System F2LP – Computing Answer Sets of First-Order Formulas

Joohyung Lee and Ravi Palla

Computer Science and Engineering
Arizona State University, Tempe, AZ, USA
{joolee,ravi.palla}@asu.edu

Abstract. We present an implementation of the general language of stable models proposed by Ferraris, Lee and Lifschitz. Under certain conditions, system F2LP turns a first-order theory under the stable model semantics into an answer set program, so that existing answer set solvers can be used for computing the general language. Quantifiers are first eliminated and then the resulting quantifier-free formulas are turned into rules. Based on the relationship between stable models and circumscription, F2LP can also serve as a reasoning engine for general circumscriptive theories. We illustrate how to use F2LP to compute the circumscriptive event calculus.

1 Introduction

One advantage of classical logic over logic programs is that the former allows us to encode knowledge in a complex formula, which is often more convenient than encoding in conjunctive normal form only. While the input languages of answer set solvers have evolved to allow various constructs for facilitating encoding efforts, such as choice rules, cardinality constraints and aggregates, the syntax is still limited to rule forms and does not allow quantifiers and connectives nested arbitrarily as in classical logic.

Recently, there have been some efforts in lifting the syntactic restriction by extending the stable model semantics to arbitrary first-order formulas, under which an answer set program is viewed as the conjunction of the implications corresponding to the rules. The generality of the language allows to view choice rules and cardinality constraints as abbreviations of first-order formulas under this semantics without references to grounding [1].

System F2LP [1] is a step towards implementing this general language. It translates an arbitrary first-order formula under the stable model semantics into an answer set program. By calling existing answer set solvers on the resulting program, we can compute Herbrand stable models of a first-order formula. The system extends the previous version described in [2], which computes stable models of an arbitrary propositional formula. The translation implemented in F2LP is based on the following recent theoretical results.

[1] http://reasoning.eas.asu.edu/f2lp .

- Every first-order formula is strongly equivalent to its prenex form [2, Theorem 2] and can be also rewritten as a universal formula under certain conditions at the price of introducing new predicate constants [5, Proposition 3].
- Every quantifier-free formula (including propositional formula) is strongly equivalent to a logic program [3, 4, 2].

We expect that F2LP will facilitate encoding efforts. It can also serve as a tool for computing general circumscriptive theories, in view of the relationship between the stable models and circumscription described in [5]. We illustrate how F2LP can be used for computing circumscriptive event calculus [6, 7], whose syntax is not necessarily in the rule form. System CIRC2DLP [8] is another implementation of circumscription using answer set solvers, which can even handle prioritized circumscription and allows varied constants. On the other hand, F2LP allows more general syntax.

2 Review: Stable Models for First-Order Formulas

We follow the definition of a stable model from [9], a journal version of [10]. The definition is also reproduced in [11]. There stable models are defined using "stable model operator SM" with "intensional predicates," similar to circumscription.

Let $\mathbf{p}$ be a list of distinct predicate constants $p_1, \dots, p_n$ other than equality. For any first-order sentence F, by $\mathrm{SM}[F; \mathbf{p}]$ we denote the second-order sentence

$$F \wedge \neg\exists \mathbf{u}((\mathbf{u} < \mathbf{p}) \wedge F^*(\mathbf{u})),$$

where $\mathbf{u}$ is a list of n distinct predicate variables $u_1, \dots, u_n$. Expression $\mathbf{u} < \mathbf{p}$ stands for a formula expressing that $\mathbf{u}$ is "stronger than" $\mathbf{p}$, defined same as in circumscription. Formula $F^*(\mathbf{u})$ is defined recursively.

- $p_i(\mathbf{t})^* = u_i(\mathbf{t})$ for any tuple $\mathbf{t}$ of terms;
- $F^* = F$ for any atomic F that does not contain members of $\mathbf{p}$;
- $(F \odot G)^* = (F^* \odot G^*), \quad \odot \in \{\wedge, \vee\}$;
- $(F \to G)^* = (F^* \to G^*) \wedge (F \to G)$;
- $(QxF)^* = QxF^*, \quad Q \in \{\forall, \exists\}$.

A model of F (in the sense of first-order logic) is *stable* (relative to the set $\mathbf{p}$ of *intensional* predicates) if it satisfies $\mathrm{SM}[F; \mathbf{p}]$. Let $\sigma(F)$ be the signature consisting of the object, function and predicate constants occurring in F. If F contains at least one object constant, an Herbrand interpretation of $\sigma(F)$ that satisfies $\mathrm{SM}[F; \mathbf{p}]$ where $\mathbf{p}$ is the list of all predicate constants occurring in F, is called an *answer set* of F. The answer sets of a logic program Π are defined as the answer sets of the FOL-representation of Π (i.e., the conjunction of the universal closure of implications corresponding to the rules). It turns out that this definition, applied to the syntax of logic programs, is equivalent to the traditional definition of answer sets based on grounding and fixpoint construction [10].

3 Quantifier Elimination

Given a set of formulas, F2LP first eliminates all quantifiers and then applies the transformation defined in [4] that turns the resulting quantifier-free formulas

into logic program rules. In this section we describe how quantifier elimination is done in F2LP.

Obviously, if the domain is known and finite, quantifiers can be replaced with multiple disjunctions and conjunctions. For instance, consider the formula

$$r \wedge \neg\exists x(p(x) \wedge q(x)) \rightarrow s \tag{1}$$

occurring in a program that contains n object constants $\{a_1, \ldots, a_n\}$. Replacing $\exists x(p(x) \wedge q(x))$ with multiple disjunctions and then turning the result into a logic program yields 2^n rules. Also this translation is not modular as it depends on the underlying domain, so that the multiple disjunctions need to be updated when the domain changes. Alternatively, we can introduce a new predicate constant p', and turn (1) into

$$s \leftarrow r, \mathit{not}\ p'$$
$$p' \leftarrow p(x), q(x)$$

which does not involve grounding so that the translation is not dependent on the domain.

Under the general stable model semantics, maximal negative occurrences of $\exists$ and maximal positive occurrences of $\forall$ in the formula can be dropped in view of the fact that the standard prenex normal form conversion turns such occurrences into outermost $\forall$ without affecting strong equivalence [2]. As shown in the example above, positive occurrences of $\exists$ can be eliminated using new predicate constants if the quantified formula is in the scope of negation. This condition is further generalized in the proposition below. We say that an occurrence of a predicate constant in a formula F is *strictly positive* if that occurrence is not in the antecedent of any implication. (For instance, in $(p \rightarrow q) \rightarrow r$, only r has a strictly positive occurrence.) About a formula F, we say that it is *negative* on a tuple $\mathbf{p}$ of predicate constants if members of $\mathbf{p}$ have no strictly positive occurrences in F [11]. The following proposition is a slight generalization of [5, Proposition 3] in view of Theorem on Double Negations from [11].

Proposition 1. *Let F be a sentence, let $\mathbf{p}$ be a list of distinct predicate constants and let q be a predicate constant that does not belong to the signature of F. For any positive occurrence of a subformula $\exists x G(x, \mathbf{y})$ of F where $\mathbf{y}$ is the list of all free variables in $\exists x G(x, \mathbf{y})$, let F' be the formula obtained from F by replacing that occurrence with $\neg\neg q(\mathbf{y})$. If the occurrence of $G(x, \mathbf{y})$ is in a subformula of F that is negative on $\mathbf{p}$, then the models of*

$$\mathrm{SM}[F' \wedge \forall x\mathbf{y}(G(x, \mathbf{y}) \rightarrow q(\mathbf{y})); \mathbf{p}, q]$$

restricted to the signature of F are precisely the models of $\mathrm{SM}[F; \mathbf{p}]$.

Negative occurrences of $\forall$ can also be eliminated using the proposition by first rewriting $\forall x G$ as $\neg\exists x \neg G$.

For example, $\exists x(p(x) \wedge q(x))$ in formula (1) is contained in a negative formula (relative to any set of intensional predicates). According to Proposition 1 $\mathrm{SM}[(1);\ p, q, r, s]$ has the same models as

$$\mathrm{SM}[(r \wedge \neg\neg\neg p' \rightarrow s) \wedge \forall x(p(x) \wedge q(x) \rightarrow p');\ p, q, r, s, p']$$

by disregarding p'.

These ideas lead to the following procedure for quantifier elimination, which is implemented in F2LP.

Definition 1. *Given a formula F, repeat the following until there are no occurrences of quantifiers remaining:*

Select a maximal occurrence of $QxG(x, \mathbf{y})$ in F where Q is $\forall$ or $\exists$ and $\mathbf{y}$ is the list of all free variables in $QxG(x, \mathbf{y})$.

(a) If Q is $\exists$ and the occurrence of $QxG(x, \mathbf{y})$ in F is negative, or if Q is $\forall$ and the occurrence of $QxG(x, \mathbf{y})$ in F is positive, then set F to be the formula obtained from F by replacing the occurrence of $QxG(x, \mathbf{y})$ with $G(z, \mathbf{y})$ where z is a new variable.

(b) If Q is $\exists$ and the occurrence of $QxG(x, \mathbf{y})$ in F is positive, then set F to be

$$F' \wedge (G(x, \mathbf{y}) \rightarrow p_G(\mathbf{y}))$$

where F' is the formula obtained from F by replacing the occurrence of $QxG(x, \mathbf{y})$ with $\neg\neg p_G(\mathbf{y})$ where p_G is a new predicate constant.

(c) If Q is $\forall$ and the occurrence of $QxG(x, \mathbf{y})$ in F is negative, then set F to be the formula obtained from F by replacing the occurrence of $QxG(x, \mathbf{y})$ with $\neg\exists x\neg G(x, \mathbf{y})$.

4 F2LP Implementation

Formulas can be encoded in the language of F2LP using the following ASCII characters.

Symbol	$\neg$	$\wedge$	$\vee$	$\rightarrow$	$\bot$	$\top$	$\forall xyz$	$\exists xyz$
ASCII	`-`	`&`	`|`	`->`	`false`	`true`	`![X,Y,Z]:`	`?[X,Y,Z]:`

F2LP turns a formula into the corresponding LPARSE program.[2] The usual LPARSE encoding is also allowed in F2LP: it is simply copied to the output. The LPARSE program returned by F2LP can be passed to ASP grounders and solvers that accept LPARSE language. While function symbols are allowed in the input language of F2LP, it is left to the grounder to handle them.

The current version of F2LP does not check if the condition to apply quantifier elimination (Proposition 1) is satisfied, which is left to the users. Also F2LP does not check if the given formula is safe (according to [1]), and may turn a safe formula into an unsafe program. For instance, F2LP turns the safe formula

```
p(X) -> ((q(Y)->r(Y)) | s(X)).
```

into an unsafe program

```
r(Y)|s(X) :- q(Y),p(X).
s(X) :- {not q(Y)}0,not r(Y),p(X).
```

However, this may not be a serious limitation since we usually declare variables using the `#domain` directive in LPARSE language, which is the same as appending domain predicates to the body of each rule.

[2] `http://www.tcs.hut.fi/Software/smodels` .

5 Computing Circumscriptive Theories

Kim *et al.* [5] show that for a certain class of formulas called "canonical," circumscription and the general stable model semantics coincide. This allows F2LP to be used for computing circumscription of canonical formulas. For example, consider the formula

$$F = \exists x(p(x) \wedge r(x)) \rightarrow q(b)$$

and the intensional predicates $\{p, q\}$. According to [5], the formula is "canonical" relative to $\{p, q\}$ so that CIRC$[F; p, q]$ is equivalent to SM$[F; p, q]$, and furthermore to SM$[F \wedge \forall x(r(x) \vee \neg r(x)); p, q, r]$. Formula $F \wedge \forall x(r(x) \vee \neg r(x))$ can be encoded in the language of F2LP (In addition, let us assume that the domain is $\{a, b, c\}$):

```
objects(a;b;c).
#domain objects(X).
?[X]:(p(X)&r(X)) -> q(b).
{r(X)}.
```

Canonical theories cover a wide range of action formalisms based on circumscription, such as circumscriptive event calculus. Here we illustrate how to use F2LP to compute an event calculus description.

A circumscriptive event calculus domain description is defined as

$$\text{CIRC}[\Sigma\ ;\ \mathit{Initiates}, \mathit{Terminates}, \mathit{Releases}] \wedge\ \text{CIRC}[\Delta\ ;\ \mathit{Happens}] \wedge \Xi. \quad (2)$$

where Σ, Δ, Ξ are first-order sentences such that all positive occurrences of $\exists xG$ in these formulas are contained in subformulas that are negative on $\{\mathit{Initiates}, \mathit{Terminates}, \mathit{Releases}, \mathit{Happens}\}$. Theorem 1 from [5] shows that this theory can be turned into

$$\text{SM}[\Sigma \wedge \Delta \wedge \Xi \wedge \mathit{Choice}(\mathbf{p} \setminus \{\mathit{Initiates}, \mathit{Terminates}, \mathit{Releases}, \mathit{Happens}\}); \mathbf{p}] \quad (3)$$

where $\mathbf{p}$ is the set of all predicates occurring in the description. (By $\mathit{Choice}(\mathbf{p})$ we denote the conjunction of "choice formulas" $\forall \mathbf{x}(p(\mathbf{x}) \vee \neg p(\mathbf{x}))$ for all predicate constants p in $\mathbf{p}$ where $\mathbf{x}$ is a list of distinct object variables whose length is the same as the arity of p.) Note that the condition on Σ, Δ, Ξ above satisfies the condition for eliminating existential quantifiers in Proposition 1.

In view of Theorem 1 from [5], F2LP can be used for computing the models of (2). To compute the models, a user can encode

$$\Sigma \wedge \Delta \wedge \Xi \wedge \mathit{Choice}(\mathbf{p} \setminus \{\mathit{Initiates}, \mathit{Terminates}, \mathit{Releases}, \mathit{Happens}\})$$

in (3) in the language of F2LP, and run F2LP to turn it into an answer set program. For instance, an action precondition axiom (in Ξ) for the Blocks World can be encoded in F2LP as

```
T < maxstep & happens(pickUp(X),T)
  -> holdsAt(clear(X),T) & X != table & -?[Y]:holdsAt(holding(Y),T).
```

("picking up X is possible only if X is clear, the agent is not already holding another object and the object being picked up is not the table.")

F2LP turns the axiom into the following rules.

```
holdsAt(clear(X),T) :- T<maxstep,happens(pickUp(X),T).
 :- {not holdsAt(holding(NV1),T)}0,T<maxstep,happens(pickUp(X),T).
 :- X=table,T<maxstep,happens(pickUp(X),T).
```

A full encoding of the Blocks World in the language of F2LP is available on the F2LP webpage (Footnote 1).

Acknowledgements

We are grateful to the anonymous referees for their useful comments on this paper. This work was partially supported by the National Science Foundation under Grant IIS-0839821.

References

1. Lee, J., Lifschitz, V., Palla, R.: A reductive semantics for counting and choice in answer set programming. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI). (2008) 472–479
2. Lee, J., Palla, R.: Yet another proof of the strong equivalence between propositional theories and logic programs. In: Working Notes of the Workshop on Correspondence and Equivalence for Nonmonotonic Theories. (2007)
3. Cabalar, P., Ferraris, P.: Propositional theories are strongly equivalent to logic programs. TPLP **7**(6) (2007) 745–759
4. Cabalar, P., Pearce, D., Valverde, A.: Reducing propositional theories in equilibrium logic to logic programs. In: Proceedings of Portuguese Conference on Artificial Intelligence (EPIA). (2005) 4–17
5. Kim, T.W., Lee, J., Palla, R.: Circumscriptive event calculus as answer set programming. In: Proceedings of International Joint Conference on Artificial Intelligence (IJCAI). (2009) To appear.
6. Shanahan, M.: A circumscriptive calculus of events. Artif. Intell. **77**(2) (1995) 249–284
7. Mueller, E.: Commonsense reasoning. Morgan Kaufmann (2006)
8. Oikarinen, E., Janhunen, T.: circ2dlp - translating circumscription into disjunctive logic programming. In: Proceedings of the Eighth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR). (2005) 405–409
9. Ferraris, P., Lee, J., Lifschitz, V.: Stable models and circumscription. Artificial Intelligence (2010) To appear.
10. Ferraris, P., Lee, J., Lifschitz, V.: A new perspective on stable models. In: Proceedings of International Joint Conference on Artificial Intelligence (IJCAI). (2007) 372–379
11. Ferraris, P., Lee, J., Lifschitz, V., Palla, R.: Symmetric splitting in the general theory of stable models. In: Proceedings of International Joint Conference on Artificial Intelligence (IJCAI). (2009) To appear.