Lloyd-Topor Completion and General Stable Models

Vladimir Lifschitz and Fangkai Yang
Department of Computer Science
The University of Texas at Austin

AAAI Spring Symposium 2014 on KR and Robotics

- AAAI Spring Symposium on Knowledge Representation, Reasoning and Robotics
- Mar 24 – 26, 2014, Palo Alto, USA
- Organizers
  - Esra Erdem (Sabanci University)
  - Volkan Patoglu (Sabanci University)
  - Mohan Sridharan (Texas Tech Univeristy)
  - Fangkai Yang (University of Texas at Austin)
- Invited speaker includes Michael Gelfond (Texas Tech University).

Stable Models and Program Completion

- (Ferraris et al, 2011) defines general stable models.
- The new definition proposes an operator SM that turns a first-order sentence F w.r.t a set of intensional predicates $\mathbf{p}$ into a stronger second order sentence $\mathrm{SM}_{\mathbf{p}}[F]$. The stable model of F relative to $\mathbf{p}$ is the arbitrary models of $\mathrm{SM}_{\mathbf{p}}[F]$
- Logic program without strong negation can be identified as sentences, so that SM is applicable to logic program as well.
- Question: can we use program completion to characterize general stable models?
- Since the definition of general stable models refers to programs that may contain quantifiers, we will use the generalization of completion from (Lloyd & Topor, 1978).
- Our results can be applied to program more general than the rules used in (Lloyd & Topor, 1978).

Lloyd-Topor Program, Choice Rules and Constraints

- A *Lloyd-Topor rule* is a rules of the form

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

It can be identified as formula $\widetilde{\forall}(G \rightarrow p(\mathbf{t}))$.

- A *constraint* is a rule of the form

$$\bot \leftarrow G,$$

identified as $\neg G$.

- A *choice rule* is a rule of the form

$$p(\mathbf{x}) \vee \neg p(\mathbf{x}) \leftarrow G.$$

It is abbreviated as

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

and can be identified as $\widetilde{\forall}(G \rightarrow p(\mathbf{x}) \vee \neg p(\mathbf{x}))$.

General Stable Models

- (Ferraris et al, 2011) defines operator SM that turns a first order sentence F into a stronger second order sentence.
- The *stable models* of F are arbitrary models of $\mathrm{SM}[F]$.
- Logic programs are identified with first order sentences. Example:

$$p(a),$$
$$q(b),$$
$$p(x) \leftarrow q(x)$$

is viewed as sentence F:

$$p(a) \wedge q(b) \wedge \forall x(q(x) \rightarrow p(x)),$$

and $\mathrm{SM}[F]$ is

$$p(a) \wedge q(b) \wedge \forall x(q(x) \rightarrow p(x)) \wedge \text{ some second order sentence.}$$

Lloyd-Topor Program and Completion

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

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

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

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

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

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

where $\mathbf{x}$ is a list of distinct variables not appearing in any of the rules (1), and $\mathbf{y}^i$ is the list of free variables in (1).

- The *completion* of Π, denoted by $\mathrm{Comp}[\Pi]$, is the conjunction of the completed definitions of all predicate constants.

Example

The completion of the program

$$p(a),$$
$$q(b),$$
$$p(x) \leftarrow q(x)$$

is

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

which is equivalent to

$$\forall x(p(x) \leftrightarrow x = a \vee q(x)),$$
$$\forall x(q(x) \leftrightarrow x = b).$$

Predicate Dependency Graph

- (Ferraris et al, 2011) extends the definition of the predicate dependency graph of a logic program to arbitrary first-order sentences. The vertices of the graph are predicate symbols. A formula is *tight* if its predicate dependency graph is acyclic.
- From the results of the paper it follows that for a tight Lloyd-Topor program Π, $\mathrm{SM}[\Pi]$ is equivalent to $\mathrm{Comp}[\Pi]$.
- For instance, the predicate dependency graph of program Π_1

$$p(a), \quad q(b),$$
$$p(x) \leftarrow q(x),$$

contains only one edge $p \longrightarrow q$ and is acyclic. So $\mathrm{SM}[\Pi_1]$ is equivalent to $\mathrm{Comp}[\Pi_1]$.

Need for Generalization

- Consider program Π_2 containing a single rule

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

  We will see that $\mathrm{SM}[\Pi_2]$ is equivalent to $\mathrm{Comp}[\Pi_2]$. But this program is not tight according to (Ferraris et al, 2011).

- Instead of the predicate dependency graph, (Lee & Meng, 2011) use the first order dependency graph. It allows them to define *atomic tightness*, which is more general than tightness. But this program is not even atomic-tight.

- Similar examples appear in applications of ASP to knowledge representation.

Conditional Equivalence between SM and Comp

- Consider program Π_3

$$\begin{aligned} p(x) &\leftarrow q(x),\\ q(a) &\leftarrow p(b). \end{aligned}$$

- $\mathrm{SM}[\Pi_3]$ is stronger than its completion $\mathrm{Comp}[\Pi_3]$

$$\forall x(p(x) \leftrightarrow q(x)),$$
$$\forall x(q(x) \leftrightarrow x = a \land p(b)).$$

- However,

$$a \neq b \models \mathrm{SM}[\Pi_3] \leftrightarrow \mathrm{Comp}[\Pi_3].$$

  It follows that the stable models of

$$\Pi_3 \cup \{\leftarrow a = b\}$$

  are characterized by the first order formula $\mathrm{Comp}[\Pi_3] \land a \neq b$.

Auxiliary Definitions

- A subformula G of a first order formula is *positive* if the number of implications containing G in the antecedent is even. ($\neg F$ stands for $F \rightarrow \bot$).
- A subformula G is *nonnegated* if it does not belong to a subformula of the form $\neg F$.

Rule Dependency Graph

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

- rules of Π, with variables 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.

Example: the rule dependency graph of program

$$p(a, b),$$
$$q(x, y) \leftarrow p(y, x) \wedge \neg p(x, y)$$

has the edges

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

for arbitrary pairs of distinct variables x_1 and y_1.

Chains

A finite path in the rule dependency graph of Π is a *chain* if the rules at its vertices have no common variables. Example: for the program with the rules

$$p(x) \leftarrow q(x) \qquad q(x) \leftarrow r(x) \qquad r(x) \leftarrow s(x)$$

chains of length 2 have the form

$$\begin{array}{c} p(x_1) \leftarrow q(x_1) \\ \big\downarrow q(x_1) \\ q(x_2) \leftarrow r(x_2) \\ \big\downarrow r(x_2) \\ r(x_3) \leftarrow s(x_3) \end{array}$$

where x_1, x_2, x_3 are pairwise distinct variables.

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

Chain Formula

Let C be a chain

$$
\begin{array}{c}
p_0(\mathbf{t}^0) \leftarrow Body_0 \\
\big\downarrow p_1(\mathbf{s}^1) \\
p_1(\mathbf{t}^1) \leftarrow Body_1 \\
\big\downarrow p_2(\mathbf{s}^2) \\
\cdots\cdots\cdots\cdots \\
\big\downarrow p_n(\mathbf{s}^n) \\
p_n(\mathbf{t}^n) \leftarrow Body_n
\end{array}
\tag{2}
$$

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

$$\left(\bigwedge_{i=1}^{n} \mathbf{s}^i = \mathbf{t}^i\right) \wedge \left(\bigwedge_{i=0}^{n} Body_i\right).$$

Example

If C is

$$q(x_1, y_1) \leftarrow p(y_1, x_1) \wedge \neg p(x_1, y_1)$$
$$\downarrow p(y_1, x_1)$$
$$p(a, b)$$

then the chain formula F_C is

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

Main Theorem

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,

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

Any tight program is trivially Γ-tight for any Γ.

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

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

Example

Consider program Π_2:

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

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

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

and the chain formula

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

is contradictory. So $\widetilde{\forall} \neg F_C$ is logically valid for any chain C of length 1.

Moving Objects (1)

Logic programs can be used to describe effects of actions. Consider logic program M describing moving objects from one location to another.

- Facts

$$step(\widehat{0}),\ step(\widehat{1}),\ \dots\ step(\widehat{k});$$
$$next(\widehat{0},\widehat{1}),\ next(\widehat{1},\widehat{2}),\ \dots,\ next(\widehat{k-1},\widehat{k});$$

- Rules

$$\{at(x,y,0)\} \leftarrow object(x) \wedge place(y)$$
$$at(x,y,t_2) \leftarrow move(x,y,t_1) \wedge next(t_1,t_2)$$
$$\{at(x,y,t_2)\} \leftarrow at(x,y,t_1) \wedge next(t_1,t_2)$$

describe initial states, effects of actions, and the commonsense law of inertia.

Moving Objects (2)

- Unique name constraints

$$\leftarrow \widehat{i} = \widehat{j} \quad (1 \leq i < j \leq k)$$

- constraints describing the arguments of at and $move$

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

- the uniqueness of location constraint

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

- the existence of location constraint

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

Intensional predicates: $\mathbf{p} = \{next, step, at\}$.

Characterizing the Stable Models of M

H: the conjunction of the constraints from M written as first order sentences.

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

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

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

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

Plan of the Proof (1)

To use Main Theorem, we will consider Lloyd-Topor program Π_4:

$$
\begin{array}{rl}
 & step(\widehat{0}),\ step(\widehat{1}),\ \dots\ step(\widehat{k}); \\
 & next(\widehat{0},\widehat{1}),\ next(\widehat{1},\widehat{2}),\ \dots,\ next(\widehat{k-1},\widehat{k}); \\
at(x,y,t_2) & \leftarrow move(x,y,t_1) \wedge next(t_1,t_2) \\
at(x,y,0) & \leftarrow object(x) \wedge place(y) \wedge \neg\neg at(x,y,0) \\
at(x,y,t_2) & \leftarrow at(x,y,t_1) \wedge next(t_1,t_2) \wedge \neg\neg at(x,y,t_2) \\
object(x) & \leftarrow \neg\neg object(x), \\
place(y) & \leftarrow \neg\neg place(y), \\
move(x,y,z) & \leftarrow \neg\neg move(x,y,z).
\end{array}
$$

From the results of (Ferraris et al, 2011) it easily follows that $\mathrm{SM}_{\mathbf{p}}[M]$ is equivalent to $\mathrm{SM}[\Pi_4] \wedge H$.

Plan of the Proof (2)

Fact Program Π_4 is H-tight.

By Main Theorem, it follows that

$$H \models \mathrm{SM}[\Pi_4] \leftrightarrow \mathrm{Comp}[\Pi_4].$$

Consequently

$$\mathrm{SM}_{\mathbf{p}}[M] \Leftrightarrow \mathrm{SM}[\Pi_4] \wedge H \Leftrightarrow \mathrm{Comp}[\Pi_4] \wedge H.$$

Conclusion

- We proposed a new method for representing $\mathrm{SM}[F]$ in the language of first-order logic. It is more general than the methods of (Ferraris et al, 2011). Relationship with (Lee & Meng, 2011) requires further study.
- This method allows us to prove the equivalence between some ASP descriptions of dynamic domains to axiomatizations based on successor state axioms.
- Proof of Main Theorem is based on the theory of stable models of infinitary propositional formulas developed by Mirek Truszczynski.