or defining equations

Proposition 3 characterized the stable models of a propositional program in terms of the collection of all propositional valuations of the underlying set of atoms. In this section, we give an alternative characterization in terms of the models of P, only. The proof of this characterization uses Proposition 3, but relates stable models of finite propositional programs P to models of theories of size $O(|P|)$. This is in contrast to Proposition 3 since the set of defining equations is, in general, of size exponential in $|P|$.
A *subequation* for an atom p is either a formula $\neg p$ or a formula

$$p \Leftrightarrow \neg S$$

where S is a support of a proof scheme for p. Here if $S = \emptyset$, then by convention we interpret $p \Leftrightarrow \neg S$ to be simply the atom p. The idea is that a subequation either asserts absence of the atom p in the putative stable model or provides the reason for the presence of p in the putative stable model. A *candidate theory* for program P is the union of P and a set of subequations, one for each $p \in At$. C_P is the class of all candidate theories for the program P.
The key to our algorithm is the following result.

Proposition 11. *1. Let $T \in C_P$. If T is consistent, then every propositional model of T is a stable model for P.*
2. For every stable model M of P, there is a theory $T \in C_P$ such that M is a model for T.

Proposition 11, similarly to [LZ02] characterizes stable models of logic programs via propositional satisfiability, except that theories are smaller.
Next we give an example of our approach to reducing the computation of stable models to satisfiability of propositional theories. It will be clear from this example that our approach avoids having to compute the completion of the program and thus significantly reduces the size of the input theories.

Example 4. Let P be a propositional program as follows:

$p \leftarrow t, \neg q$
$p \leftarrow \neg r$
$q \leftarrow \neg s$
$t \leftarrow$

Let us observe that the atom p has two supports of minimal proof schemes: $\{q\}$ and $\{r\}$. The atom q has just one support: $\{s\}$, the atom t has a single support - the empty set. The atoms r and s have no support at all.
Thus there are *three* subequations for p:

$p \Leftrightarrow \neg q$
$p \Leftrightarrow \neg r$
$\neg p$

Now, q has just two subequations: $q \Leftrightarrow \neg s$, and $\neg q$, t has also two subequations, t and $\neg t$, but this second one leads to contradiction whenever chosen. Finally each of r and s have just one defining equation, $\neg r$, and $\neg s$, respectively.
First let us choose for p, the subequation $\neg p$, and for q, the subequation $q \Leftrightarrow \neg s$. The remaining subequations are forced to t, $\neg r$, and $\neg s$. The resulting theory has nine clauses, when we write our program in propositional form:

$$S = \{\neg p, \neg r, \neg s, t, q \Leftrightarrow \neg s\} \cup \{\neg t \vee p \vee q, r \vee p, s \vee q, t\}.$$

It is quite obvious that this theory is inconsistent. However, if we choose for p, the subequation $p \Leftrightarrow \neg r$ and for q, the subequation $q \Leftrightarrow \neg s$, then the resulting theory written out in propositional form is

$$S = \{p \Leftrightarrow \neg r, \neg r, \neg s, t, q \Leftrightarrow \neg s\} \cup \{\neg t \vee p \vee q, r \vee p, s \vee q, t\}.$$

In this case, $\{p, q, t\}$ is a model of S and hence, $\{p, q, t\}$ is a stable model of P. $\Box$

Let us observe that our discussion above implies an algorithm for computing stable models. In this algorithm, we fix an order of propositional variables (atoms) and we

1. systematically generate proof-schemes for atoms,
2. then generate subequations (one per each atom), and
3. then submit the resulting theories to a SAT solver.

The algorithm described above can be implemented as a *two-tier backtracking search*, with the on-line computation of supports of proof schemes using resolution to collect the negative information derived from clauses, and the usual backtracking scheme of DPLL. This second backtracking can be implemented using any DPLL-based SAT-solver. Proposition 11 implies that the algorithm we outlined is both sound and complete. Indeed, if the SAT solver returns a model M of a theory T, then M is a stable model of P by Proposition 11(1). Otherwise we generate another candidate theory and loop through this process until one satisfying assignment is found. Proposition 11(2) guarantees the completeness of our algorithm.
Our algorithm is not using loop formulas like the algorithms of Lin and Zhao [LZ02] or Giunchiglia, Lierare and Maratea [GLM06], but systematically searches for supports of proof schemes, thus providing supports for atoms in the putative model. It also differs from the modified loop formulas approach of Ferraris, Lee and Lifschitz [FLL06] in that we do not consider loops of the call-graph of P at all. Instead, we compute systematically proof schemes and their supports for atoms. While the time-complexity of our algorithm is significant, the space complexity is $O(|P|)$. This is the effect of not looking at loop formulas at all ([LR06]). The issue of the feasibility of practical implementation of the above algorithm is not clear at the time of writing of this paper.

6 Extensions to CC-programs

In [SNS02] Niemelä and coauthors defined a significant extension of logic programming with stable semantics which allows for programming with cardinality constraints,

and, more generally, with weight constraints. This extension has been further studied in [MR04,MNT08]. To keep things simple, 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 an atom lXu is that a set of atoms M satisfies lXu if and only if $l \leq |M \cap X| \leq u$. When $l = 0$, we do not write it, and, likewise, when $u \geq |X|$, we omit it, too. Thus an atom p has the same meaning as $1\{p\}$ while $\neg p$ has the same meaning as $\{p\}0$.
The stable semantics for CC-programs is defined via fixpoints of an analogue of the Gelfond-Lifschitz operator GL_P; 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 $CCGL_P$ which is antimonotone and whose fixpoints are stable models of P. This is done as follows.

Given a clause C

$$p \leftarrow l_1X_1u_1, \ldots, l_mX_mu_m,$$

we transform it into the clause

$$p \leftarrow l_1X_1, \ldots, l_mX_m, X_1u_1, \ldots, X_mu_m \tag{4}$$

[MNT08]. We say that a clause C of the form (4) is a CC-Horn clause if it is of the form

$$p \leftarrow l_1X_1, \ldots, l_mX_m. \tag{5}$$

A CC-Horn program is a CC-program all of whose clauses are of the form (5). If P is a CC-Horn program, we can define the analogue of the one step provability operator T_P by defining that for a set of atom M,

$$T_P(M) = \{p : (\exists C = p \leftarrow l_1X_1, \ldots, l_mX_m)(\forall i \in \{1, \ldots m\})(|X_i \cap M| \geq l_i)\} \tag{6}$$

It is easy to see that T_P is monotone operator and the least fixed point of T_P is given by

$$\mathit{lfp}(T_P) = \bigcup_{n \geq 0} T_P^n(\emptyset). \tag{7}$$

We can define the analogue of the Gelfond-Lifschitz reduct of a CC-program, which we call the NSS-reduct of P, as follows. Let $\bar{P}$ denote the set of all transformed clauses derived from P. Given a set of atoms M, we eliminate from $\bar{P}$ those clauses where some upper-constraint (X_iu_i) is not satisfied by M, i.e. $|M \cap X_i| > u_i$. In the remaining clauses, the constraints of the form X_iu_i are eliminated altogether. This leaves us with a CC-Horn program P_M. We then define $CCGL_P(M)$ to be the least fixed point of T_{P_M} and say that M is a CC-stable model if M is a model of P and $M = CCGL_P(M)$. The equivalence of this construction and the original construction in [SNS02] for normal CC-programs is shown in [MNT08].
Next we define the analogues of P-proof schemes for normal CC-programs, i.e. programs which consists entirely of clauses of the form (4). This is done by induction as follows. When

$$C = p \leftarrow X_1u_1, \ldots, X_ku_k$$

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 P-CC-proof scheme with support $\{X_1 u_1, \ldots, X_k u_k\}$. Likewise, when

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

is a P-CC-proof scheme,

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

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-CC-proof scheme with support $U \cup \{X_1 u_1, \ldots X_m u_m\}$. The notion of admittance of a P-CC-proof scheme is similar to the notion of admittance of P-proof scheme for normal programs P. That is, if $\mathcal{S} = \langle\langle C_1, p_1\rangle, \ldots, \langle C_n, p_n\rangle, \langle C, p\rangle, U\rangle$ is a CC-proof scheme with support $U = \{X_1 u_1, \ldots X_n u_n\}$, then $\mathcal{S}$ is admitted by M if for every $X_i u_i \in U$, $M \models X_i u_i$, i.e. $|M \cap X_i| \leq u_i$.
Similarly, we can associate a propositional formula ϕ_U so that M admits $\mathcal{S}$ if and only if $M \models \phi_U$ as follows:

$$\phi_U = \bigwedge_{i=1}^{n} \bigvee_{W \subseteq X_i, |W| = |X_i| - u_i} \neg W. \tag{8}$$

Then we can define a partial ordering on the set of possible supports of proof scheme by defining $U_1 \preceq U_2 \iff \phi_{U_2} \models \phi_{U_1}$. For example if $U_1 = \langle\{1,2,3\}2, \{4,5,6\}2\rangle$ and $U_2 = \langle\{1,2,3,4,5,6\}, 4\rangle$, then

$$\begin{aligned}\phi_{U_1} &= (\neg 1 \vee \neg 2 \vee \neg 3) \wedge (\neg 4 \vee \neg 5 \vee \neg 6)\\ \phi_{U_2} &= \bigvee_{1 \leq i < j \leq 6} (\neg i \wedge \neg j).\end{aligned}$$

Then clearly $\phi_{U_1} \models \phi_{U_2}$ so that $U_2 \preceq U_1$. We then define a normal propositional CC-program to be a *FPS CC-program* if for each $p \in At$, there are finitely many $\preceq$-minimal supports of P-CC-proof schemes with conclusion p.
We can also define analogue of the defining equation $CCEq_p^P$ of p relative to a normal CC-program P as

$$p \Leftrightarrow (\phi_{U_1} \vee \phi_{U_2} \vee \cdots) \tag{9}$$

where $\langle U_1, U_2, \ldots\rangle$ is a list of supports of all P-CC-proofs schemes with conclusion p. Again up to a total ordering of possible finite supports, this formula is unique. Let Φ_P be the set $\{CCEq_p^P : p \in At\}$. Similarly, we define the *reduced defining equation* for p relative to P to be the formula

$$p \Leftrightarrow (\neg\phi_{U_1} \vee \neg\phi_{U_2} \vee \ldots) \tag{10}$$

where U_i range over *$\preceq$-minimal* supports of P-CC-proof schemes for the atom p.
Then we have the following analogues of Propositions 2 and 3.

Proposition 12. *For every normal propositional CC-program P and every set M of atoms, M is a CC-stable model of P if and only if the following two conditions hold:*

(i) for every $p \in M$, there is a P-CC-proof scheme S with conclusion p such that M admits S and
(ii) for every $p \notin M$, there is no P-CC-proof scheme S with conclusion p such that M admits S.

Proposition 13. *Let P be a normal propositional CC-program. Then CC-stable models of P are precisely the propositional models of the theory Φ_P.*

We also can prove the analogues of Propositions 6 and 7.

Proposition 14. *For any CC-program P, the operator $CCGL_P$ is antimonotone.*

Proposition 15. *Let P be a normal propositional CC-program and M be a set of atoms. Then*

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

We can also prove that analogue of Proposition 8.

Proposition 16. *For every normal CC-program P, the operator $CCGL_P$ is lower-half continuous.*

However, we can only prove the analogue of the first half of Proposition 10.

Proposition 17. *Let P be a normal propositional CC-program. Then if P is an* FSP-*program, the operator $CCGL_P$ is upper-half continuous, i.e.*

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

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

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 for disjunctions in the bodies of clauses of [LTT99]. Such reductions, of course, lead to an exponential blow up in the size of the representation.

7 Conclusions

In this paper, we have explored the applications of P-proof schemes. We have shown that the Gelfond-Lifschitz operator GL_P is upper-half continuous if and only if for each atom p, there are only finitely many minimal supports of P-proof schemes for p. We also show how we can use P-proofs schemes to associate a natural defining equation for each atom of p and how we can use proof schemes to generate candidate theories

whose propositional models correspond to stable models. This leads to an algorithm for finding stable models where we submit candidate theories to SAT solvers.
We note that the investigations of proof systems in a related area, SAT, have played 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. In particular, are there proof systems for ASP that can be used to develop a deeper understanding of the complexity issues related to finding stable models? The P-proof schemes described in this paper represent one possible candidate of such a proof system for ASP.

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 0654060.

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, 2004.

[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.

[Cl78] Clark, K. Negation as failure. In *Logic and data bases*, H. Gallaire and J. Minker, Eds. Plenum Press, pages 293–322, 1978.

[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*:45-74, 2005.

[FLL06] Ferraris, P., Lee, J. and Lifschitz, V. A generalization of Lin-Zhao theorem. *Annals of Mathematics and Artificial Intelligence 47*:79–101, 2006.

[GS07] Gebser, M. and Schaub, T., Generic Tableaux for Answer Set Programming, *Proceedings of International Conference on Logic Programming, 2007* pages 119–133, 2007.

[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*, pages 1070–1080, 1988.

[GLM06] Giunchiglia, E., Lierler, Y., and Maratea, M. Answer Set Programming Based on Propositional Satisfiability, *Journal of Automated Reasoning* 36:345-377, 2006.

[JO07] Järvisalo, M. and Oikarinen, E., Extended ASP Tableaux and Rule Redundancy in Normal Logic Programs, *Proceedings of International Conference on Logic Programming, 2007* pages 134–148, 2007.

[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.

[LR06] V. Lifschitz and A. Razborov. Why are there so many loop formulas. *Annals of Mathematics and Artificial Intelligence 7*:261–268, 2006.

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

[LZ02] F. Lin and Y. Zhao. ASSAT: Computing answer sets of a logic program by SAT solvers. *Proceedings of AAAI 2002*, pages 112–117. 2002

[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*:129-154, 1994.

[MNT08] Marek, V.W., Niemelä, I. and Truszczynski, M. Logic programs with monotone abstract constraint atoms, *Theory and Practice of Logic Programming* 8:167–199, 2008.

[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*, pages 154–167, Springer-Verlag, 2004.

[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*:1-34, 2005.

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