TLP **11** (1): 111–123, 2011. © Cambridge University Press 2010
doi:10.1017/S1471068410000062 First published online 24 March 2010

Guarded resolution for Answer Set Programming

V. W. MAREK
Department of Computer Science, University of Kentucky, Lexington, KY 40506, USA
(*e-mail:* marek@cs.uky.edu)

J. B. REMMEL*
Departments of Mathematics and Computer Science, University of California at San Diego, La Jolla, CA 92903, USA
(*e-mail:* jremmel@ucsd.edu)

submitted 24 April 2009; revised 15 November 2009; accepted 1 January 2010

Abstract

We investigate a proof system based on a *guarded resolution rule* and show its adequacy for the stable semantics of normal logic programs. As a consequence, we show that Gelfond–Lifschitz operator can be viewed as a proof-theoretic concept. As an application, we find a propositional theory E_P whose models are precisely stable models of programs. We also find a class of propositional theories $\mathscr{C}_P$ with the following properties. Propositional models of theories in $\mathscr{C}_P$ are precisely stable models of P, and the theories in $\mathscr{C}_T$ are of the size linear in the size of P.

KEYWORDS: guarded resolution, proof-theory for Answer Set Programming

1 Introduction

In this note, we introduce a rule of proof, called *guarded unit resolution*. Guarded unit resolution is a generalization of a special case of the resolution rule of proof, namely *positive unit resolution*. In positive unit resolution, one of the inputs is an atom unit clause. Positive unit resolution is complete for Horn clauses, specifically, given a consistent Horn theory T and an atom p, the atom p belongs to the least model of T, $lm(T)$, if and only if there is a positive unit resolution proof of p from T (Dowling and Gallier 1984).

The modification we introduce in this note concerns *guarded atoms* and *guarded Horn clauses*. Guarded atoms are strings of the form: $p : \{r_1, \ldots, r_m\}$ where $p, r_1, \ldots, r_m$ are propositional atoms. Guarded Horn clauses are strings of the form $p \leftarrow q_1, \ldots, q_n : \{r_1, \ldots, r_m\}$ again with $p, q_1, \ldots, q_n, r_1, \ldots, r_m$ propositional atoms.

* Partially supported by NSF grant DMS 0654060.

These guarded atoms and guarded rules will be used to obtain a characterization of stable models of normal logic programs. There are many characterizations of stable models of logic programs. In fact, in Lifschitz (2008), Lifschitz lists 12 different characterizations of stable models of logic programs. The characterization of stable models that we present in this paper has a distinctly proof-theoretic flavor and makes it easy to prove some basic results on Answer Set Programming (Marek and Truszczyński 1999; Niemelä 1999; Baral 2003) such as Fages' theorem (Fages 1994), Erdem–Lifschitz theorem (Erdem and Lifschitz 2003), and Dung's theorem (Dung and Kanchansut 1989).

It should be observed that Dung and Kanchansut (Dung and Kanchansut 1998) consider the so-called quasi-interpretations which, in the formalism of our paper, can be viewed as collections of guarded atoms. The difference between our approach and that of Dung and Kanchansut (1998) is that we elucidate the proof theoretic content of the Gelfond–Lifschitz operator and show how this technique allows for uniform proof of various results in the theory of stable models of programs.

The outline of this paper is as follows. First, we introduce the definition of the guarded resolution rule of proof and then derive its connections with the Gelfond–Lifschitz operator (Gelfond and Lifschitz 1988). Once we do this, we will obtain the desired lifting of the classical result on the completeness of positive unit resolution for Horn theories (Dowling and Gallier 1984) to the context of the stable semantics of logic programs. In Section 3, we show how guarded resolution proofs can be used to prove various standard results in the theory of stable models of propositional programs. Finally, in Section 4, we show how the theory developed in this paper can be used to obtain an algorithm for the computation of stable models that does not use loop formulas and runs in polynomial space in the size of the program.

2 Guarded resolution and Stable Semantics

By a *logic program clause*, we mean a string of the form

$$C = p \leftarrow q_1, \ldots, q_n, not\ r_1, \ldots, not\ r_m. \tag{1}$$

A program P is a set of logic program clauses.

We will interpret program clause C given in (1) as a guarded Horn clause:

$$g(C) = p \leftarrow q_1, \ldots, q_n : \{r_1, \ldots, r_m\}.$$

We define $g(P) = \{g(C) : C \in P\}$. Observe that when we interpret a logic program clause as a guarded Horn clause, the polarity of atoms appearing negatively in the body of the program clause changes in its representation as the guarded Horn clause. That is, they occurred *negatively* in the body of clause and they now appear *positively* in the guard. By convention, we think of a propositional atom as a guarded atom with an empty guard.

We now introduce our guarded resolution rule as follows. It has two arguments: the first is a guarded Horn clause and the second is a guarded atom $q : \{r_1, \ldots, r_n\}$. The guarded atom q must occur in the body of the guarded Horn clause. The result of the application of the rule is a guarded Horn clause whose body is the body of the original guarded Horn clause minus the atom q. The guard of the resulting

https://doi.org/10.1017/S1471068410000062 Published online by Cambridge University Press

guarded Horn clause is the union of the guard of the guarded atom and the guard of the original guarded Horn clause. Formally, our guarded resolution rule has the following form:

$$\frac{p \leftarrow q_1, \ldots, q_n : \{r_1, \ldots, r_m\} \quad q_j : \{s_1, \ldots, s_h\}}{p \leftarrow q_1, \ldots, q_{j-1}, q_{j+1}, \ldots, q_n : \{r_1, \ldots, r_m, s_1, \ldots, s_h\}}.$$

Next, we discuss the Gelfond–Lifschitz operator associated with a normal propositional program. Given a set of atoms M and a normal logic program P, we first define the Gelfond–Lifschitz reduct P^M of P. P^M is constructed according to the following two-step process. First, if

$$C = p \leftarrow q_1, \ldots, q_n, not\ r_1, \ldots, not\ r_m$$

is a clause in P and $r_j \in M$ for some $1 \leqslant j \leqslant m$, then we eliminate C. Second, if C is not eliminated after the first step, then we replace C by

$$p \leftarrow q_1, \ldots, q_n.$$

Clearly, P^M is a Horn program. Thus P^M has a least model N_M. The Gelfond–Lifschitz operator assigns to M the set of atoms N_M.

Our guarded unit resolution rule naturally leads to the notion of a guarded resolution proof $\mathscr{P}$ of a guarded atom $p : S$ from the program P. Here S is a, possibly empty, set of atoms. That is, a guarded resolution proof of $p : S$ is a labeled tree such that every node that is not a leaf has two parents, one labeled with a guarded Horn clause and the other labeled with a guarded atom, where the label of the node is the result of executing the guarded unit resolution rule on the labels of the parents. Each leaf is either a guarded Horn clause $p \leftarrow q_1, \ldots, q_n : \{r_1, \ldots, r_m\}$ such that $p \leftarrow q_1, \ldots, q_n, not\ r_1, \ldots, not\ r_m$ is in P or a guarded atom $q : \{r_1, \ldots, r_m\}$ such that $q \leftarrow not\ r_1, \ldots, not\ r_m$ is in P. In the special case where the tree consist of a single node, we assume that the node is labeled with a guarded atom $q : \{r_1, \ldots, r_m\}$ where $q \leftarrow not\ r_1, \ldots, not\ r_m$ is in P. Note that in a guarded resolution proof, guards only grow as we proceed down the tree. That is, as we resolve, the guards are summed up. For this reason, the guard of the root of the proof contains the guards of *every* label in the tree.

We say that a set of atoms M *admits* a guarded atom $p : S$, if $M \cap S = \emptyset$ and that M admits a guarded resolution proof $\mathscr{P}$ if it admits the label of the root of $\mathscr{P}$. The following statement follows from the containment properties of guards in a guarded resolution proof.

Lemma 2.1
If M admits the guarded resolution proof $\mathscr{P}$, then M admits every guarded atom occurring as a label in $\mathscr{P}$ and M is disjoint from the guard of every guarded clause in $\mathscr{P}$.

We then have the following proposition.

Proposition 2.1
Let P be a propositional logic program and let M be a set of atoms. Then $GL_P(M)$ consists exactly of atoms p such that there exists a set of atoms Z where the guarded

https://doi.org/10.1017/S1471068410000062 Published online by Cambridge University Press

atom $p : Z$ is the conclusion of a guarded resolution proof $\mathscr{P}$ from $g(P)$ admitted by M.

Proof
Let $Q = P_M$ and assume that $p \in GL_P(M)$. Then by definition, $p \in T_Q^\omega$ where T_Q is the standard one-step provability operator for Q. We claim that we can prove by induction on $n \in N$ that whenever $p \in T_Q^n$, then there exists a set of atoms Z such that $p : Z$ possesses a guarded resolution proof from $g(P)$ admitted by M. If $n = 1$, then it must be the case that the $p \leftarrow$ belongs to Q. But then, for some $r_1, \ldots, r_m$,

$$p \leftarrow not\ r_1, \ldots, not\ r_m$$

belongs to P and $\{r_1, \ldots, r_m\} \cap M = \emptyset$. Therefore the guarded atom $p : \{r_1, \ldots, r_m\}$ is admitted by M and it possesses a guarded resolution proof from $g(P)$, namely, the one that consists of the root labeled by $p : \{r_1, \ldots, r_m\}$. Now, let us assume $p \in T_Q^{n+1}$. Then there is a clause $C = p \leftarrow q_1, \ldots, q_s$ in Q such that $q_i \in T_Q^n$ for $i = 1, \ldots, s$. Thus by induction, there are sets of atoms S_i, $1 \leqslant i \leqslant n$, such that $q_i : S_i$ possesses a guarded resolution proof from $g(P)$ admitted by M. As $p \leftarrow q_1, \ldots, q_n$ belongs to Q, there must exist atoms $r_1, \ldots, r_m \notin M$ such that

$$p \leftarrow q_1, \ldots, q_n, not\ r_1, \ldots, not\ r_m$$

is a clause in P. It is easy to combine the guarded resolution proofs of $q_i : S_i$, $1 \leqslant i \leqslant n$ and the guarded clause $p \leftarrow q_1, \ldots, q_n : \{r_1, \ldots, r_m\}$ to obtain a guarded resolution proof from $g(P)$ of the following guarded atom:

$$p : S_1 \cup \ldots \cup S_n \cup \{r_1, \ldots, r_m\}.$$

As all the sets occurring in the guard of this guarded atom are disjoint from M, the resulting guarded resolution proof is admitted by M. This shows the inclusion $\subseteq$.

Conversely, suppose $p : S$ has a guarded resolution proof $\mathscr{P}$ from $g(P)$ admitted by M. By the lemma, all the guards occurring in $\mathscr{P}$ are disjoint from M. We can then prove by induction on the height of the tree $\mathscr{P}$ that $p \in GL_P(M)$. If the height of $\mathscr{P}$ is 0, then it must be the case that

$$p \leftarrow not\ r_1, \ldots, not\ r_m$$

belongs to P where $S = \{r_1, \ldots, r_m\}$. But since $M \cap S = \emptyset$, the clause $p \leftarrow$ belongs to Q. Hence $p \in GL_P(M)$.

Now, for the inductive step, assume that whenever $q : S$ has a guarded resolution proof from $g(P)$ that is admitted by M of height less than or equal to n, then $q \in GL_P(M)$. We now show that the same property holds for all guarded atoms $p : U$ which have a guarded resolution proof from $G(P)$ that is admitted by M of the height $n + 1$. What does such a guarded resolution proof look like? First, the root must be the result of a guarded unit resolution of the form

$$\frac{p \leftarrow q : Z_1 \quad q : S_0}{p : Z_1 \cup S_0}.$$

As $(Z_1 \cup S_0) \cap M = \emptyset$, $Z_1 \cap M = \emptyset$ and $S_0 \cap M = \emptyset$. Now, $q : S_0$ has a guarded resolution proof from $g(P)$ that is admitted by M of height at most n and, hence,

https://doi.org/10.1017/S1471068410000062 Published online by Cambridge University Press

$q \in GL_P(M)$ by our inductive assumption. Since as we progress down the proof tree, the body of guarded clauses only get smaller and the guards of guarded clauses only get bigger, there must exist a path starting at the guarded clause $p \leftarrow q : Z_1$ which consists of guarded clauses of the form

$$\begin{aligned} &p \leftarrow q, q_1, \ldots, q_t : Z_{t+1} \\ &\vdots \\ &p \leftarrow q, q_1 : Z_2 \\ &p \leftarrow q : Z_1, \end{aligned}$$

where $Z_{t+1} \subseteq Z_t \subseteq \cdots \subseteq Z_1$ and for each i, there is a node in the tree of the form $q_i : S_i$ such that the resolution of $p \leftarrow q, q_1, \ldots, q_i : Z_{i+1}$ and $q_i : S_i$ results in the clause $p \leftarrow q, q_1, \ldots, q_{i-1} : Z_i$. Now each $q_i : S_i$ is the root of a guarded resolution proof from $g(P)$ that is admitted by M of height less than or equal to n and, hence, q_i is in $GL_P(M)$ for $i = 1, \ldots, t$.

Since $p \leftarrow q, q_1, \ldots, q_p : Z_{t+1}$ is a leaf, there must be a clause

$$p \leftarrow q, q_1, \ldots, q_t, not\ r_1, \ldots, not\ r_m$$

in P where $Z_{t+1} = \{r_1, \ldots, r_m\}$. Since M admits the proof tree, it must be the case that $\{r_1, \ldots, r_m\} \cap M = \emptyset$ and, hence, $p \leftarrow q, q_1, \ldots, q_t$ is in Q. But then, since $q, q_1, \ldots, q_t$ are in $GL_P(M)$, it follows that $p \in GL_P(M)$. □

Proposition 2.1 tells us that the Gelfond–Lifschitz operator GL is, essentially, a proof-theoretic construct. Here is one consequence, this time a semantic one.

Corollary 2.1
Let P be a propositional logic program and let M be a set of atoms. Then M is a stable model of P if and only if

(1) for every $p \in M$, there is a set of atoms S such that there is a guarded resolution proof of $p : S$ from $g(P)$ admitted by M and
(2) for every $p \notin M$, there is no set of atoms S such that there is a guarded resolution proof of $p : S$ from $g(P)$ admitted by M.

When P is a Horn program Corollary 2.1 reduces to the classical fact (Dowling and Gallier 1984) that the elements of the least model of the Horn programs are precisely those that can be proved out of clausal representation of P using positive unit resolution.

Given a finite set of atoms S, we write $\neg S$ for the conjunction $\bigwedge_{q \in S} \neg q$. Next, let us call S such that $p : S$ has a guarded resolution proof from $g(P)$ a *support* of p with respect to P. We can then form an *equation* for p with respect to P, $eq_P(p)$, as follows:

$$p \Leftrightarrow (\neg S_1 \vee \neg S_2 \vee \ldots),$$

where $S_1, S_2, \ldots$ is the list of all supports of p with respect to P. If the only support of p is the empty set, then we let $eq_P(p) = p$ and if there are no supports for p, then we let $eq_P(p) = \neg p$. Next, let E_P be the propositional theory consisting of $eq_P(p)$

https://doi.org/10.1017/S1471068410000062 Published online by Cambridge University Press

for all $p \in At$. We then have the following theorem resembling Clark's completion theorem, except we get it for stable models, not supported models.

Proposition 2.2
Let P be a propositional program and let M be a set of atoms. Then M is a stable model of P if and only if $M \models E_P$.

Proof
First, assume that M is a stable model of P. Then if $p \in M$, it follows from Corollary 2.1 that there is an S such that $p : S$ has a guarded resolution proof admitted by M. Hence $M \cap S = \emptyset$ and $M \models \neg S$. Thus M satisfies both p and one of the disjuncts on the right-hand side of $eq_P(p)$. Hence $M \models eq_P(p)$. Next assume that $p \notin M$. Then there is no Z such that $p : Z$ has a guarded resolution proof admitted by M. It follows that either $eq_P(p)$ equals $\neg p$ or M satisfies both negation of p and of the negation of every disjunct on the right-hand side of $eq_P(p)$. Thus again $M \models eq_P(p)$. This shows "if" part of the theorem.

For the other direction, suppose that $M \models eq_P(p)$. Then if $p \in M$, either $eq_P(p) = p$ or $eq_P(p) = p \Leftrightarrow (\neg S_1 \vee \neg S_2 \vee \ldots)$. In the former case, this means that the tree consisting of a single node $p : \emptyset$ is a guarded resolution proof and, hence, $p \leftarrow$ is a clause in P. Thus p must be in M. In the latter case, there must be some S_i such that $M \models \neg S_i$. But by definition, $p : S_i$ is the root of some guarded resolution proof $\mathscr{P}$ for $g(P)$ and since every guard in such a guarded resolution proof is contained in S_i, it must be the case that M admits $\mathscr{P}$. But then we have shown that $p \in GL_P(M)$. Thus $M \subseteq GL_P(M)$.

On the other hand, if $p \notin M$, then either $eq_P(p) = \neg p$ or $eq_P(p) = p \Leftrightarrow (\neg S_1 \vee \neg S_2 \vee \ldots)$. In the former case, there must be be no guarded resolution proofs of p and, hence, p is not in M. In the latter case, it must be that M does not satisfy any $\neg S_i$. This means that there is no guarded resolution proof from $g(P)$ whose root is of the form $p : S$ such that M admits p and, hence, $p \notin GL_M(P)$. This implies $GL_P(M) \subseteq M$ and, hence, $GL_p(M) = M$. Thus M is a stable model of P. □

If we look carefully at the structure of any resolution proof tree of a guarded atom $p : S$, we see that S collects a set atoms which guarantee that $p \in N_M$ whenever $M \cap S = \emptyset$. Thus in defining $eq_P(p)$, we essentially unfold the atoms in S to conjunctions of negated atoms $\neg S$ (cf. Brass and Dix 1999).

We observe that, in principle, when the program P is infinite, the theory E_P may be infinitary. Specifically, the formulas $eq_P(p)$ may be infinitary formulas, since the disjunction on the right-hand side of the equivalence may be over an infinite set of finite conjunctions. But the semantics for infinite propositional logic is well-known (Karp 1964) and can be applied here. The authors studied the necessary and sufficient conditions for E_P to be finitary in (Marek and Remmel 2010).

3 Some applications

In this section we will use the results of Section 2 to derive the result of Erdem and Lifschitz (Erdem and Lifschitz 2003). This result generalizes a theorem by Fages (Fages 1994) which is useful as a preprocessing step for the computation of stable

https://doi.org/10.1017/S1471068410000062 Published online by Cambridge University Press

models of programs. We will also prove a result of Dung (Dung and Kanchansut 1989) on stable models of purely negative programs.

We say that a set of atoms M *has levels* with respect to a program P if

(1) M is a model of P, and
(2) there is a function $rk : M \to Ord$ such that, for every $p \in M$, there is a clause C such that
   (a) $p = head(C)$,
   (b) $M \models body(C)$, and
   (c) For all $q \in posBody(C)$, $rk(q) < rk(p)$.

Clearly, the least model of a Horn program has levels since the function which assigns to an atom $p \in M$, the least integer n such that $p \in T_P^n(\emptyset)$ is the desired rank function for condition (2).

We now prove the following proposition.

Proposition 3.1
Let P be a propositional logic program and $M \subseteq At$. Then M is a stable model of P if and only if M has levels with respect to P.

Proof
Clearly, when M is a stable model of P, then M has levels with respect to P. Namely, the rank function whose existence is postulated in (2) is the rank function inherited from the Horn program P_M.

Converse implication can be proved in a variety of ways. Our proof, in the spirit of the proof-theoretic approach of this paper, uses guarded resolution. That is, assume that M has levels with respect to P where rk is the rank function in condition (2). Our goal is to prove that $M = GL_P(M)$. First, let us observe that since $M \models P$, $GL_P(M) \subseteq M$. Thus, all we need to show is that $M \subseteq GL_P(M)$. By Corollary 2.1, we need only show that for given any $p \in M$, there is a Z such that $p : Z$ possesses a guarded resolution proof from $g(P)$ that is admitted by M. We construct the desired set Z and guarded resolution proof by using the rank function rk. Let $S = \{rk(p) : p \in M\}$, i.e. S is the range of rank function. We proceed by transfinite induction on the elements of S. Let p be an atom in M such that $rk(p)$ is the least element of S. By assumption, there must exist a clause C in P such that $M \models body(C)$, $p = head(C)$ and for all $q \in posBody(C)$, $rk(q) < rk(p)$. Since $M \models body(C)$, there can be no qs in positive part of the body of C because any such q must be in M and have rank strictly less than p. Thus the clause C has the following form:

$$p \leftarrow not\ r_1, \ldots not\ r_m.$$

As $M \models body(C)$, $r_1, \ldots, r_m \notin M$. But then $p : \{r_1, \ldots, r_m\}$ is a guarded atom admitted by M and so $p : \{r_1, \ldots, r_m\}$ has a guarded resolution proof from $g(P)$ which consists of a single node labeled with $p : \{r_1, \ldots, r_m\}$.

Now, assume that whenever $\beta \in S$, $\beta < \alpha$, and $rk(q) = \beta$, then there is a guarded resolution proof of $q : S$ from $g(P)$ admitted by M for some set S. Let us assume that $p \in M$ and $rk(p) = \alpha$. By our assumption, there is a clause C

$$p \leftarrow q_1, \ldots, q_n, not\ r_1, \ldots not\ r_m$$

https://doi.org/10.1017/S1471068410000062 Published online by Cambridge University Press

with $M \models body(C)$ and $rk(q_1) < rk(p), \ldots, rk(q_n) < rk(p)$. By inductive assumption, for every q_i, $1 \leqslant i \leqslant n$, there is a finite set of atoms Z_i such that there is guarded resolution proof $\mathscr{D}_i$ from $g(P)$ of $q_i : Z_i$ admitted by M. In particular, $Z_i \cap M = \emptyset$. We can then easily combine the guarded resolution proofs for $q_i : Z_i$ with n applications of guarded unit resolution starting with the leaf

$$p \leftarrow q_1, \ldots, q_n : \{r_1, \ldots, r_m\}$$

to produce a guarded resolution proof of

$$p : Z$$

from $g(P)$ where $Z = Z_1 \cup \ldots \cup Z_n \cup \{r_1, \ldots, r_n\}$. Since all Z_is are disjoint from M and $M \cap \{r_1, \ldots, r_m\} = \emptyset$, it follows that $M \cap Z = \emptyset$. Thus the resulting resolution proof is admitted by M. This completes the inductive argument and thus the proof of the Proposition 3.1. □

We observe that, in fact, it is sufficient to limit the functions rk in the definition of M having levels respect to P to those rank functions that take values in N, the set of natural numbers.

We get, as promised, several corollaries. One of these is the result of Erdem and Lifschitz (2003). Following Erdem and Lifschitz (2003), we say that a program P is *tight on a set of atoms* M if there is a rank function rk defined on M such that whenever C is a clause in P and $head(C) \in M$, then for all $q \in posBody(C)$, $rk(q) < rk(head(C))$. Here is the result of Erdem and Lifschitz.

Corollary 3.1 (*Erdem and Lifschitz*)
If P is tight on M and M is a supported model of P, then M is a stable model of P.

Proof
Indeed, tightness on M requires that for any $p \in M$, there is a support for p and that *all* clauses C that provide the support for the presence of p in M have the property that the atoms in the positive part of the body of C have smaller rank. In Proposition 3.1, we showed that it is enough to have just one such clause. Since tightness on M implies existence of such a supported clause, the corollary follows. □

Since all stable models are supported (Gelfond and Lifschitz 1988), one can express Erdem–Lifschitz theorem in a more succinct way.

Corollary 3.2 (*Erdem–Lifschitz*)
If for every supported model M of a program P, P is tight on M, then the classes of supported and stable models of P coincide.

Fages theorem (Fages 1994) is a weaker version of Corollary 3.1 (but with assumptions that are easier to test). Specifically, we say that a program P is *tight* if there is a rank function rk such that for every clause C of P, the ranks of the atoms occurring in the positive part of the body of C are smaller than the rank of the head of C. Clearly, if P is tight, then P is tight on any of its models M since rk will also witness that P is tight on M. Thus one has the following corollary.

https://doi.org/10.1017/S1471068410000062 Published online by Cambridge University Press

Corollary 3.3 (*Fages*)
If P is tight, then the classes of supported and stable models of P coincide.

Tightness is a syntactic property that can be checked in polynomial time by inspection of the positive call-graph of P. This is not the case for the stronger assumptions of Proposition 3.1 and Corollary 3.2.

Let $Stab(P)$ be the set of all stable models of P. We say that programs P, P' are *equivalent* if $Stab(P) = Stab(P')$. This notion and its strengthenings were studied by ASP community (Lifschitz *et al.* 2001), (Truszczynski 2006). We have the following fact.

Lemma 3.1
If P, P' prove the same guarded atoms, then P and P' are equivalent.

As a corollary we get the following result due to Dung (Dung and Kanchansut 1989)

Corollary 3.4 (*Dung*)
For every program P, there is purely negative program P' such that P, $P'$are equivalent.

The program P' is quite easy to construct. That is, for each atom p, if

$$eq_P(p) = p \Leftrightarrow (\neg S_1 \vee \neg S_2 \vee \ldots),$$

then we add to P', all clauses of the form

$$p \leftarrow not\ r_{i,1}, \ldots, not\ r_{i,m_i},$$

where $S_i = \{r_{i,1}, \ldots, r_{i,m_i}\}$. If $eq_P(p) = p$, then we add $p \leftarrow$ to P'. Finally, if $eq_P(p) = \neg p$, then we add nothing to P'. It is then easy to see that $E_P = E_{P'}$ and hence P and P' are equivalent. □

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

Proposition 2.2 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 suitably chosen propositional theories. The proof of this characterization uses Proposition 2.2, but relates stable models of finite propositional programs P to models of theories of size $O(|P|)$. This is in contrast to Proposition 2.2 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 guarded resolution proof of p from 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

https://doi.org/10.1017/S1471068410000062 Published online by Cambridge University Press

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 $p \in At$. By $\mathscr{C}_P$ we denote the set of candidate theories for P.

Proposition 4.1

(1) Let T be a candidate theory for P (i.e. $T \in \mathscr{C}_P$). 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 theory $T \in \mathscr{C}_P$ such that M is a model for T.

Proof

(1) Let T be a candidate theory for P and suppose that $M \models T$. We need to show that M is a stable model for P. In other words, we need to show that

$$GL_P(M) = M.$$

The inclusion $GL_P(M) \subseteq M$ follows from the fact that M is a model of P. That is, since M is a model of P, it immediately follows that M is a model of the Gelfond–Lifschitz transform of P, P_M. Since $GL_P(M)$ is the unique minimal model of P_M, it automatically follows that $GL_P(M) \subseteq M$.

To show that $M \subseteq GL_P(M)$, suppose that $p \in M$. Then the subequation for p that is in T can not be $\neg p$. Therefore it is of the form

$$p \Leftrightarrow \neg U_p,$$

where there is some guarded resolution proof $\mathscr{P}$ of $p : U_p$ from $g(P)$. Since $M \models T$ and $p \in M$, $M \models \neg U_p$. But then $M \cap U_p = \emptyset$ so that M admits $\mathscr{P}$. Hence by Corollary 2.1, $p \in GL_P(M)$.

(2) Let M be a stable model of P. We construct a candidate theory T so that M is a model of T. To this end, for each $p \notin M$, we select $\neg p$ as a subequation for p. For each $p \in M$, we select a guarded resolution proof of some $p : U_p$ from $g(P)$ that admitted by M. We then add to T the formula

$$p \Leftrightarrow \neg U_p.$$

Clearly, T is a candidate theory, and $M \models T$, as desired. □

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 4.1

Let P be the following propositional program:

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

https://doi.org/10.1017/S1471068410000062 Published online by Cambridge University Press

Let us observe that the atom p has two inclusion-minimal supports, namely $\{q\}$ and $\{r\}$. The atom q has just one support, namely $\{s\}$, and the atom t also has just one support, namely $\emptyset$. 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 only two subequations: $q \Leftrightarrow \neg s$, and $\neg q$. Similarly, t has only two subequations, t and $\neg t$, but the second one automatically leads to contradiction whenever it is 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. □

It should be clear that Proposition 4.1 implies an algorithm for the computation of stable models. Namely, we generate candidate theories and find their *propositional* models.

Let us observe that the algorithm described above can be implemented as a *two-tier backtracking search*, with the on-line computation of supports of guarded resolution proofs, and the usual backtracking scheme of DPLL (Davis *et al.* 1962). This second backtracking can be implemented using any DPLL-based SAT solver (Moskewicz *et al.* 2001). Proposition 4.1 implies that the algorithm we outlined is both sound and complete. Indeed, if the SAT solver returns a model M of theory T, then, by Proposition 4.1(1), M is a stable model for P. Otherwise we generate another candidate theory and loop through this process until one satisfying assignment is found. Proposition 4.1(2) guarantees the completeness of our algorithm.

Our algorithm is not using loop formulas like the algorithm of Lin and Zhao (2004) 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 (Ferraris *et al.* 2006) 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 because there may be exponentially many supports for any given atom p, the space complexity is $|P|$. This is the effect of not looking at loop formulas at all (Lifschitz and Razborov 2006).

https://doi.org/10.1017/S1471068410000062 Published online by Cambridge University Press

5 Conclusions and further work

We have shown that guarded unit resolution, a proof system that is a nonmonotonic version of unit resolution, is adequate for description of the Gelfond–Lifschitz operator GL_P and its fixpoints. That is, we can characterize the stable models of logic programs in terms of guarded resolution.

There are several natural questions concerning extensions of guarded resolution in the context of Answer Set Programming. For example:

(1) *Is there an analogous proof system for the disjunctive version of logic programming*?
or
(2) *Are there analogous proof systems for logic programming with constraints?*

We believe that availability of such proof systems could help with finding a variety of results on the complexity of reasoning in nonmonotonic logics. An interesting case is that of propositional Default Logic. We will show in a subsequent paper that that we can develop a similar guarded resolution proof scheme for propositional Default Logic. The main difference is that we must allow proof trees where the leaves can be tautologies rather than just guarded atoms or guarded clauses that are derived from the given program P as in this paper.

References

Baral, C. 2003. *Knowledge Representation, Reasoning and Declarative Problem Solving*, Cambridge University Press.

Brass, J. and Dix, J. 1999. Semantics of (Disjunctive) logic programs based on partial evaluation. *Journal of Logic Programming 40*, 1–46.

Davis, M., Logemann, G. and Loveland, D. W. 1962. A machine program for theorem-proving. *Communications of the ACM 5* (7), 394–397.

Dowling, W. F. and Gallier, J. H. 1984. Linear time algorithms for testing satisfiability of propositional Horn formulae. *Journal of Logic Programming 1*, 267–284.

Dung, P. M. and Kanchansut, K. 1989. A fixpoint approach to declarative semantics of logic programs. In *Proc. of North American Conference on Logic Programming*, E. L. Lusk and R. A. Overbeek, Eds. MIT Press, 604–625.

Erdem, E. and Lifschitz, V. 2003. Tight logic programs. *Theory and Practice of Logic Programming 3*, 499–518.

Fages, F. 1994. Consistency of Clark's completion and existence of stable models. *Journal of Methods of Logic in Computer Science 1*, 51–60.

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

Gelfond, M. and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Proc. of the Fifth International Conference and Symposium on Logic Programming*. R. A. Kowalski and K. A. Bowen, Eds. MIT Press, 1070–1080.

Karp, C. 1964. *Languages with Expressions of Infinitary length*. North-Holland, Amsterdam.

Lifschitz, V. 2008. Twelve definitions of a stable model. In *24th International Conference on Logic Programming, ICLP 2008*, M. Garcia de la Banda and E. Pontelli, Eds. Lecture Notes in Computer Science, vol. 5366. Springer, 37–51.

Lifschitz, V., Pearce, D. and Valverde, A. 2001. Strongly equivalent logic programs. *ACM Transactions on Computational Logic 2*, 526–541.

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

Lin, F. and Zhao, Y. 2004. ASSAT: Computing answer sets of a logic program by SAT solvers. *Artificial Intelligence 157* (1-2), 115–137.

Marek, V. W. and Remmel, J. B. 2010. An application of proof-theory in answer set programming. Preliminary version at `arXiv:0905.4076`.

Marek, V. W. and Truszczyński, M. 1999. Stable models and an alternative logic programming paradigm. In *The Logic Programming Paradigm*, Series Artificial Intelligence, Springer-Verlag, 375–398.

Moskewicz, M. W., Madigan, C. F., Zhao, Y., Zhang, L. and Mali, S. 2001. Chaff: Engineering an efficient SAT solver. In *Proc. of the 38th Design Automation Conference, DAC 2001*, ACM Press, 530–535.

Niemelä, I. 1999. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence 25* (3,4), 241–273.

Truszczynski, M. 2006. Strong and uniform equivalence of nonmonotonic theories – an algebraic approach. *Annals of Mathematics and Artificial Intelligence 48* (3-4), 245–265.