Translating First-Order Theories into Logic Programs

Heng Zhang[*], **Yan Zhang**[†], **Mingsheng Ying**[*,‡], **Yi Zhou**[†]

[*]Department of Computer Science and Technology, Tsinghua University, Beijing, China

[†]School of Computing and Mathematics, University of Western Sydney, NSW, Australia

[‡]Faculty of Engineering and Information Technology, University of Technology, Sydney, NSW, Australia

Abstract

This paper focuses on computing first-order theories under either stable model semantics or circumscription. A reduction from first-order theories to logic programs under stable model semantics over finite structures is proposed, and an embedding of circumscription into stable model semantics is also given. Having such reduction and embedding, reasoning problems represented by first-order theories under these two semantics can then be handled by using existing answer set solvers. The effectiveness of this approach in computing hard problems beyond NP is demonstrated by some experiments.

1 Introduction

For a knowledge representation language, the encoding naturalness and implementation efficiency may be regarded as the two faces of a coin. For instance, higher-order logic is natural to represent almost all kinds of mathematical knowledge, but no practicable solver is available; on the other hand, logic programming can be implemented efficiently, which is obviously achieved at the expense of its naturalness to represent knowledge. It is a challenging task to design a language with a smart tradeoff between these two aspects.

An appropriate candidate for such a language may be the first-order language under stable model semantics [Ferraris *et al.*, 2011; Lin and Zhou, 2011], which provides a unified framework for answer set programming (ASP), a flourishing paradigm of declarative programming emerged over the last decade. First-order quantifiers are introduced so that more flexible ways for encoding become possible. Interestingly, with this language, we can easily represent a class of knowledge for which no natural encodings in logic programs are known.[1] However, this also presents a challenge to develop an answer set solver for first-order theories, which is the main task of this paper. Another language that we will consider is first-order circumscription [McCarthy, 1980], a well-known non-monotonic formalism for knowledge representation.

The main contributions of this paper are as follows. Firstly, we show how first-order theories under stable model semantics over finite structures can be reduced to logic programs, which enables us to compute them by using existing answer set solvers; we also prove that introducing auxiliary predicates in the reduction is essential. Secondly, we discover an embedding of circumscription into stable model semantics involving no auxiliary predicates. With such an embedding, we can then compute first-order theories of circumscription over finite structures via answer set programming. Finally, by undertaking some experiments, we demonstrate the effectiveness of our approach in computing some hard problems beyond NP. As far as we know, under either stable model semantics or circumscription, our approach is the first one to compute arbitrary first-order theories over finite structures.

[1]For details, please refer to Section 5 of this paper.

2 Preliminaries

We assume *vocabularies* are finite sets of predicate constants and function constants. Every constant is equipped with a natural number, its *arity*. For technical reasons, following constants are reserved: a binary predicate constant succ, and two nullary function constants (i.e. individual constants) $\min$ and $\max$. All these constants are called *successor constants*.

Logic symbols are defined as usual, including a countable set of predicate variables and a countable set of individual variables. Predicate constants and variables are simply called *predicates* if no confusion occurs. Terms, formulae, and sentences of a vocabulary σ (or shortly, σ-terms, σ-formulae, and σ-sentences) are built from σ, equality, and variables in a standard way. The only thing which may be special is that we treat $\neg\varphi$ as a shorthand of $\varphi \to \bot$, and $\varphi \leftrightarrow \psi$ as the conjunction of $\varphi \to \psi$ and $\psi \to \varphi$. A *(finite) first-order theory* is a (finite) set of first-order sentences. Clearly, a finite first-order theory can be regarded as a first-order sentence, i.e., a finite conjunction of the sentences in that set. A first-order formula is in *prenex normal form* if it is of the form $Q_1x_1 \cdots Q_nx_n\varphi$, where $Q_i (i = 1, \ldots, n)$ is $\forall$ or $\exists$ and φ is quantifier-free. Given a natural number n, a Σ_n-formula (Π_n-formula) is a first-order formula in prenex normal form that has a prefix of n alternating blocks of quantifiers starting with an existential (universal) block, followed by a quantifier-free formula.

Every *structure* $\mathfrak{A}$ of a vocabulary σ (or shortly, *σ-structure* $\mathfrak{A}$) is accompanied by a nonempty set A, the *domain* of $\mathfrak{A}$, and interprets each n-ary predicate constant P of σ as an n-ary relation $\mathfrak{A}(P)$ on A, and interprets each n-ary function constant f of σ as an n-ary function $\mathfrak{A}(f)$ on A. Every *assignment* in a structure $\mathfrak{A}$ is a function α that maps each individual variable

to an element of A and that maps each n-ary predicate variable to an n-ary relation on A. Given a σ-formula φ and an assignment α in $\mathfrak{A}$, we write $\mathfrak{A} \models \varphi[\alpha]$ if α *satisfies* φ in $\mathfrak{A}$ in the standard way. Specially, if φ is a sentence, $[\alpha]$ may be dropped, and $\mathfrak{A}$ is said to be a *model* of φ, or in other words, $\mathfrak{A}$ *satisfies* φ. Let $\varphi(\bar{x})$ be a formula with free variables $\bar{x}$. We simply write $\mathfrak{A} \models \varphi(\bar{a})$ if there is an assignment α in $\mathfrak{A}$ that assigns $\bar{x}$ to $\bar{a}$ and that satisfies $\varphi(\bar{x})$ in $\mathfrak{A}$.

A structure is *finite* if its domain is finite. A finite structure $\mathfrak{A}$ is a *successor structure* if following statements hold: (i) the vocabulary σ of $\mathfrak{A}$ contains all the successor constants; and (ii) there is an isomorphism between $\mathfrak{A}$ and the number system $\mathbb{Z}_n$ (where $n = |A|$) such that $\mathfrak{A}(\mathrm{succ})$ is corresponding to the successor relation on $\mathbb{Z}_n$ (i.e. the set $\{0, \ldots, n-1\}$), and $\mathfrak{A}(\mathrm{min})$ and $\mathfrak{A}(\mathrm{max})$ are corresponding to 0 and $n-1$ respectively. A *restriction* of a structure $\mathfrak{A}$ to a vocabulary σ, written by $\mathfrak{A} \restriction \sigma$, is the structure obtained from $\mathfrak{A}$ by discarding all interpretations for constants which do not belong to σ. Furthermore, a σ-structure $\mathfrak{A}$ is said to be an *expansion* of a σ_0-structure $\mathfrak{B}$ if $\sigma \supseteq \sigma_0$ and $\mathfrak{B}$ is a restriction of $\mathfrak{A}$ to σ_0.

2.1 Circumscription

We follow the notion of parallel circumscription in [Lifschitz, 1985].[2] Let φ be a first-order sentence and $\bar{P}$ a tuple of predicate constants. For each predicate constant P that occurs in $\bar{P}$ (or simply $P \in \bar{P}$) we introduce a new predicate variable X_P of the same arity, and let $\bar{X}_{\bar{P}}$ be short for the tuple $(X_P)_{P \in \bar{P}}$. Moreover, we write $\bar{X}_{\bar{P}} = \bar{P}$ for the conjunction of formulae $\forall \bar{x}(X_P(\bar{x}) \leftrightarrow P(\bar{x}))$ for all predicate constants $P \in \bar{P}$, and write $\bar{X}_{\bar{P}} \leq \bar{P}$ for the conjunction of formulae $\forall \bar{x}(X_P(\bar{x}) \rightarrow P(\bar{x}))$ for all $P \in \bar{P}$. The *circumscription* $\mathrm{CIRC}(\varphi; \bar{P})$ for φ is defined to be the second-order sentence $\varphi \wedge \forall \bar{X}_{\bar{P}}(\bar{X}_{\bar{P}} < \bar{P} \rightarrow \neg\varphi(\bar{X}_{\bar{P}}))$, where $\varphi(\bar{X}_{\bar{P}})$ is the formula obtained from φ by substituting the variables $\bar{X}_{\bar{P}}$ for the constants $\bar{P}$, and $\bar{X}_{\bar{P}} < \bar{P}$ short for the formula $(\bar{X}_{\bar{P}} \leq \bar{P}) \wedge \neg(\bar{X}_{\bar{P}} = \bar{P})$. A structure $\mathfrak{A}$ is called a *$\bar{P}$-minimal model* of φ if it is a model of $\mathrm{CIRC}(\varphi; \bar{P})$.

2.2 Stable Models

Similarly, stable model semantics is defined by a syntax translation SM. Given a first-order sentence φ and a tuple $\bar{P}$ of predicate constants, let $\mathrm{SM}(\varphi; \bar{P})$ stand for the second-order sentence $\varphi \wedge \forall \bar{X}_{\bar{P}}(\bar{X}_{\bar{P}} < \bar{P} \rightarrow \neg\mathrm{St}(\varphi; \bar{P}))$, where the formula $\mathrm{St}(\varphi; \bar{P})$ is defined recursively as follows:

- $\mathrm{St}(P(\bar{t}); \bar{P}) = X_P(\bar{t})$ if P is a predicate constant in $\bar{P}$.
- $\mathrm{St}(\psi; \bar{P}) = \psi$ if ψ is an atom not in the previous case.
- $\mathrm{St}(\psi \circ \chi; \bar{P}) = \mathrm{St}(\psi; \bar{P}) \circ \mathrm{St}(\chi; \bar{P})$ if $\circ \in \{\wedge, \vee\}$.
- $\mathrm{St}(\psi \rightarrow \chi; \bar{P}) = (\psi \rightarrow \chi) \wedge (\mathrm{St}(\psi; \bar{P}) \rightarrow \mathrm{St}(\chi; \bar{P}))$.
- $\mathrm{St}(Qx\psi; \bar{P}) = Qx\mathrm{St}(\psi; \bar{P})$ if $Q \in \{\forall, \exists\}$.

A structure $\mathfrak{A}$ is called a *$\bar{P}$-stable model* of φ if it is a model of $\mathrm{SM}(\varphi; \bar{P})$. A predicate constant is said to be *intensional* if it occurs in $\bar{P}$. Otherwise, it is *extensional*. An *extensional database* of φ is a σ-structure such that σ consists of the set of all the extensional predicate constants and function constants. For more information, please refer to [Ferraris *et al.*, 2011].

[2] Since they can be efficiently eliminated by a method in [Cadoli *et al.*, 1992], we will not consider varying predicates in this paper.

3 Quantifier Elimination

In classical logic, first-order existential quantifiers can be easily removed by introducing some Skolem functions, and this process is known as skolemization. But according to the definition, it may be impossible to remove existential quantifiers in stable model semantics by such an approach. In this section, we present a translation to eliminate quantifiers in first-order theories under stable model semantics. We should note that our translation works only for finite structures.

As shown in [Cabalar *et al.*, 2005], there is also a translation that converts every universal theory to a strongly equivalent[3] disjunctive logic program. Combining these two translations, we can then implement a solver for first-order theories which first reduces the input first-order theory to a disjunctive logic program; then invokes a traditional ASP solver that uses the logic program and an extensional database as its input.

Our main idea of quantifier elimination under stable model semantics is intuitively as follows. Suppose φ is a formula of the form $\forall x \exists y \vartheta(x, y)$ where ϑ is quantifier-free, and $\bar{P}$ a tuple of predicates. Then $\mathrm{SM}(\varphi; \bar{P})$ is actually the formula:

$$\forall x \exists y \vartheta(x, y) \wedge \neg\exists \bar{X}_{\bar{P}}(\bar{X}_{\bar{P}} < \bar{P} \wedge \forall x \exists y \mathrm{St}(\vartheta(x, y); \bar{P}) \quad (1)$$

To remove $\exists y$ from the formula, we simulate $\forall x \exists y \vartheta(x, y)$ by a formula $\exists S \gamma$ and simulate $\forall x \exists y \mathrm{St}(\vartheta(x, y); \bar{P})$ by a formula $\exists X_T \varrho$. Then, by some encoding techniques, we carefully encode γ and ϱ by a universal formula ψ such that $\exists S \exists T \mathrm{SM}(\psi; \bar{P}, S, T)$ is equivalent to $\mathrm{SM}(\varphi; \bar{P})$.

To carry out the simulations mentioned above, we use Eiter *et at.*'s idea [1996], which is employed to simulate existential quantifier in classical logic. Informally, their idea is as follows. Suppose P is a property on a finite domain D equipped with a linear order. Define S as an auxiliary property on D such that a has property S iff there exists b no less than a such that b has property P. Then if we want to answer the problem whether there is an element with property P, we need only check whether the least element has property S. For more details, please refer to Theorem 2.1 in [Eiter *et al.*, 1996].

The main difficulty in constructing the translation is to find a proper encoding of γ and ϱ. But we have found one. The translation τ is defined as follows. Given a first-order sentence φ of the form $\forall \bar{x} \exists \bar{y} \vartheta(\bar{x}, \bar{y})$, we let $\tau(\varphi)$ stand for the conjunction of all sentences which are obtained from the following formulae by applying the universal closure:

$$\neg\neg S(\bar{x}, \overline{\mathrm{min}}) \quad (2)$$

$$(\overline{\mathrm{succ}}(\bar{y}, \bar{z}) \wedge S(\bar{x}, \bar{z})) \vee \vartheta^{\neg\neg}(\bar{x}, \bar{y}) \rightarrow S(\bar{x}, \bar{y}) \quad (3)$$

$$T(\bar{x}, \overline{\mathrm{min}}) \vee \vartheta(\bar{x}, \overline{\mathrm{min}}) \quad (4)$$

$$\neg(\overline{\mathrm{succ}}(\bar{y}, \bar{z}) \wedge S(\bar{x}, \bar{z})) \wedge S(\bar{x}, \bar{y}) \rightarrow (T(\bar{x}, \overline{\mathrm{max}}) \leftrightarrow \vartheta(\bar{x}, \bar{y})) \quad (5)$$

$$\overline{\mathrm{succ}}(\bar{y}, \bar{z}) \rightarrow (T(\bar{x}, \bar{y}) \leftrightarrow \vartheta(\bar{x}, \bar{z}) \vee T(\bar{x}, \bar{z})) \quad (6)$$

where S and T are two predicates of arity $|\bar{x}|+|\bar{y}|$ and have no occurrence in φ, $\vartheta^{\neg\neg}(\bar{x}, \bar{y})$ is obtained from $\vartheta(\bar{x}, \bar{y})$ by substituting $\neg\neg P(\bar{t})$ for all atoms $P(\bar{t})$, $\overline{\mathrm{min}}$ and $\overline{\mathrm{max}}$ denote the

[3] Given two arbitrary first-order sentences φ and ψ, φ is *strongly equivalent* to ψ if, for any first-order sentence γ and any tuple $\bar{P}$ of predicate constants, $\mathrm{SM}(\gamma; \bar{P})$ is equivalent to $\mathrm{SM}(\gamma_0; \bar{P})$, where γ_0 is obtained from γ by replacing some occurrences of φ by ψ.

$|\bar{y}|$-tuples $(\min,\ldots,\min)$ and $(\max,\ldots,\max)$ respectively, and $\overline{\mathrm{succ}}$ stands for a formula that describes the successor relation on tuples of length $|\bar{y}|$. (Clearly, such a formula can be easily built from predicate constant succ.)

Remark 1. In above translation, let us assume that φ is a Π_k-sentence for some $k \geq 1$. By Theorem 6.4 in [Pearce and Valverde, 2005] and Theorem 8 in [Ferraris *et al.*, 2011], it is not difficult to see that $\tau(\varphi)$ can be written as a strongly equivalent Π_{k-1}-sentence. So, every first-order sentence in prenex normal form can be converted to a universal sentence by applying translation τ and the above procedure repeatedly.

The following proposition says that translation τ is faithful.

Proposition 1. *Let φ be any first-order sentence of the form $\forall\bar{x}\exists\bar{y}\vartheta(\bar{x},\bar{y})$ without occurrence of any successor constants. Let $\bar{P}$ be any tuple of predicates. Then, over successor structures, $\exists S\exists T\mathrm{SM}(\tau(\varphi);\bar{P},S,T)$ is equivalent to $\mathrm{SM}(\varphi;\bar{P})$, where S and T are the auxiliary predicates introduced by τ.*

Proof. Let σ be the vocabulary consisting of all constants occurring in φ and of all successor constants. Let $\widehat{\sigma}$ be the vocabulary of $\tau(\varphi)$. Let $k = |\bar{x}|$ and $l = |\bar{y}|$. Given any σ-structure $\mathfrak{A}$, define $\mathcal{R}(\mathfrak{A})$ as a $(k+l)$-ary relation on A such that: for all tuples $\bar{a} \in A^k$ and $\bar{b} \in A^l$, $(\bar{a},\bar{b}) \in \mathcal{R}(\mathfrak{A})$ iff there is a tuple $\bar{c} \geq \bar{b}$ (wrt the successor relation) such that $\vartheta(\bar{a},\bar{c})$ is satisfied by $\mathfrak{A}$; for each tuple $\bar{a} \in A^k$, let $\Im(\mathfrak{A},\bar{a})$ denote the largest one of all tuples $\bar{b} \in A^l$ such that $(\bar{a},\bar{b}) \in \mathcal{R}(\mathfrak{A})$.

We first assume that $\mathfrak{A}$ is a finite successor σ-structure that satisfies $\mathrm{SM}(\varphi;\bar{P})$. Let $\mathfrak{B}$ be a $\widehat{\sigma}$-expansion of $\mathfrak{A}$ that interprets S as the relation $\mathcal{R}(\mathfrak{A})$ and T as the relation A^{k+l}. Then we want to show that $\mathfrak{B}$ satisfies $\mathrm{SM}(\tau(\varphi);\bar{P},S,T)$. It is not difficult to check $\mathfrak{B}$ satisfies $\tau(\varphi)$. Let β be any assignment in $\mathfrak{B}$ that satisfies $\bar{X}_{\bar{P}ST} < \bar{P}ST$. To obtain a contradiction, assume that β satisfies $\mathrm{St}(\tau(\varphi);\bar{P},S,T)$. Note that the formula $\forall\bar{x}\bar{y}\bar{z}\mathrm{St}((3);\bar{P},S,T)$ is clearly a logical consequence of the formula $\mathrm{St}(\tau(\varphi);\bar{P},S,T)$ in classical first-order logic, and β satisfies the former in $\mathfrak{B}$ iff β satisfies the formula

$$\forall\bar{x}\bar{y}\bar{z}\left((\overline{\mathrm{succ}}(\bar{y},\bar{z}) \wedge X_S(\bar{x},\bar{z})) \vee \vartheta^{\neg\neg}(\bar{x},\bar{y}) \rightarrow X_S(\bar{x},\bar{y})\right)$$

in $\mathfrak{B}$. So we must have that $\beta(X_S) = \mathfrak{B}(S)$, and this implies that $\bar{X}_{\bar{P}} < \bar{P}$ should be satisfied by β. Otherwise, for all tuples $\bar{a} \in A^k$, $\mathrm{St}(\vartheta(\bar{a},\Im(\mathfrak{A},\bar{a}));\bar{P})$ has the same truth with $\vartheta(\bar{a},\Im(\mathfrak{A},\bar{a}))$ (and so is true) in β. Consequently $(\bar{a},\mathfrak{A}(\overline{\max}))$ belongs to $\beta(X_T)$ since the universal closure of

$$\neg(\overline{\mathrm{succ}}(\bar{y},\bar{z}) \wedge S(\bar{x},\bar{z})) \wedge X_S(\bar{x},\bar{y}) \rightarrow (X_T(\bar{x},\overline{\max}) \leftrightarrow \mathrm{St}(\vartheta(\bar{x},\bar{y});\bar{P}))$$

should be satisfied by β in $\mathfrak{B}$ (Note that $\mathrm{St}(\tau(\varphi);\bar{P},S,T)$ is satisfied by β in $\mathfrak{B}$ by the assumption). Since the formula

$$\forall\bar{x}\bar{y}\bar{z}(\overline{\mathrm{succ}}(\bar{y},\bar{z}) \rightarrow (X_T(\bar{x},\bar{y}) \leftrightarrow \mathrm{St}(\vartheta(\bar{x},\bar{z});\bar{P}) \vee X_T(\bar{x},\bar{z})))$$

is clearly satisfied by β in $\mathfrak{B}$, the above conclusion leads to $\beta(X_T) = \mathfrak{B}(T)$, a contradiction. Therefore, we have that β satisfies $\bar{X}_{\bar{P}} < \bar{P}$. Let α denote the assignment in $\mathfrak{A}$ obtained by restricting β to variables in $\bar{X}_{\bar{P}}$. Clearly, α satisfies $\bar{X}_{\bar{P}} < \bar{P}$ in $\mathfrak{A}$. On the other hand, as β satisfies $\mathrm{St}(\tau(\varphi);\bar{P},S,T)$, for each tuple $\bar{a} \in A^k$, one of the following cases should be true: (i) $\mathrm{St}(\vartheta(\bar{a},\Im(\mathfrak{A},\bar{a}));\bar{P},S,T)$ is satisfied by β in $\mathfrak{B}$; (ii) there is at least one tuple $\bar{b} \in A^l$ such that $\mathrm{St}(\vartheta(\bar{a},\bar{b});\bar{P},S,T)$ is satisfies by β in $\mathfrak{B}$. In either case, we have that β satisfies $\mathrm{St}(\exists\bar{y}\vartheta(\bar{a},\bar{y});\bar{P},S,T)$ in $\mathfrak{B}$, which will imply that α satisfies $\mathrm{St}(\forall\bar{x}\exists\bar{y}\vartheta(\bar{x},\bar{y});\bar{P})$ in $\mathfrak{A}$. But this is impossible since, by the assumption, $\mathfrak{A}$ is a model of $\mathrm{SM}(\varphi;\bar{P})$.

Conversely, let us assume that $\mathfrak{B}$ is a $\widehat{\sigma}$-structure that satisfies $\mathrm{SM}(\tau(\varphi);\bar{P},S,T)$. Let $\mathfrak{A}$ be the restriction of $\mathfrak{B}$ to σ. Now our task is to show $\mathfrak{A}$ is a model of $\mathrm{SM}(\varphi;\bar{P})$. Since $\mathfrak{B}$ satisfies the universal closure of formula (3), it is true that $\mathfrak{B}(S) \supseteq \mathcal{R}(\mathfrak{A})$. Moreover, we can show that $\mathfrak{B}(S)$ equals to $\mathcal{R}(\mathfrak{A})$. Otherwise, there is an assignment β in $\mathfrak{B}$ such that: (i) $\beta(X_S) = \mathcal{R}(\mathfrak{A})$ and $\beta(X_T) = A^{k+l}$, and (ii) β satisfies both $\bar{X}_{\bar{P}ST} < \bar{P}ST$ and $\mathrm{St}(\tau(\varphi);\bar{P},S,T)$ in $\mathfrak{B}$. Obviously, these contradicts with the assumption. Note also that formula $\forall\bar{x}(2)$ is clearly satisfied by $\mathfrak{B}$. So, for each tuple $\bar{a} \in A^k$, it must hold that $\mathfrak{B}(S)$ contains $(\bar{a},\mathfrak{B}(\overline{\min}))$, or equivalently, $(\bar{a},\mathfrak{B}(\overline{\min})) \in \mathcal{R}(\mathfrak{A})$. By the definition, there should exist at least one tuple $\bar{c} \in A^l$ such that $\vartheta(\bar{a},\bar{c})$ is true in $\mathfrak{A}$. Immediately, we can obtain that φ is satisfied by $\mathfrak{A}$.

Let α be any assignment in $\mathfrak{A}$ that satisfies $\bar{X} < \bar{P}$. To complete the proof, we need show that $\mathrm{St}(\varphi;\bar{P})$ is unsatisfied by α in $\mathfrak{A}$. To obtain a contradiction, assume this is not true. Let β be an assignment in $\mathfrak{B}$ such that: (i) for all P in $\bar{P}$, $\beta(X_P)$ equals to $\alpha(X_P)$; (ii) $\beta(X_S) = \mathcal{R}(\mathfrak{A})$; and (iii) for all $\bar{a} \in A^k$ and all $\bar{b} \in A^l$, $(\bar{a},\bar{b}) \in \beta(X_T)$ iff $(\bar{a},\Im(\mathfrak{A},\bar{a})) \in \mathcal{R}(\mathfrak{A})$ or there is some tuple $\bar{c} > \bar{b}$ such that $\mathrm{St}(\varphi(\bar{a},\bar{c});\bar{P})$ is satisfied by α in $\mathfrak{A}$. Then it is not difficult to verify that β satisfies $\mathrm{St}(\tau(\varphi);\bar{P},S,T)$. Note that $\bar{X}_{\bar{P}ST} < \bar{P}ST$ is clearly satisfied by β. So the conclusion will imply that $\mathfrak{B}$ is not a model of $\mathrm{SM}(\tau(\varphi);\bar{P},S,T)$, a contradiction. □

According to Corollary 6.5 in [Pearce and Valverde, 2005] and Theorem 8 in [Ferraris *et al.*, 2011], under stable model semantics, there exist some translations which convert every first-order sentence into a strongly equivalent sentence in prenex normal form. Let τ_0 be one of those which translate $\tau(\varphi)$ into a Π_{k-1}-sentence for all integers $k \geq 1$ and all Π_k-sentences φ. Let τ^* be a translation that applies τ_0 and τ repeatedly until the resulting sentence is universal. So, by Proposition 1 and Remark 1, we have the faithfulness of τ^*:

Theorem 1. *Let φ be any first-order sentence that has no occurrence of any successor constants. Let $\bar{P}$ be any tuple of predicates. Then $\exists\bar{Q}\mathrm{SM}(\tau^*(\varphi);\bar{P},\bar{Q})$ is equivalent to $\mathrm{SM}(\varphi;\bar{P})$ over finite successor structures, where $\bar{Q}$ is the tuple of auxiliary predicates introduced by τ^*.*

In fact, this theorem can be generalized to finite structures.

Corollary 1. *For every first-order sentence φ, there are a universal first-order sentence ψ and a tuple $\bar{Q}$ of new predicates such that $\exists\bar{Q}\mathrm{SM}(\psi;\bar{P},\bar{Q})$ is equivalent to $\mathrm{SM}(\varphi;\bar{P})$ over finite structures for all tuples $\bar{P}$ of predicate constants.*

Proof. (Sketch) Given a first-order sentence φ, let ϖ be a first-order sentence asserting that a finite σ-structure $\mathfrak{A}$ satisfies $\exists\bar{O}\mathrm{SM}(\varpi;\bar{O})$ iff $\mathfrak{A}$ is a successor structure,[4] where $\sigma = \{\mathrm{succ},\min,\max\}$ and $\bar{O}$ is the tuple of all predicate constants which occur in ϖ and which have no occurrences in both φ and σ. Let $\psi = \varpi \wedge \tau^*(\varphi)$, and let $\bar{P}$ be any

[4] One can find such a sentence in [Eiter *et al.*, 1997].

tuple of predicate constants. By Theorem 1, there is a tuple $\bar{Q}$ of auxiliary predicates such that $\mathrm{SM}(\varphi;\bar{P})$ is equivalent to $\exists\bar{Q}\mathrm{SM}(\tau^*(\varphi);\bar{P},\bar{Q})$. Also by Splitting Lemma in [Ferraris *et al.*, 2009], $\mathrm{SM}(\psi;\bar{O},\bar{P},\bar{Q})$ is equivalent to the formula $\mathrm{SM}(\psi;\bar{O})\wedge\mathrm{SM}(\psi;\bar{P},\bar{Q})$. By the definition, the latter is equivalent to the formula $\mathrm{SM}(\varpi;\bar{O})\wedge\mathrm{SM}(\tau^*(\varphi);\bar{P},\bar{Q})$. Then, it is not difficult to verify that $\mathrm{SM}(\varphi;\bar{P})$ is equivalent to $\exists\bar{O}\exists\bar{Q}\mathrm{SM}(\psi;\bar{O},\bar{P},\bar{Q})$ over finite structures. □

Remark 2. As seen in the above corollary, it is not necessary to equip any structure with a successor relation. Notice that almost all traditional ASP solvers (e.g., CLASP, DLV, etc.) support a set of built-in arithmetic functions or predicates which can be used to define the successor relation. Therefore, when we implement a solver for first-order theories, it is enough to consider a translation over successor structures.

Remark 3. An exponential growth in the formula size seems inevitable in the worst case when we convert a universal sentence to a strongly equivalent logic program. It is because we have to apply rules like "distributive law" frequently (cf. [Cabalar *et al.*, 2005]). The situation will slightly improve for a theory since we can directly apply Cabalar *et al.*'s translation to each sentence in the theory. So we should make each sentence in resulting theory of quantifier elimination as short as possible. τ^* is not good for this purpose. But fortunately, we can improve it by the following property, which can be obtained by a slight modification of the proof for Proposition 1.

Proposition 2. *Let φ and ψ be any first-order sentences without occurrences of successor constants, and in particular φ is of the form $\forall\bar{x}\exists\bar{y}\vartheta(\bar{x},\bar{y})$. Let $\bar{P}$ be any tuple of predicates. Then, over successor structures, $\mathrm{SM}(\varphi\wedge\psi;\bar{P})$ is equivalent to $\exists S\exists T\mathrm{SM}(\tau(\varphi)\wedge\psi;\bar{P},S,T)$, where S and T are auxiliary predicates introduced by τ and have no occurrences in ψ.*

Having this proposition, we can directly apply τ to each conjunct of the input formula. Based on this idea, we then devise a new translation $\tau^\diamond$ which works as follows. For every first-order sentence φ, translation $\tau^\diamond$ repeatedly do stages 1–3 until the resulting formula is universal already:

1. Lift the conjunctions in φ as many as possible. For instance, $(\psi\wedge\varrho)\vee\gamma$ will be converted to $(\psi\vee\gamma)\wedge(\varrho\vee\gamma)$.
2. Convert each conjunct of the result of stage 1 to a formula in prenex normal form.
3. Apply τ to each conjunct of the result of stage 2.

Due to the space limit, we simply illustrate it by an example.

Example 1. Let $\varphi = \exists x\forall y\theta(x,y)$ be a sentence, where $\theta(x,y)$ is a quantifier-free formula to which stage 1 cannot be applied anymore. Then, on input φ, $\tau^\diamond$ will do stages 1–3 twice. In the first round, φ will not change at stages 1–2, but after stage 3 we will obtain the following theory:

$$\neg\neg S_1(\min) \quad (7)$$

$$\forall xu((\mathrm{succ}(x,u)\wedge S_1(u))\vee\forall y\theta^{\neg\neg}(x,y)\rightarrow S_1(x)) \quad (8)$$

$$T_1(\min)\vee\forall y\theta(\min,y) \quad (9)$$

$$\forall xu(\neg(\mathrm{succ}(x,u)\wedge S_1(u))\wedge S_1(x) \rightarrow (T_1(\max)\leftrightarrow\forall y\theta(x,y))) \quad (10)$$

$$\forall xu(\mathrm{succ}(x,u)\rightarrow(T_1(x)\leftrightarrow\forall y\theta(u,y)\vee T_1(u))) \quad (11)$$

In the second round, we only explain how the translation $\tau^\diamond$ works on sentence (8). After stage 1, we will obtain:

$$\forall xu(\mathrm{succ}(x,u)\wedge S_1(u)\rightarrow S_1(x)) \quad (12)$$

$$\forall x(\forall y\theta^{\neg\neg}(x,y)\rightarrow S_1(x)) \quad (13)$$

And we will have (12) and $\forall x\exists y(\theta^{\neg\neg}(x,y)\rightarrow S_1(x))$ after stage 2. Furthermore, the resulting theory of stage 3 will consist of (12) and the following sentences:

$$\forall x\neg\neg S_2(x,\min) \quad (14)$$

$$\forall xyu((\mathrm{succ}(y,u)\wedge S_2(x,u))\vee\lambda(x,y)\rightarrow S_2(x,y)) \quad (15)$$

$$\forall x(T_2(x,\min)\vee\eta(x,\min)) \quad (16)$$

$$\forall xyu(\neg(\mathrm{succ}(y,u)\wedge S_2(x,u))\wedge S_2(x,y) \rightarrow (T_2(x,\max)\leftrightarrow\eta(x,y))) \quad (17)$$

$$\forall xyu(\mathrm{succ}(y,u)\rightarrow(T_2(x,y)\leftrightarrow\eta(x,u)\vee T_2(x,u))) \quad (18)$$

where $\lambda(x,y)$ stands for the formula $\theta^{\neg\neg}(x,y)\rightarrow\neg\neg S_1(x)$ and $\eta(x,y)$ for the formula $\theta^{\neg\neg}(x,y)\rightarrow S_1(x)$.

Remark 4. As seen in the above example, each sentence in the resulting theory is of size[5] $O(k+n)$, where k is the number of quantifier alternating blocks in the original theory and n the maximum size of conjuncts in the original theory. Since k is usually very small, this improvement assures the resulting logic program of the reduction will be of a reasonable size.

With the above results, a natural problem may be: is there a translation that does not use auxiliary predicates? The following theorem asserts the nonexistence of such a translation even if we only consider the case of finite structures.

Theorem 2. *Over finite structures, there exists a first-order sentence φ involving only one binary predicate constant R such that, for any universal first-order sentence ψ of the same vocabulary, $\mathrm{SM}(\varphi;R)$ is not equivalent to $\mathrm{SM}(\psi;R)$.*

Proof. (Sketch) Let σ be a vocabulary consisting of a binary predicate constant R. Let ODD* be the class of finite σ-structures $\mathfrak{A}$ such that: (i) $|A|$ is odd or equals to 2, and (ii) $\mathfrak{A}(R)=A^2$. Let ζ be the conjunction of following sentences:

$$\forall xR(x,x)\wedge\forall x\forall y(R(x,y)\rightarrow R(y,x)) \quad (19)$$

$$\forall x\forall y\forall z(R(x,y)\wedge R(y,z)\rightarrow R(x,z)) \quad (20)$$

$$\forall x\exists y(x\neq y\wedge R(x,y)) \quad (21)$$

$$\forall x\forall y\forall z(x\neq y\wedge x\neq z\wedge y\neq z\wedge R(x,y)\wedge R(x,z) \rightarrow\forall u\forall v(R(u,v)\vee\neg R(u,v))) \quad (22)$$

Let ξ be the sentence $\forall w\forall w'(\neg\neg R(w,w')\wedge(R(w,w')\vee\zeta))$. Then we can show that $\mathrm{SM}(\xi;R)$ encodes exactly ODD*, i.e., a finite σ-structure $\mathfrak{A}$ is an R-stable model of ξ iff $\mathfrak{A}\in$ ODD*.

On the other hand, we can show there is no universal first-order σ-sentence ψ such that ODD* is exactly the class of finite R-stable models of ψ. Since for every universal first-order σ-sentence ψ, $\mathrm{SM}(\psi;R)$ can be written as a second-order sentence of the form $\forall\bar{X}\forall\bar{x}\exists\bar{y}\vartheta$, where $\bar{X}$ is a tuple of predicate variables, $\bar{x},\bar{y}$ two tuples of individual variables, and ϑ a quantifier-free formula. By a slight modification to the proof for Theorem 2.2 of [Eiter *et al.*, 1996], we can show that ODD* cannot be defined by any of such sentences. □

[5] We count the occurrences of quantifiers, connectives and atoms.

4 From Circumscription to Stable Models

According to Lin's transformation (cf. [Lin and Zhou, 2011]), stable model semantics can be translated to circumscription in some sense. In this section, we give a translation that embeds circumscription into stable model semantics. Surprisingly, it does not involve any auxiliary predicates, and works for arbitrary structures. Together with the reduction in previous section, we can then translate first-order theories of circumscription into logic programs under stable model semantics.

Let φ be a first-order sentence, and $\bar{P}$ a tuple of predicates. Clearly, φ can be converted to a sentence $\widehat{\varphi}$ in negational normal form without implication (i.e, a formula built from literals by $\wedge$, $\vee$, $\forall$ and $\exists$).[6] Then, we define the translation $\pi_{\bar{P}}$ as a mapping that maps each first-order formula φ to the formula

$$\widehat{\varphi}^{\neg\neg} \wedge \left(\widetilde{\varphi} \vee \bigwedge_{R \in \bar{P}} \forall \bar{u} \left(R(\bar{u}) \vee \neg R(\bar{u}) \right) \right) \tag{23}$$

where $\widehat{\varphi}^{\neg\neg}$ is same as in the previous section, and $\widetilde{\varphi}$ obtained from $\widehat{\varphi}$ by substituting $P(\bar{t}) \rightarrow \bigwedge_{R \in \bar{P}} \forall \bar{u} \left(R(\bar{u}) \vee \neg R(\bar{u}) \right)$ (denoted by $P^{\sim}(\bar{t})$) for each literal $\neg P(\bar{t})$ such that $P \in \bar{P}$.

Theorem 3. *$\mathrm{CIRC}(\varphi; \bar{P})$ is equivalent to $\mathrm{SM}(\pi_{\bar{P}}(\varphi); \bar{P})$ for any first-order sentence φ and any tuple $\bar{P}$ of predicates.*

Proof. Clearly, we have that $\mathrm{CIRC}(\varphi; \bar{P})$ is equivalent to $\mathrm{CIRC}(\widehat{\varphi}; \bar{P})$. By definition, $\mathrm{CIRC}(\widehat{\varphi}; \bar{P})$ is exactly the sentence $\widehat{\varphi} \wedge \forall \bar{X}_{\bar{P}} (\bar{X}_{\bar{P}} < \bar{P} \rightarrow \neg \widehat{\varphi}(\bar{X}_{\bar{P}}))$ and $\mathrm{SM}(\pi_{\bar{P}}(\varphi); \bar{P})$ is the sentence $\pi_{\bar{P}}(\varphi) \wedge \forall \bar{X}_{\bar{P}} (\bar{X}_{\bar{P}} < \bar{P} \rightarrow \neg \mathrm{St}(\pi_{\bar{P}}(\varphi); \bar{P}))$. By a routine simplification, the latter is equivalent to

$$\widehat{\varphi} \wedge \forall \bar{X}_{\bar{P}} \left(\bar{X}_{\bar{P}} < \bar{P} \rightarrow \neg \left(\mathrm{St}(\widetilde{\varphi}; \bar{P}) \vee \delta \right) \right) \tag{24}$$

where δ denotes $\bigwedge_{R \in \bar{P}} \forall \bar{u} \left(X_R(\bar{u}) \vee \neg R(\bar{u}) \right)$. Let σ be the vocabulary of φ, and let $\mathfrak{A}$ be a σ-structure and α an assignment in $\mathfrak{A}$ that satisfies $\bar{X}_{\bar{P}} < \bar{P}$. It is easy to see that δ is always false in α. As $\widehat{\varphi}$ is in negational normal form without implication, it is not difficult to see that α satisfies $\mathrm{St}(\widetilde{\varphi}; \bar{P})$ iff α satisfies $\widehat{\varphi}(\bar{X}_{\bar{P}})$ (Notice that α satisfies $\mathrm{St}(P^{\sim}(\bar{t}); \bar{P})$ iff α satisfies $\neg X_P(\bar{t})$ since δ is false in α). So we can conclude that formula (24) is equivalent to $\mathrm{CIRC}(\widehat{\varphi}; \bar{P})$, which completes the proof immediately. □

As we have seen in the definition of circumscription, no varying predicates are considered. For some applications of circumscription (e.g., model update), however, varying predicates may be important. In this case, we should firstly use the translation given in [Cadoli *et al.*, 1992] to eliminate varying predicates, and then transform the result to a logic program.

5 Benchmarks and Experiments

Based on the translation approach described earlier, we have implemented a prototype solver T2LP for first-order ASP. The input of our solver contains an arbitrary first-order theory and a class of finite extensional databases. The solver firstly "compiles" (so only once) the theory to a logic program, and then computes stable models by invoking a traditional ASP solver on the logic program and on each extensional database.

Two benchmarks are used to test T2LP. The first one is the clique coloring problem and the second one the quantified boolean satisfiability problem. Both benchmarks can be easily encoded in first-order theories with existential quantifiers. To the best of our knowledge, no natural encodings in disjunctive logic programs are available for these problems.

5.1 Clique Coloring Problem

The problem of clique coloring is Σ_2^p-complete and has been thoroughly studied in graph theory. Here we consider only the version of 2-color that still remains Σ_2^p-complete (cf. [Schaefer and Umans, 2002]). The problem is defined as follows. Given a graph $G = (V, E)$, does G have a 2-clique-coloring? Herein, a 2-clique-coloring is a function from V to $\{0, 1\}$ such that each maximal clique of G contains two vertices of different colors. Let Γ denote the following first-order theory:

$$\forall x(\mathrm{col}(x) \vee \underline{\mathrm{col}}(x)) \wedge \forall x(\mathrm{set}(x) \vee \underline{\mathrm{set}}(x)) \tag{25}$$
$$\exists x \exists y(\mathrm{set}(x) \wedge \mathrm{set}(y) \wedge \mathrm{col}(x) \wedge \underline{\mathrm{col}}(y)) \rightarrow \mathrm{ok} \tag{26}$$
$$\exists x \exists y(\mathrm{set}(x) \wedge \mathrm{set}(y) \wedge x \neq y \wedge \neg \mathrm{edg}(x, y)) \rightarrow \mathrm{ok} \tag{27}$$
$$\exists x(\underline{\mathrm{set}}(x) \wedge \forall y(\underline{\mathrm{set}}(y) \vee \mathrm{edg}(x, y))) \rightarrow \mathrm{ok} \tag{28}$$
$$\forall x \forall y(x \neq y \rightarrow \underline{\mathrm{set}}(x) \vee \underline{\mathrm{set}}(y)) \rightarrow \mathrm{ok} \tag{29}$$
$$(\mathrm{ok} \rightarrow \forall x(\mathrm{set}(x) \wedge \underline{\mathrm{set}}(x))) \wedge \neg\neg \mathrm{ok} \tag{30}$$

where the vocabulary of Γ consists of unary predicate constants col, $\underline{\mathrm{col}}$, set, $\underline{\mathrm{set}}$, of a binary predicate constant edg, of a nullary predicate constant ok. Let $\bar{P}$ denote the tuple of above predicate constants of arity less than 2. Every graph $G = (V, E)$ is encoded to be a $\{\mathrm{edg}\}$-structure $\mathfrak{A}_G$ such that the domain is V and edg is interpreted as E. By a reason similar to that in Examples 2 of [Eiter *et al.*, 1997], $\mathfrak{A}_G$ is a model of $\exists \bar{P} \mathrm{SM}(\Gamma; \bar{P})$ iff G has a 2-clique-coloring.

5.2 QBF Satisfiability Problem

It is well-known that the following problem is Σ_2^p-complete: Given a quantified boolean formula $\forall \bar{p} \varphi$ where $\bar{p}$ is a tuple of propositional variables and φ is quantifier-free and in disjunctive normal form, is the formula satisfiable? To show our solver works well for theories with more quantifier alternations, we modify the original problem by allowing φ to be a conjunction of a CNF-formula and a DNF-formula. Clearly, the new problem remains Σ_2^p-complete. Let tr, $\underline{\mathrm{tr}}$ and un be unary predicate constants, let pc, nc, pd and nd be binary predicate constants, and let ok be a nullary predicate constant. Then Δ is defined as the following first-order theory:

$$\forall x(\mathrm{tr}(x) \vee \underline{\mathrm{tr}}(x)) \tag{31}$$
$$\forall x \exists y((\mathrm{pc}(x, y) \wedge \mathrm{tr}(y)) \vee (\mathrm{nc}(x, y) \wedge \underline{\mathrm{tr}}(y))) \rightarrow \mathrm{ok} \tag{32}$$
$$\exists x \forall y((\mathrm{pd}(x, y) \rightarrow \mathrm{tr}(y)) \wedge (\mathrm{nd}(x, y) \rightarrow \underline{\mathrm{tr}}(y))) \rightarrow \mathrm{ok} \tag{33}$$
$$\forall x(\mathrm{ok} \wedge \mathrm{un}(x) \rightarrow \mathrm{tr}(x) \wedge \underline{\mathrm{tr}}(x)) \wedge \neg\neg \mathrm{ok} \tag{34}$$

Let σ denote the set of predicate constants $\mathrm{un}, \mathrm{pc}, \mathrm{nc}, \mathrm{pd}, \mathrm{nd}$. For every input, i.e., a formula $\forall \bar{p} \varphi$ of the form mentioned previously, we can construct a σ-structure $\mathfrak{A}$ to encode it in the following way: (i) the domain A of $\mathfrak{A}$ is the integer set $\mathbb{Z}_n = \{0, \ldots, n-1\}$, where n is the largest of the number of clauses of the CNF-formula, the number of clauses

[6] By the definition, the equivalence between formulae in classical first-order logic implies that in circumscription. So, it suffices to find such a translation in classical first-order logic, which clearly exists.

of the DNF-formula, and the number of propositional variables occurring in φ; (ii) $(i,k) \in \mathfrak{A}(\mathrm{pc})$ ($(i,k) \in \mathfrak{A}(\mathrm{nc})$) if the k-th variable has a positive (negative) occurrence in the i-th clause of the CNF-formula; (iii) $(i,k) \in \mathfrak{A}(\mathrm{pd})$ ($(i,k) \in \mathfrak{A}(\mathrm{nd})$) if the k-th variable has a positive (negative) occurrence in the i-th clause of the DNF-formula; (iv) $i \in \mathfrak{A}(\mathrm{un})$ iff $p_i \in \bar{p}$. Similarly, it is easy to show that $\mathfrak{A}$ satisfies $\exists \mathrm{tr} \exists \underline{\mathrm{tr}} \exists \mathrm{ok} \mathrm{SM}(\Delta; \sigma, \mathrm{tr}, \underline{\mathrm{tr}}, \mathrm{ok})$ iff $\forall \bar{p} \varphi$ is satisfiable.

5.3 Experimental Results

Experimental results for these two benchmarks are presented in Tables 1 and 2 respectively. The parameter n in Table 1 (Table 2) denotes the number of vertices in the graph G (the number of propositional variables in the formula φ). Each instance for either experiment is randomly generated. In particular, in the experiment of QBF satisfiability, both the number of CNF clauses and the number of DNF clauses equals to n. Non-integer real numbers in the tables figure the run time (in seconds) of the solver to compute the first stable model. If the time exceeds one hour, we simply write it as "$> 1h$" in the table. All experiments were run on a 2.7GHz PC on Linux.

As we can see in the tables, each instance of each benchmark is computed twice by our translation calling different ASP solvers: CLASPD and DLV. In general, the solver using CLASPD is faster than that using DLV. Though our translation is of general purpose and both benchmarks are Σ_2^p-complete, the experimental results seem quite optimistic.

n	Instance 1		Instance 2		Instance 3		Instance 4		Instance 5	
	T2LP+ CLASPD	T2LP+ DLV	T2LP+ CLASPD	T2LP+ DLV	T2LP+ CLASPD	T2LP+ DLV	T2LP+ CLASPD	T2LP+ DLV	T2LP+ CLASPD	T2LP+ DLV
20	0.087	2.017	0.138	1.573	0.118	7.399	0.111	13.739	0.047	3.110
30	0.526	3.140	0.497	283.430	0.363	378.220	0.459	1125.382	0.312	299.324
40	0.690	$> 1h$	1.369	3153.779	1.462	$> 1h$	1.436	$> 1h$	1.290	$> 1h$
50	2.882	$> 1h$	1.855	$> 1h$	4.484	$> 1h$	3.111	$> 1h$	3.201	$> 1h$
60	9.693	$> 1h$	4.325	$> 1h$	9.202	$> 1h$	6.233	$> 1h$	6.532	$> 1h$
70	11.274	$> 1h$	8.990	$> 1h$	14.094	$> 1h$	20.276	$> 1h$	11.105	$> 1h$
80	20.213	$> 1h$	23.434	$> 1h$	31.515	$> 1h$	28.726	$> 1h$	25.913	$> 1h$
90	37.344	$> 1h$	51.002	$> 1h$	33.364	$> 1h$	66.847	$> 1h$	56.597	$> 1h$
100	77.961	$> 1h$	88.103	$> 1h$	54.882	$> 1h$	82.426	$> 1h$	85.403	$> 1h$

Table 1: Experimental Results for Clique Coloring

n	Instance 1		Instance 2		Instance 3		Instance 4		Instance 5	
	T2LP+ CLASPD	T2LP+ DLV	T2LP+ CLASPD	T2LP+ DLV	T2LP+ CLASPD	T2LP+ DLV	T2LP+ CLASPD	T2LP+ DLV	T2LP+ CLASPD	T2LP+ DLV
20	0.088	0.135	0.146	0.721	0.121	0.595	0.090	0.091	0.161	1.465
30	0.243	0.232	0.348	181.403	0.158	4.853	0.209	0.700	0.457	0.238
40	0.404	$> 1h$	0.883	82.879	1.426	123.955	1.542	3099.264	0.417	0.414
50	0.627	1.223	1.502	$> 1h$	0.619	0.904	0.689	2.728	0.661	1.293
60	2.306	$> 1h$	0.867	3.027	1.001	1.678	16.643	$> 1h$	1.863	$> 1h$
70	2.318	2.965	1.335	4.445	1.812	$> 1h$	1.547	383.764	1.338	4.374
80	2.102	8.958	1.935	$> 1h$	6.718	$> 1h$	3.869	1682.686	2.773	7.252
90	2.598	17.494	2.585	4.126	6.801	$> 1h$	6.234	$> 1h$	7.035	$> 1h$
100	3.354	19.555	3.926	$> 1h$	11.701	$> 1h$	16.919	$> 1h$	4.656	$> 1h$
110	4.265	23.991	4.552	13.251	4.545	35.358	4.790	38.355	4.297	7.030
120	14.947	37.958	176.136	$> 1h$	5.527	$> 1h$	5.458	74.476	5.776	14.701

Table 2: Experimental Results for QBF satisfiability

6 Conclusion and Related Work

In this paper, we proposed an approach to reduce first-order theories to logic programs under stable model semantics over finite structures. The effectiveness of our approach was demonstrated by both theoretical analysis and experiments. We also discovered an embedding of circumscription into stable model semantics. These results provide useful insights for developing practical solvers for arbitrary first-order theories under either stable model semantics or circumscription.

Two different translations from first-order theories to logic programs were presented in [Kim *et al.*, 2009; Lee and Palla, 2009] and [Cabalar, 2009] respectively. However, their translations only work on some fragments, and it is not clear how their work can be soundly extended to the general case (or fragments that contain the benchmarks in this paper). In addition, even restricted to those fragments, our translation is significantly different from theirs. Also, the translation from circumscription to logic programs was considered by Janhunen and Oikarinen [2004] in the propositional case, which, in our opinion, seems not easy to be extended to the first-order case.

References

[Cabalar *et al.*, 2005] P. Cabalar, D. Pearce, and A. Valverde. Reducing propositional theories in equilibrium logic to logic programs. In *Proc. EPIA'05*, pages 4–17, 2005.

[Cabalar, 2009] P. Cabalar. Existential quantifiers in the rule body. In *Proc. WLP'09*, pages 59–74, 2009.

[Cadoli *et al.*, 1992] M. Cadoli, T. Eiter, and G. Gottlob. An efficient method for eliminating varying predicates from a circumscription. *Artificial Intelligence*, 54:397–410, 1992.

[Eiter *et al.*, 1996] T. Eiter, G. Gottlob, and Y. Gurevich. Normal forms for second-order logic over finite structures, and classification of NP optimization problems. *Annals of Pure and Applied Logic*, 78:111–125, 1996.

[Eiter *et al.*, 1997] T. Eiter, G. Gottlob, and H. Mannila. Disjunctive datalog. *ACM Transactions on Database Systems,*, 22:364–418, 1997.

[Ferraris *et al.*, 2009] P. Ferraris, J. Lee, V. Lifschitz, and R. Palla. Symmetric splitting in the general theory of stable models. In *Proc. IJCAI'09*, pages 797–803, 2009.

[Ferraris *et al.*, 2011] P. Ferraris, J. Lee, and V. Lifschitz. Stable models and circumscription. *Artificial Intelligence*, 175:236–263, 2011.

[Janhunen and Oikarinen, 2004] T. Janhunen and E. Oikarinen. Capturing parallel circumscription with disjunctive logic programs. In *Proc. JELIA'04*, pages 134–146, 2004.

[Kim *et al.*, 2009] T.-W. Kim, J. Lee, and R. Palla. Circumscriptive event calculus as answer set programming. In *Proc. IJCAI'09*, pages 823–829, 2009.

[Lee and Palla, 2009] J. Lee and R. Palla. System F2LP – computing answer sets of first-order formulas. In *Proc. LPNMR'09*, pages 515–521, 2009.

[Lifschitz, 1985] V. Lifschitz. Computing circumscription. In *Proc. IJCAI'85*, pages 121–127, 1985.

[Lin and Zhou, 2011] F. Lin and Y. Zhou. From answer set logic programming to circumscription via logic of GK. *Artificial Intelligence*, 175:264–277, 2011.

[McCarthy, 1980] J. McCarthy. Circumscription – A form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980.

[Pearce and Valverde, 2005] D. Pearce and A. Valverde. A first order nonmonotonic extension of constructive logic. *Studia Logica*, 80:321–346, 2005.

[Schaefer and Umans, 2002] M. Schaefer and C. Umans. Completeness in the polynomial-time hierarchy: A compendium. *SIGACT News*, 33:32–49, 2002.