Definitions in Answer Set Programming

Selim T. Erdoğan and Vladimir Lifschitz

Department of Computer Sciences
University of Texas, Austin, USA
{selim, vl}@cs.utexas.edu

Abstract. In answer set programming, combinatorial search problems are solved by writing logic programs the answer sets of which correspond to solutions. Such programs often contain auxiliary atoms, "defined" in terms of atoms introduced earlier. To prove that the answer sets of a program containing definitions correspond to the solutions of the problem we want to solve, we need to understand how adding definitions affects the collection of answer sets. In particular, it is useful to be able to describe the effects of adding definitions to a program with nested expressions, in view of the relation of this class of programs to the input language of the answer set programming system SMODELS. In this paper we generalize the splitting set theorem to programs with nested expressions and show how this generalization can be used to prove program correctness in answer set programming. We also show that, under certain conditions, adding explicit and recursive definitions to a program with nested expressions extends its answer sets conservatively.

1 Introduction

The work described in this paper is motivated by our interest in the methodology of answer set programming [Marek and Truszczyński, 1999, Niemelä, 1999]. The idea of this form of logic programming is to solve a problem by writing a program the answer sets of which correspond to solutions.

The basic strategy for writing "answer set programs" is as follows. First a group of rules are written the answer sets of which correspond to potential solutions (an easily described superset of the set of solutions). Then some more rules ("constraints") are added which eliminate the "bad" potential solutions, so that the answer sets of the program with the newly added rules exactly correspond to solutions of the problem. This two-step approach is called the "generate and test" (or "guess and check" in [Eiter *et al.*, 2000]) approach, with the part of the program which generates the potential answer sets being the "generate" part and the part which eliminates some of those being the "test" part.

Before writing the test part of a program, it is often convenient to define some auxiliary atoms in terms of the atoms which occur in the generate part. Then these atoms can be used in constraints. The part of the program where the new atoms are defined can be called the "define" part.

To prove that the answer sets of a program designed according to this plan correspond to the solutions of the problem we want to solve, we need to know three things. First, we need to know the answer sets of the generate part of the program. This is usually straightforward due to the special syntactic form of rules used in the generate part. Second, we need to understand how the answer sets of the generate part are affected when we add definitions of auxiliary atoms. Third, we need to know how adding the constraints in the test part affects the answer sets. This last part is often straightforward also, due to the special syntactic form of constraints. In this paper we mainly concentrate on the second, most difficult problem, understanding how adding definitions affects the answer sets of a program. The splitting set theorem [Lifschitz and Turner, 1994] allows us, under certain conditions, to split a logic program into two parts and then determine how the answer sets of the first part are affected by adding the second part. In the work presented here, we propose to use splitting to determine how adding the define part of an "answer set program" affects the answer sets of its generate part.

Answer set semantics was originally introduced in [Gelfond and Lifschitz, 1988] and [Gelfond and Lifschitz, 1991] for logic programs consisting of rules of a simple syntactic form. Then [Lifschitz *et al.*, 1999] extended the answer set semantics to allow negation as failure (*not*), conjunction (,) and disjunction (;) to be nested arbitrarily, in both heads and bodies of rules. Nested expressions are important because many constructs used in the input language of LPARSE, the front-end to answer set programming systems SMODELS[1] and CMODELS[2], can be viewed as nested expressions [Ferraris and Lifschitz, 2003]. The splitting set theorem in [Lifschitz and Turner, 1994] applies to simple programs only, and here we extend it to programs with nested expressions.

The paper is organized as follows. We start off by reviewing the syntax and semantics of programs with nested expressions. In Section 3 we present the splitting set theorem for programs with nested expressions and give an example of its use. Section 4 describes our approach to proving program correctness in answer set programming, and the next section presents an example of this approach: proving the correctness of a program for computing Hamiltonian cycles. The theorem stated in Section 6 shows that adding definitions to a program extends it in a conservative way. Finally, Section 7 contains proofs of theorems.

2 Programs with Nested Expressions

A *literal* is a propositional atom possibly prefixed with the classical negation sign $\neg$. *Elementary formulas* are literals and the symbols $\bot$ ("false") and $\top$ ("true"). *Formulas* are built from elementary formulas using the unary connective *not* (negation as failure) and the binary connectives , (conjunction) and ; (disjunction). A *rule (with nested expressions)* has the form

$$Head \leftarrow Body$$

[1] `http://www.tcs.hut.fi/Software/smodels/` .

[2] `http://www.cs.utexas.edu/users/tag/cmodels.html` .

where both $Body$ and $Head$ are formulas. A rule of the form $Head \leftarrow \top$ can be abbreviated as $Head$, and a rule of the form $\bot \leftarrow Body$ can be abbreviated as $\leftarrow Body$. A *program (with nested expressions)* is a set of rules.

The semantics of such programs is characterized by defining when a consistent set Z of literals is an answer set for a program Π. As a preliminary step, we define when a set Z of literals *satisfies* a formula F (symbolically, $Z \models F$), as follows:

- for a literal L, $Z \models L$ if $L \in Z$,
- $Z \models \top$,
- $Z \not\models \bot$,
- $Z \models (F, G)$ if $Z \models F$ and $Z \models G$,
- $Z \models (F; G)$ if $Z \models F$ or $Z \models G$,
- $Z \models not\ F$ if $Z \not\models F$.

We say that Z *satisfies* a program Π (symbolically, $Z \models \Pi$) if, for every rule $Head \leftarrow Body$ in Π, $Z \models Head$ whenever $Z \models Body$.

The *reduct* F^Z of a formula F with respect to a set Z of literals is obtained by replacing every maximal occurrence of a subformula of the form $not\ G$ in F (that is, every occurrence of $not\ G$ that isn't in the range of another not)

- with $\bot$ if $Z \models G$,
- with $\top$, otherwise.

The *reduct* Π^Z of a program Π with respect to a set Z of literals is obtained by replacing every rule $F \leftarrow G$ in Π by the rule $F^Z \leftarrow G^Z$.

The concept of an answer set is defined first for programs not containing negation as failure: a consistent set Z of literals is an *answer set* for such a program Π if Z is a minimal set satisfying Π. For an arbitrary program Π, Z is an *answer set* for Π if Z is an answer set for the reduct Π^Z.

Here is an example. If $p_1, \dots, p_n$ are atoms, then

$$\{p_1, \dots, p_n\}^c \tag{1}$$

stands for the "choice formula"

$$(p_1; not\ p_1), \dots, (p_n; not\ p_n).$$

It's easy to see that the answer sets for the one-rule program (1) are arbitrary subsets of $\{p_1, \dots, p_n\}$. In answer set programming, rules like this are often used in the generate part of a program.

3 The Splitting Set Theorem for Programs with Nested Expressions

An occurrence of a formula F in a formula or a rule is called *singular* if the symbol before this occurrence is $\neg$; otherwise it is called *regular* [Lifschitz *et al.*, 1999]. Since classical negation can occur only in front of an atom, all occurrences of nonatomic formulas are regular.

We will use $head(r)$ to denote the set of all the literals that occur regularly in the head of a rule r, and $lit(r)$ to denote the set of all the literals that occur regularly in the whole rule. Of a program Π, $head(\Pi)$ will denote the union of $head(r)$ for all rules r in the program and $lit(\Pi)$ will denote the union of $lit(r)$ for all rules r.

A *splitting set* for a program Π is any set U of literals such that, for every rule $r \in \Pi$, if $head(r) \cap U \neq \emptyset$ then $lit(r) \subseteq U$. The set of rules $r \in \Pi$ such that $lit(r) \subseteq U$ is called the *bottom* of Π relative to U and denoted by $b_U(\Pi)$. The set $\Pi \setminus b_U(\Pi)$ is called the *top* of Π relative to U. It is clear that

$$head(\Pi \setminus b_U(\Pi)) \subseteq lit(\Pi) \setminus U. \tag{2}$$

As an example, consider the program Π consisting of the rules

$$\{a, b, c\}^c \tag{3}$$

and

$$d \leftarrow a, not\ b. \tag{4}$$

The latter can be viewed as a definition of d in terms of a and b. The set $U = \{a, b, c\}$ is a splitting set for this program. The bottom $b_U(\Pi)$ is (3) and the top $\Pi \setminus b_U(\Pi)$ is (4).

The function e_U defined below represents the process of "partial evaluation" of a formula. Consider two sets of literals U, X and a formula F. For each regular occurrence of a literal L in F such that $L \in U$, if $L \in X$ replace L with $\top$, otherwise replace L with $\bot$. The new formula obtained will be denoted by $e_U(F, X)$. For a program Π, we will denote by $e_U(\Pi, X)$ the program obtained by replacing each rule $F \leftarrow G$ of Π by $e_U(F, X) \leftarrow e_U(G, X)$.[3]

The following theorem generalizes the splitting set theorem from [Lifschitz and Turner, 1994] to programs with nested expressions.

Splitting Set Theorem. *Let U be a splitting set for a program Π. A consistent set of literals is an answer set for Π iff it can be written as $X \cup Y$ where X is an answer set for $b_U(\Pi)$ and Y is an answer set for $e_U(\Pi \setminus b_U(\Pi), X)$.*

Let us illustrate the use of the splitting set theorem by finding the answer sets for program (3), (4). We need to first find the answer sets X for (3), then "evaluate" (4) relative to each X, and find the answer sets Y of the resulting programs $e_U((4), X)$. According to the example at the end of Section 2, the

[3] In [Lifschitz and Turner, 1994], formula $e_U(\Pi, X)$ is formed from Π by dropping some parts of the program; in the definition above, the procedure consists of replacing some parts of the program by $\top$ and $\bot$. This is similar to the difference between the original definition of an answer set and the corresponding definition for programs with nested expressions.

answer sets for (3) are arbitrary subsets of $\{a, b, c\}$. The following table shows $e_U((4), X)$ for each of these sets.

X	$e_U((4), X)$
$\{\}$	$d \leftarrow \bot, not\ \bot$
$\{a\}$	$d \leftarrow \top, not\ \bot$
$\{b\}$	$d \leftarrow \bot, not\ \top$
$\{c\}$	$d \leftarrow \bot, not\ \bot$
$\{a, b\}$	$d \leftarrow \top, not\ \top$
$\{a, c\}$	$d \leftarrow \top, not\ \bot$
$\{b, c\}$	$d \leftarrow \bot, not\ \top$
$\{a, b, c\}$	$d \leftarrow \top, not\ \top$

Each program $e_U((4), X)$ has a unique answer set Y, which is $\{d\}$ if $a \in X$ and $b \notin X$, and $\{\}$ otherwise. Therefore, (3), (4) has 8 answer sets:

$$\{\}, \{a, d\}, \{b\}, \{c\}, \{a, b\}, \{a, c, d\}, \{b, c\}, \{a, b, c\}. \qquad (5)$$

Proposition 1. *Let U be a splitting set for a program Π, X an answer set for $b_U(\Pi)$, and Y an answer set for $e_U(\Pi \setminus b_U(\Pi), X)$. Then $X \subseteq lit(\Pi) \cap U$ and $Y \subseteq lit(\Pi) \setminus U$.*

This proposition shows that the sets X and Y from the statement of the splitting set theorem are always disjoint.

4 Proving Program Correctness

In answer set programming, to prove the correctness of a program means to show that its answer sets correspond exactly to the solutions of the problem we want to solve. In this section, we illustrate this process by applying it to a small program whose form is similar to the generate-define-test structure discussed in the introduction.

We have seen so far how answer sets of a generate part, such as (3), can be described and how we can describe the effect of adding a define part, such as (4). The rules added in the test part of a program are usually rules with a special form: $\leftarrow F$ where F is a formula. Such a rule is called a *constraint*. The following proposition explains how adding constraints to a program affects its answer sets.

Proposition 2. *For any program Π and formula F, a set Z of literals is an answer set for $\Pi \cup \{\leftarrow F\}$ iff Z is an answer set for Π and does not satisfy F.*

This proposition is similar to Proposition 2 in [Lifschitz *et al.*, 1999], and, as we'll see, can be easily derived from it.

As an example, consider the program

$$\begin{array}{l} \{a, b, c\}^c \\ d \leftarrow a, not\ b \\ \quad \leftarrow not\ d \end{array} \qquad (6)$$

which is obtained by adding a constraint to program (3), (4). According to Proposition 2 its answer sets can be characterized as the answer sets (5) of program (3), (4) that do not satisfy the body *not* d of that constraint. Consequently, the answer sets for (6) are the sets from list (5) that contain d: $\{a, d\}$ and $\{a, c, d\}$.

This method can be applied, in particular, to programs written in the language of LPARSE. That language is less expressive than the language defined in Section 2 but it includes "choice rules" (1). For instance the example program (6) can be written as

```
{a, b, c}.
d :- a, not b.
  :- not d.
```

in the input language of LPARSE.

5 Example: Hamiltonian Cycles

Now we will show how the method described above can be used to prove the correctness of a program for computing the Hamiltonian cycles in a directed graph. The program is reproduced, with minor changes, from [Marek and Truszczyński, 1999].

Let G be a finite directed graph $\langle V, E\rangle$ and let $u_0 \in V$ be a vertex in G. The set of atoms of the program Π consists of the expressions $in(u, v)$ for all edges $\langle u, v\rangle$ of G and the expressions $reachable(u)$ for all vertices u. The program consists of rules of the following forms:

$$\{in(u, v)\}^c \tag{7}$$

$$\begin{aligned} reachable(u) &\leftarrow in(u_0, u) \\ reachable(v) &\leftarrow reachable(u), in(u, v) \end{aligned} \tag{8}$$

$$\begin{aligned} &\leftarrow in(u, v), in(u, w) \quad (v \neq w) \\ &\leftarrow in(u, w), in(v, w) \quad (u \neq v) \\ &\leftarrow not\ reachable(u). \end{aligned} \tag{9}$$

In this program, (7) is the generate part, (8) is the define part and (9) is test part.

We'll identify every edge $\langle u, v\rangle$ with the atom $in(u, v)$.

Correctness Theorem for the Hamiltonian Cycle Program. *A set $X \subseteq E$ is a Hamiltonian cycle in G iff it can be represented in the form $Z \cap E$ for some answer set Z for Π. Furthermore, every Hamiltonian cycle X can be represented in this form in only one way.*

The first half of the statement shows that if we "hide" the atoms of the form $reachable(u)$ in the output of SMODELS, we will get the Hamiltonian cycles in G. The second half shows that the output will contain every Hamiltonian cycle only once.

The proof of this proposition illustrates the methodology for proving program correctness described in Section 4. In the proof we apply the splitting set theorem to the non-constraint part (7), (8) of the program. Let us call this part Π_1.

Given a set X of edges, let R_X denote the set of atoms $\mathit{reachable}(u)$ for all vertices u to which there is a nonempty path from vertex u_0 over edges in X.

Lemma. *A set Z of literals is an answer set for Π_1 iff $Z = X \cup R_X$ for some $X \subseteq E$.*

Proof. The set E of all $in(u,v)$ atoms is clearly a splitting set for Π_1. The bottom $b_E(\Pi_1)$ is (7), and the top $\Pi_1 \setminus b_E(\Pi_1)$ is (8).

The answer sets for $b_E(\Pi_1)$ are arbitrary subsets of E. For any such subset X the program $e_E(\Pi_1 \setminus b_E(\Pi_1), X)$ is

$$\begin{array}{ll}
\mathit{reachable}(u) & \text{if } in(u_0,u) \in X \\
\mathit{reachable}(u) \leftarrow \bot & \text{if } in(u_0,u) \notin X \\
\mathit{reachable}(v) \leftarrow \mathit{reachable}(u), \top & \text{if } in(u,v) \in X \\
\mathit{reachable}(v) \leftarrow \mathit{reachable}(u), \bot & \text{if } in(u,v) \notin X
\end{array}$$

which can be rewritten as

$$\begin{array}{ll}
\mathit{reachable}(u) & \text{if } in(u_0,u) \in X \\
\mathit{reachable}(v) \leftarrow \mathit{reachable}(u) & \text{if } in(u,v) \in X.
\end{array}$$

The only answer set for this program is the smallest set that satisfies it, which is R_X. By the splitting set theorem, it follows that a set of literals is an answer set for Π_1 iff it has the form $X \cup R_X$ for some $X \subseteq E$.

Proof of the Correctness Theorem for the Hamiltonian Cycle Program. A set X of edges is a Hamiltonian cycle iff it satisfies the following conditions:

(i) X does not contain two different edges leaving the same vertex.
(ii) X does not contain two different edges ending at the same vertex.
(iii) For every vertex v, there exists a nonempty path in X leading from u_0 to v.

Condition (i) can be expressed by saying that X does not satisfy the formulas

$$in(u,v), in(u,w) \qquad (v \neq w). \tag{10}$$

Condition (ii) can be expressed by saying that X does not satisfy the formulas

$$in(u,w), in(v,w) \qquad (u \neq v). \tag{11}$$

Condition (iii) can be expressed by saying that R_X does not satisfy the formulas

$$not\ \mathit{reachable}(u). \tag{12}$$

By Proposition 2, the condition

X has the form $Z \cap E$, where Z is an answer set for Π

is equivalent to

$$\begin{gathered}X \text{ has the form } Z \cap E,\\ \text{where } Z \text{ is an answer set for } \Pi_1 \text{ which doesn't satisfy (10)–(12)}\end{gathered}$$

because Π is the union of Π_1 with the constraints (9), the bodies of which are (10)–(12). By the lemma, this condition is equivalent to

$$\begin{gathered}X \text{ has the form } (X' \cup R_{X'}) \cap E\\ \text{for some } X' \subseteq E \text{ such that } X' \cup R_{X'} \text{ doesn't satisfy (10)–(12).}\end{gathered}$$

Since $(X' \cup R_{X'}) \cap E = X'$, this is further equivalent to the condition

$$X \cup R_X \text{ doesn't satisfy (10)–(12)}$$

which is equivalent to saying that X is a Hamiltonian cycle.

To prove the second claim of the correctness theorem, assume that for answer sets Z_1 and Z_2 of Π,

$$Z_1 \cap E = Z_2 \cap E. \tag{13}$$

By Proposition 2, Z_1 and Z_2 are answer sets for Π_1 as well. By the lemma, it follows that

$$Z_1 = X_1 \cup R_{X_1},\ \ Z_2 = X_2 \cup R_{X_2} \tag{14}$$

for some $X_1, X_2 \subseteq E$. Then

$$Z_1 \cap E = X_1,\ \ Z_2 \cap E = X_2.$$

By (13), it follows that $X_1 = X_2$ and, by (14), $Z_1 = Z_2$.

6 Definitional Extensions are Conservative

We expect the rules in the define part of a program to not add or remove potential solutions, but just to extend each of them by adding the defined atoms appropriately. It turns out that this can be guaranteed when certain syntactic conditions are satisfied. The proposition below shows that in logic programming, just as in classical logic, adding definitions produces a "conservative extension."

Proposition 3. *Let Π_1 be a program and Q be a set of atoms that do not occur in Π_1. Let Π_2 be a program that consists of rules of the form*

$$q \leftarrow F$$

where $q \in Q$, and F does not contain any element of Q in the scope of negation as failure. Then $Z \mapsto Z \setminus Q$ is a 1-1 correspondence between the answer sets for $\Pi_1 \cup \Pi_2$ and the answer sets for Π_1.

Notice that the bodies of the rules of Π_2 are allowed to contain atoms from Q as long as those atoms do not occur in the scope of negation as failure. In other words, Π_2 may contain recursive definitions in addition to explicit definitions.

For instance, let Π_1 be the generate part (7) of the program from the previous section, let Q be the set of atoms $reachable(u)$ for all vertices u, and let Π_2 be the define part (8) of the program. The rules of Π_2 are such that their heads are elements of Q and their bodies do not contain negation as failure. Proposition 3 describes a 1-1 correspondence between the answer sets of (7), (8) and the answer sets of (7), which is the same as the correspondence described in the statement of the lemma from Section 5.

Proposition 3 is similar to several results proved earlier: Proposition 2.1 in [Gelfond and Przymusinska, 1991], Proposition 1 in [Lifschitz and Turner, 1994] and Lemma on Explicit Definitions in [Ferraris and Lifschitz, 2003, Section 6.2]. It differs from the first two in that it covers programs with nested expressions, and from the last in that it allows definitions to be recursive.

7 Proofs

It is convenient to prove Proposition 1 before the splitting set theorem.

7.1 Proof of Proposition 1

We will need the following two facts.

Fact 1. *Given a formula F without negation as failure and two sets Z, Z' of literals such that $Z' \subseteq Z$, if $Z' \models F$ then $Z \models F$.*

(This is Lemma 1 from [Erdem and Lifschitz, 2003].)

For any formula F, by $lit(F)$ we denote the set of all the literals that occur regularly in F.

Fact 2. *For any formula F and any set Z of literals*

$$Z \models F \text{ iff } Z \cap lit(F) \models F.$$

(This is Lemma 4 from [Erdem and Lifschitz, 2003].)

Lemma 1. *For any answer set Z for a program Π, $Z \subseteq head(\Pi)$.*

Proof. We first prove this lemma for programs without negation as failure. Let Z be an answer set for such a program Π and let $Z' = Z \cap head(\Pi)$. Take any rule $F \leftarrow G$ in Π and assume that $Z' \models G$. Then, by Fact 1, $Z \models G$. Z is an answer set for Π so $Z \models F$ and, by Fact 2, $Z \cap lit(F) \models F$. Since

$$Z \cap lit(F) \subseteq Z \cap head(\Pi) = Z',$$

by Fact 1 it follows that $Z' \models F$. Since Z is a minimal set satisfying Π and Z' is a subset of Z satisfying Π, $Z = Z' \subseteq head(\Pi)$.

Now consider any arbitrary program Π. If Z is an answer set for Π, it is also an answer set for the reduct Π^Z. Since Π^Z does not contain negation as failure,

$$Z \subseteq head(\Pi^Z) \subseteq head(\Pi).$$

Proposition 1. *Let U be a splitting set for a program Π, X an answer set for $b_U(\Pi)$, and Y an answer set for $e_U(\Pi \setminus b_U(\Pi), X)$. Then $X \subseteq lit(\Pi) \cap U$ and $Y \subseteq lit(\Pi) \setminus U$.*

Proof. By Lemma 1,

$$X \subseteq head(b_U(\Pi)) \tag{15}$$

and

$$Y \subseteq head(e_U(\Pi \setminus b_U(\Pi), X)) \subseteq head(\Pi \setminus b_U(\Pi)). \tag{16}$$

By the definition of $b_U(\Pi)$,

$$head(b_U(\Pi)) \subseteq lit(b_U(\Pi)) \subseteq lit(\Pi) \cap U. \tag{17}$$

By (15), it follows that

$$X \subseteq lit(\Pi) \cap U.$$

By (16) and (2),

$$Y \subseteq lit(\Pi) \setminus U.$$

7.2 Proof of the Splitting Set Theorem

Fact 3. *For any formula F and set X, $X \models F$ iff $X \models F^X$.*

(This is part (i) of Lemma 3 from [Erdem and Lifschitz, 2003].)

Lemma 2. *Let U, X, Y, Y_1 be sets of literals such that $X \subseteq U$, $Y \cap U = \emptyset$ and $Y_1 \cap U = \emptyset$. (a) For any formula F,*

$$X \cup Y \models F^{X \cup Y_1} \text{ iff } Y \models e_U(F, X)^{Y_1}.$$

(b) For any program Π,

$$X \cup Y \models \Pi^{X \cup Y_1} \text{ iff } Y \models e_U(\Pi, X)^{Y_1}.$$

Proof. We'll prove part (a) by structural induction.

- F is a literal.

$$F^{X \cup Y_1} = F$$
$$e_U(F, X)^{Y_1} = e_U(F, X) = \begin{cases} \top, \text{if } F \in X \\ \bot, \text{if } F \in U \setminus X \\ F, \text{if } F \notin U \end{cases}$$

$$
\begin{aligned}
X \cup Y \models\ F^{X \cup Y_1} &\Leftrightarrow X \cup Y \models F \\
&\Leftrightarrow F \in X \cup Y \\
&\Leftrightarrow F \in X \text{ or } F \in Y \\
&\Leftrightarrow F \in X \text{ or } (F \in Y \text{ and } F \notin U) \\
&\Leftrightarrow F \in X \text{ or } (Y \models F \text{ and } F \notin U) \\
&\Leftrightarrow Y \models e_U(F,X)^{Y_1}
\end{aligned}
$$

- F is $\top$. Then $F^{X \cup Y_1} = \top$ and $e_U(F,X)^{Y_1} = \top$. So $X \cup Y \models F^{X \cup Y_1}$ and $Y \models e_U(F,X)^{Y_1}$.
- F is $\bot$. Similar to the previous case.
- F is (G,H) and the assertion of the lemma holds for G and H.

$$
\begin{aligned}
X \cup Y \models\ F^{X \cup Y_1} &\Leftrightarrow X \cup Y \models\ (G,H)^{X \cup Y_1} \\
&\Leftrightarrow X \cup Y \models (G^{X \cup Y_1}, H^{X \cup Y_1}) \\
&\Leftrightarrow X \cup Y \models G^{X \cup Y_1} \text{ and } X \cup Y \models H^{X \cup Y_1} \\
&\Leftrightarrow Y \models e_U(G,X)^{Y_1} \text{ and } Y \models e_U(H,X)^{Y_1} \\
&\Leftrightarrow Y \models (e_U(G,X)^{Y_1}, e_U(H,X)^{Y_1}) \\
&\Leftrightarrow Y \models (e_U(G,X), e_U(H,X))^{Y_1} \\
&\Leftrightarrow Y \models e_U((G,H),X)^{Y_1} \\
&\Leftrightarrow Y \models e_U(F,X)^{Y_1}
\end{aligned}
$$

- F is $(G;H)$ and the lemma holds for G and H. Similar to the previous case.
- F is *not* G and the assertion of the lemma holds for G. Using Fact 3,

$$
\begin{aligned}
X \cup Y \models\ F^{X \cup Y_1} &\Leftrightarrow F^{X \cup Y_1} = \top \\
&\Leftrightarrow (\textit{not}\ G)^{X \cup Y_1} = \top \\
&\Leftrightarrow X \cup Y_1 \not\models G \\
&\Leftrightarrow X \cup Y_1 \not\models G^{X \cup Y_1} \\
&\Leftrightarrow Y_1 \not\models e_U(G,X)^{Y_1} \\
&\Leftrightarrow Y_1 \not\models e_U(G,X) \\
&\Leftrightarrow Y_1 \models \textit{not}\ e_U(G,X) \\
&\Leftrightarrow Y_1 \models (\textit{not}\ e_U(G,X))^{Y_1} \\
&\Leftrightarrow (\textit{not}\ e_U(G,X))^{Y_1} = \top \\
&\Leftrightarrow e_U(\textit{not}\ G,X)^{Y_1} = \top \\
&\Leftrightarrow e_U(F,X)^{Y_1} = \top \\
&\Leftrightarrow Y \models e_U(F,X)^{Y_1}
\end{aligned}
$$

To prove part (b), observe that the left-hand side of the equivalence to be proved can be stated as follows: For each rule $F \leftarrow G$ in Π,

$$X \cup Y \models F^{X \cup Y_1} \text{ whenever } X \cup Y \models G^{X \cup Y_1}.$$

The right-hand side of the equivalence to be proved expresses that for each rule $F \leftarrow G$ in Π,

$$Y \models e_U(F,X)^{Y_1} \text{ whenever } Y \models e_U(G,X)^{Y_1}.$$

Part (a) shows that these two conditions are equivalent to each other.

Splitting Set Theorem. *Let U be a splitting set for a program Π. A set of literals is an answer set for Π iff it can be written as $X \cup Y$ where X is an answer set for $b_U(\Pi)$ and Y is an answer set for $e_U(\Pi \setminus b_U(\Pi), X)$.*

Proof. Right to Left: Assume that X is an answer set for $b_U(\Pi)$ and Y is an answer set for $e_U(\Pi \setminus b_U(\Pi), X)$. We need to show that $X \cup Y$ is an answer set for Π, which means that it is an answer set for $\Pi^{X\cup Y}$.

By the definition of $b_U(\Pi)$, all literals in $b_U(\Pi)$ are in U. By Proposition 1, $Y \cap U = \emptyset$. So $b_U(\Pi)^{X\cup Y} = b_U(\Pi)^X$. Consequently,

$$\Pi^{X\cup Y} = b_U(\Pi)^{X\cup Y} \cup (\Pi \setminus b_U(\Pi))^{X\cup Y} = b_U(\Pi)^X \cup (\Pi \setminus b_U(\Pi))^{X\cup Y}. \quad (18)$$

Take any rule r in $\Pi^{X\cup Y}$ and assume that $X \cup Y$ satisfies the body of r. We want to show that $X \cup Y$ satisfies the head of r also. By (18), r belongs either to $b_U(\Pi)^X$ or to $(\Pi \setminus b_U(\Pi))^{X\cup Y}$.

Case 1: r belongs to $b_U(\Pi)^X$. Then r has the form $F^X \leftarrow G^X$ for some rule $F \leftarrow G$ in $b_U(\Pi)$. Since $X \cup Y \models G^X$, by Fact 2, $X \models G^X$ and, since X is an answer set for $b_U(\Pi)^X$, $X \models F^X$. Consequently, by Fact 1, $X \cup Y \models F^X$.

Case 2: r belongs to $(\Pi \setminus b_U(\Pi))^{X \cup Y}$. Since Y is an answer set for the program $e_U(\Pi \setminus b_U(\Pi),\ X)^Y$, $Y \models e_U(\Pi \setminus b_U(\Pi),\ X)^Y$. It follows from Lemma 2(b) that $X \cup Y \models (\Pi \setminus b_U(\Pi))^{X \cup Y}$. Consequently $X \cup Y$ satisfies the head of r.

We have shown that $X \cup Y$ satisfies $\Pi^{X\cup Y}$. Now we need to show it is minimal. Take any subset of $X \cup Y$ satisfying $\Pi^{X\cup Y}$. We can represent this subset in the form $X_1 \cup Y_1$ so that $X_1 \subseteq X$, $Y_1 \subseteq Y$. We'll prove that $X_1 = X$, $Y_1 = Y$.

Since $X_1 \cup Y_1 \models \Pi^{X\cup Y}$, by (18), $X_1 \cup Y_1 \models b_U(\Pi)^X$. This means that, for any rule $F^X \leftarrow G^X$ in $b_U(\Pi)^X$, if $X_1 \cup Y_1 \models G^X$ then $X_1 \cup Y_1 \models F^X$. By Fact 2, this is equivalent to saying that, if $X_1 \models G^X$ then $X_1 \models F^X$. We know that X is an answer set for $b_U(\Pi)^X$, so no proper subset of it can satisfy $b_U(\Pi)^X$. Therefore $X_1 = X$.

Now we will show that Y_1 satisfies $e_U(\Pi \setminus b_U(\Pi),\ X)^Y$. From the fact that $X_1 \cup Y_1 \models \Pi^{X \cup Y}$ we conclude by (18) that $X_1 \cup Y_1 \models (\Pi \setminus b_U(\Pi))^{X \cup Y}$. Since $X_1 = X$, $X \cup Y_1 \models (\Pi \setminus b_U(\Pi))^{X \cup Y}$. From Lemma 2(b) it follows that Y_1 satisfies $e_U(\Pi \setminus b_U(\Pi), X)^Y$. Since Y is an answer set for $e_U(\Pi \setminus b_U(\Pi),\ X)^Y$, no proper subset of it can satisfy $e_U(\Pi \setminus b_U(\Pi),\ X)^Y$. Therefore, $Y_1 = Y$.

Left to Right: Assume that A is an answer set for Π. We need to show that $A = X \cup Y$ for two sets X and Y such that X is an answer set for $b_U(\Pi)$ and Y is an answer set for $e_U(\Pi \setminus b_U(\Pi), X)$. Take $X = A \cap U$ and $Y = A \setminus U$. Then $A = X \cup Y$. Furthermore, by (18),

$$X \cup Y \models \Pi^{X\cup Y} = b_U(\Pi)^X \cup (\Pi \setminus b_U(\Pi))^{X\cup Y}. \quad (19)$$

First we will show that X is an answer set for $b_U(\Pi)$. Take any rule $F^X \leftarrow G^X$ in $b_U(\Pi)^X$ and assume that $X \models G^X$. Then, by Fact 1, $X \cup Y \models G^X$. By (19), $X \cup Y$ satisfies $b_U(\Pi)^X$, so $X \cup Y \models F^X$ also. By Fact 2, this means that $X \models F^X$.

We showed that X satisfies $b_U(\Pi)^X$. Now we need to show that it is minimal. Take any subset X_1 of X which satisfies $b_U(\Pi)^X$ and take any rule r in $\Pi^{X \cup Y}$. Assume that $X_1 \cup Y$ satisfies the body of r. We want to show that $X_1 \cup Y$ satisfies the head of r also. By (19), r belongs either to $b_U(\Pi)^X$ or to $(\Pi \setminus b_U(\Pi))^{X \cup Y}$.

Case 1: r belongs to $b_U(\Pi)^X$. Then r has the form $F^X \leftarrow G^X$. From the fact that $X_1 \cup Y \models G^X$ we can conclude by Fact 2 that $X_1 \models G^X$. Because X_1 satisfies $b_U(\Pi)^X$, $X_1 \models F^X$. Consequently, by Fact 1, $X_1 \cup Y \models F^X$.

Case 2: r belongs to $(\Pi \setminus b_U(\Pi))^{X \cup Y}$. Then r has the form $F^{X \cup Y} \leftarrow G^{X \cup Y}$. Since $X_1 \cup Y \models G^{X \cup Y}$, by Fact 1, $X \cup Y \models G^{X \cup Y}$. Because $X \cup Y$ satisfies $(\Pi \setminus b_U(\Pi))^{X \cup Y}$, $X \cup Y \models F^{X \cup Y}$. By Fact 2,

$$(X \cup Y) \cap lit(F^{X \cup Y}) \models F^{X \cup Y}. \tag{20}$$

By (2),

$$lit(F^{X \cup Y}) \subseteq head(\Pi \setminus b_U(\Pi)) \subseteq lit(\Pi) \setminus U.$$

Since $X \subseteq U$, $(X \cup Y) \cap lit(F^{X \cup Y}) = Y \cap lit(F^{X \cup Y})$. By (20) and Fact 2, $Y \models F^{X \cup Y}$. So, by Fact 1, $X_1 \cup Y \models F^{X \cup Y}$.

We have shown that $X_1 \cup Y$ satisfies $\Pi^{X \cup Y}$. We know that $X \cup Y$ is an answer set for $\Pi^{X \cup Y}$, so no proper subset of it can satisfy $\Pi^{X \cup Y}$. Therefore, $X_1 = X$. Since X is a minimal set satisfying $b_U(\Pi)^X$, it is an answer set for $b_U(\Pi)^X$.

Now we will show that Y is an answer set for $e_U(\Pi \setminus b_U(\Pi), X)$. By (19), $X \cup Y \models (\Pi \setminus b_U(\Pi))^{X \cup Y}$. So, by Lemma 2(b), $Y \models e_U(\Pi \setminus b_U(\Pi), X)^Y$. To show that Y is minimal among the sets satisfying $e_U(\Pi \setminus b_U(\Pi), X)^Y$, take any subset Y_1 of Y such that $Y_1 \models e_U(\Pi \setminus b_U(\Pi), X)^Y$. By Lemma 2(b), $X \cup Y_1 \models (\Pi \setminus b_U(\Pi))^{X \cup Y}$. We know that $X \cup Y$ is an answer set for $\Pi^{X \cup Y}$, so no proper subset of it can satisfy $\Pi^{X \cup Y}$. Therefore, $Y_1 = Y$. Since Y is a minimal set satisfying $e_U(\Pi \setminus b_U(\Pi), X)^Y$, it is an answer set for $e_U(\Pi \setminus b_U(\Pi), X)^Y$.

7.3 Proof of Proposition 2

Proposition 2. *For any program Π and formula F, a set Z of literals is an answer set for $\Pi \cup \{\leftarrow F\}$ iff Z is an answer set for Π and does not satisfy F.*

Proof. By Proposition 2 from [Lifschitz *et al.*, 1999], Z is an answer set for $\Pi \cup \{\leftarrow F\}$ iff Z is an answer set for Π and does not satisfy F^Z. By Fact 3, this is equivalent to the assertion to be proved.

7.4 Proof of Proposition 3

Lemma 3. *If the head of every rule of a program Π is an atom, and the bodies of the rules of Π do not contain atoms in the scope of negation as failure, then Π has a unique answer set.*

Proof. The equivalent transformations listed in Proposition 4 from [Lifschitz *et al.*, 1999] allow us to convert any formula that does not contain atoms into either $\bot$ or $\top$. Consequently we can assume, without loss of generality, that Π does not contain negation as failure. Let X be the intersection of all sets of literals that satisfy Π. It is easy to check, using Fact 1, that X satisfies Π. Consequently, X is the only answer set of Π.

Proposition 3. *Let Π_1 be a program and Q be a set of atoms that do not occur in Π_1. Let Π_2 be a program that consists of rules of the form*

$$q \leftarrow F$$

where $q \in Q$, and F does not contain any element of Q in the scope of negation as failure. Then $Z \mapsto Z \setminus Q$ is a 1-1 correspondence between the answer sets for $\Pi_1 \cup \Pi_2$ and the answer sets for Π_1.

Proof. Take

$$U = lit(\Pi_1 \cup \Pi_2) \setminus Q.$$

It is easy to see that U is a splitting set for $\Pi_1 \cup \Pi_2$, and that $b_U(\Pi_1 \cup \Pi_2) = \Pi_1$. Since Π_1 and Π_2 are disjoint, it follows that

$$(\Pi_1 \cup \Pi_2) \setminus b_U(\Pi_1 \cup \Pi_2) = (\Pi_1 \cup \Pi_2) \setminus \Pi_1 = \Pi_2.$$

For any set X of literals, program $e_U(\Pi_2, X)$ satisfies the conditions of Lemma 3. Denote the unique answer set of this program by $f(X)$. By Lemma 1,

$$f(X) \subseteq head(e_U(\Pi_2, X)) = head(\Pi_2) \subseteq Q.$$

According to the splitting set theorem, a consistent set of literals is an answer set for $\Pi_1 \cup \Pi_2$ iff it can be written as $X \cup f(X)$ where X is an answer set for Π_1. Let X be an answer set for Π_1. By Lemma 1, every literal in X occurs in Π_1. Consequently, X is disjoint from Q, so that

$$(X \cup f(X)) \setminus Q = (X \setminus Q) \cup (f(X) \setminus Q) = X. \tag{21}$$

It follows that the function $Z \mapsto Z \setminus Q$ maps every answer set for $\Pi_1 \cup \Pi_2$ to an answer set for Π_1. Every answer set X for Π_1 can be represented in the form $Z \setminus Q$ where Z is an answer set for $\Pi_1 \cup \Pi_2$. Indeed, we can take $Z = X \cup f(X)$; this set is consistent, because X is consistent and $f(X)$ is a set of atoms that don't occur in X. Finally, the function is 1-1: if

$$(X_1 \cup f(X_1)) \setminus Q = (X_2 \cup f(X_2)) \setminus Q$$

then, by (21), $X_1 = X_2$.

Acknowledgements

We are grateful to Paolo Ferraris for his comments and suggestions. This work was partially supported by the Texas Higher Education Coordinating Board under Grant 003658-0322-2001.

References

[Eiter *et al.*, 2000] Thomas Eiter, Wolfgang Faber, Nicola Leone, and Gerald Pfeifer. Declarative problem-solving using the DLV system. In Jack Minker, editor, *Logic-Based Artificial Intelligence*, pages 79–103. Kluwer, 2000.

[Erdem and Lifschitz, 2003] Esra Erdem and Vladimir Lifschitz. Tight logic programs. *Theory and Practice of Logic Programming*, 2003. To appear.

[Ferraris and Lifschitz, 2003] Paolo Ferraris and Vladimir Lifschitz. Weight constraints as nested expressions. In progress, 2003.

[Gelfond and Lifschitz, 1988] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert Kowalski and Kenneth Bowen, editors, *Logic Programming: Proc. Fifth Int'l Conf. and Symp.*, pages 1070–1080, 1988.

[Gelfond and Lifschitz, 1991] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.

[Gelfond and Przymusinska, 1991] Michael Gelfond and Halina Przymusinska. Definitions in epistemic specifications. In Anil Nerode, Victor Marek, and V. S. Subramanian, editors, *Logic Programming and Non-monotonic Reasoning: Proceedings of the First Int'l Workshop*, pages 245–259, 1991.

[Lifschitz and Turner, 1994] Vladimir Lifschitz and Hudson Turner. Splitting a logic program. In Pascal Van Hentenryck, editor, *Proc. Eleventh Int'l Conf. on Logic Programming*, pages 23–37, 1994.

[Lifschitz *et al.*, 1999] Vladimir Lifschitz, Lappoon R. Tang, and Hudson Turner. Nested expressions in logic programs. *Annals of Mathematics and Artificial Intelligence*, 25:369–389, 1999.

[Marek and Truszczyński, 1999] Victor Marek and Mirosław Truszczyński. Stable models and an alternative logic programming paradigm. In *The Logic Programming Paradigm: a 25-Year Perspective*, pages 375–398. Springer Verlag, 1999.

[Niemelä, 1999] Ilkka Niemelä. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence*, 25:241–273, 1999.