Lloyd-Topor Completion and General Stable Models

Vladimir Lifschitz and Fangkai Yang

Department of Computer Science
The University of Texas at Austin
{vl,fkyang}@cs.utexas.edu

Abstract. We investigate the relationship between the generalization of program completion defined in 1984 by Lloyd and Topor and the generalization of the stable model semantics introduced recently by Ferraris *et al.* The main theorem can be used to characterize, in some cases, the general stable models of a logic program by a first-order formula. The proof uses Truszczynski's stable model semantics of infinitary propositional formulas.

1 Introduction

The theorem by François Fages [1] describing a case when the stable model semantics is equivalent to program completion is one of the most important results in the theory of stable models. It was generalized in [2–4], it has led to the invention of loop formulas [5], and it has had a significant impact on the design of answer set solvers.

The general stable model semantics defined in [6] characterizes the stable models of a first-order sentence F as arbitrary models of a certain second-order sentence, denoted by SM$[F]$;[1] logic programs are viewed there as first-order sentences written in "logic programming notation." In this note we define an extension of Fages' theorem that can be used as a tool for transforming SM$[F]$, in some cases, into an equivalent first-order formula. That extension refers to the version of program completion introduced by John Lloyd and Rodney Topor in [7]. Their definition allows the body of a rule to contain propositional connectives and quantifiers.

Earlier work in this direction is reported in [6] and [8]. These papers do not mention completion in the sense of Lloyd and Topor explicitly. Instead, they discuss ways to convert a logic program to "Clark normal form" by strongly equivalent transformations [9, 10] and completing programs in this normal form by replacing implications with equivalences. But this is essentially what Lloyd-Topor completion does.

The following examples illustrate some of the issues involved. Let F be the program

$$\begin{array}{l} p(a), \\ q(b), \\ p(x) \leftarrow q(x), \end{array} \tag{1}$$

[1] To be precise, the definition of SM in that paper requires that a set of "intensional predicates" be specified. In the examples below, we assume that all predicate symbols occurring in F are intensional.

or, in other words, the sentence

$$p(a) \wedge q(b) \wedge \forall x(q(x) \to p(x)).$$

The Clark normal form of (1) is tight in the sense of [6], and Theorem 11 from that paper shows that SM$[F]$ in this case is equivalent to the conjunction of the completed definitions of p and q:

$$\begin{array}{l} \forall x(p(x) \leftrightarrow x = a \vee q(x)), \\ \forall x(q(x) \leftrightarrow x = b). \end{array} \tag{2}$$

Let now F be the program

$$\begin{array}{l} p(x) \leftarrow q(x), \\ q(a) \leftarrow p(b). \end{array} \tag{3}$$

This program is not tight in the sense of [6], so that the above-mentioned theorem is not applicable. In fact, SM$[F]$ is stronger in this case than the conjunction of the completed definitions

$$\begin{array}{l} \forall x(p(x) \leftrightarrow q(x)), \\ \forall x(q(x) \leftrightarrow x = a \wedge p(b)). \end{array} \tag{4}$$

A counterexample is provided by any interpretation that treats each of the symbols p, q as a singleton such that its element is equal to both a and b. Such a (non-Herbrand) interpretation satisfies (4), but it is not a stable model of (3). (In stable models of (3) both p and q are empty.)

Program (3) is, however, atomic-tight in the sense of [8, Section 5.1.1]. Corollary 5 from that paper allows us to conclude that the equivalence between SM$[F]$ and (4) is entailed by the unique name assumption $a \neq b$. It follows that the result of applying SM to the program obtained from (4) by adding the constraint

$$\leftarrow a = b$$

is equivalent to the conjunction of the completion sentences (4) with $a \neq b$. This example illustrates the role of a property more general than the logical equivalence between SM$[F]$ and the completion of F: it may be useful to know when the equivalence between these two formulas is entailed by a certain set of assumptions. This information may be relevant if we are interested in a logic program obtained from F by adding constraints.

The result of applying SM to the program

$$\begin{array}{r l} p(a) & \leftarrow p(b), \\ q(c) & \leftarrow q(d), \\ & \leftarrow a = b, \\ & \leftarrow c = d \end{array} \tag{5}$$

is equivalent to the conjunction of the formulas

$$\begin{array}{c} \forall x(p(x) \leftrightarrow x = a \wedge p(b)), \\ \forall x(q(x) \leftrightarrow x = c \wedge q(d)), \\ a \neq b, \\ c \neq d. \end{array} \tag{6}$$

This claim cannot be justified, however, by a reference to Corollary 5 from [8]. The program in this example is atomic-tight, but it does not contain constraints corresponding to some of the unique name axioms, for instance $a \neq c$. We will show how our claim follows from the main theorem stated below.

We will discuss also an example illustrating limitations of earlier work that is related to describing dynamic domains in answer set programming (ASP). The program in that example is not atomic-tight because of rules expressing the commonsense law of inertia. We will show nevertheless that the process of completion can be used to characterize its stable models by a first-order formula.

The class of tight programs is defined in [6] in terms of predicate dependency graphs; that definition is reproduced in Section 3 below. The definition of an atomic-tight program in [8] refers to more informative "first-order dependency graphs." Our approach is based on an alternative solution to the problem of making predicate dependency graphs more informative, "rule dependency graphs."

After reviewing some background material in Sections 2 and 3, we define rule dependency graphs in Section 4, and state the main theorem and give examples of its use in Sections 5 and 6. Section 7 reviews Truszczynski's theory of stable models of infinitary propositional formulas, which is used in the proof of the main theorem, and then the proof is presented in Section 8 and in the appendix.

2 Review: Operator SM, Lloyd-Topor Programs, and Completion

In this paper, a *formula* is a first-order formula that may contain the propositional connectives $\bot$ (logical falsity), $\wedge$, $\vee$, and $\rightarrow$, and the quantifiers $\forall$, $\exists$. We treat $\neg F$ as an abbreviation for $F \rightarrow \bot$; $\top$ stands for $\bot \rightarrow \bot$; $F \leftrightarrow G$ stands for $(F \rightarrow G) \wedge (G \rightarrow F)$.

For any first-order sentence F and any tuple $\mathbf{p}$ of distinct predicate constants (other than equality) $\mathrm{SM}_{\mathbf{p}}[F]$ is the conjunction of F with a second-order "stability condition"; see [6, Section 2] for details. The members of $\mathbf{p}$ are called *intensional*, and the other predicate constants are *extensional*. We will drop the subscript in the symbol $\mathrm{SM}_{\mathbf{p}}$ when $\mathbf{p}$ is the list of all predicate symbols occurring in F. For any sentence F, a $\mathbf{p}$-*stable* (or simply *stable*) *model* of F is an interpretation of the underlying signature that satisfies $\mathrm{SM}_{\mathbf{p}}[F]$.

A *Lloyd-Topor program* is a finite set of rules of the form

$$p(\mathbf{t}) \leftarrow G, \tag{7}$$

where **t** is a tuple of terms, and G is a formula. We will identify a program with the sentence obtained by conjoining the formulas

$$\widetilde{\forall}(G \rightarrow p(\mathbf{t}))$$

for all its rules (7). ($\widetilde{\forall} F$ stands for the universal closure of F.)

Let Π be a Lloyd-Topor program, and p a predicate constant (other than equality). Let

$$p(\mathbf{t}^i) \leftarrow G_i \qquad (i = 1, 2, \dots) \tag{8}$$

be all rules of Π that contain p in the head. The *definition of* p *in* Π is the rule

$$p(\mathbf{x}) \leftarrow \bigvee_i \exists \mathbf{y}^i (\mathbf{x} = \mathbf{t}^i \wedge G_i), \tag{9}$$

where **x** is a list of distinct variables not appearing in any of the rules (8), and $\mathbf{y}^i$ is the list of free variables of (8).[2] The *completed definition of* p *in* Π is the formula

$$\forall \mathbf{x} \left(p(\mathbf{x}) \leftrightarrow \bigvee_i \exists \mathbf{y}^i (\mathbf{x} = \mathbf{t}^i \wedge G_i) \right). \tag{10}$$

For instance, the completed definitions of p and q in program (1) are the formulas

$$\forall x_1 (p(x_1) \leftrightarrow x_1 = a \vee \exists x (x_1 = x \wedge q(x))),$$
$$\forall x_1 (q(x_1) \leftrightarrow x_1 = b),$$

which can be equivalently rewritten as (2).

By Comp$[\Pi]$ we denote the conjunction of the completed definitions of all predicate constants p in Π. This sentence is similar to the completion of Π in the sense of [7, Section 2], except that it does not include Clark equality axioms.

3 Review: Tight Programs

We will review now the definition of tightness from [6, Section 7.3]. In application to a Lloyd-Topor program Π, when all predicate constants occurring in Π are treated as intensional, that definition can be stated as follows.

An occurrence of an expression in a first-order formula is *negated* if it belongs to a subformula of the form $\neg F$ (that is, $F \rightarrow \bot$), and *nonnegated* otherwise. The *predicate dependency graph of* Π is the directed graph that has

– all predicate constants occurring in Π as its vertices, and
– an edge from p to q whenever Π contains a rule (7) with p in the head such that its body G has a positive[3] nonnegated occurrence of q.

We say that Π is *tight* if the predicate dependency graph of Π is acyclic.

For example, the predicate dependency graph of program (1) has a single edge, from p to q. The predicate dependency graph of program (3) has two edges, from p to q and from q to p. The predicate dependency graph of the program

$$\begin{array}{l} p(a,b) \\ q(x,y) \leftarrow p(y,x) \wedge \neg p(x,y) \end{array} \tag{11}$$

[2] By $\mathbf{x} = \mathbf{t}^i$ we denote the conjunction of the equalities between members of the tuple **x** and the corresponding members of the tuple $\mathbf{t}^i$.

[3] Recall that an occurrence of an expression in a first-order formula is called *positive* if the number of implications containing that occurrence in the antecedent is even, and *negative* otherwise.

has a single edge, from q to p (because one of the occurrences of p in the body of the second rule is nonnegated). The predicate dependency graph of the program

$$\begin{aligned} p(x) &\leftarrow q(x), \\ q(x) &\leftarrow r(x), \\ r(x) &\leftarrow s(x) \end{aligned} \tag{12}$$

has 3 edges:

$$p \longrightarrow q \longrightarrow r \longrightarrow s.$$

Programs (1), (11) and (12) are tight; program (3) is not.

Proposition 1. *If a Lloyd-Topor program Π is tight then* SM$[\Pi]$ *is equivalent to* Comp$[\Pi]$.

This is an easy corollary to the theorem from [6] mentioned in the introduction. Indeed, consider the set Π' of the definitions (9) of all predicate constants p in Π. It can be viewed as a formula in Clark normal form in the sense of [6, Section 6.1]. It is tight, because it has the same predicate dependency graph as Π. By Theorem 11 from [6], SM$[\Pi']$ is equivalent to the completion of Π' in the sense of [6, Section 6.1], which is identical to Comp$[\Pi]$. It remains to observe that Π is intuitionistically equivalent to Π', so that SM$[\Pi]$ is equivalent to SM$[\Pi']$ [6, Section 5.1].

4 Rule Dependency Graph

We are interested in conditions on a Lloyd-Topor program Π ensuring that the equivalence

$$\text{SM}[\Pi] \leftrightarrow \text{Comp}[\Pi]$$

is entailed by a given set of assumptions Γ. Proposition 1 gives a solution for the special case when Γ is empty. The following definition will help us answer the more general question.

The *rule dependency graph* of a Lloyd-Topor program Π is the directed graph that has

– rules of Π, with variables (both free and bound) renamed arbitrarily, as its vertices, and
– an edge from a rule $p(\mathbf{t}) \leftarrow G$ to a rule $p'(\mathbf{t}') \leftarrow G'$, labeled by an atomic formula $p'(\mathbf{s})$, if $p'(\mathbf{s})$ has a positive nonnegated occurrence in G.

Unlike the predicate dependency graph, the rule dependency graph of a program is usually infinite. For example, the rule dependency graph of program (11) has the vertices $p(a, b)$ and

$$q(x_1, y_1) \leftarrow p(y_1, x_1) \wedge \neg p(x_1, y_1) \tag{13}$$

for arbitrary pairs of distinct variables x_1, y_1. It has an edge from each vertex (13) to $p(a, b)$, labeled $p(y_1, x_1)$. The rule dependency graph of program (12) has edges of two kinds:

– from $p(x_1) \leftarrow q(x_1)$ to $q(x_2) \leftarrow r(x_2)$, labeled $q(x_1)$, and
– from $q(x_1) \leftarrow r(x_1)$ to $r(x_2) \leftarrow s(x_2)$, labeled $r(x_1)$

for arbitrary variables x_1, x_2.

The rule dependency graph of a program is "dual" to its predicate dependency graph, in the following sense. The vertices of the predicate dependency graph are predicate symbols, and the presence of an edge from p to q is determined by the existence of a rule that contains certain occurrences of p and q. The vertices of the rule dependency graph are rules, and the presence of an edge from R_1 to R_2 is determined by the existence of a predicate symbol with certain occurrences in R_1 and R_2.

There is a simple characterization of tightness in terms of rule dependency graphs:

Proposition 2. *A Lloyd-Topor program Π is tight iff there exists n such that the rule dependency graph of Π has no paths of length n.*

Proof. Assume that Π is tight, and let n be the number of predicate symbols occurring in Π. Then the rule dependency graph of Π has no paths of length $n+1$. Indeed, assume that such a path exists:

$$R_0 \xrightarrow{p_1(\dots)} R_1 \xrightarrow{p_2(\dots)} R_2 \xrightarrow{p_3(\dots)} \cdots \xrightarrow{p_{n+1}(\dots)} R_{n+1}.$$

Each of the rules R_i ($1 \leq i \leq n$) contains p_i in the head and a positive nonnegated occurrence of p_{i+1} in the body. Consequently the predicate dependency graph of Π has an edge from p_i to p_{i+1}, so that $p_1, \dots, p_{n+1}$ is a path in that graph; contradiction. Now assume that Π is not tight. Then there is an infinite path $p_1, p_2, \dots$ in the predicate dependency graph of Π. Let R_i be a rule of Π that has p_i in the head and a positive nonnegated occurrence of p_{i+1} in the body. Then the rule dependency graph of Π has an infinite path of the form

$$R_1 \xrightarrow{p_2(\dots)} R_2 \xrightarrow{p_3(\dots)} \cdots .$$

The main theorem, stated in the next section, refers to finite paths in the rule dependency graph of a program Π that satisfy an additional condition: the rules at their vertices have no common variables (neither free nor bound). Such paths will be called *chains*.

Corollary 1. *A Lloyd-Topor program Π is tight iff there exists n such that Π has no chains of length n.*

Indeed, any finite path in the rule dependency graph of Π can be converted into a chain of the same length by renaming variables.

5 Main Theorem

Let C be a chain

$$\begin{array}{c} p_0(\mathbf{t}^0) \leftarrow \mathit{Body}_0 \\ \big\downarrow p_1(\mathbf{s}^1) \\ p_1(\mathbf{t}^1) \leftarrow \mathit{Body}_1 \\ \big\downarrow p_2(\mathbf{s}^2) \\ \cdots\cdots\cdots \\ \big\downarrow p_n(\mathbf{s}^n) \\ p_n(\mathbf{t}^n) \leftarrow \mathit{Body}_n \end{array} \tag{14}$$

in a Lloyd-Topor program Π. The corresponding *chain formula* F_C is the conjunction

$$\bigwedge_{i=1}^{n} \mathbf{s}^i = \mathbf{t}^i \ \wedge \bigwedge_{i=0}^{n} \mathit{Body}_i.$$

For instance, if C is the chain

$$\begin{array}{c} q(x_1, y_1) \leftarrow p(y_1, x_1) \wedge \neg p(x_1, y_1) \\ \big\downarrow p(y_1, x_1) \\ p(a, b) \end{array}$$

in program (11) then F_C is

$$y_1 = a \wedge x_1 = b \wedge p(y_1, x_1) \wedge \neg p(x_1, y_1).$$

Let Γ be a set of sentences. About a Lloyd-Topor program Π we will say that it is *tight relative to* Γ, or Γ*-tight*, if there exists a positive integer n such that, for every chain C in Π of length n,

$$\Gamma, \mathrm{Comp}[\Pi] \models \widetilde{\forall} \neg F_C.$$

Main Theorem. *If a Lloyd-Topor program Π is Γ-tight then*

$$\Gamma \models \mathrm{SM}[\Pi] \leftrightarrow \mathrm{Comp}[\Pi].$$

Corollary 1 shows that every tight program is trivially Γ-tight even when Γ is empty. Consequently the main theorem can be viewed as a generalization of Proposition 1.

Tightness in the sense of Section 3 is a syntactic condition that is easy to verify; Γ-tightness is not. Nevertheless, the main theorem is useful because it may allow us to reduce the problem of characterizing the stable models of a program by a first-order formula to verifying an entailment in first-order logic.

Here are some examples. In each case, to verify Γ-tightness we take $n = 1$. We will check the entailment in the definition of Γ-tightness by deriving a contradiction from (some subset of) the assumptions Γ, $\mathrm{Comp}[\Pi]$, and F_C.

Example 1. The one-rule program

$$p(a) \leftarrow p(x) \wedge x \neq a$$

is tight relative to $\emptyset$. Indeed, any chain of length 1 has the form

$$\begin{array}{c} p(a) \leftarrow p(x_1) \wedge x_1 \neq a \\ \big\downarrow p(x_1) \\ p(a) \leftarrow p(x_2) \wedge x_2 \neq a. \end{array}$$

The corresponding chain formula

$$x_1 = a \wedge p(x_1) \wedge x_1 \neq a \wedge p(x_2) \wedge x_2 \neq a.$$

is contradictory.

Thus the stable models of this program are described by its completion, even though the program is not tight (and not even atomic-tight).

Example 2. Let Π be the program consisting of the first 2 rules of (5):

$$\begin{array}{c} p(a) \leftarrow p(b), \\ q(c) \leftarrow q(d). \end{array}$$

To justify the claim about (5) made in the introduction, we will check that Π is tight relative to $\{a \neq b, c \neq d\}$. There are two chains of length 1:

$$\begin{array}{c} p(a) \leftarrow p(b) \\ \big\downarrow p(b) \\ p(a) \leftarrow p(b) \end{array}$$

and

$$\begin{array}{c} q(c) \leftarrow q(d) \\ \big\downarrow q(d) \\ q(c) \leftarrow q(d). \end{array}$$

The corresponding chain formulas are

$$b = a \wedge p(b) \wedge p(b)$$

and

$$d = c \wedge q(d) \wedge q(d).$$

Each of them contradicts Γ.

Example 3. Let us check that program (3) is tight relative to $\{a \neq b\}$. Its chains of length 1 are

$$\begin{array}{c} p(x_1) \leftarrow q(x_1) \\ \big\downarrow q(x_1) \\ q(a) \leftarrow p(b) \end{array}$$

and

$$\begin{array}{c} q(a) \leftarrow p(b) \\ \big\downarrow q(b) \\ p(x_1) \leftarrow q(x_1) \end{array}$$

for an arbitrary variable x_1. The corresponding chain formulas include the conjunctive term $p(b)$. Using the completion (4) of the program, we derive $b = a$, which contradicts Γ.

6 A Larger Example

Programs found in actual applications of ASP usually involve constructs that are not allowed in Lloyd-Topor programs, such as choice rules and constraints. Choice rules have the form

$$\{p(\mathbf{t})\} \leftarrow G.$$

We view this expression as shorthand for the sentence

$$\widetilde{\forall}(G \rightarrow p(\mathbf{t}) \vee \neg p(\mathbf{t})).$$

A constraint $\leftarrow G$ is shorthand for the sentence $\widetilde{\forall}\neg G$. Such sentences do not correspond to any rules in the sense of Section 2.

Nevertheless, the main theorem stated above can sometimes help us characterize the stable models of a "realistic" program by a first-order formula. In this section we discuss an example of this kind.

The logic program M described below encodes commonsense knowledge about moving objects from one location to another. Its signature consists of

– the object constants $\widehat{0}, \dots, \widehat{k}$, where k is a fixed nonnegative integer;
– the unary predicate constants *object*, *place*, and *step*; they correspond to the three types of individuals under consideration;
– the binary predicate constant *next*; it describes the temporal order of steps;
– the ternary predicate constants *at* and *move*; they represent the fluents and actions that we are interested in.

The predicate constants *step*, *next*, and *at* are intensional; the other three are not. (The fact that some predicates are extensional is the first sign that M is not a Lloyd-Topor program.) The program consists of the following rules:

(i) the facts

$$\begin{array}{l} \mathit{step}(\widehat{0}),\ \mathit{step}(\widehat{1}),\ \dots\ \mathit{step}(\widehat{k}); \\ \mathit{next}(\widehat{0},\widehat{1}),\ \mathit{next}(\widehat{1},\widehat{2}),\ \dots,\ \mathit{next}(\widehat{k-1},\widehat{k}); \end{array}$$

(ii) the unique name constraints

$$\leftarrow \widehat{i} = \widehat{j} \qquad (1 \le i < j \le k);$$

(iii) the constraints describing the arguments of *at* and *move*:

$$\leftarrow at(x,y,z) \wedge \neg(object(x) \wedge place(y) \wedge step(z))$$

and

$$\leftarrow move(x,y,z) \wedge \neg(object(x) \wedge place(y) \wedge step(z));$$

(iv) the uniqueness of location constraint

$$\leftarrow at(x,y_1,z) \wedge at(x,y_2,z) \wedge y_1 \neq y_2;$$

(v) the existence of location constraint

$$\leftarrow object(x) \wedge step(z) \wedge \neg\exists y\, at(x,y,z);$$

(vi) the rule expressing the effect of moving an object:

$$at(x,y,u) \leftarrow move(x,y,z) \wedge next(z,u);$$

(vii) the choice rule expressing that initially an object can be anywhere:

$$\{at(x,y,0)\} \leftarrow object(x) \wedge place(y);$$

(viii) the choice rule expressing the commonsense law of inertia:[4]

$$\{at(x,y,u)\} \leftarrow at(x,y,z) \wedge next(z,u).$$

Program M is not atomic-tight, so that methods of [8] are not directly applicable to it. Nevertheless, we can describe the stable models of this program without the use of second-order quantifiers. In the statement of the proposition below, **p** stands for the list of intensional predicates *step*, *next* and *at*, and H is the conjunction of the universal closures of the formulas

$$\begin{array}{l}
\widehat{i} \neq \widehat{j} \qquad (1 \leq i < j \leq k),\\
at(x,y,z) \rightarrow object(x) \wedge place(y) \wedge step(z),\\
move(x,y,z) \rightarrow object(x) \wedge place(y) \wedge step(z),\\
at(x,y_1,z) \wedge at(x,y_2,z) \rightarrow y_1 = y_2,\\
object(x) \wedge step(z) \rightarrow \exists y\, at(x,y,z).
\end{array}$$

Proposition 3. $\mathrm{SM}_{\mathbf{p}}[M]$ *is equivalent to the conjunction of* H *with the universal closures of the formulas*

$$step(z) \leftrightarrow \bigvee_{i=0}^{k} z = \widehat{i}, \tag{15}$$

$$next(z,u) \leftrightarrow \bigvee_{i=0}^{k-1} (z = \widehat{i} \wedge u = \widehat{i+1}), \tag{16}$$

$$at(x,y,\widehat{i+1}) \leftrightarrow (move(x,y,\widehat{i}) \vee (at(x,y,\widehat{i}) \wedge \neg\exists w\, move(x,w,\widehat{i}))) \qquad (i = 0,\dots,k-1). \tag{17}$$

[4] This representation of inertia follows the example of [11, Figure 1].

Recall that the effect of adding a constraint to a logic program is to eliminate its stable models that violate that constraint [6, Theorem 3]. An interpretation satisfies H iff it does not violate any of the constraints (ii)–(v). So the statement of Proposition 3 can be summarized as follows: the contribution of rules (i) and (vi)–(viii), under the stable model semantics, amounts to providing explicit definitions for *step* and *next*, and "successor state formulas" for *at*.

The proof of Proposition 3 refers to the Lloyd-Topor program Π consisting of rules (i), (vi),

(vii′) $\mathit{at}(x, y, 0) \leftarrow \mathit{object}(x) \wedge \mathit{place}(y) \wedge \neg\neg \mathit{at}(x, y, 0)$,
(viii′) $\mathit{at}(x, y, u) \leftarrow \mathit{at}(x, y, z) \wedge \mathit{next}(z, u) \wedge \neg\neg \mathit{at}(x, y, t_2)$,

and

$$\begin{aligned} \mathit{object}(x) &\leftarrow \neg\neg \mathit{object}(x), \\ \mathit{place}(y) &\leftarrow \neg\neg \mathit{place}(y), \\ \mathit{move}(x, y, z) &\leftarrow \neg\neg \mathit{move}(x, y, z). \end{aligned} \tag{18}$$

It is easy to see that $\mathrm{SM}_{\mathbf{p}}[M]$ is equivalent to $\mathrm{SM}[\Pi] \wedge H$. Indeed, consider the program M' obtained from M by adding rules (18). These rules are strongly equivalent to the choice rules

$$\{\mathit{object}(x)\},\ \{\mathit{place}(y)\},\ \{\mathit{move}(x, y, z)\}.$$

Consequently $\mathrm{SM}_{\mathbf{p}}[M]$ is equivalent to $\mathrm{SM}[M']$ [6, Theorem 2]. It remains to notice that (vii) is strongly equivalent to (vii′), and (viii) is strongly equivalent to (viii′).

Furthermore—and this is the key step in the proof of Proposition 3—the second-order formula $\mathrm{SM}[\Pi] \wedge H$ is equivalent to the first-order formula $\mathrm{Comp}[\Pi] \wedge H$, in view of our main theorem and the following fact:

Lemma 1. *Program Π is H-tight.*

To derive Proposition 3 from the lemma, we only need to observe that (15) and (16) are the completed definitions of *step* and *next* in Π, and that the completed definition of *at* can be transformed into (17) under assumptions (15), (16), and H.

Proof of Lemma 1. Consider a chain in Π of length $k + 2$:

$$R_0 \xrightarrow{p_1(\dots)} R_1 \xrightarrow{p_2(\dots)} \cdots \xrightarrow{p_{k+1}(\dots)} R_{k+1} \xrightarrow{p_{k+2}(\dots)} R_{k+2}. \tag{19}$$

Each R_i is obtained from one of the rules (i), (vi), (vii′), (viii′), (18) by renaming variables. Each p_i occurs in the head of R_i and has a positive nonnegated occurrence in R_{i-1}. Since there are no nonnegated predicate symbols in the bodies of rules (i) and (18), we conclude that $R_0, \dots, R_{k+1}$ are obtained from other rules of Π, that is, from (vi), (vii′), and (viii′). Since the predicate constant in the head of each of these three rules is *at*, each of $p_1, \dots, p_{k+1}$ is the symbol *at*. Since there are no nonnegated occurrences of *at* in the bodies of (vi) and (vii′), we conclude that $R_0, \dots, R_k$ are obtained by renaming variables in (viii′). This means that chain (18) has the form

$$\begin{array}{c}
at(x_0, y_0, u_0) \leftarrow at(x_0, y_0, z_0) \wedge next(z_0, u_0) \wedge \neg\neg at(x_0, y_0, u_0) \\
\big\downarrow at(x_0, y_0, z_0) \\
at(x_1, y_1, u_1) \leftarrow at(x_1, y_1, z_1) \wedge next(z_1, u_1) \wedge \neg\neg at(x_1, y_1, u_1) \\
\big\downarrow at(x_1, y_1, z_1) \\
\dots \\
\big\downarrow at(x_{k-1}, y_{k-1}, z_{k-1}) \\
at(x_k, y_k, u_k) \leftarrow at(x_k, y_k, z_k) \wedge next(z_k, u_k) \wedge \neg\neg at(x_k, y_k, u_k) \\
\big\downarrow at(x_k, y_k, z_k) \\
R_{k+1} \\
\big\downarrow \cdots \\
R_{k+2}.
\end{array}$$

The corresponding chain formula contains the conjunctive terms

$$z_0 = u_1, z_1 = u_2, \dots, z_{k-1} = u_k$$

and

$$next(z_0, u_0), next(z_1, u_1), \dots, next(z_k, u_k).$$

From these formulas we derive

$$next(u_1, u_0), next(u_2, u_1), \dots, next(u_{k+1}, u_k), \tag{20}$$

where u_{k+1} stands for z_k. Using the completed definition of *next*, we conclude:

$$u_i = \widehat{0} \vee \dots \vee u_i = \widehat{k} \qquad (0 \leq i \leq k+1).$$

Consider the case when

$$u_i = \widehat{j_i} \qquad (0 \leq i \leq k+1)$$

for some numbers $j_0, \dots, j_{k+1} \in \{0, \dots, k\}$. There exists at least one subscript i such that $j_i \neq j_{i+1} + 1$, because otherwise we would have

$$j_0 = j_1 + 1 = j_2 + 2 = \dots = j_{k+1} + k + 1,$$

which is impossible because $j_0, j_{k+1} \in \{0, \dots, k\}$. By the choice of i, from the completed definition of *next* and the unique name assumption (included in H) we can derive $\neg next(\widehat{j_{i+1}}, \widehat{j_i})$. Consequently $\neg next(u_{i+1}, u_i)$, which contradicts (20).

7 Review: Stable Models of Infinitary Formulas

Our proof of the main theorem employs the method proposed (for a different purpose) by Miroslaw Truszczynski [12], and in this section we review some of the definitions and results of that paper. The stable model semantics of propositional formulas due to Paolo Ferraris [13] is extended there to formulas with infinitely long conjunctions and disjunctions, and that generalization is related to the operator SM.

Let $\mathcal{A}$ be a set of propositional atoms. The sets $\mathcal{F}_0, \mathcal{F}_1, \dots$ are defined as follows:

– $\mathcal{F}_0 = \mathcal{A} \cup \{\bot\}$;
– $\mathcal{F}_{i+1}$ consists of expressions $\mathcal{H}^\vee$ and $\mathcal{H}^\wedge$, for all subsets $\mathcal{H}$ of $\mathcal{F}_0 \cup \ldots \cup \mathcal{F}_i$, and of expressions $F \to G$, where $F, G \in \mathcal{F}_0 \cup \ldots \cup \mathcal{F}_i$.

An *infinitary formula* (over $\mathcal{A}$) is an element of $\bigcup_{i=0}^{\infty} \mathcal{F}_i$.

A *(propositional) interpretation* is a subset of $\mathcal{A}$. The satisfaction relation between an interpretation and an infinitary formula is defined in a natural way. The definition of the reduct F^I of a formula F relative to an interpretation I proposed in [13] is extended to infinitary formulas as follows:

– $\bot^I = \bot$.
– For $A \in \mathcal{A}$, $A^I = \bot$ if $I \not\models A$; otherwise $A^I = A$.
– $(\mathcal{H}^\wedge)^I = \bot$ if $I \not\models \mathcal{H}^\wedge$; otherwise $(\mathcal{H}^\wedge)^I = \{G^I : G \in \mathcal{H}\}^\wedge$.
– $(\mathcal{H}^\vee)^I = \bot$ if $I \not\models \mathcal{H}^\vee$; otherwise $(\mathcal{H}^\vee)^I = \{G^I : G \in \mathcal{H}\}^\vee$.
– $(G \to H)^I = \bot$ if $I \not\models G \to H$; otherwise $(G \to H)^I = G^I \to H^I$.

(Note that according to this definition F^I is $\bot$ whenever $I \not\models F$.) An interpretation I is a *stable model* of an infinitary formula F if I is a minimal model of F^I. An interpretation I satisfies F^I iff it satisfies F [12, Proposition 1], so that stable models of F are models of F.

Infinitary formulas are used to encode first-order sentences as follows. For any interpretation I in the sense of first-order logic, let $\mathcal{A}$ be the set of ground atoms formed from the predicate constants of the underlying signature and the "names" ξ^* of elements ξ of the universe $|I|$ of I—new objects constants that are in a 1–1 correspondence with elements of $|I|$. By I^r we denote the set of atoms from $\mathcal{A}$ that are satisfied by I. In the definition below, t^I stands for the value assigned to the ground term t by the interpretation I. The *grounding* of a first-order sentence F relative to I (symbolically, $gr_I(F)$) is the infinitary formula over A constructed as follows:

– $gr_I(\bot) = \bot$.
– $gr_I(p(t_1, \ldots, t_k)) = p((t_1^I)^*, \ldots, (t_k^I)^*)$.
– $gr_I(t_1 = t_2) = \top$, if $t_1^I = t_2^I$, and $\bot$ otherwise.
– If $F = G \vee H$, $gr_I(F) = gr_I(G) \vee gr_I(H)$ (the case of $\wedge$ is analogous).
– If $F = G \to H$, $gr_I(F) = gr_I(G) \to gr_I(H)$.
– If $F = \exists x G(x)$, $gr_I(F) = \{gr_I(G(u^*)) : u \in |I|\}^\vee$.
– If $F = \forall x G(x)$, $gr_I(F) = \{gr_I(G(u^*)) : u \in |I|\}^\wedge$.

It is easy to check that gr_I is a faithful translation in the following sense: I satisfies a first-order sentence F iff I^r satisfies $gr_I(F)$.

This transformation is also faithful in the sense of the stable model semantics: I satisfies $\mathrm{SM}[F]$ iff I^r is a stable model of $gr_I(F)$ [12, Theorem 5]. This is why infinitary formulas can be used for proving properties of the operator SM.

8 Plan of the Proof

In the statement of the main theorem, the implication left-to-right

$$\mathrm{SM}[\Pi] \to \mathrm{Comp}[\Pi]$$

is logically valid for any Lloyd-Topor program Π. This fact follows from [6, Theorem 11] by the argument used in the proof of Proposition 1 above. To prove the theorem in the other direction, we need to establish the following:

$$\begin{array}{c}\textit{If a Lloyd-Topor program } \Pi \textit{ is } \Gamma\textit{-tight,}\\ \textit{and an interpretation } I \textit{ satisfies both } \Gamma \textit{ and } \mathrm{Comp}[\Pi],\\ \textit{then } I \textit{ satisfies } \mathrm{SM}[\Pi].\end{array} \tag{21}$$

This assertion follows from three lemmas, stated in this section and proved in the appendix. The first of them expresses a Fages-style property of infinitary formulas similar to Theorem 1 from [3]. It deals with *infinitary programs*—conjunctions of (possibly infinitely many) implications $G \to A$ with $A \in \mathcal{A}$. Such an implication will be called an *(infinitary) rule* with the *head* A and *body* G, and we will write it as $A \leftarrow G$. For instance, if Π is a Lloyd-Topor program then, for any interpretation I, $gr_I(\Pi)$ is an infinitary program. We say that an interpretation I is *supported* by an infinitary program Π if each atom $A \in I$ is the head of a rule $A \leftarrow G$ of Π such that $I \models G$. The lemma shows that under some condition the stable models of an infinitary program Π can be characterized as the models that are supported by Π.

The condition refers to the set of *positive nonnegated atoms* of an infinitary formula. This set, denoted by $\mathrm{Pnn}(F)$, and the set of *negative nonnegated atoms* of F, denoted by $\mathrm{Nnn}(F)$, are defined recursively, as follows:

- $\mathrm{Pnn}(\bot) = \emptyset$.
- For $A \in \mathcal{A}$, $\mathrm{Pnn}(A) = \{A\}$.
- $\mathrm{Pnn}(\mathcal{H}^\wedge) = \mathrm{Pnn}(\mathcal{H}^\vee) = \bigcup_{H\in\mathcal{H}} \mathrm{Pnn}(H)$.
- $\mathrm{Pnn}(G \to H) = \begin{cases} \emptyset & \text{if } H = \bot, \\ \mathrm{Nnn}(G) \cup \mathrm{Pnn}(H) & \text{otherwise.} \end{cases}$

- $\mathrm{Nnn}(\bot) = \emptyset$,
- For $A \in \mathcal{A}$, $\mathrm{Nnn}(A) = \emptyset$.
- $\mathrm{Nnn}(\mathcal{H}^\wedge) = \mathrm{Nnn}(\mathcal{H}^\vee) = \bigcup_{H\in\mathcal{H}} \mathrm{Nnn}(H)$.
- $\mathrm{Nnn}(G \to H) = \begin{cases} \emptyset & \text{if } H = \bot, \\ \mathrm{Pnn}(G) \cup \mathrm{Nnn}(H) & \text{otherwise.} \end{cases}$

Let Π be an infinitary program, and I a propositional interpretation. About atoms $A, A' \in I$ we say that A' is a *parent of* A *relative to* Π *and* I if Π has a rule $A \leftarrow G$ with the head A such that $I \models G$ and A' is a positive nonnegated atom of G. We say that Π is *tight on* I if there is no infinite sequence $A_0, A_1, \dots$ of elements of I such that for every i, A_{i+1} is a parent of A_i relative to F and I.

Lemma 2. *For any model I of an infinitary program Π such that Π is tight on I, I is stable iff I is supported by Π.*

The next lemma relates the Γ-tightness condition from the statement of the main theorem to tightness on an interpretation defined above.

Lemma 3. *If a Lloyd-Topor program Π is Γ-tight, and an interpretation I satisfies both Γ and* $\mathrm{Comp}[\Pi]$, *then $gr_I(\Pi)$ is tight on I^r.*

Finally, models of Comp$[\Pi]$ can be characterized in terms of satisfaction and supportedness.

Lemma 4. *For any Lloyd-Topor program Π, an interpretation I satisfies* Comp$[\Pi]$ *iff I^r satisfies $gr_I(\Pi)$ and is supported by $gr_I(\Pi)$.*

To derive assertion (21) from these lemmas, assume that Π is a Γ-tight Lloyd-Topor program, and that I is an interpretation satisfying both Γ and Comp$[\Pi]$. By Lemma 3, $gr_I(\Pi)$ is tight on I^r. By Lemma 4, I^r satisfies $gr_I(\Pi)$ and is supported by $gr_I(\Pi)$. By Lemma 2, it follows that I^r is a stable model of $gr_I(\Pi)$. By Theorem 5 from [12], quoted at the end of Section 7, it follows that I satisfies SM$[\Pi]$.

9 Conclusion

We proposed a new method for representing SM$[F]$ in the language of first-order logic. It is more general than the approach of [6]. Its relationship with the ideas of [8] requires further study. This method allows us, in particular, to prove the equivalence of some ASP descriptions of dynamic domains to axiomatizations based on successor state axioms.

The use of the stable model semantics of infinitary formulas [12] in the proof of the main theorem illustrates the potential of that semantics as a tool for the study of the operator SM.

Acknowledgements

We are grateful to Joohyung Lee and to the anonymous referees for useful comments.

Appendix: Proofs of Lemmas 2–4

Proof of Lemma 2

In this section, Π is an arbitrary infinitary program. For any model I of Π, the reduct Π^I consists of (i) the rules $A \leftarrow G^I$ for all rules $A \leftarrow G$ of Π such that $A \in I$, and (ii) tautologies $\bot \leftarrow \bot$. We will disregard these tautologies and think of Π^I as a program.

Lemma A. *A model I of Π is supported by Π iff it is supported by Π^I.*

Proof. A model I of Π is supported by Π^I iff for every atom $A \in I$ there exists a rule $A \leftarrow G$ in Π such that $I \models G^I$. By Proposition 1 from [12], $I \models G^I$ iff $I \models G$.

Lemma B. *Any stable model of Π is supported by Π.*

Proof. By Lemma A, it is sufficient to check that any stable model I of Π is supported by Π^I. Take an atom $A \in I$. Since I is a stable model of Π, I is minimal among the models of Π^I. Therefore $I \setminus \{A\}$ does not satisfy Π^I, that is to say, for some rule $A' \leftarrow G$ of Π such that

$$A' \in I, \tag{22}$$

$I \setminus \{A\}$ does not satisfy the corresponding rule $A' \leftarrow G^I$ of Π^I. Then

$$I \setminus \{A\} \models G^I \tag{23}$$

and

$$A' \notin I \setminus \{A\}. \tag{24}$$

From (23), $I \models G$ (because otherwise G^I would be $\bot$), and consequently $I \models G^I$. From (22) and (24), $A' = A$. Thus $A' \leftarrow G^I$ is a rule of Π^I such that its head is A and its body is satisfied by I.

Lemma C. *For any infinitary formula F and any interpretation I,*

$$\text{Pnn}(F^I) \subseteq \text{Pnn}(F), \quad \text{Nnn}(F^I) \subseteq \text{Nnn}(F).$$

Proof: Straightforward, by strong induction on the rank of F (defined as the value of i for which $F \in \mathcal{F}_i$).

Lemma D. *For any model I of Π, if Π is tight on I then so is Π^I.*

Proof. Assume that Π^I is not tight on I, and let $A_0, A_1, \ldots$ be an infinite sequence of elements of I such that A_{i+1} is a parent of A_i relative to Π^I and I. Consider the rule of Π^I justifying this property. That rule has the form $A \leftarrow G^I$ for some rule $A \leftarrow G$ of Π such that $A \in I$, and it satisfies the following conditions:

$$A = A_i, \quad I \models G^I, \quad A_{i+1} \in I \cap \text{Pnn}(G^I).$$

Then $I \models G$ and, in view of Lemma C,

$$A_{i+1} \in I \cap \text{Pnn}(G^I) \subseteq I \cap \text{Pnn}(G).$$

Consequently, for every i, A_{i+1} is a parent of A_i relative to I and Π, contrary to the assumption that Π is tight on I.

The statement of the following lemma refers to the set of *strictly positive atoms* of an infinitary formula F, denoted by $\text{SPos}(F)$, which is defined as follows:

- $\text{SPos}(\bot) = \emptyset$.
- For $A \in \mathcal{A}$, $\text{SPos}(A) = \{A\}$.
- $\text{SPos}(\mathcal{H}^\wedge) = \bigcup_{H \in \mathcal{H}} \text{SPos}(H)$.
- $\text{SPos}(\mathcal{H}^\vee) = \bigcup_{H \in \mathcal{H}} \text{SPos}(H)$.
- $\text{SPos}(G \rightarrow H) = \text{SPos}(H)$.

Lemma E. *For any infinitary formula F,* $\text{SPos}(F) \subseteq \text{Pnn}(F)$.

Proof: Straightforward, by induction on the rank of F.

Lemma F. *Let I be a model of an infinitary formula F. If F can be represented in the form G^I for some infinitary formula G then any interpretation J such that*

$$I \cap \text{SPos}(F) \subseteq J$$

is a model of F as well.

Proof. By induction on the rank of G we can show that if $I \models G^I$ (or, equivalently, $I \models G$) and $I \cap \text{SPos}(G^I) \subseteq J$ then $J \models G^I$. Consider the more difficult case when G has the form $H_1 \rightarrow H_2$. Since $I \models G$, and G^I is $H_1^I \rightarrow H_2^I$. We can distinguish between two subcases: (i) $I \not\models H_1$ and (ii) $I \models H_2$. In the first case, H_1^I is $\bot$, so that G^I is tautological, and the assertion $J \models G^I$ is trivial. Assume now that $I \models H_2$. Since

$$I \cap \text{SPos}(G^I) = I \cap \text{SPos}(H_2^I) \subseteq J,$$

we can conclude from the induction hypothesis that $J \models H_2^I$. Consequently $J \models G^I$.

Proof of Lemma 2. The only if part is immediate from Lemma B. Let I be a supported model of Π such that Π is tight on I. To prove the stability of I, we need to show that no proper subset of I satisfies Π^I. Take a proper subset J of I. There is an atom A in $I \setminus J$ that has no parent in $I \setminus J$ relative to Π^I and I. Indeed, if every atom in $I \setminus J$ has a parent relative to Π^I and I that belongs to $I \setminus J$ then there exists an infinite sequence $A_0, A_1, \dots$ of elements of $I \setminus J$ such that A_{i+1} is a parent of A_i, so that Π^I is not tight on I; this is impossible by Lemma D. Consider such an atom A. By Lemma A, I is supported by Π^I. It follows that there is a rule $A \leftarrow F$ in Π^I such that $I \models F$. By the definition of the parent relation, all elements of $I \cap \text{Pnn}(F)$ are parents of A relative to Π^I and I. By the choice of A, no parent of A relative to Π^I and I belongs to $I \setminus J$. Consequently $I \cap \text{Pnn}(F)$ is disjoint from $I \setminus J$, so that

$$I \cap \text{Pnn}(F) \subseteq J.$$

In view of Lemma E, it follows that

$$I \cap \text{SPos}(F) \subseteq J.$$

Since $A \leftarrow F$ is a rule of Π^I, F has the form G^I for some formula G. By Lemma F, it follows that $J \models F$. Since $A \in I \setminus J$, we conclude that J does not satisfy $A \leftarrow F$ and therefore is not a model of Π^I.

Proof of Lemma 3

Lemma 3 relates the Γ-tightness of a Lloyd-Topor program Π (defined in Section 5) to the tightness of $gr_I(\Pi)$ on I^r in the sense of Section 8. As a preliminary step, we will describe a relationship between positive nonnegated atomic subformulas of a first-order formula F, referred to in the definition of the rule dependency graph, and the positive nonnegated atoms of the infinitary formula $gr_I(F)$.

In the following lemma, I is an interpretation in the sense of first-order logic, and F is a first-order sentence that may contain the names ξ^* of elements ξ of the universe of I. If $\mathbf{u}$ is a tuple $\xi_1, \dots, \xi_k$ of elements of the universe then $\mathbf{u}^*$ stands for the corresponding tuple of names $\xi_1^*, \dots, \xi_k^*$. If $\mathbf{t}$ is a tuple $t_1, \dots, t_k$ of ground terms then $gr_I(\mathbf{t})$ stands for the tuple $(t_1^I)^*, \dots, (t_k^I)^*$ of the names of their values.

Lemma G. *For any ground atom of the form $p(\mathbf{u}^*)$,* *(i) if* $p(\mathbf{u}^*) \in \mathrm{Pnn}(gr_I(F))$ *then* $\mathbf{u}^*$ *has the form* $gr_I(\mathbf{t}(\mathbf{v}^*))$ *for some tuple* $\mathbf{t}(\mathbf{x})$ *of terms such that* $p(\mathbf{t}(\mathbf{x}))$ *has a positive nonnegated occurrence in* F*, and some tuple* $\mathbf{v}$ *of elements of the universe;*

(ii) if $p(\mathbf{u}^*) \in \mathrm{Nnn}(gr_I(F))$ *then* $\mathbf{u}^*$ *has the form* $gr_I(\mathbf{t}(\mathbf{v}^*))$ *for some tuple* $\mathbf{t}(\mathbf{x})$ *of terms such that* $p(\mathbf{t}(\mathbf{x}))$ *has a negative nonnegated occurrence in* F*, and some tuple* $\mathbf{v}$ *of elements of the universe.*

Proof. The proof is by induction on the size of F. We will consider three cases: when F atomic, when F is an implication, and when F begins with the universal quantifier.

If F is an atomic formula that does not contain p then $gr(F)$ does not contain atoms of the form $p(\mathbf{u}^*)$, and assertions (i) and (ii) are trivial. Assume that F is $p(\mathbf{t})$, so that $gr_I(p(\mathbf{t})) = p(gr_I(\mathbf{t}))$, $\mathrm{Pnn}(gr_I(F)) = \{p(gr_I(\mathbf{t}))\}$, and $\mathrm{Nnn}(gr_I(F)) = \emptyset$. If $p(\mathbf{u}^*) \in \mathrm{Pnn}(gr_I(F))$ then $\mathbf{u}^* = gr_I(\mathbf{t})$; $p(\mathbf{u}^*) \in \mathrm{Nnn}(gr_I(F))$ is impossible.

If F is $G \to H$ then $gr_I(F)$ is $gr_I(G) \to gr_I(H)$. Assume that $gr_I(H)$ is different from $\perp$ (otherwise both $\mathrm{Pnn}(gr_I(F))$ and $\mathrm{Nnn}(gr_I(F))$ are empty). Then

$$\mathrm{Pnn}(gr_I(F)) = \mathrm{Nnn}(gr_I(G)) \cup \mathrm{Pnn}(gr_I(H)),$$
$$\mathrm{Nnn}(gr_I(F)) = \mathrm{Pnn}(gr_I(G)) \cup \mathrm{Nnn}(gr_I(H)).$$

To prove (i), assume that $p(\mathbf{u}^*) \in \mathrm{Pnn}(gr_I(F))$. Then

$$p(\mathbf{u}^*) \in \mathrm{Nnn}(gr_I(G)) \;\text{ or }\; p(\mathbf{u}^*) \in \mathrm{Pnn}(gr_I(H)).$$

By the induction hypothesis, it follows that $\mathbf{u}^*$ has the form $gr_I(\mathbf{t}(\mathbf{v}^*))$ for some tuple $\mathbf{t}(\mathbf{x})$ of terms such that $p(\mathbf{t}(\mathbf{x}))$ has a negative nonnegated occurrence in G or a positive nonnegated occurrence in H. Since $gr_I(H)$ is not $\perp$, H is not $\perp$ either. Consequenltly $p(\mathbf{t}(\mathbf{x}))$ has a positive nonnegated occurrence in $G \to H$. The proof of (ii) is similar.

If F is $\forall z G(z)$ then

$$gr_I(F) = \{gr_I(G(w^*)) \,:\, w \in |I|\}^\wedge.$$

To prove (i), assume that $p(\mathbf{u}^*) \in \mathrm{Pnn}(gr_I(F))$. Since

$$\mathrm{Pnn}(gr_I(F)) = \bigcup_{w \in |I|} \mathrm{Pnn}(gr_I(G(w^*))),$$

$p(\mathbf{u}^*) \in \mathrm{Pnn}(gr_I(G(w^*)))$ for some $w \in |I|$. By the induction hypothesis, it follows that $\mathbf{u}^*$ has the form $gr_I(\mathbf{t}(\mathbf{v}^*))$ for some tuple $\mathbf{t}(\mathbf{x})$ of terms such that, for some $w \in |I|$, $p(\mathbf{t}(\mathbf{x}))$ has a positive nonnegated occurrence in $G(w^*)$. Without loss of generality we can assume that every member of $\mathbf{x}$ occurs in $\mathbf{t}(\mathbf{x})$. *Case 1:* z is not a member of $\mathbf{x}$. Let $p(\mathbf{t}'(\mathbf{x}, z))$ be the part of $G(z)$ from which the occurrence of $p(\mathbf{t}(\mathbf{x}))$ in $G(w^*)$ is obtained by substituting w^* for z. This part has a positive nonnegated occurrence in $G(z)$, and consequently in $F(z)$. On the other hand, $t(\mathbf{x})$ is $t'(\mathbf{x}, w^*)$, so that $t(\mathbf{v}^*)$ is $t'(\mathbf{v}^*, w^*)$, and

$$\mathbf{u}^* = gr_I(t(\mathbf{v}^*)) = gr_I(\mathbf{t}'(\mathbf{v}^*, w^*)).$$

Case 2: z is a member of $\mathbf{x}$. Then $p(\mathbf{t}(\mathbf{x}))$ contains z, which is only possible if all occurrences of z in the part of $F(z)$ from which the occurrence of $p(\mathbf{t}(\mathbf{x}))$ is obtained by substitution are bound. Then that part of $F(z)$ is not affected by the substitution and equals $p(\mathbf{t}(\mathbf{x}))$. Thus $p(\mathbf{t}(\mathbf{x}))$ has a positive nonnegated occurrence in $F(z)$, and $\mathbf{u}^*$ is $gr_I(t(\mathbf{v}^*))$. The proof of part (ii) is similar.

Proof of Lemma 3. Assume that Π is Γ-tight, that an interpretation I satisfies both Comp$[\Pi]$ and Γ, and that $gr_I(\Pi)$ is not tight on I^r. Then there exists an infinite sequence $A_0, A_1, \dots$ of atoms such that each A_{i+1} is a parent of A_i relative to $gr_I(\Pi)$ and I^r. In other words, there exist rules

$$p_i(gr_I(\mathbf{t}^i(\mathbf{c}_i^*))) \leftarrow gr_I(G_i(\mathbf{c}_i^*)) \qquad (i = 0, 1, \dots)$$

of $gr_I(\Pi)$, obtained by grounding from rules

$$p_i(\mathbf{t}^i(\mathbf{x}^i)) \leftarrow G_i(\mathbf{x}^i) \tag{25}$$

of Π, such that A_i is $p_i(gr_I(\mathbf{t}^i(\mathbf{c}_i^*)))$,

$$I^r \models gr_I(G_i(\mathbf{c}_i^*)), \tag{26}$$

and $A_{i+1} \in \text{Pnn}(gr_I(G_i(\mathbf{c}_i^*)))$. Atom A_{i+1} can be written as $p_{i+1}(\mathbf{u}^*)$, where $\mathbf{u}^*$ is $gr_I(\mathbf{t}^{i+1}(\mathbf{c}_{i+1}^*))$. By Lemma G,

$$gr_I(\mathbf{t}^{i+1}(\mathbf{c}_{i+1}^*)) \text{ is } gr_I(\mathbf{s}^{i+1}(\mathbf{d}_i^*)) \tag{27}$$

for some atom $p_{i+1}(\mathbf{s}^{i+1}(\mathbf{z}^i))$ that has a positive nonnegated occurrence in $G_i(\mathbf{c}_i^*)$, and some tuple $\mathbf{d}_i$ of elements of the universe. That occurrence of $p_{i+1}(\mathbf{s}^{i+1}(\mathbf{z}^i))$ is the result of substituting $\mathbf{c}_i^*$ for $\mathbf{x}^i$ in some atom $p_{i+1}(\mathbf{r}^{i+1}(\mathbf{x}^i, \mathbf{z}^i))$ that has a positive nonnegated occurrence in $G_i(\mathbf{x}^i)$, so that $\mathbf{s}^{i+1}(\mathbf{z}^i)$ is $\mathbf{r}^{i+1}(\mathbf{c}_i^*, \mathbf{z}^i)$. From (27) we conclude that

$$gr_I(\mathbf{t}^{i+1}(\mathbf{c}_{i+1}^*)) \text{ is } gr_I(\mathbf{r}^{i+1}(\mathbf{c}_i^*, \mathbf{d}_i^*)). \tag{28}$$

Since Π is Γ-tight and I satisfies Comp$[\Pi]$ and Γ, there exists n such that, for every chain C in Π of length n, $I \models \widetilde{\forall} \neg F_C$. Consider rules (25) for $i = 0, \dots, n$. Let

$$p_i(\mathbf{t}^i(\widehat{\mathbf{x}^i})) \leftarrow \widehat{G_i}(\widehat{\mathbf{x}^i}) \tag{29}$$

be those rules with variables renamed so that different rules have no common variables. (Formula $\widehat{G_i}(\widehat{\mathbf{x}^i})$ is the result of renaming bound variables in $G_i(\widehat{\mathbf{x}^i})$.) Then $p_{i+1}(\mathbf{r}^{i+1}(\widehat{\mathbf{x}^i}, \widehat{\mathbf{z}^i}))$ has a positive nonnegated occurrence in $\widehat{G_i}(\widehat{\mathbf{x}^i})$, for some tuple $\widehat{\mathbf{z}^i}$

of variables. Let C be the chain

$$
\begin{array}{c}
p^0(\mathbf{t}^0(\widehat{\mathbf{x}^0})) \leftarrow \widehat{G^0}(\widehat{\mathbf{x}^0}) \\
\big\downarrow p^1(\mathbf{r}^1(\widehat{\mathbf{x}^0}, \widehat{\mathbf{z}^0})) \\
p^1(\mathbf{t}^1(\widehat{\mathbf{x}^1})) \leftarrow \widehat{G^1}(\widehat{\mathbf{x}^1}) \\
\big\downarrow p^2(\mathbf{r}^2(\widehat{\mathbf{x}^1}, \widehat{\mathbf{z}^1})) \\
p^2(\mathbf{t}^2(\widehat{\mathbf{x}^2})) \leftarrow \widehat{G^2}(\widehat{\mathbf{x}^2}) \\
\big\downarrow p^3(\mathbf{r}^3(\widehat{\mathbf{x}^2}, \widehat{\mathbf{z}^2})) \\
\cdots \\
\big\downarrow p^n(\mathbf{r}^n(\widehat{\mathbf{x}^{n-1}}, \widehat{\mathbf{z}^{n-1}})) \\
p^n(\mathbf{t}^n(\widehat{\mathbf{x}^n})) \leftarrow \widehat{G^n}(\widehat{\mathbf{x}^n}).
\end{array}
$$

The corresponding chain formula F_C is

$$\bigwedge_{i=0}^{n-1} \mathbf{t}^{i+1}(\widehat{\mathbf{x}^{i+1}}) = \mathbf{r}^{i+1}(\widehat{\mathbf{x}^i}, \widehat{\mathbf{z}^i}) \wedge \bigwedge_{i=0}^{n} \widehat{G_i}(\widehat{\mathbf{x}^i}).$$

Since interpretation I satisfies $\widetilde{\forall}\neg F_C$, it satisfies also

$$\neg\left(\bigwedge_{i=0}^{n-1} \mathbf{t}^{i+1}(\widehat{\mathbf{c}^*_{i+1}}) = \mathbf{r}^{i+1}(\widehat{\mathbf{c}^*_i}, \widehat{\mathbf{d}^*_i}) \wedge \bigwedge_{i=0}^{n} \widehat{G_i}(\widehat{\mathbf{c}^*_i})\right),$$

so that I^r satisfies

$$\neg\left(\bigwedge_{i=0}^{n-1} gr_I(\mathbf{t}^{i+1}(\widehat{\mathbf{c}^*_{i+1}}) = \mathbf{r}^{i+1}(\widehat{\mathbf{c}^*_i}, \widehat{\mathbf{d}^*_i})) \wedge \bigwedge_{i=0}^{n} gr_I(\widehat{G_i}(\widehat{\mathbf{c}^*_i}))\right).$$

In view of (28), each of the formulas $gr_I(\mathbf{t}^{i+1}(\widehat{\mathbf{c}^*_{i+1}}) = \mathbf{r}^{i+1}(\widehat{\mathbf{c}^*_i}, \widehat{\mathbf{d}^*_i}))$ is $\top$, so that I^r satisfies

$$\neg\left(\bigwedge_{i=0}^{n} gr_I(\widehat{G_i}(\widehat{\mathbf{c}^*_i}))\right).$$

This is impossible by (26).

Proof of Lemma 4

Recall that the rules of a Lloyd-Topor program Π have the form

$$p(\mathbf{t}(\mathbf{y})) \leftarrow G(\mathbf{y})$$

(with all free variables of the rule explictly shown), and that the rules of the infinitary program $gr_I(\Pi)$ have the form

$$p(gr_I(\mathbf{t}(\mathbf{v}^*))) \leftarrow gr_I(G(\mathbf{v}^*)) \tag{30}$$

for all tuples $\mathbf{v}$ of elements of $|I|$. For any Lloyd-Topor program Π, Comp$[\Pi]$ is equivalent to the conjunction of Π with the universal closures of the definitions (9) of all predicate constants p. To prove Lemma 4, we need to check that the condition: for all p,

$$I \models \forall \mathbf{x} \left(p(\mathbf{x}) \rightarrow \bigvee_i \exists \mathbf{y}^i (\mathbf{x} = \mathbf{t}^i(\mathbf{y}^i)) \wedge G^i(\mathbf{y}^i) \right), \tag{31}$$

is equivalent to the assertion that $gr_I(\Pi)$ is supported by I^r. Note first that (31) is equivalent to the condition:

$$I^r \models \left\{ gr_I \left(p(\mathbf{u}^*) \rightarrow \bigvee_i \exists \mathbf{y}^i (\mathbf{u}^* = \mathbf{t}^i(\mathbf{y}^i) \wedge G_i(\mathbf{y}^i)) \right) \,:\, \mathbf{u} \in |I|^k \right\}^{\wedge},$$

where k is the arity of p. The conjunctive terms $gr_I(\cdots)$ can be written as

$$p(\mathbf{u}^*) \rightarrow \bigvee_i \left\{ gr_I(\mathbf{u}^* = \mathbf{t}^i(\mathbf{v}^*)) \wedge gr_I(G_i(\mathbf{v}^*))) \,:\, \mathbf{v} \in |I|^{l_i} \right\}^{\vee},$$

where l_i is the length of the tuple $\mathbf{y}^i$. Therefore (31) is equivalent to following condition: for every $\mathbf{u} \in |I|^k$ such that $p(\mathbf{u}^*) \in I^r$,

$$\text{there exist } i \text{ and } \mathbf{v} \text{ such that } \mathbf{u}^* \text{ is } gr_I(\mathbf{t}^i(\mathbf{v}^*)), \text{ and } I^r \models gr_I(G_i(\mathbf{v}^*)). \tag{32}$$

Condition (32) is equivalent to saying that $p(\mathbf{u}^*)$ is the head of one of the rules (30) whose body is satisfied by I^r.

References

1. Fages, F.: Consistency of Clark's completion and existence of stable models. Journal of Methods of Logic in Computer Science **1** (1994) 51–60
2. Lifschitz, V.: Foundations of logic programming. In Brewka, G., ed.: Principles of Knowledge Representation. CSLI Publications (1996) 69–128
3. Erdem, E., Lifschitz, V.: Tight logic programs. Theory and Practice of Logic Programming **3** (2003) 499–518
4. Lin, F., Zhao, J.: On tight logic programs and yet another translation from normal logic programs to propositional logic. In: Proceedings of International Joint Conference on Artificial Intelligence (IJCAI). (2003) 853–864
5. Lin, F., Zhao, Y.: ASSAT: Computing answer sets of a logic program by SAT solvers. Artificial Intelligence **157** (2004) 115–137
6. Ferraris, P., Lee, J., Lifschitz, V.: Stable models and circumscription. Artificial Intelligence **175** (2011) 236–263
7. Lloyd, J., Topor, R.: Making Prolog more expressive. Journal of Logic Programming **1** (1984) 225–240

8. Lee, J., Meng, Y.: First-order stable model semantics and first-order loop formulas. Journal of Artificial Inteligence Research (JAIR) **42** (2011) 125–180
9. Lifschitz, V., Pearce, D., Valverde, A.: Strongly equivalent logic programs. ACM Transactions on Computational Logic **2** (2001) 526–541
10. Lifschitz, V., Pearce, D., Valverde, A.: A characterization of strong equivalence for logic programs with variables. In: Procedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR). (2007) 188–200
11. Bartholomew, M., Lee, J.: Stable models of formulas with intensional functions. In: Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR). (2012)
12. Truszczyński, M.: Connecting first-order ASP and the logic FO(ID) through reducts. In: Correct Reasoning: Essays on Logic-Based AI in Honor of Vladimir Lifschitz. Springer (2012)
13. Ferraris, P.: Answer sets for propositional theories. In: Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR). (2005) 119–131