Loop Formulas for Disjunctive Logic Programs

Joohyung Lee and Vladimir Lifschitz

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

Abstract. We extend Clark's definition of a completed program and the definition of a loop formula due to Lin and Zhao to disjunctive logic programs. Our main result, generalizing the Lin/Zhao theorem, shows that answer sets for a disjunctive program can be characterized as the models of its completion that satisfy the loop formulas. The concept of a tight program and Fages' theorem are extended to disjunctive programs as well.

1 Introduction

Among the theories proposed to explain the semantics of "negation as failure" in logic programming, influential are the completion semantics [Clark, 1978] and the stable model or the answer set semantics [Gelfond and Lifschitz, 1988]. It is well known that an answer set for a logic program is also a model of its completion, while the converse, generally, does not hold.

When are the two semantics equivalent to each other? François Fages [1994] showed that a certain syntactic condition, which is now called "tightness", is sufficient for establishing the equivalence between them. Esra Erdem and Vladimir Lifschitz [2003] generalized Fages' theorem and extended it to programs with nested expressions (in the sense of [Lifschitz *et al.*, 1999]) in the bodies of rules.

In view of Fages' theorem, answer sets for a tight program can be computed by a satisfiability solver (SAT solver) [Babovich *et al.*, 2000]. CMODELS[1], a system based on this idea, uses SAT solvers to compute answer sets for finite nondisjunctive logic programs that are tight or that can be turned into tight programs by simple equivalent transformations. In several benchmark examples, it found answer sets faster than SMODELS[2], a general-purpose answer set solver.

On the other hand, Fangzhen Lin and Yuting Zhao [2002] were interested in the relationship between completion and answer sets from a different point of view. They showed that a rather simple extension of a program's completion gives a set of formulas that describes the program's answer sets correctly in all cases—even when the program is not tight. We just need to extend the completion by adding what they call "loop formulas." According to their main theorem, models

[1] `http://www.cs.utexas.edu/users/tag/cmodels.html` .
[2] `http://www.tcs.hut.fi/Software/smodels/` .

of the extended completion are identical to the program's answer sets. Their system ASSAT[3] is similar to CMODELS in the sense that it uses SAT solvers for computing answer sets, but it is not limited to tight programs.

However, the concepts of completion and a loop formula have been defined so far for nondisjunctive programs only. In some cases disjunctive rules can be equivalently turned into nondisjunctive rules (see, for instance, [Erdem and Lifschitz, 1999]). But a complexity result tells us that there is no efficient translation from arbitrary disjunctive programs to nondisjunctive programs: the problem of the existence of an answer set for a disjunctive program is Σ_2^P-hard [Eiter and Gottlob, 1993, Corollary 3.8], while the same problem for a nondisjunctive program is in class NP.

In this note, we extend Clark's completion and the definition of a loop formula due to Lin and Zhao to disjunctive logic programs and, more generally, to arbitrary programs with nested expressions. Our main theorem generalizes the Lin/Zhao theorem mentioned above and shows that answer sets for a disjunctive program can be characterized as the models of its completion that satisfy the loop formulas. We extend the concept of a tight program to disjunctive programs as well. For a tight program, adding the loop formulas does not affect the models of completion. This fact leads to extending Fages' theorem to disjunctive programs.

A puzzling feature of the Lin and Zhao's extension of completion with loop formulas is that even though their procedure is essentially a reduction of one NP-complete problem to another NP-complete problem, it is not a polynomial time reduction. The number of loop formulas can grow exponentially in the worst case. Our result explains this phenomenon: since a small modification of the definition of a loop formula is applicable to a Σ_2^P-hard problem, the possibility of exponentially many loop formulas is natural.

Another reason why our extension of the Lin/Zhao theorem may be interesting is that it provides an alternative method of computing answer sets for programs more general than those that are currently handled by ASSAT. Yuliya Babovich and Marco Maratea are using this idea to adapt the ASSAT algorithm to programs with cardinality constraints (personal communication). It would be interesting to extend their implementation to disjunctive programs and compare its performance with that of DLV[4], an existing answer set solver that can deal with disjunctive programs.

We begin with a review of the definition of answer sets in the next section, and extend Clark's completion to disjunctive programs in Section 3. The concepts of a loop formula and the main result of [Lin and Zhao, 2002] are extended to disjunctive programs in Section 4, and the definition of tightness is extended to disjunctive programs in Section 5. Related work is discussed in Section 6, and the proofs are presented in Section 7.

---

[3] `http://assat.cs.ust.hk/` .

[4] `http://www.dbai.tuwien.ac.at/proj/dlv/` .

2 Review of the Answer Set Semantics

This section is a review of the answer set semantics for finite programs with nested expressions, but without classical negation.[5]

We first define the syntax of formulas, rules and logic programs. *Formulas* are formed from propositional atoms and 0-place connectives $\top$ and $\bot$ using negation (*not*), conjunction (,) and disjunction (;). If $m = 0$ then $F_1, \dots, F_m$ is understood as $\top$ and $F_1; \dots; F_m$ is understood as $\bot$.

A *rule* is an expression of the form

$$Head \leftarrow Body$$

where *Head* and *Body* are formulas. A rule of the form $Head \leftarrow \top$ can be abbreviated as *Head*.

A *(logic) program* is a finite set of rules.

The *satisfaction relation* $X \models F$ between a set X of atoms and a formula F is defined recursively, as follows:

- for an atom a, $X \models a$ if $a \in X$
- $X \models \top$
- $X \not\models \bot$
- $X \models (F, G)$ if $X \models F$ and $X \models G$
- $X \models (F; G)$ if $X \models F$ or $X \models G$
- $X \models not\ F$ if $X \not\models F$.

This definition is equivalent to the usual definition of satisfaction in propositional logic if we agree (as we do in this note) to identify '*not*' with '$\neg$', ',' with '$\wedge$', and ';' with '$\vee$'.

We say that a set X of atoms *satisfies* a rule $Head \leftarrow Body$ if $X \models Head$ whenever $X \models Body$, and say that X *satisfies* a program Π (symbolically, $X \models \Pi$) if X satisfies every rule of Π.

The *reduct* F^X of a formula F with respect to a set X of atoms is defined recursively, as follows:

- if F is an atom or a 0-place connective, then $F^X = F$
- $(F, G)^X = F^X, G^X$
- $(F; G)^X = F^X; G^X$
- $(not\ F)^X = \begin{cases} \bot\,, & \text{if } X \models F, \\ \top\,, & \text{otherwise.} \end{cases}$

The *reduct* Π^X of a program Π with respect to X is the set of rules

$$F^X \leftarrow G^X$$

[5] The syntax of a program defined in [Lifschitz *et al.*, 1999] is more general than the syntax defined here in that a program can be infinite and can contain classical negation ($\neg$). Classical negation can be easily eliminated by introducing auxiliary atoms as explained in [Lifschitz *et al.*, 2001, Section 5].

for all rules $F \leftarrow G$ in Π.

Finally, a set X of atoms is an *answer set* for a program Π if X is minimal among the sets of atoms that satisfy the reduct Π^X [Lifschitz *et al.*, 1999].[6]

A program Π_1 is *strongly equivalent* to a program Π_2 if, for every logic program Π, $\Pi_1 \cup \Pi$ has the same answer sets as $\Pi_2 \cup \Pi$ [Lifschitz *et al.*, 2001]. Section 4 of [Lifschitz *et al.*, 1999] contains a number of examples of strongly equivalent programs. (The equivalence relation defined in that paper is even stronger than strong equivalence.)

We understand the term *clause* to mean a disjunction of distinct atoms $a_1; \dots; a_n$ $(n \geq 0)$, and we identify a clause with the corresponding set $\{a_1, \dots, a_n\}$.

Using Propositions 3–6 of [Lifschitz *et al.*, 1999], it is easy to prove the following fact.

Fact 1 *Any rule is strongly equivalent to a finite set of rules of the form*

$$A \leftarrow Body \tag{1}$$

$(n \geq 0)$ *where A is a clause and $Body$ is a formula.*

The form of rules can be made even more special:

Fact 2 *Any rule is strongly equivalent to a finite set of rules of the form*

$$A \leftarrow B, F \tag{2}$$

where A is a clause, B is a conjunction of atoms and F is a formula in which every occurrence of each atom is in the scope of negation as failure.

We can further specialize the form of F and require that it be a formula of the form

$$not\ a_1, \dots, not\ a_m, not\ not\ a_{m+1}, \dots, not\ not\ a_n$$

$(0 \leq m \leq n)$, where a_i $(0 \leq i \leq n)$ are atoms.

3 Completion

Let Π be a logic program whose rules have the form (1). The *completion* of Π, $Comp(\Pi)$, is defined to be the set of propositional formulas that consists of the implication

$$Body \supset A \tag{3}$$

for every rule (1) in Π, and the implication

$$a \supset \bigvee_{\substack{A \leftarrow Body\ \in\ \Pi \\ a \in A}} \left(Body \wedge \bigwedge_{p \in A \setminus \{a\}} \neg p \right) \tag{4}$$

[6] In this definition, minimality is understood as minimality relative to set inclusion. In other words, we say that X is an answer set for Π if X satisfies Π^X but no proper subset of X satisfies Π^X.

for each atom a. When the head of every rule of Π is a single atom, $Comp(\Pi)$ is equivalent to the propositional case of completion defined in [Lloyd and Topor, 1984].

For instance, let Π_1 be the program:

$$p \; ; \; q.$$

The answer sets for Π_1 are $\{p\}$ and $\{q\}$. $Comp(\Pi_1)$ is

$$\begin{array}{l} p \vee q \\ p \supset \neg q \\ q \supset \neg p \end{array}$$

and its models are $\{p\}$ and $\{q\}$ also.[7]

Π_2 is the following program that adds two rules to Π_1:

$$\begin{array}{l} p \; ; \; q \\ p \leftarrow q \\ q \leftarrow p \end{array}$$

The only answer set for Π_2 is $\{p, q\}$. $Comp(\Pi_2)$ is

$$\begin{array}{l} p \vee q \\ q \supset p \\ p \supset q \\ p \;\supset\; \neg q \vee q \\ q \;\supset\; \neg p \vee p, \end{array}$$

and its only model is $\{p, q\}$ also.

The following proposition generalizes the property of completion familiar from [Marek and Subrahmanian, 1989] to disjunctive programs:

Proposition 1 *For any program Π whose rules have the form (1) and any set X of atoms, if X is an answer set for Π, then X is a model of $Comp(\Pi)$.*

It is well known that the converse of Proposition 1 does not hold; the one-rule program $p \leftarrow p$ is a standard counterexample. The following program Π_3 is another example of this kind, which contains a disjunctive rule:

$$\begin{array}{l} p \; ; \; r \leftarrow q \\ q \leftarrow p \\ p \leftarrow not\ r \\ r \leftarrow r. \end{array}$$

[7] We identify an interpretation with the set of atoms that are true in it.

The only answer set for Π_3 is $\{p, q\}$. $Comp(\Pi_3)$ is

$$\begin{aligned} q &\supset p \vee r \\ p &\supset q \\ \neg r &\supset p \\ r &\supset r \\ p &\supset (q \wedge \neg r) \vee \neg r \\ q &\supset p \\ r &\supset (q \wedge \neg p) \vee r \end{aligned} \tag{5}$$

and its models are $\{p, q\}$ and $\{r\}$.

In the next section, the method of strengthening the completion defined in [Lin and Zhao, 2002] is extended to arbitrary programs whose rules have the form (2).

4 Loop Formulas

For any formula F, by $pa(F)$ we denote the set of its "positive atoms": the set of all atoms a such that at least one occurrence of a in F is not in the scope of negation as failure.

Let Π be a program. The *positive dependency graph* of Π is the directed graph G such that

- the vertices of G are the atoms occurring in Π, and
- for every rule $Head \leftarrow Body$ in Π, G has an edge from each atom in $pa(Head)$ to each atom in $pa(Body)$.

For instance, if all rules of Π have the form (2), then the edges of G go from atoms a_i to atoms b_j.

A nonempty set L of atoms is called a *loop* of Π if, for any pair a_1, a_2 of atoms in L, there exists a path of nonzero length from a_1 to a_2 in the positive dependency graph of Π such that all vertices in this path belong to L. For example, for the programs in Section 3, Π_1 has no loops; Π_2 has one loop: $\{p, q\}$; Π_3 has two loops: $\{p, q\}$ and $\{r\}$.

The definition of a loop is essentially equivalent to the corresponding definition from [Lin and Zhao, 2002] in the case when the program is nondisjunctive.[8]

The rest of this section assumes that every rule of Π has the form (2).

For any loop L of Π, by $R(L)$ we denote the set of formulas

$$B \wedge F \wedge \bigwedge_{p \in A \setminus L} \neg p$$

for all rules (2) in Π such that $A \cap L \neq \emptyset$ and $pa(B) \cap L = \emptyset$. By $CLF(L)$ we denote the *conjunctive loop formula* for L:

$$CLF(L) = \bigwedge L \supset \bigvee R(L).^9 \tag{6}$$

[8] A program is *nondisjunctive* if the head of every rule in it is either an atom or $\bot$.

[9] The expression $\bigwedge L$ in the antecedent stands for the conjunction of all elements of L. The expression in the consequent has a similar meaning.

By $CLF(\Pi)$ we denote the set of all conjunctive loop formulas for Π:

$$CLF(\Pi) = \{CLF(L)\ :\ L \textit{ is a loop of } \Pi\}.$$

For instance, $CLF(\Pi_1)$ is $\emptyset$ because Π_1 has no loops. For the only loop $L = \{p, q\}$ of Π_2, $R(L)$ is $\{\top\}$, so that $LF(\Pi_2) = \{p \wedge q \supset \top\}$, which is tautological. For the two loops $L_1 = \{p, q\}$ and $L_2 = \{r\}$ of Π_3, $R(L_1)$ is $\{\neg r\}$ and $R(L_2)$ is $\{q \wedge \neg p\}$, so that $CLF(\Pi_3) = \{p \wedge q \supset \neg r,\ r \supset q \wedge \neg p\}$.

An alternative (stronger) definition of a loop formula replaces $\bigwedge$ in the antecedent of (6) with $\bigvee$. Let's call this formula $DLF(L)$ (*disjunctive loop formula*):

$$DLF(L) = \bigvee L\ \supset\ \bigvee R(L). \tag{7}$$

This second definition is a direct generalization of the Lin/Zhao definition of a loop formula to disjunctive programs. $DLF(\Pi)$ is the set of all disjunctive loop formulas for Π.

Syntactically, the loop formulas (6), (7) are somewhat similar to completion formulas (4). One difference is that their antecedent may contain several atoms, rather than a single atom. Accordingly, the multiple conjunction in the consequent of a loop formula extends over the atoms in $A \setminus L$, instead of the atoms in $A \setminus \{a\}$. Second, the multiple disjunction in the consequent of a loop formula is limited to the rules satisfying the condition $pa(B) \cap L = \emptyset$; this condition has no counterpart in the definition of completion.

Intuitively, each disjunct in the consequent of a completion formula (4) characterizes a source of support for the atom a, whereas each disjunct in the consequent of loop formulas (6), (7) characterizes a source of support for the atoms in $A \cap L$ where the support comes from outside of L.

We are now ready to state our main theorem.

Theorem 1 *For any program Π whose rules have the form (2) and any set X of atoms, the following conditions are equivalent:*

(a) X is an answer set for Π,
(b) X is a model of $Comp(\Pi) \cup CLF(\Pi)$,
(c) X is a model of $Comp(\Pi) \cup DLF(\Pi)$.

The theorem tells us that answer sets for any logic program Π can be found by computing the models of the corresponding propositional theory $Comp(\Pi) \cup CLF(\Pi)$ (or $Comp(\Pi) \cup DLF(\Pi)$). For example, the answer sets for each of the programs Π_1, Π_2 can be found by computing the models of its completion. The only answer set for Π_3 is the first of the two models $\{p, q\}$ and $\{r\}$ of its completion (5) because that model satisfies the loop formulas $p \wedge q \supset \neg r$, $r \supset q \wedge \neg p$; the second model of the completion does not have this property.

From the fact that (*b*) and (*c*) are equivalent to each other we can infer that the antecedent of a loop formula can be more general. For any loop L, let F_L be a formula formed from atoms in L using conjunctions and disjunctions. For instance, F_L can be any single atom that belongs to L or a conjunction or disjunction of several atoms from L. It is clear that F_L entails $\bigvee L$ and is entailed by $\bigwedge L$. We conclude:

Corollary 1 *For any program Π whose rules have the form (2) and any set X of atoms, the condition that X is a model of*

$$Comp(\Pi) \cup \{F_L \supset \bigvee R(L) \ : \ L \text{ is a loop of } \Pi\}$$

is equivalent to each of conditions (a)–(c) from Theorem 1.

5 Tight Programs

About a program we say that it is *absolutely tight* if it has no loops. For instance, program Π_1 from Section 3 is absolutely tight, and programs Π_2, Π_3 are not. In the case when the program is nondisjunctive, this definition has the same meaning as the definition from [Erdem and Lifschitz, 2003]. On the other hand, the definition from [Erdem and Lifschitz, 2003] is more general in the sense that it is applicable to infinite programs and programs with classical negation.

It is clear from Theorem 1 that for any absolutely tight program Π whose rules have the form (2), X is an answer set for Π iff X satisfies $Comp(\Pi)$. This corollary can be extended to programs of a more general form:

Proposition 2 *For any absolutely tight program Π whose rules have the form (1) and any set X of atoms, X is an answer set for Π iff X satisfies $Comp(\Pi)$.*

This proposition is a generalization of Corollary 6 from [Erdem and Lifschitz, 2003] to disjunctive programs. It shows that the method of computing answer sets based on completion [Babovich *et al.*, 2000] can be extended to tight programs whose rules have the form (1). In view of the following proposition, the restriction on the form of rules of the program is not essential.

Proposition 3 *Any absolutely tight program is strongly equivalent to an absolutely tight program whose rules have the form (1).*

Besides absolute tightness, Erdem and Lifschitz [2003] define also tightness relative to a set. That definition can be extended to disjunctive programs as follows.

Let Π be a program, and let X be a set of atoms. By Π_X we denote the set of rules of Π whose bodies are satisfied by X. We say that Π is *tight* on a set X of atoms if the subgraph of the positive dependency graph of Π_X induced by X has no loops.

This definition is equivalent to the corresponding definition from [Erdem and Lifschitz, 2003] in the case when the program is nondisjunctive. The following two propositions are similar to Propositions 2, 3.

Proposition 4 *For any program Π whose rules have the form (1) and any set X of atoms such that Π is tight on X, X is an answer set for Π iff X satisfies $Comp(\Pi)$.*

Proposition 5 *For any set X of atoms, any program tight on X is strongly equivalent to a program tight on X whose rules have the form (1).*

Proposition 4 is a generalization of Corollary 2 from [Erdem and Lifschitz, 2003] to disjunctive programs.

6 Related Work

Baral and Gelfond [1994] and Inoue and Sakama [1998] extended the definition of a supported set to the case of disjunctive programs. In application to programs whose rules have the form (1), their definition can be stated as follows: a set X of atoms is *supported by Π* if, for each atom $a \in X$, there exists a rule (1) in Π such that $X \cap A = \{a\}$ and $X \models Body$. It is easy to check that X satisfies $Comp(\Pi)$ iff X satisfies Π and is supported by Π. In view of this fact, Proposition 1 is similar to Proposition 4.1 from [Baral and Gelfond, 1994] and to Theorem 5.3 from [Inoue and Sakama, 1998], which assert that every answer set is supported.

Rachel Ben-Eliyahu and Rina Dechter [1996] showed that a certain class of disjunctive programs (called head-cycle-free), that includes all nondisjunctive programs, can be translated into propositional theories. Their translation may introduce extra atoms, while ours does not, just as the original Lin/Zhao translation.

7 Proofs

The following facts are stated in [Erdem and Lifschitz, 2003] as Lemma 1 and Lemma 3(i).

Fact 3 *Given a formula F without negation as failure and two sets X, Y of atoms such that $Y \subseteq X$, if $Y \models F$ then $X \models F$.*

Fact 4 *For any formula F and any set X of atoms, $X \models F$ iff $X \models F^X$.*

7.1 Proof of Proposition 1

Lemma 1 *If X is a minimal set of atoms that satisfies a set Γ of clauses, then for each $a \in X$ there exists a clause A in Γ such that*

$$X \cap A = \{a\}$$

Proof If this is not the case, then there exists an atom $a \in X$ such that every clause in Γ that contains a also includes another atom that is in X. It follows that $X \setminus \{a\}$ also satisfies Γ, which contradicts with the minimality of X. ∎

Lemma 2 *Let Π be a program whose rules have the form (1), let X be a set of atoms, and let Γ be the set of the heads of the rules in Π whose bodies are satisfied by X. If X is an answer set for Π, then X is a minimal set of atoms that satisfies Γ.*

Proof Let X be an answer set for Π. First we will prove that X satisfies Γ. Take any clause A in Γ and consider a corresponding rule $A \leftarrow Body$ in Π such that $X \models Body$. By Fact 4, $X \models Body^X$. Since X is an answer set for Π, X satisfies $A \leftarrow Body^X$. Consequently, $X \models A$. It follows that X satisfies Γ.

Next we will prove that X is minimal among the sets of atoms that satisfies Γ. Let Y be any subset of X that satisfies Γ. We will show that Y satisfies Π^X. Take any rule $A \leftarrow Body$ from Π, and assume that Y satisfies the body of the corresponding rule

$$A \leftarrow Body^X \tag{8}$$

from Π^X. By Fact 3, $X \models Body^X$. By Fact 4, it follows that $X \models Body$, so that $A \in \Gamma$. Since Y satisfies Γ, we conclude that Y satisfies the head A of (8).

Since Y is a subset of an answer set for Π^X and Y satisfies Π^X, $Y = X$. ∎

Proposition 1 *For any program Π whose rules have the form (1) and any set X of atoms, if X is an answer set for Π, then X is a model of $Comp(\Pi)$.*

Proof Let X be an answer set for Π. Take any rule $A \leftarrow Body$ in Π, and assume that X satisfies the antecedent $Body$ of the corresponding formula (3). Then A belongs to set Γ from the statement of Lemma 2. According to the lemma, it follows that X satisfies the consequent A of (3).

Now assume that X satisfies an atom a; we need to show that X satisfies the consequent of the corresponding formula (4). According to Lemma 2, X is a minimal set of atoms satisfying Γ. By Lemma 1, it follows that there exists a clause A in Γ such that $X \cap A = \{a\}$. By the definition of Γ, there is a rule $A \leftarrow Body$ in Π such that $X \models Body$. Then X satisfies the disjunctive term in the consequent of (4)

$$Body \wedge \bigwedge_{p \in A \setminus \{a\}} \neg p$$

that corresponds to this rule. ∎

7.2 Proof of Theorem 1

Theorem 1 *For any program Π whose rules have the form (2) and any set X of atoms, the following conditions are equivalent:*

(a) X is an answer set for Π,
(b) X is a model of $Comp(\Pi) \cup CLF(\Pi)$,
(c) X is a model of $Comp(\Pi) \cup DLF(\Pi)$.

Proof *From* (c) *to* (b): use the fact that $DLF(\Pi) \models CLF(\Pi)$.

From (a) *to* (c): Let X be an answer set for Π. Then X is a model of $Comp(\Pi)$ by Proposition 1. It remains to show that X satisfies every formula in $DLF(\Pi)$. Assume that X satisfies the antecedent $\bigvee L$ of a disjunctive loop formula (7); we want to show that X satisfies the consequent of this formula. Since X is an answer set for Π^X, its proper subset $X \setminus L$ does not satisfy Π^X. Consequently, there is a rule (2) in Π such that $X \setminus L$ satisfies the body of the corresponding rule

$$A \leftarrow B, F^X \tag{9}$$

of Π^X, but does not satisfy its head. From Fact 3, X satisfies the body of (9) also. Consequently, X satisfies the head of (9), that is,

$$X \cap A \neq \emptyset.$$

But $X \setminus L$ does not satisfy the head of (9), that is

$$(X \setminus L) \cap A = \emptyset. \tag{10}$$

From these two facts we can conclude that $A \cap L \neq \emptyset$. On the other hand, from the fact that $X \setminus L$ satisfies the body of (9), we can conclude that all atoms in B belong to $X \setminus L$, so that $pa(B) \cap L = \emptyset$. Consequently, the formula

$$B \wedge F \wedge \bigwedge_{p \in A \setminus L} \neg p$$

belongs to $R(L)$, and is one of the disjunctive terms in the consequent of (7). We will now check that X satisfies this formula. From the fact that X satisfies the body of (9) and Fact 4, we see that X satisfies both B and F. Formula (10) can be rewritten as

$$X \cap (A \setminus L) = \emptyset$$

Consequently, X doesn't contain atoms from $A \setminus L$, so that it satisfies $\bigwedge_{p \in A \setminus L} \neg p$.

From (b) to (a): Let X be a model of $Comp(\Pi) \cup CLF(\Pi)$. First, we will prove that $X \models \Pi^X$. Take any rule

$$A \leftarrow Body^X \tag{11}$$

in Π^X such that $X \models Body^X$. Then $X \models Body$ by Fact 4. Since X is a model of $Comp(\Pi)$, X satisfies every implication (3). Consequently, $X \models A$, so that X satisfies (11). It follows that X satisfies Π^X.

Next, we will prove that X is minimal among the sets of atoms that satisfy Π^X. Suppose that this is not the case, and consider a proper subset Y of X that satisfies Π^X. We will show that there exists an infinite sequence $p_1, p_2, \ldots$ of elements of $X \setminus Y$ with the following two properties for every $k > 0$:

(i) the positive dependency graph of Π contains a path of nonzero length from p_k to p_{k+1} such that all atoms in the path belong to $\{p_1, \ldots, p_{k+1}\}$;
(ii) if L is a loop of Π such that

$$p_k \in L \subseteq \{p_1, \ldots, p_k\} \tag{12}$$

then $p_{k+1} \notin L$.

The sequence $p_1, p_2, \ldots$ with these properties is defined recursively, as follows. Recall that Y is a proper subset of X; take p_1 to be any element of $X \setminus Y$. To select p_{k+1} given the sequence $p_1, \ldots, p_k$ such that $p_1, \ldots, p_k \in X \setminus Y$, consider two cases.

Case 1: Π has a loop satisfying condition (12). Take the union L_{max} of all such loops L. It is clear that L_{max} is a loop, and

$$p_k \in L_{max} \subseteq \{p_1, \ldots, p_k\}. \tag{13}$$

Since $p_1, \ldots, p_k \in X \setminus Y$, X satisfies $\bigwedge L_{max}$. On the other hand, by the choice of X, X satisfies all conjunctive loop formulas of Π, including $CLF(L_{max})$. Consequently, X satisfies one of the disjunctive terms

$$B \wedge F \wedge \bigwedge_{p \in A \setminus L_{max}} \neg p \tag{14}$$

of $R(L_{max})$. For the corresponding rule (2) of Π,

$$A \cap L_{max} \neq \emptyset \tag{15}$$

and

$$pa(B) \cap L_{max} = pa(B, F) \cap L_{max} = \emptyset. \tag{16}$$

Since X satisfies (14),

$$X \models B, F \tag{17}$$

and

$$Y \cap (A \setminus L_{max}) \subseteq X \cap (A \setminus L_{max}) = \emptyset. \tag{18}$$

From (13),

$$Y \cap L_{max} \subseteq Y \cap \{p_1, \ldots, p_k\} \subseteq Y \cap (X \setminus Y) = \emptyset,$$

so that $Y \cap L_{max} = \emptyset$. In combination with (18), this formula shows that

$$Y \cap A \subseteq Y \cap ((A \setminus L_{max}) \cup L_{max}) = (Y \cap (A \setminus L_{max})) \cup (Y \cap L_{max}) = \emptyset,$$

so that $Y \cap A = \emptyset$, which means that $Y \not\models A$. But by the choice of Y, Y satisfies every rule in Π^X, including

$$A \leftarrow B, F^X.$$

Consequently,

$$Y \not\models B, F^X. \tag{19}$$

Since all occurrences of atoms in F are in the scope of negation as failure, F^X doesn't contain atoms. It follows that F^X is either satisfied by every set of atoms or is not satisfied by any of them. From (17) and Fact 4 we see that F^X is satisfied by X. Consequently, F^X is satisfied by Y also. By (19), we can conclude that $Y \not\models B$, that is to say, one of the atoms in B doesn't belong to Y. Take this atom to be p_{k+1}. By (17), every atom in B belongs to X, so that p_{k+1} is indeed an element of $X \setminus Y$.

Case 2: Π does not have a loop satisfying condition (12). Consider the implication (4) in $Comp(\Pi)$ whose antecedent is p_k. We know that $p_k \in X \setminus Y \subseteq X$ and X satisfies $Comp(\Pi)$. It follows that X satisfies one of the disjunctive terms

$$B \wedge F \wedge \bigwedge_{p \in A \setminus \{p_k\}} \neg p \tag{20}$$

of this implication, so that (17) holds and

$$X \cap (A \setminus \{p_k\}) = \emptyset. \tag{21}$$

Since $Y \subseteq X$, $Y \cap (A \setminus \{p_k\}) = \emptyset$ also; since $p_k \in X \setminus Y$, $Y \cap \{p_k\} = \emptyset$. Consequently, $Y \cap A = \emptyset$, which means that $Y \not\models A$. By the same reasoning as in Case 1, we can conclude that one of the atoms in B belongs to $X \setminus Y$. Take this atom to be p_{k+1}.

Let us check that the sequence $p_1, p_2, \ldots$ satisfies (i). Assume that p_{k+1} was chosen according to Case 1. By (15), there is an atom p that belongs both to the head A of (2) and to L_{max}. Since $p_{k+1} \in pa(B, F)$, the positive dependency graph of Π contains an edge from p to p_{k+1}. Since both p_k and p belong to the loop L_{max}, this graph contains a path from p_k to p such that all atoms in the path belong to $\{p_1, \ldots, p_k\}$. Consequently, the graph contains a path of nonzero length from p_k to p_{k+1} such that all atoms in the path belong to $\{p_1, \ldots, p_{k+1}\}$.

Assume now that p_{k+1} was chosen according to Case 2. Since X satisfies (20) and the formula (3) corresponding to the rule $A \leftarrow B, F$, X satisfies A, that is, $X \cap A \neq \emptyset$. In view of (21), it follows that $p_k \in A$. But p_{k+1} was chosen as one of conjunctive terms of B. Consequently, there is an edge from p_k to p_{k+1} in the positive dependency graph of Π.

Next we will prove that the sequence satisfies (ii). If a loop L of Π satisfies condition (12), then p_{k+1} was chosen according to Case 1. Since $p_{k+1} \in pa(B)$, from (16) we conclude that $p_{k+1} \notin L_{max}$. Since $L \subseteq L_{max}$, it follows that $p_{k+1} \notin L$.

Notice that an atom may belong to X only if it occurs in the head of one of the rules of Π. Indeed, for any other atom a, the consequent of (4) is the empty disjunction $\bot$, so that X satisfies $a \supset \bot$. Since Π has finitely many rules, it follows that X is finite. Since $p_1, p_2, \ldots \in X \setminus Y$, this sequence must contain repetitions. Assume that $p_j = p_{k+1}$ for $j \leq k$. From property (i) we see that the positive dependency graph of Π contains a path from p_j to p_k whose vertices belong to $\{p_1, \ldots, p_{k+1}\}$. This property implies also that there is such a path of nonzero length from p_k to p_{k+1}. Since $p_j = p_{k+1}$, it follows that this graph contains a loop L such that

$$p_k, p_{k+1} \in L \subseteq \{p_1, \ldots, p_{k+1}\} = \{p_1, \ldots, p_k\},$$

which contradicts property (ii). ∎

7.3 Proof of Propositions 2, 3

In logic programming, any formula can be converted to "disjunctive and conjunctive normal forms" in the following sense. A *simple conjunction* is a formula of the form

$$p_1, \ldots, p_k, \mathit{not}\ p_{k+1}, \ldots, \mathit{not}\ p_m, \mathit{not}\ \mathit{not}\ p_{m+1}, \ldots, \mathit{not}\ \mathit{not}\ p_n$$

and a *simple disjunction* is a formula of the form

$$p_1; \dots; p_k; \mathit{not}\ p_{k+1}; \dots; \mathit{not}\ p_m; \mathit{not}\ \mathit{not}\ p_{m+1}; \dots; \mathit{not}\ \mathit{not}\ p_n$$

where $0 \leq k \leq m \leq n$ and all p_i are atoms.

The following fact is essentially Proposition 5 from [Lifschitz *et al.*, 1999].

Fact 5 *Any formula can be equivalently rewritten as*

(i) a formula of the form $F_1; \dots; F_n$ $(n \geq 0)$ where each F_i is a simple conjunction, and
(ii) a formula of the form $F_1, \dots, F_n$ $(n \geq 0)$ where each F_i is a simple disjunction.

Proposition 2 *For any absolutely tight program Π whose rules have the form (1) and any set X of atoms, X is an answer set for Π iff X satisfies $Comp(\Pi)$.*

Proof If the rules of Π have the form (2), then the set of loop formulas of Π is empty, and the assertion to be proved follows from Theorem 1. On the other hand, any program whose rules have the form (1) can be rewritten as a strongly equivalent program whose rules have the form (2) by the following procedure: convert each body to a disjunctive normal form (Fact 5(i)) and transform the rule into a set of rules whose bodies are simple conjunctions ([Lifschitz *et al.*, 1999, Proposition 6(ii)]). This can be done without changing the positive dependency graph, so that the resulting program will be absolutely tight. ∎

Proposition 3 *Any absolutely tight program is strongly equivalent to an absolutely tight program whose rules have the form (1).*

Proof Any program can be rewritten as a strongly equivalent program whose rules have the form (1) by the following procedure: convert each head to a conjunctive normal form (Fact 5(ii)) and transform the rule into a set of rules whose heads are simple disjunctions ([Lifschitz *et al.*, 1999, Proposition 6(i)]). The resulting program can be rewritten as a set of rules of the form (1) ([Lifschitz *et al.*, 1999, Proposition 6(iii,iv)]). This can be done without changing the positive dependency graph, so that the resulting program will be absolutely tight. ∎

We do not include a proof of Proposition 4 because of the limit on the length of the paper. The proof of Proposition 5 is similar to the proof of Proposition 3.

Acknowledgements

We are grateful to Fangzhen Lin and Yuting Zhao for providing us with access to the journal version of [Lin and Zhao, 2002], to Yuliya Babovich, Paolo Ferraris, Michael Gelfond and Hudson Turner for useful discussions related to the subject of this paper. This work was partially supported by the Texas Higher Education Coordinating Board under Grant 003658-0322-2001.

References

[Babovich *et al.*, 2000] Yuliya Babovich, Esra Erdem, and Vladimir Lifschitz. Fages' theorem and answer set programming.[10] In *Proc. Eighth Int'l Workshop on Non-Monotonic Reasoning*, 2000.

[Baral and Gelfond, 1994] Chitta Baral and Michael Gelfond. Logic programming and knowledge representation. *Journal of Logic Programming*, 19,20:73–148, 1994.

[Ben-Eliyahu and Dechter, 1996] Rachel Ben-Eliyahu and Rina Dechter. Propositional semantics for disjunctive logic programs. *Annals of Mathematics and Artificial Intelligence*, 12:53–87, 1996.

[Clark, 1978] Keith Clark. Negation as failure. In Herve Gallaire and Jack Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, New York, 1978.

[Eiter and Gottlob, 1993] Thomas Eiter and Georg Gottlob. Complexity results for disjunctive logic programming and application to nonmonotonic logics. In Dale Miller, editor, *Proc. ILPS-93*, pages 266–278, 1993.

[Erdem and Lifschitz, 1999] Esra Erdem and Vladimir Lifschitz. Transformations of logic programs related to causality and planning. In *Logic Programming and Non-monotonic Reasoning: Proc. Fifth Int'l Conf. (Lecture Notes in Artificial Intelligence 1730)*, pages 107–116, 1999.

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

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

[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.

[Inoue and Sakama, 1998] Katsumi Inoue and Chiaki Sakama. Negation as failure in the head. *Journal of Logic Programming*, 35:39–78, 1998.

[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.

[Lifschitz *et al.*, 2001] Vladimir Lifschitz, David Pearce, and Agustin Valverde. Strongly equivalent logic programs. *ACM Transactions on Computational Logic*, 2:526–541, 2001.

[Lin and Zhao, 2002] Fangzhen Lin and Yuting Zhao. ASSAT: Computing answer sets of a logic program by SAT solvers. In *Proc. AAAI-02*, 2002.

[Lloyd and Topor, 1984] John Lloyd and Rodney Topor. Making Prolog more expressive. *Journal of Logic Programming*, 3:225–240, 1984.

[Marek and Subrahmanian, 1989] Victor Marek and V.S. Subrahmanian. The relationship between logic program semantics and non-monotonic reasoning. In Giorgio Levi and Maurizio Martelli, editors, *Logic Programming: Proc. Sixth Int'l Conf.*, pages 600–617, 1989.

[10] `http://arxiv.org/abs/cs.ai/0003042` .