Nested Expressions in Logic Programs

Vladimir Lifschitz and Lappoon R. Tang

Department of Computer Science
University of Texas at Austin
Austin, TX 78712, USA

Hudson Turner

Department of Computer Science
University of Minnesota at Duluth
Duluth, MN 55812, USA

November 9, 2001

Goal

The goal of this presentation is to present the new syntax and semantics introduced in the paper, and to show how the new semantics fit in with our conception of answer sets.

Introduction

The authors extend the answer set semantics to include programs with nested expressions permitted in the heads and the bodies of rules.

Structure

The talk is structured as follows:

- review of the current syntax and semantics of logic programs
- introduce the expansion of the syntax and semantics to include nested expressions
- show how the new semantics relates to our current concept of answer sets
- discuss the relationship between the new semantics and the Lloyd-Topor semantics

Current Syntax

As of now, we define a logic program as a set of rules of the form:

$$L_0 \leftarrow L_1, \ldots, L_m, \mathit{not}\ L_{m+1}, \ldots, \mathit{not}\ L_n$$

where the L's are literals. Rules of this form are read as follows: "L_0 is true if we know $L_1, \ldots, L_m$ to be true and we have *no reason to believe* in $L_{m+1}, \ldots, L_n$". Programs whose rules do not contain negation as failure are known as basic programs.

Current Semantics

The semantics of a logic program is defined in terms of answer sets as follows:

1. The answer set of a basic program Π, is a minimal set S of ground literals which satisfies the following two conditions:
   - S is closed under the rules of Π
   - If S contains contrary literals, then the answer set of Π is equal to the set of all ground literals.

2. Otherwise, given a set of literals S, we obtain the basic program Π^S from Π by:
   - removing all rules in Π containing $not\, f$, where $f \in S$
   - removing all other premises containing not

S is an answer set of Π, if and only if S is an answer set of Π^S

Syntax of Formulas

The authors define *elementary formulas* as literals, and the connectives $\top$ ("true"), and $\bot$ ("false").

Elementary formulas, are *formulas*. The negation as failure (*not*), of a formula is a formula, and conjunctions ',', and disjunctions ';' of formulas are themselves formulas.

Syntax of Conditional Expressions

For any formulas F, G, and H, we write the formula $(F, G); (not\ \ F, H)$ as follows:

$$F \to G; H$$

Formulas of this form are read as follows: "if F then G, else H".

Examples

The following are all valid formulas under the new syntax:

- (F, G)
- $(F; G)$
- $not(F, G)$
- $not(not(F, G); (H \rightarrow I; J))$

Syntax of Programs

A program is redefined as a set of rules of the form:

$$Head \leftarrow Body$$

where $Head$ and $Body$ are formulas.

Rules of the form $F \leftarrow \top$ are known as *facts* and are written as $F \leftarrow$.

Rules of the form $\bot \leftarrow G$ are known as *constraints* and are written as $\leftarrow G$.

Formulas, rules and program that do not contain negation as failure are known as *basic*.

New Semantics

The authors define when a consistent set X of literals *satisfies* a basic formula F, denoted as $X \models F$, recursively as follows:

- for elementary formula F, $X \models F$ if $F \in X$ or $F = \top$
- $X \models (F, G)$ if $X \models F$ and $X \models G$
- $X \models (F; G)$ if $X \models F$ or $X \models G$

Now let Π be a basic program. A consistent set X of literals is *closed* under Π if, for every rule $(F \leftarrow G) \in \Pi$, $X \models F$ whenever $X \models G$.

X is an answer set for Π, if X is minimal among the consistent sets of literals closed under Π.

Examples

1. Consider the basic program $q \leftarrow (p; \neg p)$

   As there are no rules with p or $\neg p$ in their heads, it is clear that the only answer set of this program is $\emptyset$.

2. Let us add the rule $p \leftarrow$ to the above program.

   $q \leftarrow (p; \neg p)$
   $p \leftarrow$

   It is clear that the answer set of the program is $\{p, q\}$.

The *reduct* of a formula, rule or program with respect to a consistent set of literals X is defined recursively, as follows:

- for an elementary F, $F^X = F$
- $(F, G)^X = (F^X, G^X)$
- $(F; G)^X = (F^X; G^X)$
- $(not\ F)^X = \begin{cases} \bot \text{ if } X \models F^X \\ \top \text{ otherwise} \end{cases}$
- $(F \leftarrow G)^X = F^X \leftarrow G^X$
- $\Pi^X = \{(F \leftarrow G)^X : F \leftarrow G \in \Pi\}$

X is an answer set of Π, if and only if X is an answer set of Π^X.

Examples

Let us consider the program Π:

$$p \leftarrow (q \rightarrow r; not\ s).$$

equivalently written as:

$$p \leftarrow (q, r); (not\ q, not\ s).$$

Take $X = \{p\}$. We compute Π^X as follows:

$$\begin{aligned}
\Pi^X &= p^X \leftarrow ((q, r); (not\ q, not\ s))^X \\
&= p \leftarrow (q, r)^X; (not\ q, not\ s)^X \\
&= p \leftarrow (q^X, r^X); ((not\ q)^X, (not\ s)^X) \\
&= p \leftarrow (q, r); (\top, \top)
\end{aligned}$$

It is clear that the only answer set of Π^X is X, and hence X is an answer set of Π.

Curiously, under the expanded semantics two consecutive negation as failure operators do not cancel each other out. Let us consider the program Π:

$$p \leftarrow not\ not\ p.$$

Take $X_1 = \emptyset$. We compute Π^{X_1} as follows:

$$\begin{aligned} \Pi^X &= p^{X_1} \leftarrow (not\ not\ p)^{X_1} \\ &= p \leftarrow (not\ \top) \\ &= p \leftarrow \bot \end{aligned}$$

It is clear that the only answer set of Π^{X_1} is $\emptyset$, and hence X_1 is an answer set of Π.

Now take $X_2 = \{p\}$. We compute Π^{X_2} as follows:

$$\begin{aligned}\Pi^X &= p^{X_2} \leftarrow (not\ not\ p)^{X_2} \\ &= p \leftarrow (not\ \bot) \\ &= p \leftarrow \top\end{aligned}$$

It is clear that the only answer set of Π^{X_2} is $\{p\}$, and hence X_2 is an answer set of Π.

Relation to Current Semantics

Proposition 1 *For a program whose rules have the form:*

$$L_1; \ldots; L_k; not\ L_{k+1}; \ldots; not\ L_l \leftarrow L_{l+1}, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n$$

where $0 \leq k \leq l \leq m \leq n$*, and all of the L's are literals, the answer sets given under the expanded definition of the semantics are exactly the consistent answer sets according to the current definition.*

Let Π be a set of constraints. A consistent set of literals X *violates* Π, if for an constraint $\leftarrow G \in \Pi, X \models G^X$.

Proposition 2 *Let* Π_1 *and* Π_2 *be programs, such that* Π_2 *is a set of constraints. A consistent set of literals is an answer set for* $\Pi_1 \cup \Pi_2$ *if and only if it is an answer set of* Π_1*, and does not violate* Π_2

Equivalent Transformations

The authors define a series of transformations from arbitrary rules with nested expression to rules of the disjunctive form previously discussed. These transformations take one of the following forms:

1. replacing a formula in the rule by another formula
2. moving a formula from the body of a rule to its head
3. moving a formula from the head of a rule to its body

This begs the question: "What does it mean when we say that two formulas are equivalent?"

A formula F is *equivalent* to a formula G, if for any consistent sets of literals X and Y, $X \models F^Y$ if and only if $X \models G^Y$.

Proposition 3 *Let* Π *be a program, and let* F *and* G *be a pair of equivalent formulas. Any program obtained from* Π *by replacing occurences of* F *by* G*, is equivalent to* Π*.*

Proposition 4 *For any formulas F, G, and H:*

1. $F, G \equiv G, F$ *and* $F; G \equiv G; F$
2. $(F, G), H \equiv F, (G, H)$ *and* $(F; G); H \equiv F; (G; H)$
3. $F, (G; H) \equiv (F, G); (F, H)$ *and* $F; (G, H) \equiv (F; G), (F; H)$
4. $not(F, G) \equiv not\ F; not\ G$ *and* $not(F; G) \equiv not\ F, not\ G$
5. $not\ not\ not\ F \equiv not\ F$
6. $F, \top \equiv F$ *and* $F; \top \equiv \top$
7. $F, \bot \equiv \bot$ *and* $F; \bot \equiv F$
8. *if P is an atom the* $p, \neg p \equiv \bot$, *and* $not\ p; not\ \neg p \equiv \top$
9. $not\ \top \equiv \bot$ *and* $not\ \bot \equiv \top$

A *simple conjunction* is a formula of the form:

$$L_1, \ldots, L_k, not\ L_{k+1}, \ldots, not\ L_m,$$
$$not\ not\ L_{m+1}, \ldots, not\ not\ L_n.$$

A simple disjunction is a formula of the form:

$$L_1; \ldots; L_k; not\ L_{k+1}; \ldots; not\ L_m;$$
$$not\ not\ L_{m+1}; \ldots, not\ not\ L_n.$$

Proposition 5 *Any formula is equivalent to:*

1. *a formula of the form* $F_1; \ldots; F_n$ *where* $n \geq 1$*, and each* F_i *is a simple conjunction, and*
2. *a forumla of the form* $F_1, \ldots, F_n$ *where* $n \geq 1$*, and each* F_i *is a simple disjunction*

Proposition 6

1. $(F, G) \leftarrow H$ *is equivalent to:*

$$F \leftarrow H,$$
$$G \leftarrow H.$$

2. $F \leftarrow (G; H)$ *is equivalent to:*

$$F \leftarrow G,$$
$$F \leftarrow H.$$

3. $F \leftarrow (G, not\ not\ H)$ *is equivalent to:*

$$(F; not\ H) \leftarrow G.$$

4. $(F; not\ not\ G) \leftarrow H$ *is equivalent to:*

$$F \leftarrow not\ G, H.$$

Proposition 7 *Any program is equivalent to a set of rules of the form:*

$$L_1; \ldots; L_k; not\ L_{k+1}; \ldots; not\ L_l \leftarrow L_{l+1}, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n$$

We will later see that rules of the above form which are generated from a program Π, do not contain negation as failure in the heads, $(k = l)$, if negation as failure in Π:

1. does not occur in the heads of rules, and
2. is not nested, that is, not applied to formulas containing negation as failure

Proof of Proposition 1

Lemma 1 *Let* Π *be a program whose rules have the form:*

$$L_1; \ldots; L_k \leftarrow L_{k+1}, \ldots, L_m$$

where $0 \leq k \leq m$*, and all of the* L*'s are literals, and let* X *be a consistent set of literals.* X *is closed under* Π *in the sense of this paper if, for every rule* $Head \leftarrow Body$ *in* Π*,* $Head \in X$ *whenever* $Body \subseteq X$*.*

Lemma 2 *Let* Π *be a program whose rules have the form:*

$$L_1; \ldots; L_k; not\ L_{k+1}; \ldots; not\ L_l \leftarrow L_{l+1}, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n$$

and let X*, and* Y *be consistent sets of literals.* Y *is closed under* Π^X *if and only* Y *is closed under the reduct of* Π *relative to* X *in the sense "our definition".*

Proof of Lemma 1: It is easy to verify that X is closed under Π in the sense of this paper if and only if for every rule of the form:

$$L_1; \ldots; L_k \leftarrow L_{k+1}, \ldots, L_m$$

in Π, X includes at least one of the literals $L_1, \ldots, L_k$ provided that X includes all of the literals $L_{k+1}, \ldots, L_m$. This is exactly what it is for X to be closed under Π.

Proof of Lemma 2: For a program Π whose rules have the form:

$$L_1; \ldots; L_k; not\ L_{k+1}; \ldots; not\ L_l \leftarrow L_{l+1}, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n$$

and for a consistent set of literals X, Π^X can be characterized as the result of replacing each subformula of the form $not\ L$ in Π by $\bot$ if $L \in X$, and by $\top$ otherwise.

On the other hand, the reduct of Π relative to X is defined as the program obtained from Π by:

- deleting every rule such that at least one of $L_{k+1}, \ldots, L_m$ is not in X or at least one of $L_{m+1}, \ldots, L_n$ is in X, and
- replacing each remaining rule by:

$$L_1; \ldots; L_k \leftarrow L_{l+1}, \ldots, L_m$$

This program can be obtained from Π^X by:

- deleting every rule sucht that its head contains $\top$ or its body contains $\bot$, and
- removing every $\bot$ in the head, and every $\top$ in the body of each remaining rule.

It is clear that these steps have no effect on whether a consistent set Y of literals is closed under that program.

Proof of Proposition 2

Lemma 3 *Let* Π *be a set of basic constraints, and let* X *be a consistent set of literals. If* X *is closed under* Π*, then every subset of* X *is closed under* Π*.*

Proof of Lemma 3: It is easy to see that for any consistent sets X, and Y of literals, and any basic formula G, if $Y \subseteq X$ and $Y \models G$ then $X \models G$.

Proof of Proposition 2: Let X be a consistent set of literals. We need to show that X is an answer set for $\Pi_1^X \cup \Pi_2^X$ if and only if it is an answer set for Π_1^X and does not violate Π_2.

Left-to-Right: Assume that X is an answer set for $\Pi_1^X \cup \Pi_2^X$. Then X is closed under both Π_1^X and Π_2^X. The second condition means that X does not violate Π_2. Now we need to check the minimality of X. Let Y be a subset of X closed under Π_1^X. Since X is closed under Π_2^X, from Lemma 3 we know that Y is closed under Π_2^X as well. Since X is minimal under the sets closed under $\Pi_1^X \cup \Pi_2^X$, it follows that $Y = X$.

Right-to-Left: Assume that X is an answer set for Π_1^X and does not violate Π_2. The second condition means that X is closed under Π_2^X. Consequently, X is closed under both Π_1^X and Π_2^X. Now we need to check the minimality of X. Let Y be a subset of X closed under $\Pi_1^X \cup \Pi_2^X$. Then in particular Y is closed under Π_1^X. Since X is minimal among such sets, it follows that $Y = X$.