First-order answer set programming as constructive proof search

Aleksy Schubert and Pawel Urzyczyn

May 31, 2018

First-order answer set programming as constructive proof search

Aleksy Schubert and Paweł Urzyczyn

University of Warsaw, Poland

(*e-mail:* `[alx,urzy]@mimuw.edu.pl`)

Abstract

We propose an interpretation of the first-order answer set programming (FOASP) in terms of intuitionistic proof theory. It is obtained by two polynomial translations between FOASP and the bounded-arity fragment of the Σ_1 level of the Mints hierarchy in first-order intuitionistic logic. It follows that Σ_1 formulas using predicates of fixed arity (in particular unary) is of the same strength as FOASP. Our construction reveals a close similarity between constructive provability and stable entailment, or equivalently, between the construction of an answer set and an intuitionistic refutation.

KEYWORDS: Answer set programming, intuitionistic logic, proof terms, lambda calculus

Contents

1 Introduction

The logic programming paradigm originates from the fundamental idea that program execution can be realized as constructive proof search done by means of SLD resolution. A solution to a computational problem is obtained in proof construction as a witness of the statement being proven. While very natural and effective, this idea is hardly applied any further than to slight variations of Horn clauses. (Works exploring the constructive approach, like (Fu and Komendantskaya 2017), are not very common.) Consider for example first-order ASP which is a form of adding negation to Datalog. The standard way in which ASP is implemented consists in encoding a program into a satisfiability problem and then applying a SAT-solver (Brewka et al. 2011). This clearly goes away from the proof search oriented motivation of logic programming.

In this paper we attempt to argue that this semantic approach to ASP is not necessary. We demonstrate a proof-theoretical interpretation of answer set programming by showing a mutual translation between first order ASP and the bounded arity Σ_1 fragment of intuitionistic predicate logic, as defined in (Schubert et al. 2016). This extends our previous work (Schubert and

Urzyczyn 2018) where we have defined a similar equivalence between propositional answer set programming and a fragment of intuitionistic propositional logic. The extension required a new approach since the techniques used in (Schubert and Urzyczyn 2018) turned out to be not adequate in the first-order case. Of course, a first-order program can be propositionalized by grounding, and then one could repeat the propositional construction to obtain a (quantifier-free) formula provable iff the given entailment holds. However, the resulting formula would be no longer polynomial in the size of the original program, as the grounding is exponential in size.

In addition, avoiding the exponential explosion is far not enough. The straightforward generalization of the propositional approach would yield a formula that does not fit our *co*-NEXPTIME fragment of Σ_1, because targets of subformulas would be of arbitrary arity. Instead we built our formula using "easy" axioms with nullary targets. While the overall scheme "unsound versus inconsistent" remains the same, the construction must be different. This is because we can no longer use proof goals to control proof construction, (in particular to discover loops in the unsound case B) because there is too few of them. Instead we must use (significantly more nondeterministic) approach, continuation-passing in spirit.

Strictly speaking, the direct correspondence between proofs and programs has a "contravariant" flavour in the case of ASP (also known as stable model semantics). Indeed, a stable model of a program does not represent a proof of a formula but rather a *refutation*, i.e., a certificate that a proof does not exist. Put it differently, provability in a fragment of first-order intuitionistic logic is equivalent to stable *entailment*, i.e., the ASP consequence relation.

The construction consists of two polynomial time translations. First, for a given first-order logic program $\mathscr{P}$ with negations, and a given atom Ω we define a formula φ so that $\mathscr{P}$ entails Ω under stable model semantics if and only if φ is an intuitionistic theorem (Proposition 1).

The formula φ is a first-order formula using only unary predicate symbols and only $\forall$ and $\rightarrow$ as the logical connectives. Since all quantifiers occur in φ at negative positions, our formula belongs to the Σ_1 fragment of Mints hierarchy (Schubert et al. 2016). In general, provability of Σ_1-formulas is NEXPSPACE-complete. However, for any fixed bound on the arity of predicate symbols, in particular for our unary object language. the provability problem turns out to be *co*-NEXPTIME complete.

Actually, the formula φ constructed in Section 3 is not yet unary. It is an *easy* formula in the sense of (Schubert et al. 2016), because all its non-atomic subformulas have nullary targets. Provability of such formulas can however be reduced to provability with unary predicates.

Our second construction reduces the question of refutability (non-provability) of a formula φ to the non-existence of a stable model of a logic program with negations. Yet differently, given a formula φ, we define $\mathscr{P}$ so that φ is not an intuitionistic theorem if and only if $\mathscr{P}$ has a stable model (Proposition 2). The form of the latter equivalence is intended to stress that the stable models of $\mathscr{P}$ correspond to intuitionistic refutations. The construction works for Σ_1 formulas using predicate symbols of any bounded arity, not necessarily unary.

The two translations together add up to a polynomial equivalence of two problems known to be NEXPTIME-complete: the existence of stable models and the provability for bounded-arity Σ_1 formulas.

As such, it does not constitute a formally new theorem concerning complexity of the formalisms, and it was not meant to do so. The purpose and contribution of this paper is to point out a natural constructive counterpart of the first-order answer set programming, and by derivation of the correspondence to expose the constructive nature of ASP. In this way we complement the purely propositional accounts (Pearce 2006; Schubert and Urzyczyn 2018) of the links between

logic programming and intuitionistic logic. In addition the current account is in terms of proof theory and its details are closely related to the original procedure of answer derivation in logic programming.

2 Answer set programming and logic

First order answer set programming is an extension of Datalog by negation understood as "fix-point" (Kolaitis and Papadimitriou 1988). The short presentation we give here is essentially following (Brewka et al. 2011).

Basic definitions: A *positive atom*, or simply an *atom*, is an atomic formula $P(\vec{x})$, where $\vec{x}$ are individual variables or constants. A *negative atom* has the form $\neg P(\vec{x})$, where $P(\vec{x})$ is a positive atom. A *ground* atom is an atom where only constants can occur (and no variables).

A *clause* is an expression of the form $\mathbf{a} :- \mathbf{a}_1, \ldots, \mathbf{a}_n$, where $\mathbf{a}$ is a positive atom and $\mathbf{a}_1, \ldots, \mathbf{a}_n$ are positive or negative atoms. A *ground* clause is one without variables. A *program* is a finite set of clauses $\mathscr{P}$, together with an associated nonempty finite set of constants $D_{\mathscr{P}}$, the *domain of* $\mathscr{P}$, which is assumed to include all constants occurring in $\mathscr{P}$.

We write $B(\mathscr{P})$ for the set of all positive ground atoms $P(\vec{c})$, where P is a predicate in the language of $\mathscr{P}$ and $\vec{c}$ is a vector of constants in $D_{\mathscr{P}}$ of appropriate length. By $ground(\mathscr{P})$ we denote the set of all ground clauses obtained from clauses of $\mathscr{P}$ as substitution instances using constants in $D_{\mathscr{P}}$. Note that $ground(\mathscr{P})$ can be seen as a propositional program with negations using ground atoms in $B(\mathscr{P})$ as propositional literals. The following definitions are therefore immediate adaptations from the propositional case.

A *model* of the language of $\mathscr{P}$ is a set $\mathfrak{M}$ of positive ground atoms. One can think of $\mathfrak{M}$ as of a Herbrand structure of domain $D_{\mathscr{P}}$, satisfying the ground atoms in $\mathfrak{M}$ and no other atom, or as of a Boolean valuation of ground atoms. Given a program $\mathscr{P}$ and a model $\mathfrak{M}$, we transform $ground(\mathscr{P})$ into a program $\mathscr{P}^{\mathfrak{M}}$ without negations. For every ground atom $\mathbf{a}$:

- If $\mathbf{a} \notin \mathfrak{M}$ then we delete $\neg\mathbf{a}$ from the rhs of all clauses of $ground(\mathscr{P})$;
- If $\mathbf{a} \in \mathfrak{M}$ then we delete all clauses of $ground(\mathscr{P})$ with $\neg\mathbf{a}$ at the rhs.

The *interpretation* of $\mathscr{P}$ under $\mathfrak{M}$, denoted $I(\mathscr{P}, \mathfrak{M})$, is the least fixed point of the operator:

$$F(\mathscr{I}) = \mathscr{I} \cup \{\mathbf{a} \mid \text{there is a clause } \mathbf{a} :- \mathbf{a}_1, \ldots, \mathbf{a}_n \text{ in } \mathscr{P}^{\mathfrak{M}} \text{ such that all } \mathbf{a}_i \text{ are in } \mathscr{I}\}.$$

A model $\mathfrak{M}$ of $\mathscr{P}$ is *stable* (or it is an *answer set* for $\mathscr{P}$) if and only if $\mathfrak{M} = I(\mathscr{P}, \mathfrak{M})$. We say that $\mathscr{P}$ *entails* an atom $\mathbf{a}$ *under SMS*, written $\mathscr{P} \models_{\text{SMS}} \mathbf{a}$, when every stable model of $\mathscr{P}$ satisfies $\mathbf{a}$. It is known (Dantsin et al. 2001; Kolaitis and Papadimitriou 1988) that the existence of a stable model and the entailment under SMS are, respectively, NExpTime and *co*-NExpTime complete problems.

Lemma 2.1
Let Ω be a nullary predicate letter not occurring in a program $\mathscr{P}$. Then $\mathscr{P} \models_{\text{SMS}} \Omega$ if and only if $\mathscr{P}$ has no stable model.

Intuitionistic logic: We consider formulas of minimal predicate logic with $\rightarrow$ and $\forall$ as the only connectives. Our formulas use neither function symbols nor equality. In particular, the only individual terms are *object variables*, written in lower case, e.g., $x, y, \ldots$ A formula is therefore

either an *atom* $P(x_1, \ldots, x_n)$, where $n \geq 0$, or an implication $\varphi \to \psi$, or it has the form $\forall x \varphi$. The convention is that $\to$ associates to the right, i.e., $\varphi \to \psi \to \vartheta$ stands for $\varphi \to (\psi \to \vartheta)$.

Quantifier scopes are marked by parentheses or dots; for example $\forall x. P(x) \to Q(x)$ is the same as $\forall x(P(x) \to Q(x))$. The set of free variables in a formula φ is defined as usual and denoted $FV(\varphi)$. The notation $\varphi[x := y]$ stands for a capture-avoiding substitution of y for all free occurrences of x in φ. We also use the notion of simultaneous substitution written $\varphi[\vec{x} := \vec{y}]$.

A clause of the form $\mathbf{a} :- \mathbf{a}_1, \ldots, \mathbf{a}_n$ (without negations) is identified with the universal formula $\forall \vec{x}(\mathbf{a}_1 \to \cdots \to \mathbf{a}_n \to \mathbf{a})$, where $\vec{x}$ are all individual variables occurring in the clause. In particular a ground clause is an implicational closed formula, and a ground program can be seen as a set of such formulas.

Let $\overline{\mathscr{P}}$ be obtained from $\mathit{ground}(\mathscr{P})$ by replacing all occurrences of negative atoms $\neg P(\vec{c})$ by $\overline{P}(\vec{c})$, where $\overline{P}$ is a new predicate symbol. Now let $\overline{\mathfrak{M}} = \{\overline{\mathbf{b}} \mid \mathbf{b} \in B(\mathscr{P}) - \mathfrak{M}\}$. The following easy lemma (where $\vdash$ is intuitionistic provability) is an immediate consequence of Lemma 4 in (Schubert and Urzyczyn 2018).

Lemma 2.2
$I(\mathscr{P}, \mathfrak{M}) = \{\mathbf{a} \in B(\mathscr{P}) \mid \overline{\mathscr{P}} \cup \overline{\mathfrak{M}} \vdash \mathbf{a}\}.$

Lambda-terms: We use lambda-terms for proof notation as e.g. in (Schubert et al. 2015). In addition to object variables, used in formulas, we also have *proof variables* occurring in proofs. By convention, proof variables are written in upper case, e.g., $X, Y, \ldots$ A finite set of declarations $(X : \varphi)$, where X is a proof variable and φ is a formula, is called an *environment* provided there is at most one declaration for any variable X. A *proof term* (or simply "term") is one of the following:

- a proof variable,
- an abstraction $\lambda X : \varphi. M$, where φ is a formula and M is a proof term,
- an abstraction $\lambda x M$, where M is a proof term,
- an application MN, where M, N are proof terms,
- an application Mx, where M is a proof term and x is an object variable.

That is, we have two kinds of lambda-abstraction: the proof abstraction $\lambda X : \varphi. M$ and the object abstraction $\lambda x M$. There are also two forms of application: the proof application MN, where N is a proof term, and the object application My, where y is an object variable. Terms (and also formulas) are taken up to alpha-conversion, i.e., the choice of bound variables is irrelevant.

The type-assignment rules in Figure 1 infer judgments of the form $\Gamma \vdash M : \varphi$, where Γ is an environment, M is a term, and φ is a formula. In rule $(\forall I)$ we require $x \notin FV(\psi)$ for every $\psi \in \Gamma$.

The formalism is used liberally. For instance, we may say that "a term M has type α" leaving the environment implicit. We also may identify proof environments with the appropriate sets of formulas.

The notion of a term in *long normal form* (lnf) is defined according to its type in a given environment.

- If N is an lnf of type φ then $\lambda x N$ is an lnf of type $\forall x \varphi$.
- If N is an lnf of type ψ then $\lambda X : \varphi. N$ is an lnf of type $\varphi \to \psi$.
- If $N_1, \ldots, N_n$ are lnf or object variables, and $X N_1 \ldots N_n$ is of an atom type, then $X N_1 \ldots N_n$ is an lnf.

The following lemma makes it possible to restrict proof search to long normal forms.

Lemma 2.3 (Schubert et al. 2015)

If φ is intuitionistically derivable from Γ then $\Gamma \vdash N : \varphi$, for some lnf N.

$$\Gamma, X : \varphi \vdash X : \varphi \quad (Ax)$$

$$\frac{\Gamma, X : \varphi \vdash M : \psi}{\Gamma \vdash \lambda X : \varphi.M \; : \; \varphi \to \psi} \; (\to I) \qquad \frac{\Gamma \vdash M : \varphi \to \psi \quad \Gamma \vdash N : \varphi}{\Gamma \vdash MN : \psi} \; (\to E)$$

$$\frac{\Gamma \vdash M : \varphi}{\Gamma \vdash \lambda x M : \forall x \varphi} \; (\forall I) \qquad \frac{\Gamma \vdash M : \forall x \varphi}{\Gamma \vdash My : \varphi[x := y]} \; (\forall E)$$

Fig. 1. Proof assignment rules

We are concerned with the Σ_1 level of the Mints hierarchy, defined by:

- $\Sigma_1 ::= \mathbf{a} \mid \Pi_1 \to \Sigma_1$;
- $\Pi_1 ::= \mathbf{a} \mid \Sigma_1 \to \Pi_1 \mid \forall x \Pi_1$,

where $\mathbf{a}$ stands for an atomic formula.

Observe that every Σ_1-formula takes the form $\varphi = \tau_1 \to \tau_2 \to \cdots \to \tau_q \to \mathbf{c}$, where all τ_i are Π_1-formulas and $\mathbf{c}$ is an atom, called the *target* of φ. The following follows from Lemma 2.3.

Lemma 2.4

If $\varphi = \tau_1 \to \tau_2 \to \cdots \to \tau_q \to \mathbf{c}$ then $\Gamma \vdash \varphi$ is derivable if and only if so is $\Gamma, \tau_1, \ldots, \tau_q \vdash \mathbf{c}$.

Proof

A long normal proof of φ (i.e., a lnf of type φ) must have the form $\lambda X_1 : \tau_1 \ldots \lambda X_q : \tau_q.N$, where the term N has type $\mathbf{c}$. $\square$

Thus to determine if a given Σ_1-formula is provable, one has to consider judgments $\Gamma \vdash \mathbf{c}$, where all members of Γ are in Π_1, i.e., of the form $\forall \vec{y}_1 (\sigma_1 \to \forall \vec{y}_2 (\sigma_2 \to \cdots \to \forall \vec{y}_k (\sigma_k \to \mathbf{b}) \ldots))$, where $\sigma_i \in \Sigma_1$, and $\mathbf{b}$ is an atom. The atom $\mathbf{b}$ is called the *target* of ψ.

From Lemmas 2.3 and 2.4 we now obtain the following "generation lemma":

Lemma 2.5

Let $\mathbf{c}$ be an atom, and let Γ consists of Π_1 formulas. Then $\Gamma \vdash \mathbf{c}$ is derivable iff there is a formula in Γ of the form $\psi = \forall \vec{y}_1 (\sigma_1 \to \forall \vec{y}_2 (\sigma_2 \to \cdots \to \forall \vec{y}_k (\sigma_k \to \mathbf{b}) \ldots))$, and variables $\vec{x} = \vec{x}_1 \ldots, \vec{x}_k$ such that $\mathbf{b}[\vec{y} := \vec{x}] = \mathbf{c}$, where $\vec{y} = \vec{y}_1 \ldots \vec{y}_k$, and, for $i = 1 \ldots, k$, if $\sigma_i = \tau_1^i \to \tau_2^i \to \cdots \to \tau_{q_i}^i \to \mathbf{a}_i$ then all judgments $\Gamma, \tau_1^i[\vec{y} := \vec{x}], \ldots, \tau_{q_i}^i[\vec{y} := \vec{x}] \vdash \mathbf{a}_i[\vec{y} := \vec{x}]$ are derivable.

Lemma 2.5 defines a Ben-Yelles style proof-search algorithm, which can be seen as a behaviour of an alternating automaton. One interprets $\Gamma \vdash \mathbf{c}$ as a configuration of the machine, where $\mathbf{c}$ is the internal state and Γ is (write-only) memory or database. A computation step consists of a nondeterministic choice of an assumption ψ from the database, together with an appropriate substitution, and of a universal split into as many computation branches as there are premises in ψ. Each of these commences with the proof goal (internal state) given by a target of one of the premises in ψ.

3 SMS entailment as provability

Given a program $\mathscr{P}$ with negations, and a ground atom Ω, we define in this section a formula φ such that $\mathscr{P}$ entails Ω under stable model semantics if and only if φ is intuitionistically provable. We begin with some preparatory considerations regarding the (now fixed) program $\mathscr{P}$.

We assume that $D_{\mathscr{P}} = \{c_1, \ldots, c_m\}$, and every clause K of $\mathscr{P}$ is written as

$$R(\vec{u}) :- P_1(\vec{v}_1), \ldots, P_r(\vec{v}_r), \neg S_1(\vec{w}_1), \ldots, \neg S_s(\vec{w}_s),$$

where $\vec{u}, \vec{v}_1, \ldots, \vec{v}_r, \vec{w}_1, \ldots, \vec{w}_s$ are some vectors of variables and constants, possibly with repetitions. The *arity* of clause K is the arity of R (the length of $\vec{u}$).

Let $\vec{x}$ be the sequence of all variables in $\vec{u}$ taken without repetitions. Then let $\vec{y}$ stand for the sequence of all variables occurring in $\vec{v}_1, \ldots, \vec{v}_r, \vec{w}_1, \ldots, \vec{w}_s$, but not occurring in $\vec{x}$, listed without repetitions in some fixed order. We may now denote the clause K by $K(\vec{x}, \vec{y})$. The vector $\vec{y}$ consists of variables that only occur at the rhs of K.

Recall that program $\overline{\mathscr{P}}$ is obtained from $ground(\mathscr{P})$ by replacing all occurrences of negative atoms $\neg P(\vec{c})$ by $\overline{P}(\vec{c})$. The program $\overline{\mathscr{P}}$ thus consists of clauses $K(\vec{a}, \vec{b})$ of the form

$$R(\vec{e}) :- P_1(\vec{c}_1), \ldots, P_r(\vec{c}_r), \overline{S}_1(\vec{d}_1), \ldots, \overline{S}_s(\vec{d}_s),$$

obtained from $K(\vec{x}, \vec{y})$ by substituting elements $\vec{a}, \vec{b}$ of $D_{\mathscr{P}}$ for $\vec{x}, \vec{y}$. Vectors $\vec{e}, \vec{c}_i$, and $\vec{d}_i$ are appropriately selected from $\vec{a}, \vec{b}$. More precisely, if S denotes the substitution $[\vec{x}, \vec{y} := \vec{a}, \vec{b}]$, then $\vec{e} = \vec{u}[S]$, $\vec{c}_i = \vec{v}_i[S]$ and $\vec{d}_i = \vec{w}_i[S]$.

The vocabulary of φ consists of $D_{\mathscr{P}}$ and the following predicate symbols:

- For every predicate symbol P in $\mathscr{P}$, there are four symbols P, $\overline{P}$, P!, and P?, each of the same arity as P.
- For every pair P, Q of predicate symbols, of arity m, n respectively, there is a predicate PQ of arity $m + n$.
- For every clause $K(\vec{x}, \vec{y})$ of arity l with $|\vec{y}| = m$, and for every $i = 0, \ldots, m$ there is a predicate K^i of arity $l + i$, and a nullary predicate $\overline{K}^i$.
- In addition we have six nullary predicates *loop*, Ω, A, B, $\circ$, and $\bullet$.

The arity of predicates occurring in φ depends on the arity of atoms in $\mathscr{P}$. However, all targets of the implication subformulas in φ are nullary. As such, φ is an "easy formula" and by (Schubert et al. 2016) it is translatable in polynomial time to a formula φ' with unary predicates so that φ' is provable if and only if so is φ.

The formula φ has the form $\psi_1 \to \cdots \to \psi_d \to loop$, where *loop* is a nullary predicate symbol, and $\psi_1, \ldots, \psi_d$ are closed formulas, called *axioms*, listed in Figure 2. We now explain how these axioms can be used to prove φ.

First we have the initial axioms (1), for every predicate letter R. These axioms can be applied towards proving the main goal *loop*. Every time an axiom of type (1) is used, the proof universally splits into two branches and at each of them a ground atom either of the form $R(\vec{c})$ or of the form $\overline{R}(\vec{c})$ is added to the proof environment. At the end of this phase we are facing the task of proving *loop* under any possible choice of positive or overlined ground facts. More precisely, every branch of the proof leads to a judgment $\Gamma \vdash loop$, where Γ contains a selection of ground atoms in addition to the axioms of φ. Ideally we could make exactly as many choices as is needed to fully determine a different model $\mathfrak{M}$ at every branch. This happens when axiom (1) is used at every branch exactly once for every predicate letter R and every substitution $\vec{z} := \vec{c}$. Then the set Γ contains either $R(\vec{c})$ or $\overline{P}(\vec{c})$, for every R and every $\vec{c}$. We could achieve this in the propositional case (Schubert and Urzyczyn 2018) by using all propositional atoms as proof goals one after one

1. $\forall \vec{z}((R(\vec{z}) \to loop) \to (\overline{R}(\vec{z}) \to loop) \to loop)$,   for every predicate symbol R of $\mathscr{P}$.
2. $\Omega \to loop$, $A \to loop$,   and $B \to loop$.
3. $\forall \vec{z}. \overline{R}(\vec{z}) \to (R!(\vec{z}) \to \bullet) \to A$,   for every predicate symbol R of $\mathscr{P}$.
4. $\forall \vec{z}. R(\vec{z}) \to (R?(\vec{z}) \to \circ) \to B$,   for every predicate symbol R of $\mathscr{P}$.
5. $\forall \vec{x}\vec{y}. R!(\vec{u}) \to (P_1!(\vec{v}_1) \to \bullet) \to \cdots \to (P_r!(\vec{v}_r) \to \bullet) \to \overline{S}_1(\vec{w}_1) \to \cdots \to \overline{S}_t(\vec{w}_s) \to \bullet$,
   for every $K(\vec{x},\vec{y})$ of the form $R(\vec{u}) :\!- P_1(\vec{v}_1),\ldots,P_r(\vec{v}_r),\neg S_1(\vec{w}_1),\ldots,\neg S_t(\vec{w}_t)$.
6. $\forall \vec{z}. R?(\vec{z}) \to (K_1^0(\vec{z}) \to \overline{K}_1^0) \to \cdots \to (K_l^0(\vec{z}) \to \overline{K}_l^0) \to \circ$,   for every predicate symbol R
   such that $K_1,\ldots,K_l$ are all clauses with target R.
7. $\forall \vec{z}_1\vec{z}_2. K^0(\vec{z}_1,d,\vec{z}_2) \to \overline{K}^0$,   for every constant $d \neq c$, and every clause K with target $R(\vec{u})$,
   where c occurs in $\vec{u}$ as the $|\vec{z}_1|+1$-st argument.
8. $\forall \vec{z}_1\vec{z}_2\vec{z}_3. K^0(\vec{z}_1,c,\vec{z}_2,d,\vec{z}_3) \to \overline{K}^0$,   for every pair c,d of different constants, and every
   clause K with target $R(\vec{u})$, where the $|\vec{z}_1|+1$-st and the $|\vec{z}_1|+1+|\vec{z}_2|+1$-st arguments in
   $\vec{u}$ are the same variable.
9. $\forall \vec{z}\vec{v}. K^i(\vec{z},\vec{v}) \to (K^{i+1}(\vec{z},\vec{v},c_1) \to \overline{K}^{i+1}) \to \cdots \to (K^{i+1}(\vec{z},\vec{v},c_m) \to \overline{K}^{i+1}) \to \overline{K}^i$,
   for every clause $K(\vec{x},\vec{y})$, and every prefix $\vec{v}$ of $\vec{y}$ such that $|\vec{v}| = i < |\vec{y}|$.
   (Recall that $c_1,\ldots,c_m$ are all the constants occurring in $\mathscr{P}$.)
10. $\forall \vec{x}\vec{y}. K^m(\vec{u},\vec{y}) \to (P(\vec{v}) \to RP(\vec{u},\vec{v}) \to \circ) \to \overline{K}^m$,   for every clause $K(\vec{x},\vec{y})$ with target $R(\vec{u})$
    such that $P(\vec{v})$ occurs at the rhs of $K(\vec{x},\vec{y})$.
11. $\forall \vec{x}\vec{y}. K^m(\vec{u},\vec{y}) \to S(\vec{w}) \to \overline{K}^m$,   for every clause $K(\vec{x},\vec{y})$ with target $R(\vec{u})$ such that $\neg S(\vec{w})$
    occurs at the rhs of $K(\vec{x},\vec{y})$.
12. $\forall \vec{z}\vec{y}\vec{w}(RP(\vec{z},\vec{y}) \to PQ(\vec{y},\vec{w}) \to (RQ(\vec{z},\vec{w}) \to \circ) \to \circ)$,   for every predicate symbols R,P,Q
    of $\mathscr{P}$ of respective arity $|\vec{z}|, |\vec{y}|, |\vec{w}|$.
13. $\forall \vec{z}(PP(\vec{z},\vec{z}) \to \circ)$,   for every predicate symbol P of $\mathscr{P}$.

Fig. 2. Axioms of the formula φ. Vectors $\vec{z}, \vec{z}_i$ always consist of different variables.

in a deterministic fashion. As we now work with nullary goals, this is impossible, and we must accept nondeterminism. So in general on any given branch we may either have a "partial model" (when neither $R(\vec{c})$ nor $\overline{R}(\vec{c})$ was chosen) or an "inconsistency" (when both are chosen). In the latter case a certain branch of the proof leads to a judgment of the form $\Gamma, R(\vec{c}), \overline{R}(\vec{c}) \vdash loop$. Observe though that such an inconsistency is not as dangerous as it may appear. It only happens when the same axiom (1) was used twice for the same ground substitution $\vec{z} := \vec{c}$. Then in addition to the "inconsistent" branch we have two other branches where both $R(\vec{c})$ and $\overline{R}(\vec{c})$ were selected in a consistent manner. A successful proof must therefore always handle all the consistent cases, i.e., apply to all models.

The goal *loop* should be provable when $\mathscr{P}$ entails Ω. The three axioms (2):

$$\Omega \to loop, \qquad\qquad A \to loop, \qquad\qquad B \to loop,$$

where A, B are fixed nullary predicate letters, correspond to the three ways in which the entailment $\mathscr{P} \models_{\text{SMS}} \Omega$ may hold in a stable model: either Ω holds, or the model is unstable because $\overline{\mathscr{P}} \cup \overline{\mathfrak{M}}$ proves too much ($\mathscr{P}$ is unsound for $\mathfrak{M}$), or the model is unstable because $\overline{\mathscr{P}} \cup \overline{\mathfrak{M}}$ does not prove what is needed ($\mathscr{P}$ is incomplete for $\mathfrak{M}$). In the first case, a proof of φ should be completed with help of the axiom $\Omega \to loop$, as the atom Ω (a member of $B(\mathscr{P})$) is available as an assumption. Otherwise the proof goal is set to A or B, respectively.

The unsound case A :   For the unsound case we have a nullary predicate $\bullet$ and a collection of axioms (3) of the form

$$\forall \vec{z}.\, \overline{R}(\vec{z}) \to (R!(\vec{z}) \to \bullet) \to A.$$

Only one of axioms (3) can be used at every branch of any proof and only once. In our automata-theoretic interpretation, it passes control from A to $\bullet$, adding $R!(\vec{c})$ to the memory, for some instance $\vec{c}$ of $\vec{z}$. The following proof should succeed if $R(\vec{c}) \in I(\mathscr{P}, \mathfrak{M})$, despite that $\overline{R}(\vec{c}) \in \overline{\mathfrak{M}}$.

The intuitive meaning of the predicate $R!(\vec{c})$ is: "the atom $R(\vec{c})$ has been visited as a proof goal in a derivation from $\overline{\mathscr{P}} \cup \overline{\mathfrak{M}}$" (cf. Lemma 2.2).

Now consider any clause of $\mathscr{P}$ with target R, e.g., $R(\vec{x}) :- P(\vec{x}), Q(\vec{x}), \neg S(\vec{x})$, where $\vec{x}$ consists only of variables and has no repetitions. (That is assumed for simplicity of the example.) With such a clause we associate an assumption of type (5) which in this case takes the form

$$\forall \vec{x}.\, R!(\vec{x}) \to (P!(\vec{x}) \to \bullet) \to (Q!(\vec{x}) \to \bullet) \to \overline{S}(\vec{x}) \to \bullet,$$

and enforces a universal split to two new tasks: (i) prove $\bullet$ under the assumption $P!(\vec{c})$, and (ii) prove $\bullet$ under the assumption $Q!(\vec{c})$, provided $R(\vec{c})$ has already been visited as a proof goal and $\overline{S}(\vec{c})$ holds in the model. Axioms of type (5) are now repeatedly applied in every branch of the proof until $\bullet$ can be derived without creating a new proof task. This happens when an assumption of the form $R!(\vec{d})$ corresponds to a clause with no positive atom at the rhs. The next lemma makes this explicit.

For a model $\mathfrak{M}$, let $\Gamma_{\mathfrak{M}}$ be the union of $\mathfrak{M} \cup \overline{\mathfrak{M}}$ and the set of all axioms of φ.

Lemma 3.1
Let Δ consist of assumptions of the form $R!(\vec{c})$. Then $\Gamma_{\mathfrak{M}}, \Delta \vdash \bullet$ holds if and only if there is an atom $R!(\vec{c}) \in \Delta$ such that $\overline{\mathscr{P}} \cup \overline{\mathfrak{M}} \vdash R(\vec{c})$.

Proof
($\Rightarrow$) Induction wrt the size of a long normal proof (cf. Lemma 2.3). To prove $\bullet$ one must use an instance of one of the axioms (5), say of the form:

$$R!(\vec{a}) \to (P_1!(\vec{c}_1) \to \bullet) \to \cdots \to (P_r!(\vec{c}_r) \to \bullet) \to \overline{S}_1(\vec{d}_1) \to \cdots \to \overline{S}_t(\vec{d}_s) \to \bullet \qquad (*)$$

Then the atoms $R!(\vec{a}), \overline{S}_1(\vec{d}_1), \ldots, \overline{S}_s(\vec{d}_s)$ are provable from $\Gamma_{\mathfrak{M}}, \Delta$, and this can only happen when $R!(\vec{a}) \in \Delta$ and $\overline{S}_1(\vec{d}_1), \ldots, \overline{S}_s(\vec{d}_s) \in \overline{\mathfrak{M}}$, as there are no other assumptions with these targets.

In addition there are proofs of $\Gamma_{\mathfrak{M}}, \Delta, P_i!(\vec{c}_i) \vdash \bullet$, for $i = 1, \ldots, r$. By the ind. hypothesis, there are atoms $R_i!(\vec{e}_i) \in \Delta, P_i!(\vec{c}_i)$ such that $\overline{\mathscr{P}} \cup \overline{\mathfrak{M}} \vdash R_i(\vec{e}_i)$. Should any of these belong to Δ, we are done. Otherwise $R_i(\vec{e}_i) = P_i(\vec{c}_i)$, for all i. (This covers the "base case" when $r = 0$ and the induction hypothesis is not used.) Then we can derive $R(\vec{a})$ from the appropriate clause in $\overline{\mathscr{P}} \cup \overline{\mathfrak{M}}$, which is $R(\vec{a}) :- P_1(\vec{c}_1), \ldots, P_r(\vec{c}_r), \overline{S}_1(\vec{d}_1), \ldots, \overline{S}_s(\vec{d}_s)$.

($\Leftarrow$) Induction with respect to the size of a long normal proof. The proof must use a clause $R(\vec{a}) :- P_1(\vec{c}_1), \ldots, P_r(\vec{c}_r), \overline{S}_1(\vec{d}_1), \ldots, \overline{S}_s(\vec{d}_s)$ which corresponds to an axiom (5) instance of the form $(*)$. Then $\overline{S}_1(\vec{d}_1), \ldots, \overline{S}_s(\vec{d}_s) \in \overline{\mathfrak{M}}$ and $\overline{\mathscr{P}} \cup \overline{\mathfrak{M}} \vdash P_i(\vec{c}_i)$, for every $i = 1, \ldots, r$. By the induction hypothesis we have $\Gamma_{\mathfrak{M}}, \Delta, P_i!(\vec{c}_i) \vdash \bullet$, for all i. This enables us to apply $(*)$ to obtain a proof of $\bullet$ from $\Gamma_{\mathfrak{M}}, \Delta$. $\square$

The following is an easy consequence of Lemmas 2.2 and 3.1.

Lemma 3.2
We have $\Gamma_{\mathfrak{M}} \vdash A$ if and only if there is $R(\vec{c}) \in I(\mathscr{P}, \mathfrak{M})$ such that $\overline{R}(\vec{c}) \in \overline{\mathfrak{M}}$. That is, $\Gamma_{\mathfrak{M}} \vdash A$ if and only if $\mathscr{P}$ is unsound for $\mathfrak{M}$.

Proof

($\Rightarrow$) The only way to prove $\Gamma_{\mathfrak{M}} \vdash A$ is by applying one of the axioms (3), for some $R(\vec{c})$. Then $\overline{R}(\vec{c}) \in \Gamma_{\mathfrak{M}}$ and there must be a proof of $\Gamma_{\mathfrak{M}}, R!(\vec{c}) \vdash \bullet$. We apply Lemma 3.1 for the one element set $\Delta = \{R!(\vec{c})\}$.

($\Leftarrow$) Let $\mathscr{P}$ be unsound for $\mathfrak{M}$. There is an atom $R(\vec{c}) \in I(\mathscr{P}, \mathfrak{M})$ such that $\overline{R}(\vec{c}) \in \overline{\mathfrak{M}}$. Then we have $\overline{\mathscr{P}} \cup \overline{\mathfrak{M}} \vdash R(\vec{c})$ by Lemma 2.2 whence $\Gamma_{\mathfrak{M}}, R!(\vec{c}) \vdash \bullet$ by Lemma 3.1. To prove A one uses an instance of axiom (3). $\square$

The incomplete case B*:* The atom B can be derived with the help of (4):
$$\forall \vec{z}. R(\vec{z}) \to (R?(\vec{z}) \to \circ) \to B.$$
This axiom can be used when $R(\vec{c}) \in \mathfrak{M}$, for some $\vec{c}$, and it yields the proof task $\Gamma_{\mathfrak{M}}, R?(\vec{c}) \vdash \circ$. The meaning of the assumption $R?(\vec{c})$ is "we claim that $R(\vec{c})$ cannot be derived from $\overline{\mathscr{P}} \cup \overline{\mathfrak{M}}$". Now if $K_1, \ldots, K_l$ are all clauses with target $R(\vec{x})$ then we can use an instance of axiom (6)
$$R?(\vec{c}) \to (K_1^0(\vec{c}) \to \overline{K}_1^0) \to \cdots \to (K_l^0(\vec{c}) \to \overline{K}_l^0) \to \circ.$$
This results in l parallel proof obligations $\Gamma_{\mathfrak{M}}, R?(\vec{c}), K_i^0(\vec{c}) \vdash \overline{K}_i^0$, for all $i \leq l$. The intuitive understanding of the assumption $K_i^0(\vec{c})$ is "clause K_i cannot be used to derive $R?(\vec{c})$".

We need some care to handle the additional free variables in $K_1, \ldots, K_l$, and this is why we need axioms of type (9). For example, if clause K is one of $K_1, \ldots, K_l$, and it has free variables y_1, y_2, y_3 occurring only at the rhs, then we need three additional assumptions to extend the vector $\vec{z}$. So we take the following steps (recall that $c_1, \ldots, c_m$ are all the constants):
$$\forall \vec{z}. K^0(\vec{z}) \to (K^1(\vec{z}, c_1) \to \overline{K}^1) \to \cdots \to (K^1(\vec{z}, c_m) \to \overline{K}^1) \to \overline{K}^0$$
$$\forall \vec{z} y_1. K^1(\vec{z}, y_1) \to (K^2(\vec{z}, y_1, c_1) \to \overline{K}^2) \to \cdots \to (K^2(\vec{z}, y_1, c_m) \to \overline{K}^2) \to \overline{K}^1$$
$$\forall \vec{z} y_1 y_2. K^2(\vec{z}, y_1, y_2) \to (K^3(\vec{z}, y_1, y_2, c_1) \to \overline{K}^3) \to \cdots (K^3(\vec{z}, y_1, y_2, c_m) \to \overline{K}^3) \to \overline{K}^2$$

Note that with m elements of the domain we could "compress" the three example formulas into one with m^3 premises, but we cannot do it in general: the number of added variables in a clause can be proportional to the size of the program. We have avoided that using n formulas with m premises each rather than one formula with m^n premises.

The auxiliary formulas, targeted $\circ, \overline{K}^1, \overline{K}^2$, generate m^3 independent processes, each with a different "valuation" of y_1, y_2, y_3. For every such valuation one of the assumptions targeted $\overline{K}^3$ must be fired to guarantee that clause K cannot be used towards a proof of $R(\vec{x})$.

For example let K has the form $R(\vec{x}) :- P(\vec{w}), Q(\vec{v}), \neg S(\vec{v})$, where $\vec{x}$ are different variables. Then all variables $\vec{w}, \vec{v}, \vec{v}$ are among $\vec{x}, \vec{y}$, and we can use axioms of type (10) and (11):
$$\forall \vec{x} \vec{y}. K^3(\vec{x}, \vec{y}) \to (P?(\vec{w}) \to RP(\vec{x}, \vec{w}) \to \circ) \to \overline{K}^3$$
$$\forall \vec{x} \vec{y}. K^3(\vec{x}, \vec{y}) \to (Q?(\vec{v}) \to RQ(\vec{x}, \vec{v}) \to \circ) \to \overline{K}^3$$
$$\forall \vec{x} \vec{y}. K^3(\vec{x}, \vec{y}) \to S(\vec{v}) \to \overline{K}^3$$
Predicates denoted $RP(\vec{x}, \vec{w}), RQ(\vec{x}, \vec{v})$ "remember" about the transition from the goal $R?(\vec{x})$ to the goal $P?(\vec{w})$, respectively $Q?(\vec{v})$. The axiom (12) ensures that this memory is transitive, and axiom (13) terminates the proof construction when a loop in memory is revealed.

An additional source of complication is that the lhs of a clause can contain constants or repeated variables. Such a clause can therefore be used only towards matching goals, and this is handled by axioms (7) and (8). For example, if clause K begins with $R(x, y, x) :- \cdots$, and an assumption $K^0(a, b, c)$ was introduced with $a \neq c$, then the proof can be immediately completed using axiom (8) of the form $\forall z. K^0(a, z, c) \to \overline{K}^0$.

We now show that a proof of B is possible if and only if our program $\mathscr{P}$ is incomplete. We begin with the "if" part.

Refutations Fix an $\mathfrak{M}$. Let $\mathbf{a} \in B(\mathscr{P})$. A *refutation for* $\mathbf{a}$ is a possibly infinite tree labeled by members of $B(\mathscr{P})$ such that the root has label $\mathbf{a}$ and as many immediate subtrees as there are clauses in *ground*$(\mathscr{P})$ with target $\mathbf{a}$. For every such clause, the corresponding subtree

- either is a refutation for $\mathbf{b}$, for some positive atom $\mathbf{b}$ occurring at the rhs of the clause,
- or consists of a single node labeled $\overline{\mathbf{b}}$, where $\mathbf{b} \in \mathfrak{M}$ and $\neg\mathbf{b}$ occurs at the rhs of the clause.

Lemma 3.3
If $R(\vec{c}) \notin I(\mathscr{P}, \mathfrak{M})$ then there exists a refutation for $R(\vec{c})$.

Proof
Suppose $R(\vec{c}) \notin I(\mathscr{P}, \mathfrak{M})$. Then for any clause of *ground*$(\mathscr{P})$ of the form
$$R(\vec{c}) :- P_1(\vec{c}_1), \ldots, P_r(\vec{c}_r), \overline{S}_1(\vec{d}_1), \ldots, \overline{S}_s(\vec{d}_s),$$
either some $P_i(\vec{c}_i)$ is not in $I(\mathscr{P}, \mathfrak{M})$, or some $S_i(\vec{d}_i)$ is in $\mathfrak{M}$. A refutation is thus defined by co-induction. $\square$

Let D be a refutation. For any node v of D we define a set of atoms Δ_v by induction. If v is the root then $\Delta_v = \varnothing$. Otherwise v is an immediate successor (child) of a node μ. If the labels of μ and v are respectively $P(\vec{a})$ and $Q(\vec{b})$ then we take $\Delta_v = \Delta_\mu \cup \{PQ(\vec{a}\vec{b})\}$. One can say that Δ_v collects the history of transitions between atoms along the path to v. Observe that, as the distance from the root increases, this history must eventually contain a repetition.

Lemma 3.4
Let node v in a refutation D be labeled $R(\vec{c})$. Then $\Gamma_{\mathfrak{M}}, \Delta_v, R?(\vec{c}) \vdash \circ$.

Proof
Let $h(v)$ be 0 if all children of v are labeled by overlined atoms of the form $\overline{Q}(\vec{d})$ or some label occurs twice on the path to v. Otherwise take $h(v)$ to be one plus the maximum of $h(\pi)$, for all children π of v. The measure $h(v)$ is well-defined, because on any infinite path labels must repeat. The proof goes by induction on $h(v)$.

First suppose that an atom $P(\vec{c})$ occurs twice at the path leading to v. Then the set Δ_v contains a sequence of assumptions of the form $PQ(\vec{c}, \vec{d})$, $QS(\vec{d}, \vec{e})$, …, $TP(\vec{a}, \vec{c})$. By repeatedly using axioms (12) we reduce our proof obligation $\Gamma_{\mathfrak{M}}, \Delta_v, R?(\vec{c}) \vdash \circ$ to proving $\circ$ from an environment containing $PP(\vec{c}, \vec{c})$ and finally we apply axiom (13).

In the other case we prove $\circ$ using an axiom of type (6). This splits our proof into as many parallel ones as as there are clauses in $\mathscr{P}$ targeted $R(\vec{u})$, for some $\vec{u}$. For every such clause $K(\vec{x}, \vec{y})$ we need to prove the judgment $\Gamma_{\mathfrak{M}}, \Delta_v, R?(\vec{c}), K^0(\vec{c}) \vdash \overline{K}^0$. Here we have two cases, depending on whether the vector $\vec{u}$ can be instantiated to $\vec{c}$ or not. The latter may happen when $\vec{u}$ contains a constant or a repetition while $\vec{c}$ does not. But then an appropriate instance of axiom (7) or (8) can be applied to do the job. Otherwise we apply axioms (9) and this leads to further branching of the proof. At the end of each of the branches we have the judgments $\Gamma_{\mathfrak{M}}, \Delta_v, R?(\vec{c}), K^0(\vec{c}), \ldots, K^m(\vec{c}, \vec{d}) \vdash \overline{K}^m$, where $\vec{d}$ are all possible choices of constants. So now we have as many judgments to prove as there are immediate successors of v in D: each one corresponds to a clause $K(\vec{c}, \vec{d})$ in *ground*$(\mathscr{P})$ targeted $R(\vec{c})$.

Suppose that the successor π of v corresponding to $K(\vec{c}, \vec{d})$ is labeled $\overline{S}(\vec{e})$, where $S(\vec{e}) \in \mathfrak{M}$. Then $S(\vec{e}) \in \Gamma_{\mathfrak{M}}$ and we can derive $\overline{K}^m$ using axiom (11).

Otherwise the subtree of D rooted at π is a refutation of some $P(\vec{e})$. Now $h(\pi) < h(v)$ whence by the induction hypothesis we have a proof of the judgment $\Gamma_{\mathfrak{M}}, \Delta_v, RP(\vec{d}, \vec{e}), P?(\vec{e}) \vdash \circ$. It remains to call an instance of axiom (10). $\square$

Lemma 3.5

If $\mathscr{P}$ is incomplete for $\mathfrak{M}$ then $\Gamma_{\mathfrak{M}} \vdash B$.

Proof

There is an atom $R(\vec{c}) \in \mathfrak{M} - I(\mathscr{P}, \mathfrak{M})$. By Lemma 3.3 we have a refutation for $R(\vec{c})$. Apply Lemma 3.4 to the root to obtain $\Gamma_{\mathfrak{M}}, R?(\vec{c}) \vdash \circ$, and use axiom (4) to derive B. $\square$

We now prove that whenever B is provable, the program must be incomplete.[1]

Derivations from $\overline{\mathscr{P}} \cup \overline{\mathfrak{M}}$: Proofs of $\overline{\mathscr{P}} \cup \overline{\mathfrak{M}} \vdash \mathbf{a}$, for $\mathbf{a} \in D(\mathscr{P})$, (cf. Lemma 2.2) are of particularly simple shape (they do not contain any lambda-abstractions). Such proofs can be represented as trees labeled with clauses of $\overline{\mathscr{P}}$ and elements of $\overline{\mathfrak{M}}$. More precisely, we say that a finite tree T is a *derivation* for $R(\vec{e})$ when:

- The root of T is labeled $K(\vec{a}, \vec{b})$, where $K(\vec{a}, \vec{b})$ is a clause in $\overline{\mathscr{P}}$ of the form
$$R(\vec{e}) :- P_1(\vec{c}_1), \ldots, P_r(\vec{c}_r), \overline{S}_1(\vec{d}_1), \ldots, \overline{S}_s(\vec{d}_s). \qquad (*)$$
- The immediate subtrees of T are derivations of $P_1(\vec{c}_1), \ldots, P_r(\vec{c}_r)$ and single nodes labeled $\overline{S}_1(\vec{d}_1), \ldots, \overline{S}_s(\vec{d}_s) \in \mathfrak{M}$.

Let $K(\vec{a}, \vec{b})$ be of the form (*) and assume that $\vec{b} = b_1 \ldots b_d$. We say that the following atoms (where $i = 0, \ldots, d$ and $j = 1, \ldots, r$) are *justified* by clause $K(\vec{a}, \vec{b})$:
$$R?(\vec{e}), \qquad\qquad K^i(\vec{e}, b_1 \ldots b_i), \qquad\qquad RP_j(\vec{e}, \vec{c}_j).$$
If a node n of T is labeled by a clause which justifies an atom $\mathbf{a}$ then we also say that the atom $\mathbf{a}$ is *justified* by the node n. We say that T contains a *return* if there are two nodes in T such that one is a descendant of the other and both justify the same atom of the form $Q?(\vec{e})$. Clearly, a return can be "pumped out" so the following lemma is quite immediate.

Lemma 3.6

An atom $Q(\vec{e})$ belongs to $I(P, \mathfrak{M})$ if and only if there exists a derivation for $Q(\vec{e})$ without returns.

Justified environments: Let T be a derivation. An atom of the form $Q?(\vec{e})$ or of the form $K^i(\vec{e}, \vec{b})$ is *justified by T* when it is justified by some clause occurring as a label in T. An atom of the form $PQ(\vec{e}, \vec{c})$ is *justified by T* when it is either justified by some clause occurring as a label in T or there are atoms $PS(\vec{e}, \vec{d})$ and $SQ(\vec{d}, \vec{c})$ justified by T. (Informally, T justifies the transitive closure of the relation induced by atoms justified by labels of T.)

Now let Δ consist of atomic formulas of forms $PQ(\vec{e}, \vec{c})$, $P?(\vec{e})$ and $K^i(\vec{e}, \vec{b})$. We say that Δ is *justified by T* when all members of Δ are justified by T.

Lemma 3.7

Let T be a derivation without returns and let Δ be justified by T. Neither $\circ$ nor any of the logical constants $\overline{K}^i$ is provable from $\Gamma_{\mathfrak{M}}, \Delta$.

---

[1] Surprisingly, the natural conjecture that every proof of B yields a refutation (a correct refuter's winning strategy) is actually false. In a sense, the refuter can win by cheating, but she cannot cheat against a winning strategy of the prover.

Proof

Suppose otherwise, i.e., suppose that either $\circ$ or some $\overline{K}^i$ has a (long normal) proof M. By induction on the size of M we show that it is impossible. We proceed by cases depending on the head variable of M. As a first example, consider the case when the proof uses an assumption

$$\forall \vec{z}.\, R?(\vec{z}) \to (K_1^0(\vec{z}) \to \overline{K}_1^0) \to \cdots \to (K_l^0(\vec{z}) \to \overline{K}_l^0) \to \circ$$

of type (6) to derive $\circ$. Then, for some $\vec{e}$, we have $\Gamma_{\mathfrak{M}}, \Delta \vdash R?(\vec{e})$ and $\Gamma_{\mathfrak{M}}, \Delta, K_i^0(\vec{e}) \vdash \overline{K}_i^0$, for all i. There is no axiom where $R?$ occurs in the target, so it must be the case that $R?(\vec{e}) \in \Delta$. Thus $R?(\vec{e})$ is justified by some clause $K_i(\vec{a}, \vec{b})$ occurring in T. Therefore also $K_i^0(\vec{e})$ is justified by T (for this particular i), whence the environment $\Delta, K_i^0(\vec{e})$ is justified by T. This way we obtain a contradiction from the induction hypothesis, because the proof of $\Gamma_{\mathfrak{M}}, \Delta, K_i^0(\vec{e}) \vdash \overline{K}_1^0$ must be shorter than our proof of $\Gamma_{\mathfrak{M}}, \Delta \vdash \circ$.

The constant $\circ$ could be proven using the transitivity scheme (12):

$$\forall \vec{z}\vec{y}\vec{w}(RP(\vec{z}, \vec{y}) \to PQ(\vec{y}, \vec{w}) \to (RQ(\vec{z}, \vec{w}) \to \circ) \to \circ)$$

There are $\vec{e}, \vec{b}, \vec{c}$ such that $\Gamma_{\mathfrak{M}}, \Delta \vdash RP(\vec{e}, \vec{b})$ and $\Gamma_{\mathfrak{M}}, \Delta \vdash PQ(\vec{b}, \vec{c})$. This can only happen when $RP(\vec{e}, \vec{b})$ and $PQ(\vec{b}, \vec{c})$ are in Δ and therefore are justified by T. By definition, also $RQ(\vec{e}, \vec{c})$ is justified, and we apply induction to the proof of $\Gamma_{\mathfrak{M}}, \Delta, RQ(\vec{e}, \vec{c}) \vdash \circ$.

If $\circ$ were proven using the axiom $\forall \vec{x}(PP(\vec{x}, \vec{x}) \to \circ)$ of type (13) then $PP(\vec{e}, \vec{e}) \in \Delta$, for some $\vec{e}$. This is a contradiction because $PP(\vec{e}, \vec{e})$ cannot be justified in a derivation without returns.

Now suppose that a proof of $\Gamma_{\mathfrak{M}}, \Delta \vdash \overline{K}^i$ uses an axiom (9):

$$\forall \vec{z}\vec{v}.\, K^i(\vec{z}, \vec{v}) \to (K^{i+1}(\vec{z}, \vec{v}, c_1) \to \overline{K}^{i+1}) \to \cdots \to (K^{i+1}(\vec{z}, \vec{v}, c_m) \to \overline{K}^{i+1}) \to \overline{K}^i,$$

where $\vec{v}$ is a vector of variables of length i, and $c_1, \ldots, c_m$ are all the constants of $\mathscr{P}$. There are vectors $\vec{e}, \vec{d}$ of constants such that $\Gamma_{\mathfrak{M}}, \Delta \vdash K^i(\vec{e}, \vec{d})$ and $\Gamma_{\mathfrak{M}}, \Delta, K^{i+1}(\vec{e}, \vec{d}, c_j) \vdash \overline{K}^{i+1}$, for all j. We must have $K^i(\vec{e}, \vec{d}) \in \Delta$, as there is no other way to prove it, whence $K^i(\vec{e}, \vec{d})$ is justified by T. Therefore a clause of the form $K(\vec{a}, \vec{b})$ occurs in T, for some $\vec{a}$ and some vector $\vec{b}$ extending $\vec{d}c_j$. Then $K^{i+1}(\vec{e}, \vec{d}, c_j)$ is justified by T and we can apply the induction hypothesis to $\Gamma_{\mathfrak{M}}, \Delta, K^{i+1}(\vec{e}, \vec{d}, c_j) \vdash \overline{K}^{i+1}$. Other cases are omitted. $\square$

Lemma 3.8

If $\Gamma_{\mathfrak{M}} \vdash B$ then $\mathscr{P}$ is unsound for $\mathfrak{M}$: there exists an atom $R(\vec{e})$ such that $R(\vec{e}) \in \mathfrak{M}$ but $R(\vec{e}) \notin I(P, \mathfrak{M})$.

Proof

If $R(\vec{e}) \in I(P, \mathfrak{M})$ then any derivation for $R(\vec{e})$ without returns justifies $R?(\vec{x})$. Therefore a proof of $\Gamma_{\mathfrak{M}}, R?(\vec{e}) \vdash \circ$ is impossible by Lemma 3.7. $\square$

Proposition 1

The formula φ is provable if and only if $\mathscr{P} \models_{\mathrm{SMS}} \Omega$.

4 Refutations into programs

Now we describe a translation in the direction opposite to that discussed in Section 3, namely from bounded-arity intuitionistic formulas to answer set programs. The title of this section alludes to the famous slogan "proofs into programs", because our translation demonstrates that ASP, as a programming paradigm, corresponds to logic understood from the refuter's, rather than prover's, point of view.

Let φ be a Σ_1 formula with atomic predicates of arity at most r. We assume that φ is written so that no free variable is bound and no variable is bound twice. From now on the variables free in φ are called *constants*, the word *variable* being reserved for the bound ones. The latter may of course become free in certain subformulas of φ.

Let n be the length of the formula φ. The number of constants and variables in φ cannot exceed n so let us assume that all the constants in φ are among $c_1, \ldots, c_n$, and that all bound variables are among $x_1, \ldots, x_n$. Observe that the number of all atoms over $c_1, \ldots, c_n, x_1, \ldots, x_n$ is polynomial in n.

A sequence of constants of length n is called a *substitution*. We write $\psi[S]$ for the result of applying a substitution $S = c_{i_1} \ldots c_{i_n}$ to a formula ψ, i.e., substituting c_{i_j} for x_j, whenever x_j is free in ψ.

Let Γ be a set of Π_1 formulas of the form $\psi[S]$, where ψ is a subformula of φ, and S, T be substitutions. A statement of the form $\Gamma \nvdash \mathbf{a}$, where $\mathbf{a}$ is a closed atom, is called a *disjudgment*. A triple of the form $\langle \psi, S, T \rangle$ is called a *question* asked at $\Gamma \nvdash \mathbf{a}$ when the formula $\psi[S]$ belongs to Γ and the target of $\psi[S][T]$ coincides with $\mathbf{a}$.

Then $\psi = \forall \vec{y}_1 (\sigma_1 \to \forall \vec{y}_2 (\sigma_2 \to \cdots \to \forall \vec{y}_k (\sigma_k \to \mathbf{b}) \ldots))$, where $\sigma_i \in \Sigma_1$, and $\mathbf{b}$ is an atom. A question $\langle \psi, S, T \rangle$ represents a possible proof attempt (an attempt to construct a term in long normal form) with a variable of type $\psi[S]$ as a head variable and with $\vec{y}_1, \ldots, \vec{y}_k$ instantiated by T. The variables $\vec{y}_1, \ldots, \vec{y}_k$ are called the *top variables of* ψ. Every σ_i is a Σ_1 formula of shape $\tau_1 \to \tau_2 \to \cdots \to \tau_q \to \mathbf{c}_i$, where $\tau_j \in \Pi_1$, and $\mathbf{c}$ is an atom. The atom $\mathbf{c}_i$ is then called the *i*-th *subgoal* in ψ and the formulas $\tau_1, \ldots, \tau_q$ are *i-th descendants* of ψ. Note that top variables of ψ may occur free in the target atom of ψ, the *i*-th descendants and subgoals of ψ. More precisely, we have $\mathrm{FV}(\mathbf{b}) \cup \mathrm{FV}(\tau_j) \cup \mathrm{FV}(\mathbf{c}_i) \subseteq \mathrm{FV}(\psi) \cup \vec{y}_1 \cup \cdots \cup \vec{y}_i$.

An *i-th answer* to the question $\langle \psi, S, T \rangle$ as above is any disjudgment of the form $\Gamma' \nvdash \mathbf{c}_i[S][T]$ such that $\Gamma, \tau_1[S][T], \ldots, \tau_q[S][T] \subseteq \Gamma'$. A question *is answered* in a set $\mathscr{Z}$ when it has an *i*-th answer in $\mathscr{Z}$, for some i.

The intuition to be associated with an *i*-th answer is that the question (proof attempt) $\langle \psi, S, T \rangle$ is challenged at the *i*-th argument: the prover is expected to fail to prove the formula $\sigma_i[S][T]$.

Finally we define a *refutation soup* for φ, for brevity called *soup*, as a set $\mathscr{Z}$ of disjudgments such that $\varnothing \nvdash \varphi$ belongs to $\mathscr{Z}$ and every question asked at any $(\Gamma \nvdash \mathbf{a}) \in \mathscr{Z}$ is answered in $\mathscr{Z}$. It is shown in (Schubert et al. 2016) that:

- If there is a soup for φ then $\nvdash \varphi$.
- If $\nvdash \varphi$ then there is a soup of size at most 2^{n^r}, where r is the maximum arity of the predicates in φ.

We now construct a program $\mathscr{P}$ such that $\mathscr{P}$ has a stable model if and only if there exists a soup for φ of size 2^{n^r}.

The domain $D_{\mathscr{P}}$ of $\mathscr{P}$ consists of all (occurrences of) subformulas of φ, all the constants $c_1, \ldots, c_n$, and two additional constants $\mathbf{0}$ and $\mathbf{1}$. The size of $D_{\mathscr{P}}$ is clearly polynomial in n. Note that we count different occurrences of the same subformula as different objects.

Every disjudgment in a soup of size 2^{n^r} can be identified by a sequence of $\mathbf{0}$s and $\mathbf{1}$s of length n^r, called an *address*. One can think of a soup as of a set of triples of the form $\xi : \Gamma \nvdash \mathbf{a}$, where ξ is the address of the disjudgment $\Gamma \nvdash \mathbf{a}$.

The vocabulary of $\mathscr{P}$: The atomic predicates occurring in $\mathscr{P}$ are written using some abbreviations. We use $\xi, \eta, \ldots$ as metavariables for addresses, i.e., $\xi, \eta, \ldots$ are sequences of length n^r,

intended to be instantiated by **0**s and **1**s. We also write $S, T, \ldots$ for "substitutions", i.e., sequences of length n to be instantiated by constants of φ.

The program $\mathscr{P}$ will use the following predicates. Each of them is accompanied below by its intuitive meaning.

- $\text{Dsc}_i(\tau, \psi, S, T, U)$, "the formula τ is an i-th **descendant** of ψ, and $\tau[U]$ is obtained from $\tau[S]$ by applying T to the top variables of ψ";
- $\text{Sbg}_{i,\mathbf{a}}(\psi, S, T)$, "the atom $\mathbf{a}$ is the result of applying T to the i-th **subgoal** in $\psi[S]$";
- $\text{Hd}_{\mathbf{a}}(\psi, S, T)$, "the atom $\mathbf{a}$ is the result of applying T to the **head** atom of $\psi[S]$";
- $\text{Gol}_{\mathbf{a}}(\xi)$, "the atom $\mathbf{a}$ is the **goal** at the address ξ";
- $\text{Env}(\psi, S, \xi)$, "the instance $\psi[S]$ of ψ occurs in the assumption **environment** at ξ";
- $\overline{\text{Env}}(\psi, S, \xi)$, "the above does not hold";
- $\text{Qst}(\psi, S, T, \xi)$, "the triple $\langle \psi, S, T \rangle$ is a **question** at ξ";
- $\text{Ans}_i(\psi, S, T, \xi, \eta)$, "the question $\langle \psi, S, T \rangle$ at ξ has an i-th **answer** at η";
- $\overline{\text{Ans}}_i(\psi, S, T, \xi, \eta)$, "the above does not hold";
- F, an auxiliary "false" for the contradiction clauses (11) and (14);
- $\text{Y}(\psi, S, T, \xi)$, an auxiliary predicate for clauses (14–15).

The clauses of $\mathscr{P}$: We begin with the most obvious clauses determined by the syntax of φ. There is a number of facts of the form

1. $\text{Dsc}_i(\tau, \psi, S, T, U) :-$;
2. $\text{Sbg}_{i,\mathbf{a}}(\psi, S, T) :-$;
3. $\text{Hd}_{\mathbf{a}}(\psi, S, T) :-$,

where τ, ψ, S, T, U, are concrete formulas and substitutions, and $\mathbf{a}$ is any atom. For example, if $n = 4$ and $\psi = \forall y_1(\text{R}(y_1, c_2) \to \forall y_2(\text{P}(y_1, c_1) \to \text{R}(c_1, y_2, y_3)))$ then $\mathscr{P}$ contains all clauses of the form $\text{Hd}_{\text{R}(c_1, c_4, c_2)}(\psi, *, *, c_2, *, *, c_4, *, *) :-$, where every asterisk can be replaced by any constant. This clause can be written as $\text{Hd}_{\text{R}(c_1, c_4, c_2)}(\psi, S, T) :-$, where $S = (*, *, c_2, *)$ and $T = (*, c_4, *, *)$. Let $\psi = \sigma \to \forall y_3((\cdots \to \tau \to \cdots \to \mathbf{a}) \to \forall y_1(\sigma' \to \mathbf{b}))$ be another example, where y_4 is free and $\tau = \forall y_2(\text{P}(y_2, y_1) \to \text{R}(y_3, y_4))$. Then $\mathscr{P}$ contains e.g., all clauses of the form $\text{Dsc}_2(\tau, \psi, S, T, U) :-$, for $S = (*, *, *, d)$, $T = (*, *, e, *)$, and $U = (*, *, e, d)$, where the asterisks can be replaced by anything. This is because variables y_1 and y_2 are bound in both ψ and τ, and therefore ignored by the substitutions. There is a lot of similar clauses but still only a polynomial number of such clauses. The remaining clauses determine the shape of the model which is supposed to represent a soup. We begin with a few facts describing the initial judgment at address $\mathbf{00} \ldots \mathbf{0}$. If $\varphi = \psi_1 \to \cdots \to \psi_n \to \mathbf{a}$ then the following clauses are in $\mathscr{P}$:

4. $\text{Gol}_{\mathbf{a}}(\mathbf{00} \ldots \mathbf{0}) :-$;
5. $\text{Env}(\psi_i, S, \mathbf{00} \ldots \mathbf{0}) :-$;
6. $\overline{\text{Env}}(\psi, S, \mathbf{00} \ldots \mathbf{0}) :-$,

where $i = 1, \ldots, n$, $\psi \notin \{\psi_1, \ldots, \psi_n\}$, and S is an arbitrary substitution (note that ψ_i have no free variables, so S does not matter).

The following clauses guarantee that any answer given in the model is correct with respect to the question it responds to.

7. $\text{Env}(\tau, U, \eta) :- \text{Ans}_i(\psi, S, T, \xi, \eta), \text{Env}(\tau, U, \xi)$;
8. $\text{Env}(\tau, U, \eta) :- \text{Ans}_i(\psi, S, T, \xi, \eta), \text{Dsc}_i(\tau, \psi, S, T, U)$;

9. $\mathrm{Gol}_{\mathbf{a}}(\eta) :- \mathrm{Ans}_i(\psi, S, T, \xi, \eta), \mathrm{Sbg}_{i,\mathbf{a}}(\psi, S, T).$

The last one of the three above clauses defines the goal at η, while the first two list the necessary assumptions at η. But the definition of a soup allows any other formula to occur as an assumption in the answer judgment. This must be reflected by any stable model. For this purpose we add to $\mathscr{P}$ the three clauses (using the special symbol F):

10. $\mathrm{Env}(\psi, S, \xi) :- \neg\overline{\mathrm{Env}}(\psi, S, \xi), \quad \overline{\mathrm{Env}}(\psi, S, \xi) :- \neg\mathrm{Env}(\psi, S, \xi),$
11. $\mathrm{F} :- \mathrm{Env}(\psi, S, \xi), \overline{\mathrm{Env}}(\psi, S, \xi), \neg\mathrm{F},$

which force that either $\mathrm{Env}(\psi, S, \xi)$ or $\overline{\mathrm{Env}}(\psi, S, \xi)$ must hold, but not both.

A similar measure is also applied for the predicates Ans_i (so far only occurring at the rhs). That is, $\mathscr{P}$ contains (for $i = 1, \ldots n$) the constructs:

12. $\mathrm{Ans}_i(\psi, S, T, \xi, \eta) :- \neg\overline{\mathrm{Ans}}_i(\psi, S, T, \xi, \eta), \mathrm{Qst}(\psi, S, T, \xi),$
    $\overline{\mathrm{Ans}}_i(\psi, S, T, \xi, \eta) :- \neg\mathrm{Ans}_i(\psi, S, T, \xi, \eta), \mathrm{Qst}(\psi, S, T, \xi).$

The next clauses (one for each **a**) define the notion of a question.

13. $\mathrm{Qst}(\psi, S, T, \xi) :- \mathrm{Env}(\psi, S, \xi), \mathrm{Hd}_{\mathbf{a}}(\psi, S, T), \mathrm{Gol}_{\mathbf{a}}(\xi).$

Clauses 12 permit an arbitrary choice between predicates $\mathrm{Ans}_i(\ldots)$ and $\overline{\mathrm{Ans}}_i(\ldots)$. We need additional clauses to ensure that this choice makes sense, i.e., that every question has an answer. For this purpose $\mathscr{P}$ includes clauses (where $i \leq n$, and Y does not occur elsewhere).

14. $\mathrm{F} :- \neg\mathrm{Y}(\psi, S, T, \xi), \mathrm{Qst}(\psi, S, T, \xi), \neg\mathrm{F};$
15. $\mathrm{Y}(\psi, S, T, \xi) :- \mathrm{Ans}_i(\psi, S, T, \xi, \eta).$

Clauses (14–15) together enforce that any stable model of the program must satisfy the following property: If $\mathrm{Qst}(\psi, S, T, \xi) \in \mathfrak{M}$ then $\mathrm{Ans}_i(\psi, S, T, \xi, \eta) \in \mathfrak{M}$, for some i and η. Indeed, in a stable model it is impossible to have $\mathrm{Qst}(\psi, S, T, \xi)$ without $\mathrm{Y}(\psi, S, T, \xi)$. And $\mathrm{Y}(\psi, S, T, \xi)$ can only be derived if $\mathrm{Ans}_i(\psi, S, T, \xi, \eta)$ holds for some i and η.[2]

At the end we need to guarantee that no judgment can address two goals. This is handled by one clause for any pair **a** and **b** of distinct atoms.

16. $:- \mathrm{Gol}_{\mathbf{a}}(\xi), \mathrm{Gol}_{\mathbf{b}}(\xi).$

Cooking a soup from a model We now show that every stable model $\mathfrak{M}$ of $\mathscr{P}$ defines a soup for φ. Clauses (10) and (11) force $\mathfrak{M}$ to assign an environment Γ_ξ to every address ξ. This Γ_ξ is uniquely defined. Indeed, for each formula $\psi[S]$ either $\mathrm{Env}(\psi, S, \xi) \in \mathfrak{M}$ or $\overline{\mathrm{Env}}(\psi, S, \xi) \in \mathfrak{M}$ by clause (10), and not both of them by clause (11). (Note that (10) alone are not enough, because Env and $\overline{\mathrm{Env}}$ may occur at the lhs of other clauses.)

Lemma 4.1
Let $\mathscr{Z}$ consists of all disjudgments $\xi : \Gamma_\xi \nvdash \mathbf{a}$ such that $\mathrm{Gol}_{\mathbf{a}}(\xi) \in \mathfrak{M}$. Then $\mathscr{Z}$ is a soup.

---

[2] This is an example of a more general pattern: clauses $\mathrm{Y}(\vec{y}) :- \mathrm{P}(\vec{x}, \vec{y})$ and $:- \neg\mathrm{A}(\vec{y})$, where Y is fresh, enforce that any stable model must satisfy $\forall\vec{y}\exists\vec{x}P(\vec{x}, \vec{y})$.

Proof
Clauses (4–6) ensure that the initial judgment is in $\mathscr{Z}$. In addition, whenever $\text{Gol}_{\mathbf{a}}(\xi) \in \mathfrak{M}$, and the triple $\langle \psi, S, T \rangle$ is a question at ξ, then this question has an answer in $\mathscr{Z}$. Indeed, clause (13) yields $\text{Qst}(\psi, S, T, \xi) \in \mathfrak{M}$, and this implies that either $\text{Ans}_i(\psi, S, T, \xi, \eta) \in \mathfrak{M}$ or $\overline{\text{Ans}}_i(\psi, S, T, \xi, \eta) \in \mathfrak{M}$ by clause (12). Because of (14) we must have $\text{Y}(\psi, S, T, \xi) \in \mathfrak{M}$ and since $\mathfrak{M}$ is stable, one of clauses (15) must have been fired, i.e., at least one $\text{Ans}_i(\psi, S, T, \xi, \eta)$ is in the model. It remains to observe the effect of clauses (7–9): the disjudgment at η is indeed an answer to the question $\langle \psi, S, T \rangle$ asked at ξ. $\square$

Boiling out a model from a soup Suppose $\mathscr{Z}$ is a soup without unnecessary ingredients, that is every judgment in $\mathscr{Z}$, except the initial judgment, is an answer to some question posed in $\mathscr{Z}$. Assume also that every judgment in the soup is given an address, with the initial judgment having the address $\mathbf{00} \ldots \mathbf{0}$. We require that any given address uniquely determines a judgment, but one judgment may have several addresses. Repetitions are permitted, because it is technically convenient that all addresses of a fixed length are in use (do actually refer to some judgments).

   We say that the address $\mathbf{00} \ldots \mathbf{0}$ has *depth* zero. Otherwise the depth of an address is 1 plus the minimal depth of a question it answers. This definition is extended to closed atoms as follows: the *depth* of a closed atom, where a single address occurs (like e.g., $\text{Env}(\psi, S, \xi)$) is the depth of that address, and in case of $\text{Ans}_i(\psi, S, T, \xi, \eta)$ it is the depth of the first address ξ.

   We define a model $\mathfrak{M}$ (recall that a model is just a set of ground atoms). The elements of $\mathfrak{M}$ are chosen as follows:

- Atoms $\text{Dsc}_i(\tau, \psi, S, T, U)$, $\text{Sbg}_{i,\mathbf{a}}(\psi, S, T)$, $\text{Hd}_{\mathbf{a}}(\psi, S, T)$ are selected according to the syntax of the main formula φ.
- Atoms $\text{Gol}_{\mathbf{a}}(\xi)$, $\text{Env}(\psi, S, \xi)$, $\overline{\text{Env}}(\psi, S, \xi)$ are selected according to the shape of judgments.
- Atoms $\text{Qst}(\psi, S, T, \xi)$, $\text{Ans}_i(\psi, S, T, \xi, \eta)$, and $\overline{\text{Ans}}_i(\psi, S, T, \xi, \eta)$ are selected according to the structure of questions and answers in the soup. But if $\langle \psi, S, T \rangle$ does not form a question at address ξ then neither $\text{Ans}_i(\psi, S, T, \xi, \eta)$ nor $\overline{\text{Ans}}_i(\psi, S, T, \xi, \eta)$ is in the model.
- Atom F is not selected.
- The atom $\text{Y}(\psi, S, T, \xi)$ is selected whenever at least one of $\text{Ans}_i(\psi, S, T, \xi, \eta)$ is selected.

Lemma 4.2
The model $\mathfrak{M}$ is a stable model of $\mathscr{P}$.

Proof
One must check that all the selected atoms are derivable, i.e., belong to $I(\mathscr{P}, \mathfrak{M})$. For the syntax-related atoms it is obvious by the choice of facts (1–3). In all other atoms there are occurrences of addresses, and one proceeds by induction with respect to depth. Suppose all atoms in $\mathfrak{M}$ of depth smaller than d are derivable, and let the depth of ξ be d. Consider the goal $\mathbf{a}$ at the address ξ. If $d = 0$ then $\text{Gol}_{\mathbf{a}}(\mathbf{00} \ldots \mathbf{0})$ is derived from clause (4). Otherwise ξ is an address of an answer to a question of depth at most $d - 1$. Thus we can instantiate clause (9) as $\text{Gol}_{\mathbf{a}}(\xi) :- \text{Ans}_i(\psi, S, T, \zeta, \xi), \text{Sbg}_{i,\mathbf{a}}(\psi, S, T)$, where ζ is of depth at most $d - 1$, and by the induction hypothesis $\text{Ans}_i(\psi, S, T, \zeta, \xi) \in I(\mathscr{P}, \mathfrak{M})$. This derives $\text{Gol}_{\mathbf{a}}(\xi)$ which is then used to derive atoms of the form $\text{Qst}(\psi, S, T, \xi)$ from (13).[3] Given $\text{Qst}(\psi, S, T, \xi)$, clause (12) yields

---

[3] This is the only way to derive $\text{Qst}(\psi, S, T, \xi)$, so it can only happen when the necessary atoms are in $\mathfrak{M}$, i.e., the question in question does actually occur in the soup.

either $\text{Ans}_i(\psi, S, T, \xi, \eta)$ or $\overline{\text{Ans}}_i(\psi, S, T, \xi, \eta)$, for any η, depending on whether an appropriate answer occurs at η or not.

The proof that no unwanted atom is derivable goes by induction with respect to derivations in $\mathscr{P}^{\mathfrak{M}}$. First observe that clauses (1–6) only introduce atoms that belong to the model $\mathfrak{M}$. Then note that in each pair of clauses (10) and (12) one of the negated atoms is already in the model. As the corresponding clause gets erased, it is impossible to use (10) or (12) to derive both the underlined and non-underlined version of the same atom. In particular, clause (11) will not derive F. Suppose an atom $\text{Env}(\tau, U, \eta)$ is derived by clause (7). Then both $\text{Ans}_i(\psi, S, T, \xi, \eta)$ and $\text{Env}(\tau, U, \xi)$ must have been derived before, and therefore belong to $\mathfrak{M}$. It follows that η provides an i-th answer to ξ in our soup, whence $\text{Env}(\tau, U, \eta) \in \mathfrak{M}$ as well. Similar argument applies to atom derived using clauses (8,9,13,15). $\quad\square$

We can now conclude with the following

Proposition 2
The program $\mathscr{P}$ has a stable model if and only if φ is refutable.

Corollary 3
SMS entailment and monadic Σ_1 provability formulas are mutually polynomial time translatable.

Conclusion

In this paper we demonstrated a mutual polynomial time translation between first-order Answer Set Programming and the bounded arity Σ_1 fragment of intuitionistic predicate logic. One of the main motivations for this work was to give an alternative point of view on ASP, namely the constructive interpretation. Constructive proofs serve to derive conclusions from assumptions, very much as logic programs derive goals, therefore we find it natural to work in the context of the ASP entailment problem. Especially because existence of stable models is nothing else than non-entailment.

While our translations may seem complicated on the first look, let us point out that they are actually quite natural. The format of the object (target) language is very simple, in particular very similar to the ASP source language. All the assumptions in the resulting formul have the form of straightforward rules where each premise is either an atom or a pseudo-negated atom of the form $\mathbf{a} \to \star$. With such rules as axioms, proof search can be seen as an algorithm which manipulates a database by expanding and querying it. The translation therefore not only yields the known complexity bound but also delineates the spectrum of adequate proof tactics.

Our work gives a complementary sight on the explanation of logic programming in terms of Howard's system **H** introduced by Fu and Komendantskaya (Fu and Komendantskaya 2017). The system interprets Horn formulas as types, and derivations for a given formula as the proof terms inhabiting the type corresponding to the formula. The authors demonstrate that different forms of resolution can be expressed in this system and relate them. In our work the operation of resolution is limited since we work with constants only. Instead, our work concentrates on the operational semantics of negation. Indeed, our translation, due to the limited number of available formula targets, has strong flavour of countinuation-passing style transformation.

Logic programming with negation undestood as in ASP has its explanation in terms of equilibrium logic (Pearce 1999; Pearce 2006). This logic is built on the foundations of intuitionistic

theory called *here and there logic*, which can be axiomatized in propositional case by the scheme

$$(\neg\alpha \to \beta) \to \varphi_P(\alpha, \beta),$$

where $\varphi_P(\alpha, \beta) = ((\beta \to \alpha) \to \beta) \to \beta$ is the Peirce's law. It is interesting to see that proof construction techniques that occur when the result of our translation is to be proved are very similar to ones used in case of *here and there logic*.

While the equilibrium logic relies on a property of a model to settle the space to represent a coherent database of inferrable basic statements, in the case of our logic, the space is located in the proof environment that evolves during the proof construction process.

Acknowledgement

References

BREWKA, G., EITER, T., AND TRUSZCZYŃSKI, M. 2011. Answer set programming at a glance. *Commun. ACM 54,* 12 (Dec.), 92–103.

DANTSIN, E., EITER, T., GOTTLOB, G., AND VORONKOV, A. 2001. Complexity and expressive power of logic programming. *ACM Comput. Surv. 33,* 3, 374–425.

FU, P. AND KOMENDANTSKAYA, E. 2017. Operational semantics of resolution and productivity in Horn clause logic. *Formal Asp. Comput. 29,* 3, 453–474.

KOLAITIS, P. G. AND PAPADIMITRIOU, C. H. 1988. Why not negation by fixpoint? In *Proceedings of the Seventh ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems. PODS'88.* ACM, New York, NY, USA, 231–239.

LIFSCHITZ, V. 2008. What is answer set programming? In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3.* AAAI'08. AAAI Press, 1594–1597.

MAREK, W. AND TRUSZCZYŃSKI, M. 1991. Autoepistemic logic. *J. ACM 38,* 3 (July), 587–618.

NICKAU, H. 1994. Hereditarily sequential functionals. In *Logical Foundations of Computer Science,* A. Nerode and Y. Matiyasevich, Eds. Lecture Notes in Computer Science, vol. 813. Springer, 253–264.

OSORIO, M., NAVARRO, J. A., AND ARRAZOLA, J. 2004. Applications of intuitionistic logic in answer set programming. *Theory Pract. Log. Program. 4,* 3 (May), 325–354.

PEARCE, D. 1999. Stable inference as intuitionistic validity. *The Journal of Logic Programming 38,* 1, 79–91.

PEARCE, D. 2006. Equilibrium logic. *Annals of Mathematics and Artificial Intelligence 47,* 1-2, 3–41.

SCHUBERT, A. AND URZYCZYN, P. 2018. Answer set programming in intuitionistic logic. *Indagationes Mathematicae 29,* 1, 276–292. L.E.J. Brouwer, fifty years later.

SCHUBERT, A., URZYCZYN, P., AND WALUKIEWICZ-CHRZĄSZCZ, D. 2015. Restricted positive quantification is not elementary. In *Proc. TYPES 2014,* H. Herbelin, P. Letouzey, and M. Sozeau, Eds. LIPIcs, vol. 39. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 251–273.

SCHUBERT, A., URZYCZYN, P., AND ZDANOWSKI, K. 2016. On the Mints hierarchy in first-order intuitionistic logic. *Logical Methods in Computer Science 12,* 4.

SØRENSEN, M. H. AND URZYCZYN, P. 2006. *Lectures on the Curry-Howard Isomorphism.* Studies in Logic and the Foundations of Mathematics, vol. 149. Elsevier.

URZYCZYN, P. 2016. Intuitionistic games: Determinacy, completeness, and normalization. *Studia Logica 104,* 5, 957–1001.