Disjunctive Programs with Set Constraints

Victor W. Marek[1] and Jeffrey B. Remmel[2]

[1] Department of Computer Science, University of Kentucky, Lexington, KY 40506 *
[2] Departments of Mathematics and Computer Science, University of California at San Diego, La Jolla, CA 92903 **

To Vladimir Lifschitz on the occasion of his 65-th birthday: in recognition of his many contributions to Nonmonotonic Reasoning which have inspired so many researchers in the field.

Abstract. We study an extension of disjunctive logic programs called *set constraint disjunctive (SCD) programs* where the clauses of the program are allowed to have a disjunction of monotone set constraints in their head and arbitrary monotone and antimonotone set constraints in their body. We introduce new class of models called selector stable models which represent all models which can be computed by an analogue the Gelfond-Lifschitz transform. We show that the stable models of disjunctive logic programs can be defined in terms of selector stable models and then extend this result to *SCD* logic programs. Finally we show that there is a natural proof theory associated with selector stable models.

1 Introduction

The answer-set semantics for disjunctive programs both resembles and differs from the answer set semantics for normal logic programs. On the one hand, it is based on the notion of the Gelfond-Lifschitz reduct [6]. On the other hand, it involves an additional element, namely, the minimality of a model. This conjoining of two seemingly different notions results of the increased expressibility. That is, propositional disjunctive programs capture the class of Σ_2^P problems [3]. In this paper, we investigate a class of programs, called *set constraint disjunctive (SCD) logic programs*, which generalize disjunctive logic programs [11] and set constraint logic programs [13]. Clauses in *SCD* logic programs are allowed to have heads which are a disjunction of monotone set constraints and bodies which are conjunctions of monotone and antimonotone set constraints.

We define a natural class of models called *selector stable models* which can be constructed via an analogue of the standard method of constructing stable models for normal logic programs based on the Gelfond-Lifschitz reduct. Selector stable models are based on an underlying *selector function* f which selects for each *SCD* clause C in a *SCD* program D, a specific set of atoms $f(C)$ such that

* email:marek@cs.uky.edu
** email: jremmel@ucsd.edu

$f(C)$ satisfies at least one monotone constraint that occurs in the head of the C. When D is an *SCD* program which has no antimonotone set constraints in the body of its clauses, then D behaves like a Horn program in that we can assign an analogue of the one-step provability operator $T_{f,D}$ to D and f. $T_{f,D}$ is always a monotone operator whose least fixpoint is a model of P. Conversely, each model M of such a Horn-like program determines a canonical function f so that the least fixpoint of $T_{f,D}$ is included in M. We can then define the notion of a selector stable model for D and f via the usual Gelfond-Lifschitz transform assuming that the selector function f satisfies a simple coherence condition which assures that $f(C_1) = f(C_2)$ whenever the Horn parts of clauses C_1 and C_2 coincide.

In the special case of disjunctive logic programs, we will show that a stable model is just a minimal selector stable model. A similar result holds for general *SCD* programs. We will also show that selector stable models have a natural proof theory associated with them.

The outline of this paper is as follows. In Section 2, we shall define the basic notions of set constraints and *SCD* programs. In Section 3, we shall show how stable models of disjunctive logic programs can be defined via selector stable models and define a proof theory of selector stable models. In Section 4, we will show how the results of Section 3 naturally extend to *SCD* logic programs. In Section 5, we will state our conclusions and perspectives for further research.

2 Preliminaries

Given any set X, we let 2^X denote the set of all subsets of X. Let At be a set of propositional variables. A *set constraint* over At is a pair $\langle X, F\rangle$ where X is a finite subset of At and $F \subseteq 2^X$. A set constraint $\langle X, F\rangle$ is called *monotone* if whenever $Y \in F$ and $Y \subseteq Z \subseteq X$, $Z \in F$. $\langle X, F\rangle$ is called *antimonotone* if whenever $Y \in F$ and $Z \subseteq Y$, $Z \in F$. Set constraints were introduced by the authors in [13] and were further studied in [9, 12]. The semantics for programs with set constraints introduced in [13] was a natural generalization of the proposal of [18]. An alternative semantics for those constraints (with the name *abstract* constraints) was studied in [19]. A more general proposal for the semantics of abstract constraints was introduced in [10] which included a set of postulates that should be satisfied by any reasonable semantics for abstract constraints.

Given a set constraint $S = \langle X, F\rangle$, we define the monotonic closure of S to be the set constraint $\overline{S} = \langle X, \overline{F}\rangle$ where $A \in \overline{F}$ if and only if $A \subseteq X$ and there is a $B \in F$ such that $B \subseteq A$. Similarly, we define the antimonotonic closure of S to be the set constraint $\underline{S} = \langle X, \underline{F}\rangle$ where $A \in \underline{F}$ if and only if there is a $B \in F$ such that $A \subseteq B$. We say a subset M of At is a model of $\langle X, F\rangle$, written $M \models \langle X, F\rangle$, if $M \cap X \in F$. One advantage of monotone constraints $\langle X, F\rangle$ is that if $M \subseteq N \subseteq At$ and $M \models \langle X, F\rangle$, then $N \models \langle X, F\rangle$, Among the monotone constraints, a special role is played by monotone cardinality constraints and by *cones* i.e., constraints of the form $C_Z = \langle X, \{Y : Z \subseteq Y \subseteq X\}\rangle$. In particular, every monotone constraint and every monotone cardinality constraint can

be represented as union of cones. An analogous result holds for antimonotone constraints [12].

Set constraints or abstract constraints are a common generalization of constraints pervasive in ASP literature such as cardinality constraints, weight constraints, parity constraints, SQL constraints (i.e., those using constructs such as min,max avg, etc.) One complaint about the practicality of set constraints is that when a set constraint is represented explicitly the size of such representation may be exponential in $|X|$. While this is true in general, there are many set constraints which have exponential size when written out explicitly, but still can be processed efficiently. For example, the monotone cardinality constraint $\{Y : |Y| \geq .5 * |X|\}$ has exponential size when written out explicitly. Nevertheless, cardinality constraints and weight constraints that have been implemented effectively. That is, we write $kX\ell$ for the set constraint such that $M \models kX\ell$ if and only if $k \leq |M \cap X| \leq \ell$. Thus $kX\ell = \langle X, F\rangle$ where F is the family of sets $A \subseteq X$ such that $k \leq |A| \leq \ell$. Even though explicitly representing $kX\ell$ in set form can be exponential in $|X|$, the reason one can build effective systems in ASP which allow for cardinality constraints is that there is is an efficient algorithm to test whether $M \models kX\ell$. That is, given a total order $<$ on At, and $X = \{x_1 < \ldots < x_n\}$, then we can represent $M \cap X$ as a sequence $s_{M\cap X} = s_1 \ldots s_n$ in $\{0,1\}^n$ where $s_i = 1$ if and only if $x_i \in M$. Given $s_{M\cap X}$ and k and ℓ, it is simple to determine if $M \models kX\ell$ by taking one pass through $s_{M\cap X}$. In fact, every set constraint $\langle X, F\rangle$ where $X = \{x_1 < \ldots < x_n\}$ can be thought of a Boolean function over $\{0,1\}^n$ where $f(s_1 \ldots s_n) = 1$ if and only if $\{x_i : s_i = 1\} \in F$. Thought of in this way, set constraints can be given a variety of representations, for instance as CNFs, DNFs, ROBDDs, Boolean polynomials, etc., which allow for efficient processing. The topic of such representations will be studied in [14].

In this paper, we shall define an answer set or stable model semantics for an extension of disjunctive logic programs [11] and set constraint programs [13] which we call set constraint disjunctive (*SCD*) logic programs. An *SCD* clause is a clause of the form

$$C = H_1 \vee \ldots \vee H_k \leftarrow K_1, \ldots, K_n, L_1, \ldots, L_m \tag{1}$$

where $H_1, \ldots, H_k, K_1, \ldots K_n$ are monotone constraints and $L_1, \ldots, L_m$ are antimonotone constraints. We refer to $H_1 \vee \ldots \vee H_k$ as the head of C and $K_1 \wedge \ldots \wedge K_n \wedge L_1 \wedge \ldots \wedge L_m$ as the body of C and write $\mathit{concl}(C) = \{H_1, \ldots, H_k\}$, $\mathit{prem}(C) = \{K_1, \ldots, K_n\}$ and $\mathit{constr}(C) = \{L_1, \ldots, L_m\}$. We say that C is a Horn clause if $\mathit{constr}(C) = \emptyset$. If $M \subseteq At$, then we say that M satisfies the body of C if and only if $M \models K_i$ and $M \models L_j$ for all i and j and M satisfies the head of C if there is at least one i such that $M \models H_i$. We say that M is model of C if and only if either M does not satisfy the body of C or M satisfies the head of C. An *SCD*-program P is a collection of *SCD* clauses. M is model of P if and only if M is model of every clause in P.

3 Selector stable models for disjunctive logic programs

A disjunctive logic programming clause is a clause of the form

$$C = a_1 \vee a_2 \vee \ldots \vee a_n \leftarrow b_1, \ldots, b_n, \neg c_1, \ldots, \neg c_m \tag{2}$$

where $a_1, \ldots, a_k, b_1, \ldots, b_n, c_1, \ldots, c_m$ are atomic formulas in a first order language. We call $a_1 \vee a_2 \vee \ldots \vee a_k$ the head of C, $b_1, \ldots, b_n$ the premises of C, $c_1, \ldots, c_m$ the constraints of C, $b_1 \wedge \ldots \wedge b_n \wedge \neg c_1 \wedge \ldots \wedge \neg c_m$ the body of C, and write $concl(C) = \{a_1, \ldots, a_k\}$, $prem(C) = \{b_1, \ldots, b_n\}$, $constr(C) = \{c_1, \ldots, c_m\}$. C is called a (disjunctive) Horn clause if $constr(C) = \emptyset$, i.e., if C has no negated atoms in its body. C is a ground clause if C has no free variables. If C has no disjunctions in the head, i.e., if $k = 1$, the C is a normal logic programming clause.

A disjunctive logic program D is a set of clauses of the form (2). D is said to be a Horn program if all its clauses are Horn clauses. A ground instance of a clause is a clause obtained by substituting ground terms (terms without free variables) for all the free variables of the clause. We let $ground(D)$ denote the disjunctive propositional logic program consisting of all the ground instances of the clauses in D. The Herbrand base of D, $H(D)$, is the set of all ground atoms that are instances of atoms that appear in D. If $M \subseteq H(D)$ and

$$C = a_1 \vee a_2 \vee \cdots \vee a_k \leftarrow b_1, \ldots, b_m, \neg c_1, \ldots, \neg c_n \in ground(D),$$

then we say that M is a model of D if either M does not satisfy the body of C or M satisfies the head of C. M is a model of D if M is a model of all clauses in $ground(D)$. Thus, as usual, one can reduce models and stable models from predicate disjunctive logic programs and to models and stable model of their grounded versions. That is, the semantics of predicate logic programs can be reduce to the semantics of propositional logic programs. Thus for the rest of this section, we shall focus on propositional disjunctive logic programs.

One of the significant differences between disjunctive logic programs and normal logic programs is that disjunctive Horn programs have multiple intended models. In disjunctive logic programming, one takes the point of view of that models which are minimal with respect to inclusion are the preferred models. We let $mm(D)$ denote the set of minimal models of D.

Example 1. Let D be the propositional disjunctive logic program consisting of the following two clauses.
$C_1 = a \vee b \leftarrow$.
$C_2 = c \vee d \leftarrow b$.
Then is is easy to check that the models of D are $M_1 = \{a\}$, $M_2 = \{b, c\}$, $M_3 = \{b, d\}$, $M_4 = \{b, c, d\}$, $M_5 = \{a, b, c\}$, $M_6 = \{a, b, d\}$, and $M_7 = \{a, b, c, d\}$. Thus $mm(D) = \{M_1, M_2, M_3\}$. □

Given a disjunctive propositional logic program D and a set $M \subseteq H(D)$, we define the Gelfond-Lifschitz reduct D^M by first removing all clauses $C \in D$

such that $constr(C) \cap M \neq \emptyset$ and then for each of the remaining clauses C, replacing C by the clause $a_1 \vee a_2 \vee \ldots \vee a_k \leftarrow b_1, \ldots, b_n$ where $a_1 \vee a_2 \vee \ldots \vee a_k$ is the conclusion of C and $prem(C) = \{b_1, \ldots, b_n\}$. Clearly D^M will always be a disjunctive logic Horn program. Then we say that M is a *stable model* (answer set) of D if $M \in mm(D^M)$.

The main goal of this section is define an alternative approach to defining models and stable models of disjunctive logic programs that can be extended to a much larger class of programs. Our approach is to use what we call selector functions.

Let us suppose that D is a disjunctive propositional logic Horn program. We say that $f : D \rightarrow 2^{H(D)}$ is a *selector function* for D if for each clause $C \in D$, $f(C)$ is a non-empty subset of $concl(C)$. This given, we can then define an analogue of the one-step provability operator relative to D and f. That is, for $M \subseteq H(D)$, we define

$$T_{f,D}(M) = \bigcup \{A : (\exists C \in D)(prem(C) \subseteq M \ \& \ A = f(C)\}.$$

The idea is that one cannot define a one-step provability operator for propositional disjunctive logic programs because if $M \subseteq H(D)$ and C is a clause of the form

$$C = a_1 \vee a_2 \vee \ldots \vee a_k \leftarrow b_1, \ldots, b_n$$

where $\{b_1, \ldots, b_n\} \subseteq M$ and $k \geq 2$, then we do not know which elements from $a_1, \ldots, a_k$ that we should put into $T_D(M)$ for the clause C. The selector function overcomes this difficulty in that it says that elements from $a_1, \ldots, a_k$ that we should put into $T_{f,D}(M)$ are precisely the elements in $f(C)$. It is easy to see that the usual proof that the one-step provability operator T_P for propositional Horn programs is monotone and continuous [20] also applies to the operators $T_{f,D}$. Thus, $T_{f,D}$ is monotone and continuous and $T_{f,D}$ reaches the fixpoint in at most ω steps. This given, then we define the *selector model* $M_{f,D}$ *of* D *relative to* f to be

$$M_{f,D} = T_{f,D} \uparrow_\omega (\emptyset) = \bigcup_{n \geq 0} T^n_{f,D}(\emptyset)$$

where for any $S \subseteq H(D)$, $T^0_{f,D}(S) = S$ and $T^{n+1}_{f,D}(S) = T_{f,D}(T^n_{f,D}(S))$.

For example, consider the program D in Example 1. We have 3 choices for the value of selector function f on C_1, namely, we can have $f(C_1) = \{a\}$, $f(C_1) = \{b\}$, or $f(C_1) = \{a, b\}$. Similarly, we have 3 choices for the value of selector function f on C_2, namely, we can have $f(C_1) = \{c\}$, $f(C_2) = \{d\}$, or $f(C_2) = \{c, d\}$. Now if $f_1(C_1) = \{a\}$, then it is easy to see that $M_{f_1,D} = \{a\} = M_1$ no matter what the value of $f_1(C_2)$ is. If $f_2(C_1) = \{b\}$, then it is easy to see that $M_{f_2,D} = \{b, c\} = M_2$ if $f_2(C_2) = \{c\}$, $M_{f_2,D} = \{b, d\} = M_3$ if $f_2(C_2) = \{d\}$, and $M_{f_2,D} = \{b, c, d\} = M_4$ if $f_2(C_2) = \{c, d\}$. If $f_3(C_1) = \{a, b\}$, then it is easy to see that $M_{f_3,D} = \{a, b, c\} = M_5$ if $f_3(C_2) = \{c\}$, $M_{f_3,D} = \{a, b, d\} = M_6$ if $f_3(C_2) = \{d\}$, and $M_{f_3,D} = \{b, c, d\} = M_7$ if $f_3(C_2) = \{c, d\}$.

Theorem 1. *Suppose that D is a propositional disjunctive logic Horn program. Then*

1. *for all selector functions* $f : D \to 2^{H(D)}$, $M_{f,D}$ *is a model of* D *and*
2. *for every minimal model* M *of* D, $M = M_{g,D}$ *where for any clause* $C \in D$, $g(C) = M \cap concl(C)$ *if* $prem(C) \subseteq M$ *and* $g(C) = concl(C)$ *otherwise.*

Proof: For (1), note that we have observed that $T_{f,D}$ is a monotone operator. Thus if $A \subseteq B \subseteq H(D)$, then $T_{f,D}(A) \subseteq T_{f,D}(B)$. It then easily follows that for all n, $T^n_{f,D}(\emptyset) \subseteq T^{n+1}_{f,D}(\emptyset)$. Now suppose that $C = a_1 \vee a_2 \vee \ldots \vee a_k \leftarrow b_1, \ldots, b_m$ is a clause in D. Now if $\{b_1, \ldots, b_m\} \subseteq M_{f,D}$, then for each i, there is stage n_i such that $b_i \in T^{n_i}_{f,D}(\emptyset)$. Thus if $n = max(\{n_1, \ldots, n_k\})$, then $\{b_1, \ldots, b_m\} \subseteq T^n_{f,D}(\emptyset)$. But then $f(C) \subseteq T^{n+1}_{f,D}(\emptyset)$. Since we are assuming that $f(C) \neq \emptyset$, it follows that $f(C) \subseteq M_{f,D} \cap \{a_1, \ldots, a_k\}$ so that $M_{f,D}$ is a model of C. Hence $M_{f,D}$ is a model of D.

For (2), it is easy to prove by induction that $T^n_{g,D}(\emptyset) \subseteq M$ for all n so that $M_{g,D} \subseteq M$. That is, $T^1_{g,D}(\emptyset) = \{g(C) : C \in D \ \& \ prem(C) \subseteq \emptyset\}$. But if $prem(C) = \emptyset$, then C must be of the form $a_1 \vee \ldots \vee a_k \leftarrow$ and since M is a model of C, $g(C) = \{a_1, \ldots, a_k\} \cap M$. Thus $g(C)$ is a nonempty subset of M. Hence $T^1_{g,D}(\emptyset) \subseteq M$. Now by induction, suppose that $T^n_{g,D}(\emptyset) \subseteq M$. Then

$$T^{n+1}_{g,D}(\emptyset) = \{g(C) : C \in D \ \& \ prem(C) \subseteq T^n_{g,D}(\emptyset)\}.$$

Now if $prem(C) \subseteq T^n_{g,D}(\emptyset)$, then $prem(C) \subseteq M$ so that $g(C) = M \cap concl(C)$. It follows that $T^{n+1}_{g,D}(\emptyset) \subseteq M$. Hence $M_{g,D} = T_{g,D} \uparrow_\omega (\emptyset) \subseteq M$. But by (1), $M_{g,D}$ is model of D so that $M_{g,D} = M$ since M is a minimal model of D. □

We note that the hypothesis that M is a minimal model in part 2 of Theorem 1 is necessary. That is, suppose that $\overline{D}$ consists of the clauses C_1 and C_2 from Example 1 plus the clause

$$C_3 = e \vee k \leftarrow g.$$

Then it is to see $M = \{a, b, c, e\}$ is model of $\overline{D}$, but that M cannot be of the form $M_{f,\overline{D}}$ for any selector function. That is, since g is not in the head of any clause of $\overline{D}$, it follows that it is impossible that e could be derived in process of computing $T_{f,\overline{D}} \uparrow_\omega (\emptyset)$ no matter how one defines the selector function f. In fact, in this case, it is easy to see that the selector models of D from Example 1 and $\overline{D}$ are the same.

We can also define selector stable models for disjunctive propositional logic programs admitting negation in the body as follows. Suppose D is such disjunctive propositional logic program. We say that $f : D \to 2^{H(D)}$ is a *selector function* for D if it satisfies the following two properties.

1. If C is a clause in D, then $f(C)$ is a non-empty subset of $concl(C)$.
2. If C_1 and C_2 are clauses in D such that $concl(C_1) = concl(C_2)$ and $prem(C_1) = prem(C_2)$, then $f(C_1) = f(C_2)$.

Now suppose that we are given a subset M of $H(P)$ and a selector function f. We define the *Gelfond-Lifschitz reduct* of D, D^M, via the following two step process. In Step 1, we eliminate all clauses $C \in D$ such that $constr(C) \cap M \neq \emptyset$. In Step 2, for each remaining clause

$$C = a_1 \vee a_2 \vee \ldots \vee a_k \leftarrow b_1, \ldots, b_n, \neg c_1, \ldots, \neg c_m,$$

we replace C by

$$C_M = a_1 \vee a_2 \vee \ldots \vee a_k \leftarrow b_1, \ldots, b_n.$$

The resulting program D^M is a disjunctive propositional Horn program. We then let f_M be the selector function for D^M defined by letting $f_M(C_M) = f(C)$. Note that condition (2) of our definition of a selector function for D ensures that f_M is a well defined function from D^M into $2^{H(D)}$. Then we say that M is a *selector stable model of D relative to f* if $M = M_{f_M, D^M}$. We say that M is a *selector stable model* if M is a selector stable model relative to f for some selector function for D. We let $SS(D)$ denote the set of selector stable models of D. Then we say that M is a *minimal selector stable model of D* if and only if M is a minimal element of $SS(D)$ relative to inclusion.

We then have the following theorem.

Theorem 2. *Let D be disjunctive logic program. Then M is a stable model of D if and only if M is a minimal selector stable model of D.*

Proof: First assume that M is a stable model of D. Then M is a minimal model of D^M. Since D^M is a disjunctive propositional logic Horn program, it follows from Theorem 1 that there is selector function g_M for M such that $M = M_{g_M, D^M}$ where for any clause $C \in D$, $g_M(C) = M \cap concl(C)$ if $prem(C) \subseteq M$ and $g_M(C) = concl(C)$ otherwise. Then we define $f : D \rightarrow 2^{H(D)}$ by letting $f(C) = g_M(E)$ if there is a clause $E \in D^M$ such that $concl(C) = concl(E)$ and $prem(C) = prem(E)$ and defining $f(C) = concl(C)$, otherwise. It is easy to see that f is a selector function for D and that $f_M = g_M$. It follows that $M_{f_M, D^M} = M$ so that M is a selector stable model.

Now suppose that N is a selector model and $N \subseteq M$. Then we know that $D^M \subseteq D^N$ and N is a model of D^N. But then N is a model of D^M. Since M is a minimal model of D^M, it follows that $N = M$. Hence M is a minimal selector stable model.

Next suppose that N is a minimal selector stable model of D. Then N is a model of D^N by Theorem 1. We claim that N is a minimal model of D^N so that N is stable model of D. That is, suppose that $M \subset N$ and M is a minimal model of D^N. Then by Theorem 1, there is a selector function g for D^N such that $M_{g, D^N} = M$. Then as above, we define $f : D \rightarrow 2^{H(D)}$ by letting $f(C) = g(E)$ if there is a clause $E \in D^N$ such that $concl(C) = concl(E)$ and $prem(C) = prem(E)$ and defining $f(C) = concl(C)$, otherwise. Then f is a selector function for D such that $f_N = g$. It follows that $M_{f_N, D^N} = M$ so that M is a selector stable model which violates the fact that N was a minimal selector stable model. Thus it must be the case that N is a minimal model of D^N so that N is a stable model. □

We view the collection of selector stable models of a disjunctive logic program D as the collection of models that can reasonably be computed from D. Since selector stable models are intrinsic to D, we can use the set of selector stable models to define alternative stable logic semantics for D. For example, one might prefer models that are minimal with respect to cardinality rather than just models that are minimal with respect to inclusion. It is easy to see that our proof of

Theorem 1 also shows that M is minimal model of a disjunctive propositional logic Horn program with respect to cardinality, then it will be of the form $M_{f,D}$ for some selector program. This allows to define "cardinality stable models" of a disjunctive logic program by defining it to a selector stable model of minimal cardinality.

One advantage of selector stable models is that there is a natural proof theory associated with them. That is, recall [15] that normal propositional logic programs P have an associated collection of P-proof schemes. That is, given a normal propositional logic program P, the notion of a P-proof scheme is defined by induction on its length n. Specifically, the set of P-proof schemes are defined inductively by declaring that

(I) $\langle\langle C_1, p_1\rangle, U\rangle$ is a P-proof scheme of length 1 if $C_1 \in P$, p_1 is the head of C_1, $prem(C_1) = \emptyset$, and $U = constr(C_1)$ and

(II) for $n > 1$, $\langle\langle C_1, p_1\rangle, \ldots, \langle C_n, p_n\rangle, U\rangle$ is a P-proof scheme of length n if $\langle\langle C_1, p_1\rangle, \ldots, \langle C_{n-1}, p_{n-1}\rangle, \bar{U}\rangle$ is a P-proof scheme of length $n-1$ and C_n is a clause in P such that p_n is the head of C_n, $prem(C_n) \subseteq \{p_1, \ldots, p_{n-1}\}$ and $U = \bar{U} \cup constr(C_n)$.

If $S = \langle\langle C_1, p_1\rangle, \ldots, \langle C_n, p_n\rangle, U\rangle$ is a P-proof scheme of length n, then we let $supp(S) = U$ and $concl(S) = p_n$.

Example 2. Let P be the normal propositional logic program consisting of the following 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 an unnecessary item, 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. □

A P-proof scheme differs from the usual Hilbert-style proofs in that it 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 a set M of atoms, we say that a P-proof scheme S is *M-applicable* or that *M admits S* if $M \cap supp(S) = \emptyset$. The fundamental connection proved in between proof schemes and stable models is given by the following proposition which is proved in [15].

Proposition 1. *For every normal propositional logic 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 with conclusion p such that M admits S and
(ii) for every $p \notin M$, there is no P-proof scheme S with conclusion p such that M admits S.

We can define an analogous notion of selector proof schemes for disjunctive logic programs. Suppose that we are given a disjunctive propositional logic program D and a selector function f for D. Then we can define a (D, f)-proof scheme by induction on its length n. Specifically, the set of (D, f)-proof schemes are defined inductively by declaring that

(I) $\langle\langle C_1, f(C_1)\rangle, U\rangle$ is a (D, f)-proof scheme of length 1 if $C_1 \in D$, $prem(C_1) = \emptyset$, and $U = constr(C_1)$ and

(II) for $n > 1$, $\langle\langle C_1, f(C_1)\rangle, \ldots, \langle C_n, f(C_n)\rangle, U\rangle$ is a (D, f)-proof scheme of length n if $\langle\langle C_1, f(C_1)\rangle, \ldots, \langle C_{n-1}, f(C_{n-1})\rangle, \bar{U}\rangle$ is a (D, f)-proof scheme of length $n-1$ and C_n is a clause in D such that $prem(C_n) \subseteq \bigcup_{i=1}^{n-1} f(C_i)$ and $U = \bar{U} \cup constr(C_n)$

If $S = \langle\langle C_1, f(C_1)\rangle, \ldots, \langle C_n, f(C_n)\rangle, U\rangle$ is a (D, f)-proof scheme of length n, then we let $supp(S) = U$ and $concl(S) = \bigcup_{i=1}^{n} f(C_i)$.

Example 3. Let D be the normal propositional logic program consisting of the following four clauses:
$C_1 = p \vee q \leftarrow$, $C_2 = a \vee b \leftarrow p, \neg r$, $C_3 = r \leftarrow a, b, \neg q$, and $C_4 = s \vee t \leftarrow \neg t$.
and $f(C_1) = \{p\}$, $f(C_2) = \{a, b\}$, $f(C_3) = \{r\}$, and $f(C_4) = \{t\}$. Then
(a) $\langle\langle C_1, \{p\}\rangle, \emptyset\rangle$ is a (D, f)-proof scheme of length 1 with conclusion $\{p\}$ and empty support.
(b) $\langle\langle C_1, \{p\}\rangle, \langle C_2, \{a, b\}\rangle, \{r\}\rangle$ is a (D, f)-proof scheme of length 2 with conclusion $\{p, a, b\}$ and support $\{r\}$.
(c) $\langle\langle C_1, \{p\}\rangle, \langle C_2, \{a, b\}\rangle, \langle C_3, \{r\}\rangle, \{q, r\}\rangle$ is a (D, f)-proof scheme of length 3 with conclusion $\{p, a, b, r\}$ and support $\{q, r\}$. □

For a set M of atoms, we say that a (D, f)-proof scheme S is *M-applicable* or that *M admits S* if $M \cap supp(S) = \emptyset$. Then we have the following analogue of Proposition 1.

Proposition 2. *For every disjunctive propositional logic program D, every selector function f for D, and every set M of atoms, $M = M_{f_M, D^M}$ is the selector stable model of D relative to the selector function f if and only if*
(i) for every $p \in M$, there is a (D, f)-proof scheme S with $p \in concl(S)$ such that M admits S and
(ii) for every $p \notin M$, there is no (D, f)-proof scheme S such that $p \in concl(S)$ and M admits S.

Proof: First suppose that $M = M_{f,D}$ is a selector stable model. It is easy to see by induction on the length of (D, f) proof schemes that if S is a (D, f)-proof scheme admitted by M, then $concl(S) \subseteq M$. That is, if $S = \langle\langle C_1, f(C_1)\rangle, U\rangle$ is a (D, f)-proof scheme of length 1 which is admitted by M, then $C_1 \in D$, $prem(C_1) = \emptyset$, and $U = constr(C_1)$ is such that $U \cap M = \emptyset$. It then follows

that $(C_1)_M$ is of the form $a_1 \vee \ldots \vee a_k \leftarrow$ and $f_M((C_1)_M) = f(C_1)$. Thus $concl(S) = f(C_1) \subseteq T^1_{f_M,D^M}(\emptyset) \subseteq M$.

Next suppose that $n > 1$ and $S = \langle\langle C_1, f(C_1)\rangle, \ldots, \langle C_n, f(C_n)\rangle, U\rangle$ is a (D, f)-proof scheme of length n admitted by M. Then

$$\overline{S} = \langle\langle C_1, f(C_1)\rangle, \ldots, \langle C_{n-1}, f(C_{n-1})\rangle, \bar{U}\rangle$$

is a (D, f)-proof scheme of length $n-1$ admitted by M and C_n is a clause in D such that $prem(C_n) \subseteq \bigcup_{i=1}^{n-1} f(C_i)$ and $U = \bar{U} \cup constr(C_n)$. By induction, $\bigcup_{i=1}^{n-1} f(C_i) \subseteq M$. Hence there is a q such that $\bigcup_{i=1}^{n-1} f(C_i) \subseteq T^q_{f_M,D^M}(\emptyset)$. Then it is easy to see that C_n will witness that $f(C_n) \subseteq T^{q+1}_{f_M,D^M}(\emptyset)$.

Vice versa, it is also easy to prove by induction that for all $n \geq 1$, if $p \in T^n_{f_M,D^M}(\emptyset)$, then there is a (D, f)-proof scheme S such that $p \in concl(S)$ and M admits S. That is, if $p \in T^1_{f_M,D^M}(\emptyset)$, there there must be a clause B of the form $a_1 \vee \ldots \vee a_k \leftarrow$ and belonging to D^M such that $p \in f_M(B)$. But then there is a clause $C \in D$ such that $C_M = B$ which means that C is of the form

$$a_1 \vee \ldots \vee a_k \leftarrow \neg c_1, \ldots, \neg c_m$$

where $M \cap \{c_1, \ldots, c_m\} = \emptyset$. Since in the case $f(C) = f_M(B)$, it follows that $S = \langle\langle C, f(C)\rangle, \{c_1, \ldots, c_m\}\rangle$ is (D, f) proof scheme of length 1 with $p \in concl(S)$.

Next assume that $p \in T^{n+1}_{f_M,D^M}(\emptyset) \setminus T^n_{f_M,D^M}(\emptyset)$. Then there must be a clause B of the form

$$a_1 \vee \ldots \vee a_k \leftarrow b_1, \ldots, b_p \in D^M$$

such that $p \in f(B)$ and $b_1, \ldots, b_p \in T^n_{f_M,D^M}(\emptyset)$. But then there are (f, D)-proof schemes $S_1, \ldots, S_p$ admitted by M such that $b_i \in f(S_i)$ and a clause C in D of the form

$$a_1 \vee \ldots \vee a_k \leftarrow b_1, \ldots, b_p, \neg c_1, \ldots, \neg c_m$$

such that $M \cap \{c_1, \ldots, c_m\} = \emptyset$ and $f(C) = f_M(B)$. It follows that we if take the proof scheme S which combines the proof schemes $S_1, \ldots, S_p$ followed by the $\langle C, f(C)\rangle, \{c_1, \ldots, c_m\} \cup \bigcup_{i=1}^{n-1} supp(S_i)\rangle$, then S will be a (D, f) admitted by M such that $p \in concl(S)$. Thus (i) and (ii) hold.

Now if (i) and (ii) hold, our arguments show that $M = T_{f_M,D^M} \uparrow_\omega (\emptyset)$ so that M is a selector stable model. □

4 A stable model semantics for *SCD* programs

In this section, we shall extend the ideas of Section 2 to define a stable model semantics for *SCD* programs.

We start by defining the notion of a selector functions. Suppose D is an *SCD* Horn program, i.e., D has no antimonotone constraints appearing in the body of any of its clauses, and $C \in D$ is an *SCD* Horn clause with head $H_1 \vee \ldots \vee H_k$ where each H_i is a monotone set constraint of the form $\langle X_i, F_i\rangle$. To avoid trivialities, we shall always assume that there is no i such that $F_i = 2^{X_i}$ since otherwise

every M is a model of $H_1 \vee \ldots \vee H_k$. Thus, in particular, we assume that $\emptyset \notin F_i$ for all i. The Herbrand base $H(D)$ of D is the set of all atoms that appear in some set constraint which occurs in a clause in D. A selector function f for D is a map from D into $2^{H(D)}$ where for each such clause C, $f(C)$ is a non-empty subset of $X_1 \cup \ldots \cup X_k$ such that there is at least one i such that $f(C) \cap X_i \in F_i$.

Suppose that D is an *SCD* propositional Horn program and $f : D \to 2^{H(D)}$ is a selector function for D. Then we can define the one-step provability operator $T_{D,f} : 2^{H(D)} \to 2^{H(D)}$ for D relative to f by defining for $S \subseteq H(D)$,

$$T_{f,D}(S) = \bigcup \{f(C) : (\exists C \in D)(S \text{ satisfies the body of } C)\}.$$

Again, it is easy to see that the usual proof that the one-step provability operator T_P for propositional Horn programs is monotone and continuous [20] also applies to the operators $T_{f,D}$. Thus, $T_{f,D}$ is monotone and continuous and $T_{f,D}$ reaches the fixpoint in at most ω steps. We then define the *selector model* $M_{f,D}$ *of* D *relative to* f to be

$$M_{f,D} = T_{f,D} \uparrow_\omega (\emptyset) = \bigcup_{n \geq 0} T^n_{f,D}(\emptyset)$$

where for any $S \subseteq H(D)$, $T^0_{f,D}(S) = S$ and $T^{n+1}_{f,D}(S) = T_{f,D}(T^n_{f,D}(S))$.

Theorem 3. *Suppose that D is an SCD propositional Horn program. Then*

1. *for all selector functions $f : D \to 2^{H(D)}$, $M_{f,D}$ is a model of D and*
2. *for every minimal model M of D, $M = M_{g,D}$ where for any clause $C \in D$ whose head is of the form $\langle X_1, F_1\rangle \vee \ldots \vee \langle X_k, F_k\rangle$, $g(C) = M \cap (X_1 \cup \ldots \cup X_k)$ if M satisfies the body of C and $g(C) = X_1 \cup \ldots \cup X_k$ otherwise.*

Proof: For (1), observe that since $T_{f,D}$ is a monotone operator, $T^n_{f,D}(\emptyset) \subseteq T^{n+1}_{f,D}(\emptyset)$ for all n. Now suppose that $C = H_1 \vee H_2 \vee \ldots \vee H_k \leftarrow K_1, \ldots, K_m$ is an *SCD* Horn clause in D and that $M_{f,D} \models K_i = \langle Y_i, G_i\rangle$ for each $i \leq m$. Then $M \cap Y_i \in G_i$ for each i. Thus there must be a stage n_i such that $M \cap Y_i \subseteq T^{n_i}_{f,D}(\emptyset)$. Thus if $n = max(\{n_1, \ldots, n_k\})$, then $T^n_{f,D}(\emptyset)$ satisfies the body of C. But then $f(C) \subseteq T^{n+1}_{f,D}(\emptyset)$. Since we are assuming that $f(C) \models H_j$ for at least one j, it follows that $M_{f,D} \models H_j$ since $f(C) \subseteq M_{f,D}$ and H_j is monotone constraint. Thus $M_{f,D}$ is a model of C. It follows that $M_{f,D}$ is a model of D.

For (2), it is easy to prove by induction that $T^n_{g,D}(\emptyset) \subseteq M$ for all n so that $M_{f,D} \subseteq M$. That is, $T^1_{g,D}(\emptyset) = \{g(C) : C \in D \;\&\; prem(C) \subseteq \emptyset\}$. But if $prem(C) = \emptyset$, then C must be of the form: $H_1 \vee \ldots \vee H_k \leftarrow$, where each H_i is a monotone constraint of the form $\langle X_i, F_i\rangle$. Since M is a model of C, $g(C) = M \cap (X_1 \cup \ldots \cup X_k)$. Thus $g(C)$ is a nonempty subset of M. Hence $T^1_{g,D}(\emptyset) \subseteq M$. Now by induction, suppose that $T^n_{g,D}(\emptyset) \subseteq M$. Then

$$T^{n+1}_{g,D}(\emptyset) = \{g(C) : C \in D \;\&\; T^n_{g,D}(\emptyset) \text{ satisfies the body of } C\}.$$

Now if $T^n_{g,D}(\emptyset)$ satisfies the body of C, then M must satisfy the body of C since all the elements in the body of C are monotone constraint. If the head of C is of the form $H_1 \vee \ldots \vee H_k \leftarrow$ where each $H_i = \langle X_i, F_i\rangle$ is a monotone

constraint, then $g(C) = M \cap (X_1 \cup \ldots \cup X_k)$. It follows that $T_{g,D}^{n+1}(\emptyset) \subseteq M$. Hence $M_{g,D} = T_{g,D} \uparrow_\omega (\emptyset) \subseteq M$. But by (1), $M_{g,D}$ is model of D so that $M_{g,D} = M$ since M is a minimal model of D. □

We define selector stable models for *SCD* propositional logic programs as follows. Suppose D is a *SCD* propositional logic program. We say that $f : D \rightarrow 2^{H(D)}$ is a selector function for D if it satisfies the following two properties.

1. If C is a clause in D whose head is of the form $\langle X_1, F_1 \rangle \vee \ldots \vee \langle X_k F_k \rangle$, then $f(C)$ is a non-empty subset of $X_1 \cup \ldots \cup X_k$ such that there is at least one i such that $f(C) \cap X_i \in F_i$.
2. If C_1 and C_2 are clauses in D such that $concl(C_1) = concl(C_2)$ and $prem(C_1) = prem(C_2)$, then $f(C_1) = f(C_2)$.

Now suppose that we are given a subset M of $H(P)$ and a selector function f. We define the *Gelfond-Lifschitz reduct* of D, D^M, via the following two step process. Suppose that C is a *SCD* clause in D of the form

$$C = H_1 \vee H_2 \vee \ldots \vee H_k \leftarrow K_1, \ldots, K_n, L_1, \ldots, L_m$$

where $H_1, \ldots, H_k, K_1, \ldots, K_m$ are monotone constraints and $L_1, \ldots, L_n$ are antimonotone constraints. In Step 1, we eliminate all clauses $C \in D$ such that M does not satisfy L_i for some i, $1 \leq i \leq m$. In Step 2, if C was not eliminated in Step I, then we replace C by

$$C_M = H_1 \vee H_2 \vee \ldots \vee H_k \leftarrow K_1, \ldots, K_n.$$

The resulting program D^M is an *SCD* propositional disjunctive Horn program. We then let f_M be the selector function for D^M defined by letting $f_M(C_M) = f(C)$. Note that condition (2) of our definition of a selector function for D ensures that f_M is a well defined function from D^M into $2^{H(D)}$. Then we say that M is a *selector stable model of D relative to f* if $M = M_{f_M, D^M}$. We say that M is a *selector stable model* if M is a selector stable model relative to f for some selector function for D. We let $SS(D)$ denote the set of selector stable models of D. Then we say that M is a *minimal selector stable model of D* if and only if M is a minimal element of $SS(D)$ relative to inclusion. Finally, we say that M is stable model of D if and only if M is minimal model of D^M.

We then have the following theorem.

Theorem 4. *Let D be SCD propositional logic program. Then M is a stable model of D if and only if M is a minimal selector stable model of D.*

Proof: First assume that M is a stable model of D. Then M is a minimal model of D^M. Since D^M is an *SCD* propositional Horn program, it follows from Theorem 3 that there is selector function g_M for D^M such that $M = M_{g_M, D^M}$ where for any clause $C \in D^M$ whose head is of the form $\langle X_1, F_1 \rangle \vee \ldots \vee \langle X_k, F_k \rangle$, $g_M(C) = M \cap (X_1 \cup \ldots \cup X_k)$ if M satisfies the body of C and $g_M(C) = X_1 \cup \ldots \cup X_k$, otherwise. Then we define $f : D \rightarrow 2^{H(D)}$ by letting $f(C) = g_M(E)$ if there is a clause $E \in D^M$ such that $concl(C) = concl(E)$ and $prem(C) = prem(E)$ and defining $f(C) = X_1 \cup \ldots \cup X_k$ if M does not satisfy the body of C and the head

of C is of the form $\langle X_1, F_1\rangle \vee \ldots \vee \langle X_k, F_k\rangle$. It is easy to see that f is a selector function for D and that $f_M = g_M$. It follows that $M_{f_M, D^M} = M$ so that M is a selector stable model.

Now suppose that N is a selector model and $N \subseteq M$. Then we know that $D^M \subseteq D^N$ and N is a model of D^N. But then N is a model of D^M. Since M is a minimal model of D^M, it follows that $N = M$. Hence M is a minimal selector stable model.

Next suppose that N is a minimal selector stable model of D. Then N is a model of D^N by Theorem 3. We claim that N is a minimal model of D^N so that N is stable model of D. That is, suppose that $M \subset N$ and M is a minimal model of D^N. Then by Theorem 3, there is a selector function g for D^N such that $M_{g,D^N} = M$. Then as above, we define $f : D \to 2^{H(D)}$ by letting $f(C) = g(E)$ if there is a clause $E \in D^N$ such that $concl(C) = concl(E)$ and $prem(C) = prem(E)$ and defining $f(C) = X_1 \cup \ldots \cup X_k$ if there is no such clause $E \in D^M$ and the head of C is of the form $\langle X_1, F_1\rangle \vee \ldots \vee \langle X_k, F_k\rangle$, otherwise. Then f is a selector function for D such that $f_N = g$. It follows that $M_{f_N, D^N} = M$ so that M is a selector stable model which violates the fact that N was a minimal selector stable model. Thus it must be the case that N is a minimal model of D^N so that N is a stable model. □

We can also define a notion of selector proof schemes for *SCD* propositional logic programs. Suppose that we are given a disjunctive propositional logic program D and a selector function f for D. Then we can define a (D, f)-proof scheme by induction on its length n. Specifically, the set of (D, f)-proof schemes are defined inductively by declaring that

(I) $\langle\langle C_1, f(C_1)\rangle, U\rangle$ is a (D, f)-proof scheme of length 1 if $C_1 \in D$, $prem(C_1) = \emptyset$, and $U = constr(C_1)$ and

(II) for $n > 1$, $\langle\langle C_1, f(C_1)\rangle, \ldots, \langle C_n, f(C_n)\rangle, U\rangle$ is a (D, f)-proof scheme of length n if $\langle\langle C_1, f(C_1)\rangle, \ldots, \langle C_{n-1}, f(C_{n-1})\rangle, \bar{U}\rangle$ is a (D, f)-proof scheme of length $n-1$ and C_n is a clause in D such that $\bigcup_{i=1}^{n-1} f(C_i)$ is a model of all the premises of C_n and $U = \bar{U} \cup constr(C_n)$

If $S = \langle\langle C_1, f(C_1)\rangle, \ldots, \langle C_n, f(C_n)\rangle, U\rangle$ is a (D, f)-proof scheme of length n, then we let $supp(S) = U$ and $concl(S) = \bigcup_{i=1}^{n} f(C_i)$.

For a set M of atoms, we say that a (D, f)-proof scheme S is *M-applicable* or that *M admits S* if M is a model of all antimonotone constraints in $supp(S)$. Then we have the following analogue of Proposition 2.

Proposition 3. *For SCD propositional logic program D, every selector function f for D, and every set M of atoms, $M = M_{f_M, D^M}$ is the selector stable model of D relative to the selector function f if and only if*

(i) for every $p \in M$, there is a (D, f)-proof scheme S with $p \in concl(S)$ such that M admits S and

(ii) for every $p \notin M$, there is no (D, f)-proof scheme S such that $p \in concl(S)$ and M admits S.

Proof: First assume that $M = M_{f,D}$ is selector stable model. It is easy to see by induction on the length of (D, f) proof schemes that if S is a (D, f)-proof

scheme admitted by M, then $concl(S) \subseteq M$. That is, if $S = \langle\langle C_1, f(C_1)\rangle, U\rangle$ is a (D, f)-proof scheme of length 1 which is admitted by M, then $C_1 \in D$, $prem(C_1) = \emptyset$, and $U = constr(C_1)$ is such that M satisfies every antimonotone constraint in U. It then follows that $(C_1)_M$ is of the form $H_1 \vee \ldots \vee H_k \leftarrow$ and $f_M((C_1)_M) = f(C_1)$. Thus $concl(S) = f(C_1) \subseteq T^1_{f_M, D^M}(\emptyset) \subseteq M$.

Next suppose that $n > 1$ and $S = \langle\langle C_1, f(C_1)\rangle, \ldots, \langle C_n, f(C_n)\rangle, U\rangle$ is a (D, f)-proof scheme of length n admitted by M. Then

$$\overline{S} = \langle\langle C_1, f(C_1)\rangle, \ldots, \langle C_{n-1}, f(C_{n-1})\rangle, \bar{U}\rangle$$

is a (D, f)-proof scheme of length $n-1$ admitted by M and C_n is a clause in D such that $prem(C_n) \subseteq \bigcup_{i=1}^{n-1} f(C_i)$ and $U = \bar{U} \cup constr(C_n)$. By induction, $\bigcup_{i=1}^{n-1} f(C_i) \subseteq M$. Hence there is a q such that $\bigcup_{i=1}^{n-1} f(C_i) \subseteq T^q_{f_M, D^M}(\emptyset)$. Then it is easy to see that C_n will witness that $f(C_n) \subseteq T^{q+1}_{f_M, D^M}(\emptyset)$.

Vice versa, it is also easy to prove by induction that for all $n \geq 1$, if $p \in T^n_{f_M, D^M}(\emptyset)$, then there is a (D, f)-proof scheme S such that $p \in concl(S)$ and M admits S. That is, if $p \in T^1_{f_M, D^M}(\emptyset)$, there there must be a clause B of the form $H_1 \vee \ldots \vee H_k \leftarrow$ belonging to D^M such that $p \in f_M(B)$. But then there is a clause $C \in D$ such that $C_M = B$ which means that C is of the form

$$H_1 \vee \ldots \vee H_k \leftarrow L_1, \ldots, L_m$$

where each L_i is an antimonotone constraint such that $M \models L_i$. Since in the case $f(C) = f_M(B)$, it follows that $S = \langle\langle C, f(C)\rangle, \{L_1, \ldots, L_m\}\rangle$ is (D, f) proof scheme of length 1 with $p \in concl(S)$.

Next assume that $p \in T^{n+1}_{f_M, D^M}(\emptyset) - T^n_{f_M, D^M}(\emptyset)$. Then there must be a clause B of the form

$$H_1 \vee \ldots \vee H_k \leftarrow K_1, \ldots, K_p \in D^M$$

such that $p \in f(B)$ and $T^n_{f_M, D^M}(\emptyset)$ is a model of $K_i = \langle Y_i, G_i\rangle$ for $i = 1, \ldots, p$. But then there are (f, D)-proof schemes $S_1, \ldots, S_r$ admitted by M such that for each b such that there exists an i with $b \in M \cap Y_i$, there exists a j with $b \in concl(S_j)$ and a clause C of the form

$$H_1 \vee \ldots \vee H_k \leftarrow K_1, \ldots, K_p, L_1, \ldots, L_m$$

where $L_1, \ldots, L_m$ are antimonotone constraints such that $M \models L_i$ for $i = 1, \ldots, m$ and $f(C) = f_M(B)$. It follows that we if take the proof scheme S which combines the proof schemes $S_1, \ldots, S_p$ followed by the $\langle\langle C, f(C)\rangle, \{L_1, \ldots, L_m\} \cup \bigcup_{i=1}^{n-1} supp(S_i)\rangle$, then S will be a (D, f) admitted by M such that $p \in concl(S)$. Thus (i) and (ii) hold.

If (i) and (ii) hold, then our proofs show that $M = T_{f,D} \uparrow_\omega (\emptyset)$ so that M is a selector stable model. □

5 Conclusions and further research

In this paper, we introduced the notion of selector stable models for a class of programs called set constraint logic (SCD) programs which are a common

generalization of disjunctive logic programs and set constraint logic programs. We defined a collection of selector stable models which we view as the set of models that can reasonably be computed from the program via natural analogues of the Gelfond-Lifschitz transform. Selector stable models have a natural proof theory and can be used to define classical stable models of disjunctive logic programs.

Selector stable models are based on the notion of selector functions which specifies of a way to satisfy the head of any *SCD* clause. A moment reflection shows that such selector functions are present even in the standard normal logic programming. In that case, the selector function just specifies the head of the clause so it is completely trivial. Moreover, it is not difficult to see in hindsight that selector functions are implicit in the paper by Niemelä and his collaborators [18] on weight constraint programs and in our generalization of their construction in [13] on set constraint programs. That is, the selector function was hidden in the translation of the SNS-reduct to the clauses with single-atom heads. But since this translation produced groups of clauses that fire simultaneously, the selector function is just the abstraction from that idea. By that same argument the selector functions generalize the approach of [13].

We believe that selector functions play a crucial role whenever constructions admitting disjunctions of conditions are studied. Moreover, our work opens up several topics for further research. For example, it would be interesting to see how the analysis of Ferraris and Lifschitz [5] of the relationship of weight constraints and nested expressions relates to the present context. Our work also suggests that a natural notion of equivalence of two *SCD* programs is that they have the same set of selector stable models. Thus it should be interesting to study analogues of the notions of equivalence of normal logic programs and its variations such as those in [8] for *SCD* programs.

Our work suggests that one can explore alternative algorithms to the standard "guess-and-check" search method to computing stable models in the context of selector stable models of *SCD* programs. For example, in the case of normal logic programs, there is a forward chaining algorithm of [16] or a Metropolis-type algorithm due to Brik and Remmel [2]. One should also study a number of complexity issues associated with *SCD* programs such as the complexity of finding stable models under limitations of the asymptotic complexity of selector function that are allowed in the process. Finally, it is possible to extend our approach to programs which allow arbitrary set constraints in the bodies and to predicate logic versions of *SCD* programs.

References

1. C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*, Cambridge University Press, 2003.
2. A. Brik and J.B. Remmel, Computing Stable Models of Logic Programs Using Metropolis Type Algorithms, Proceedings of Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP) 2011, paper no. 6, 15 pgs.

3. T. Eiter and G. Gottlob: On the Computational Cost of Disjunctive Logic Programming: Propositional Case. *Ann. Math. Artif. Intell.* 15:289-323, 1995.
4. T. Eiter, W. Faber, N. Leone, and G. Pfeifer. Declarative Problem-solving in DLV. In: J. Minker, ed. *Logic-based Artificial Intelligence*, pages 79 - 103, 2000.
5. P. Ferraris, V. Lifschitz: Weight constraints as nested expressions. *Theor. Pract. Logic Prog.* 5:45-74, 2005.
6. M. Gelfond and V. Lifschitz. The stable semantics for logic programs. *Proceedings 5th Int'l. Symp. Logic Programming*, MIT Press, pages 1070-1080, 1988.
7. M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Gen. Comput.* 9:365-385, 1991.
8. V. Lifschitz, D. Pearce, and A. Valverde: Strongly equivalent logic programs. *ACM Trans. Comput. Log.* 2:526-541, 2001.
9. L. Liu, M. Truszczynski: Properties and Applications of Programs with Monotone and Convex Constraints. *J. Artif. Intell. Res.* 27:299-334, 2006.
10. L. Liu, E. Pontelli, T.C. Son, and M. Truszczynski. Logic Programs with Abstract Constraint Atoms – the Role of Computations. *Artif. Intell.* 174:295–315, 2010
11. J. Lobo, J. Minker, and A. Rajasekar, *Foundations of Disjunctive Logic Programming*, MIT Press, 1992.
12. V.W. Marek. *Introduction to Mathematics of Satisfiability*, CRC Press, 2009.
13. V.W. Marek, J.B. Remmel: Set Constraints in Logic Programming. In: V. Lifschitz and I. Niemelä, eds. Proceedings of the 7^{th} International Conference on Logic Programming and Nonmonotonic Reasoning. Springer Lecture Notes in Computer Science 2923, pages 167 - 179, 2004.
14. V.W. Marek, J.B. Remmel. Effective Set Constraints, in preparation.
15. W. Marek, A. Nerode and J.B. Remmel. Nonmonotonic rule systems I. *Ann. Math. Artif. Intell.* 1:241-273, 1990.
16. W. Marek, A. Nerode and J.B. Remmel. Logic Programs, Well-orderings, and Forward Chaining, *Ann. Pure App. Logic* 96:231-276, 1999 .
17. J. Minker: Overview of Disjunctive Logic Programming. *Ann. Math. Artif. Intell.* 12:1-24, 1994.
18. I. Niemelä, P. Simons, and T. Soininen. Stable Model Semantics of Weight Constraint Rules. In: M. Gelfond, N. Leone and G. Pfeifer, eds. Proceedings of the 5^{th} International Conference on Logic Programming and Nonmonotonic Reasoning, Springer Lecture Notes in Computer Science 1730, pages 317 -331, 1999.
19. T.C. Son, E. Pontelli, and P.H. Tu. Answer Sets for Logic Programs with Arbitrary Abstract Constraint Atoms. *J. Artif. Intell. Res.* 29:353-389, 2007.
20. M.H. van Emden and R.A. Kowalski. The semantics of predicate logic as a programming language. *J. ACM*, 23:733–742, 1976.

Use your QR-barcode reader to get to the e-repository of first author papers.