Ordered Completion for Logic Programs with Aggregates

Vernon Asuncion, Yan Zhang and Yi Zhou
Intelligent Systems Laboratory
School of Computing, Engineering and Mathematics
University of Western Sydney, NSW, Australia

Abstract

In this paper, we show that first-order logic programs with monotone aggregates under the stable model semantics can be captured in classical first-order logic. More precisely, we extend the notion of ordered completion for logic programs with a large variety of aggregates so that every stable model of a program with aggregates corresponds to a classical model of its enhanced ordered completion, and vice versa.

Introduction

In the last three decades, Answer Set Programming (ASP) has emerged as a predominant declarative programming paradigm in the area of knowledge representation and logic programming (Baral 2003). One of the main focuses of recent advances in ASP is first-order answer set programming (Ferraris, Lee, and Lifschitz 2011; Lin and Zhou 2011), which aims to characterize the answer set or stable model semantics of logic programs directly on a first-order level. This is significantly different from the traditional approach by grounding (Gelfond and Lifschitz 1988), which is essentially propositional.

The stable model semantics of first-order logic programs is defined in second-order logic (Ferraris, Lee, and Lifschitz 2011; Lin and Zhou 2011). Interestingly, Asuncion et al. (2012) recently proposed a notion of ordered completion (a first-order sentence) for normal logic programs, and showed that the stable models of a normal logic program are exactly corresponding to the models of its ordered completion on finite structures. This work is not only theoretically interesting but also practically relevant as it initiates a new direction of ASP solver. A first implementation (Asuncion et al. 2012) shows that this new direction is promising as it performs surprisingly good for the Hamiltonian Circuit program on big instances.

However, this work is not adequate enough, at least from a practical point of view, because it cannot handle aggregates - a very important building block in ASP that is widely used in many applications. The reason why aggregates are crucial in answer set solving is twofold. First, it can simplify the representation task. For many applications, one can write a simpler and more elegant logic program by using aggregates, for instance, the job scheduling program (Pontelli, Son, and Elkabani 2004). More importantly, it can improve the efficiency of ASP solving (Gebser et al. 2009). Normally, the program using aggregates can be solved much faster (Faber, Leone, and Pfeifer 2003).

Incorporating aggregates in ordered completion is a challenging task. In the propositional case, the complexity of normal logic programs with arbitrary aggregates is beyond the class of NP (Ferraris 2011). This possibly suggests that, most likely, normal logic programs with arbitrary aggregates cannot be captured in first-order logic. Hence, we are mainly focused on (anti)monotone aggregates. Even for this case, the task is still very complicated as aggregate atoms, on one hand, can express some features of existential quantifiers, and on the other hand, contribute to the loops (Chen et al. 2006; Lee and Meng 2009) of the program.

In this paper, we show that first-order normal logic programs with a large variety of (anti)monotone aggregates (covering the types of aggregates in most benchmark programs) can indeed be captured in first-order logic. More precisely, we extend the notion of ordered completion for these kind of programs so that every stable model of a program is corresponding to a classical model of its enhanced ordered completion, and vice versa. Our proof technique is significantly different from the original proof for ordered completion in (Asuncion et al. 2012), thus also sheds new insights in first-order ASP (with aggregates).

Preliminaries

We first consider a first-order language without functions but with equality $=$. A *signature* contains a set of constants and a set of predicates. A *structure* $\mathcal{A}$ of a signature is a tuple

$$(Dom(\mathcal{A}), c_1^{\mathcal{A}}, \ldots, c_l^{\mathcal{A}}, P_1^{\mathcal{A}}, \ldots, P_m^{\mathcal{A}}),$$

where $c_i^{\mathcal{A}}, 1 \leq i \leq l$, and $P_j^{\mathcal{A}}, 1 \leq j \leq m$ are interpretations for constant c_i and predicate P_j respectively. We call a structure *finite* if $Dom(\mathcal{A})$ is finite.

Aggregates

Now we introduce the aggregate component. For this purpose, we enhance our first-order language with a background theory for numbers, similar to Satisfiability Modular Theories (SMT) (Nieuwenhuis, Oliveras, and Tinelli 1999).

An *aggregate atom* δ is of the form

$$\text{OP}\langle \mathbf{v} : \exists \mathbf{w} Bd(\delta) \rangle \preceq N, \tag{1}$$

where:

- OP $\in \{\text{CARD}, \text{SUM}, \text{PROD}, \text{MIN}, \text{MAX}\}$ is an aggregate function for cardinality, sum, product, minimum, and maximum respectively;
- $Bd(\delta)$ (the body of δ) is of the form

$$Q_1(\mathbf{y_1}) \wedge \cdots \wedge Q_s(\mathbf{y_s}) \wedge \neg R_1(\mathbf{z_1}) \wedge \cdots \wedge \neg R_t(\mathbf{z_t}), \tag{2}$$

  where each $Q_i(\mathbf{y_i})$ $(1 \leq i \leq s)$, $R_j(\mathbf{z_j})$ $(1 \leq j \leq t)$ are atoms in the first-order language;
- $\mathbf{v}$ and $\mathbf{w}$ are distinctive tuples of variables mentioned in (2), and $\mathbf{v} \cap \mathbf{w} = \emptyset$.
- $\preceq \in \{<, \leq, =, \geq, >\}$ is a comparison operator on numbers;
- finally N is a number.

For convenience, we use $Pos(\delta)$ and $Neg(\delta)$ to denote the sets $\{Q_1(\mathbf{y_1}), \ldots, Q_s(\mathbf{y_s})\}$ and $\{R_1(\mathbf{z_1}), \ldots, R_t(\mathbf{z_t})\}$ respectively.

For an aggregate atom δ of the form (1), let $\mathbf{y}$ be the set of free variables in it, i.e. the variables in δ but not in either $\mathbf{v}$ or $\mathbf{w}$. Let $\mathcal{A}$ be a structure and $\mathbf{a}$ a set of domain elements that matches $\mathbf{y}$, $\mathcal{A} \models \delta[\mathbf{y}/\mathbf{a}]$ if and only if:

1. The multiset[1]

$$M = \{\!\{\mathbf{c}[1] \mid \mathcal{A} \models Bd(\delta)[\mathbf{ywv}/\mathbf{abc}], \mathbf{b} \in X, \mathbf{c} \in Y\}\!\}$$

   is in the domain of OP, where X and Y are the sets of domain tuples that match $\mathbf{w}$ and $\mathbf{v}$ respectively;
2. $\text{OP}(M) \preceq N$.

Based on this, the satisfaction relation between structures (along with assignments on variables) and formulas (with aggregates) is defined recursively as usual in classical logic.

An aggregate atom OP is said to be: *monotone* if $M_1, M_2 \in Dom(\text{OP})$ (i.e. in the domain of OP), $M_1 \subseteq M_2$, and $\text{OP}(M_1) \preceq N$ imply that $\text{OP}(M_2) \preceq N$; and *anti-monotone* if $M_1, M_2 \in Dom(\text{OP})$, $M_2 \subseteq M_1$, and $\text{OP}(M_1) \preceq N$ imply that $\text{OP}(M_2) \preceq N$.

Stable Models Semantics for Logic Programs

In this paper, we consider normal logic programs with aggregates mentioned above. A *rule* r is of the form

$$\beta_1 \wedge \cdots \wedge \beta_n \wedge \neg\gamma_1 \wedge \cdots \wedge \neg\gamma_m \rightarrow \alpha, \tag{3}$$

where α is an atom $P(\mathbf{x})$; β_i $(1 \leq i \leq n)$ and γ_j $(1 \leq j \leq m)$ are either an equality atom $t_1 = t_2$, or an atom $P(\mathbf{t})$, or an aggregate atom of the form (1). In the literature, this form is also written as

$$\alpha \leftarrow \beta_1, \ldots, \beta_n, \mathsf{not}\, \gamma_1, \ldots, \mathsf{not}\, \gamma_m.$$

[1] In the following, $\mathbf{c}[1]$ denotes the first component (or position) of $\mathbf{c}$.

For convenience, we use $Head(r)$ and $Body(r)$ to denote α and $\beta_1 \wedge \cdots \wedge \beta_n \wedge \neg\gamma_1 \wedge \cdots \wedge \neg\gamma_m$ respectively.

A *program* is a finite set of rules. The signature of a program Π, denoted by $\tau(\Pi)$, consists of all constants and predicates occurred in Π. A predicate in a program is said to be *intensional* if it occurs in some head of the program, and *extensional* otherwise. We use $\mathcal{P}_{int}(\Pi)$ to denote the set of all intensional predicates of Π.

Example 1 Consider the following program Π with a simple aggregate atom

$$\text{CARD}\langle x : P(x) \rangle = 2 \rightarrow Q(y), \tag{4}$$
$$Q(x) \rightarrow P(x), \tag{5}$$
$$R(x) \rightarrow P(x). \tag{6}$$

Here, P and Q are intensional while R is extensional. Rule (4) states that if $P(x)$ holds exactly for two elements, then we have $Q(y)$.

The stable model semantics of programs (with aggregates) is defined by a second-order sentence. Let Π be a program and $\mathcal{P}_{int}(\Pi) = \{P_1, \ldots, P_n\}$. Let $\mathbf{U} = \{U_1, \ldots, U_n\}$ be a set of new predicates such that each $U_i, 1 \leq i \leq n$, matches the arity of P_i. Let $\rho = P(\mathbf{x})$ be an atom. By ρ^*, we denote $U_i(\mathbf{x})$ if $P = P_i$ where $P_i \in P_{int}(\Pi)$, and $P(\mathbf{x})$ itself if $P \notin \mathcal{P}_{int}(\Pi)$. Let δ be an aggregate atom of form (1). By δ^*, we denote the formula

$$(\text{OP}\langle \mathbf{v} : \exists \mathbf{w} Bd(\delta)^* \rangle \preceq N) \wedge (\text{OP}\langle \mathbf{v} : \exists \mathbf{w} Bd(\delta) \rangle \preceq N),$$

where $Bd(\delta)^* = Q_1^*(\mathbf{y_1}) \wedge \cdots \wedge Q_s^*(\mathbf{y_s}) \wedge \neg R_1(\mathbf{z_1}) \wedge \cdots \wedge \neg R_t(\mathbf{z_t})$. Now given a rule r of the form (3), by r^*, we denote the universal closure of the following formula

$$\beta_1^* \wedge \cdots \wedge \beta_m^* \wedge \neg\gamma_1 \wedge \cdots \wedge \neg\gamma_l \rightarrow \alpha^*.$$

Finally, by $SM(\Pi)$[2], we denote the following second-order sentence

$$\bigwedge_{r \in \Pi} \widehat{r} \wedge \neg\exists \mathbf{U}(\mathbf{U} < \mathcal{P}_{int}(\Pi) \wedge \bigwedge_{r \in \Pi} r^*), \tag{7}$$

where $\widehat{r}$ is the universal closure of r and $\mathbf{U} < \mathcal{P}_{int}(\Pi)$ is the abbreviation of the formula

$$\bigwedge_{1 \leq i \leq n} \forall \mathbf{x}(U_i(\mathbf{x}) \rightarrow P_i(\mathbf{x})) \wedge \neg \bigwedge_{1 \leq i \leq n} (P_i(\mathbf{x}) \rightarrow U_i(\mathbf{x})).$$

Definition 1 (stable model) *Let Π be a program. A $\tau(\Pi)$-structure $\mathcal{A}$ is said to be a* stable model *of Π if $\mathcal{A}$ is a model of $SM(\Pi)$.*

Ordered Completion

Asuncion et al. (2012) showed that the stable models semantics for normal logic programs without aggregates can be captured in classical first-order logic on finite structures. More precisely, they defined the notion of ordered completion, which is a modification of Clark's completion by adding the track of the derivation order, and showed that the

[2] Notice that this definition is essentially equivalent to the one defined in (Bartholomew, Lee, and Meng 2011) but restricted into normal programs for simplicity.

finite stable models of a normal program corresponds exactly to the models of its ordered completion.

Let Π be a program without aggregate atoms. The *modified completion* of Π, denoted by $MComp(\Pi)$, is

$$\widehat{\Pi} \wedge \bigwedge_{P \in \mathcal{P}_{int}(\Pi)} \forall \mathbf{x}(P(\mathbf{x}) \rightarrow \bigvee_{\substack{r \in \Pi \\ P(\mathbf{x}) = Head(r)}} \exists \mathbf{y}(Body(r) \wedge Pos(\widehat{r) < P}(\mathbf{x}))), \tag{8}$$

where $\widehat{\Pi} = \bigwedge_{r \in \Pi} \widehat{r}$ and $Pos(\widehat{r) < P}(\mathbf{x})$ is[3]

$$\bigwedge_{Q(\mathbf{y}) \in Pos(r), Q \in \mathcal{P}_{int}(\Pi)} (\leq_{QP} (\mathbf{yx}) \wedge \neg \leq_{PQ} (\mathbf{xy})),$$

where $\leq_{QP}$ and $\leq_{PQ}$ are new predicates called *comparison predicates* for keeping track of the derivation order. Finally, the *ordered completion* of Π, denoted by $OC(\Pi)$, is $MComp(\Pi) \wedge Trans(\Pi)$, where $Trans(\Pi)$ is

$$\bigwedge_{P,Q,R \in \mathcal{P}_{int}(\Pi)} \forall \mathbf{xyz}(\leq_{PQ} (\mathbf{xy}) \wedge \leq_{QR} (\mathbf{yz}) \rightarrow \leq_{PR} (\mathbf{xz})).$$

Theorem 1 *(Asuncion et al. 2012) Let Π be a normal logic program without aggregate atoms and $\mathcal{A}$ a finite $\tau(\Pi)$-structure. Then, $\mathcal{A}$ is a stable model of Π if and only if $\mathcal{A}$ can be expanded to a model of $OC(\Pi)$.*

Ordered Completion for Programs with Aggregates

From a theoretical point of view, ordered completion makes important progressions on understanding first-order answer set programming. Firstly, it shows that the stable model semantics is simply Clark's completion plus derivation order. Secondly, it clarifies the relationship between first-order normal ASP and classical first-order logic. More precisely, every normal answer set program can be captured by a first-order sentence with some new predicates on finite structures. Surprisingly, this fails to be true on infinite structures or if no new predicates are allowed (Asuncion et al. 2012).

Ordered completion is not only theoretically interesting but also practically important. It initiates a new direction of ASP solver by first translating a normal logic program to its ordered completion, then working on finding a model of this first-order sentence (Asuncion et al. 2012). A first implementation shows that this new direction is promising as it performs surprisingly good on the Hamiltonian Circuit program (Niemelä 1999), especially on big instances.

However, ordered completion can hardly be used beyond the Hamiltonian Circuit program because it cannot handle aggregates - a very important building block in ASP that is widely used in many applications. As far as we are concerned, most benchmark programs contain aggregate constructs (Calimeri et al. 2011). Aggregates are crucial in answer set solving because on one hand, it can simplify the representation task, and on the other hand, it can improve the efficiency of ASP solving (Gebser et al. 2009).

In this paper, we consider to incorporate aggregates in ordered completion. However, this is a challenging task. As shown in (Asuncion et al. 2012), ordered completion cannot be extended for disjunctive programs because they are not in the same complexity level. Similarly, normal answer set programs with arbitrary aggregates has the same complexity level as disjunctive programs, which is beyond the complexity of naive normal programs. This suggests that, most likely, normal logic programs with arbitrary aggregates cannot be captured in first-order logic, thus not by ordered completion.

Hence, we are mainly focused on some special classes of aggregates, for instance, (anti)monotone aggregates. Even for this case, the task is still very complicated. One observation is that aggregate atoms can express some features of existential quantifiers, which is even more complicated than disjunctions in first-order ASP. For instance, recall the program specified in Example 1. The first rule, i.e. rule (4), is actually equivalent to the following rule $\exists^{=2} P(x) \rightarrow Q(y)$, where $\exists^{=2} P(x)$ is a shorthand of

$$\exists xz(x \neq z \wedge P(x) \wedge P(z) \wedge \forall u(P(u) \rightarrow u = x \vee u = z)),$$

meaning that $P(x)$ holds exactly for 2 elements.

Another observation is that aggregate atoms that are not anti-monotone[4] contribute to the first-order loops (Chen et al. 2006; Lee and Meng 2009) of the program.[5] Again, consider the program in Example 1. If we ignore the aggregate atoms, then this program has no loops. But the stable models of the program cannot be captured by its Clark's completion (Clark 1978). This means that the aggregate atom in rule (4) indeed plays a role to form a loop of the program.

Hence, the difficult part is in identifying the gap between the aggregates that can be incorporated into ordered completion and those that cannot. In this paper, we show that a large variety of (anti)monotone aggregates can indeed be incorporated in ordered completion, which covers the types of aggregates in most benchmark programs as far as we have checked. We extend the notion of ordered completion to normal logic programs with such kind of aggregates so that every stable model of a program corresponds to a classical model of its enhanced ordered completion, and vice versa.

Formally, the aggregate functions we considered in this paper are restricted as follows.

Definition 2 *For an aggregate atom of the form (1) where* OP $\in \{$CARD, SUM, PROD, MIN, MAX$\}$,

- CARD *is a function from multisets of* domain tuples *to $\mathbb{Z}^+$, and it is defined as* 0 *on the empty multiset $\emptyset$;*
- SUM *is a function from multisets of $\mathbb{Z}^+$ to $\mathbb{Z}^+$, and it is also defined as* 0 *on $\emptyset$;*
- MIN *and* MAX *are functions from multisets of $\mathbb{Z}$ to $\mathbb{Z}$, and are undefined for $\emptyset$;*
- PROD *is a function from* multisets *of $\mathbb{N}$ to $\mathbb{N}$, and is defined as* 1 *on $\emptyset$.*

Now, we define the ordered completion for normal programs with aggregate atoms of form (1) under the restrictions in Definition 2. For convenience, we use

[3] $Pos(r)$ denotes the positive body atoms of r as usual.

[4] Aggregate atoms that can enforce a non-empty multiset.

[5] For space reasons, the readers are referred to (Chen et al. 2006) for the formal definitions about loops and some related notions.

$PosAgg(r)$ to denote the aggregate atoms from $Pos(r)$; $PosCardSumProd(r)$ to denote the cardinality, sum, and product aggregates from $Pos(r)$; and $PosMinMax(r)$ to denote the minimum and maximum aggregate atoms from $Pos(r)$ respectively.

Definition 3 (ordered completion with aggregates) *Let* Π *be a program with aggregate atoms of form (1) under the restrictions in Definition 2. The* modified completion *of* Π*, denoted by* $MComp(\Pi)$*, is*

$$\widehat{\Pi} \wedge \bigwedge_{P \in \mathcal{P}_{int}(\Pi)} \forall \mathbf{x}\{P(\mathbf{x}) \rightarrow \bigvee_{r \in \Pi, Head(r)=P(\mathbf{x})} \exists \mathbf{y}[Body(r) \wedge \widehat{Pos(r) < P}(\mathbf{x}) \wedge \widehat{PosAgg(r) < P}(\mathbf{x})]\}, \quad (9)$$

where:

- $\widehat{\Pi} = \bigwedge_{r \in \Pi} \widehat{r}$*;*
- $\widehat{Pos(r) < P}(\mathbf{x})$ *is*

$$\bigwedge_{Q(\mathbf{y}) \in Pos(r) \setminus PosAgg(r), \; Q \in \mathcal{P}_{int}(\Pi)} (\leq_{QP} (\mathbf{yx}) \wedge \neg \leq_{PQ} (\mathbf{xy}));$$

- *and* $\widehat{PosAgg(r) < P}(\mathbf{x})$ *is*

$$\bigwedge_{\delta \in PosCardSumProd(r), \; \preceq \in \{=,\geq,>\}} (\text{OP}\langle \mathbf{v} : \exists \mathbf{w}(Bd(\delta) \wedge \widehat{Pos(\delta) < P}(\mathbf{x})) \rangle \preceq N) \wedge$$

$$\bigwedge_{\delta \in PosMinMax(r), \; \preceq \in \{<,\leq,=,\geq,>\}} (\text{OP}\langle \mathbf{v} : \exists \mathbf{w}(Bd(\delta) \wedge \widehat{Pos(\delta) < P}(\mathbf{x})) \rangle \preceq N),$$

where for an aggregate atom δ *of the form (1) with body of the form (2),* $\widehat{Pos(\delta) < P}(\mathbf{x})$ *stands for*

$$\bigwedge_{1 \leq i \leq s, Q_i \in \mathcal{P}_{int}(\Pi)} (\leq_{Q_i P} (\mathbf{y_i x}) \wedge \neg \leq_{PQ_i} (\mathbf{x y_i})). \quad (10)$$

Finally, the ordered completion *of* Π*, denoted by* $OC(\Pi)$*, is again* $MComp(\Pi) \wedge Trans(\Pi)$.

Let us take a closer look at Definition 3. First of all, for non-aggregate atoms, we treat them the same way as in the original definition of ordered completion. For aggregate atoms, we distinguish two cases. For those atoms $\text{OP}\langle \mathbf{v} : \exists \mathbf{w} Bd(\delta) \rangle \preceq N$ where $\text{OP} \in \{\text{CARD}, \text{SUM}, \text{PROD}\}$ and $\preceq \in \{<, \leq\}$ and those atoms negatively occurring in the rule, we do not need to pay extra attentions. However, for the rest of the positive aggregate atoms δ, we need to enforce the comparison assertions, i.e. formula (10). This is because, for the latter kind of aggregate atoms, we need to keep track of the derivation order. However, for the former kind, this is not necessary because they do not enforce the condition of a non-empty multiset, i.e., since anti-monotone.

Example 2 (*Example 1 continued*) Consider again the program Π in Example 1. Then, $OC(\Pi)$ is

$$\forall y(\text{CARD}\langle x : P(x) \rangle = 2 \rightarrow Q(y)) \quad (11)$$
$$\wedge \forall x(Q(x) \rightarrow P(x)) \wedge \forall x(R(x) \rightarrow P(x)) \quad (12)$$
$$\wedge \forall x(P(x) \rightarrow R(x) \vee (Q(x) \wedge \widehat{Q(x) < P}(x))) \quad (13)$$
$$\wedge \forall y(Q(y) \rightarrow (\text{CARD}\langle x : P(x) \rangle = 2) \wedge \text{CARD}\langle x : P(x) \wedge \widehat{P(x) < Q}(y) \rangle = 2) \quad (14)$$
$$\wedge Trans(\Pi), \quad (15)$$

where $\widehat{Q(x) < P}(x)$ and $\widehat{P(x) < Q}(y)$ are the shorthand of $\leq_{QP} (x,x) \wedge \neg \leq_{PQ} (x,x)$ and $\leq_{PQ} (x,y) \wedge \neg \leq_{QP} (y,x)$ respectively. Notice that $OC(\Pi)$ can be represented in first-order logic as formula (11) is equivalent to $\forall y(\exists^{=2} x P(x) \rightarrow Q(y))$ and formula (14) is equivalent to $\forall y(Q(y) \rightarrow \exists^{=2} x(P(x) \wedge \widehat{P(x) < Q}(y)))$.

Consider a structure $\mathcal{A}$ where $R^{\mathcal{A}} = \emptyset$. Then, $\mathcal{A}$ is a stable model of Π iff $P^{\mathcal{A}} = Q^{\mathcal{A}} = \emptyset$. This in fact corresponds to a model of $OC(\Pi)$. Otherwise, suppose $P^{\mathcal{A}} \neq \emptyset$, say $P^{\mathcal{A}} = \{a, b\}$. Then since we have $P(a)$, by (13), $Q(a) \wedge \widehat{Q(a) < P}(a)$ holds. Then by (14), $\exists^{=2} x(P(x) \wedge \widehat{P(x) < Q}(a))$ must also hold. Then since $P^{\mathcal{A}} = \{a, b\}$ exactly contains 2 elements, then $P(a) < Q(a)$ must hold as well. This contradicts to $Trans(\Pi)$ and $\widehat{Q(a) < P}(a)$.

Consider a structure $\mathcal{A}$ with $R^{\mathcal{A}} = \{a, b\}$. Then, $\mathcal{A}$ is a stable model of Π iff $Dom(A)$ only contains two elements and $P^{\mathcal{A}} = Q^{\mathcal{A}} = \{a, b\}$. Now we show that this in fact corresponds to the model of $OC(\Pi)$ as well. On one hand, if $Dom(A)$ only contains two elements and $P^{\mathcal{A}} = Q^{\mathcal{A}} = \{a, b\}$, it can be easily extended to a model of $OC(\Pi)$ by forcing $\forall xy \widehat{P(x) < Q}(y)$ to be true. On the other hand, suppose $Dom(A)$ contains more than 2 elements. Then, $OC(\Pi)$ has no model. There are two cases:

Case 1: $P^{\mathcal{A}} = \{a, b\}$. Then, by (11), Q holds for all domain elements. By (12), P holds for all domain elements as well, a contradiction.

Case 2: $P^{\mathcal{A}} \neq \{a, b\}$. Then, by (12), $\{a, b\} \subset P^{\mathcal{A}}$. By (13), $Q^{\mathcal{A}}$ is not empty. Then by (14), $\exists^{=2} x P(x)$ holds, a contradiction.

In fact, it can be verified that a finite structure $\mathcal{A}$ is a stable model of Π iff it can be expanded to a model of $OC(\Pi)$.

In general, we have the following main theorem.

Theorem 2 (main theorem) *Let* Π *be a program with aggregate atoms of form (1) under the restrictions in Definition 2, and* $\mathcal{A}$ *a finite* $\tau(\Pi)$*-structure. Then,* $\mathcal{A}$ *is a stable model of* Π *if and only if* $\mathcal{A}$ *can be expanded to a model of* $OC(\Pi)$.

The proof of this theorem is rather technical and tedious. However, the basic ideas are simple. Here, the notion of externally supported set plays a crucial role. Roughly speaking, a set of grounded atoms[6] is externally supported if there

[6]Grounded atoms are of the form $P(\mathbf{a})$, where P is a predicate and $\mathbf{a}$ is a tuple of domain elements matching the arity of P.

exists a grounded atom in it and an associated rule that supports this atom (i.e. this atom is the head of the grounded rule) and whose positive body could be satisfied by external grounded atoms (i.e. grounded atoms not in this set). Then, we show that a structure is a stable model of a program if and only if it is a model of the program and every subset of grounded atoms included in this structure is externally supported. Furthermore, it is also equivalent to the fact that this structure can be expanded to a model of the ordered completion of the program.

For this purpose, we first introduce the following notion. For a program Π and structure $\mathcal{A}$ of signature σ such that $\tau(\Pi) \subseteq \sigma$, by $[\mathcal{P}_{int}(\Pi)]^{\mathcal{A}}$, we denote the set of grounded atoms $\{P(\mathbf{a}) \mid \mathbf{a} \in P^{\mathcal{A}}, P \in \mathcal{P}_{int}(\Pi)\}$.

Definition 4 (Externally supported set) *Let Π be a program with aggregate atoms of form (1) and $\mathcal{A}$ a structure of σ such that $\tau(\Pi) \subseteq \sigma$. We say that a set $S \subseteq [\mathcal{P}_{int}(\Pi)]^{\mathcal{A}}$ is* externally supported *(under $\mathcal{A}$ and Π) if there exist some $P(\mathbf{a}) \in S$ and rule $Body(r) \rightarrow P(\mathbf{x}) \in \Pi$ with local variables $\mathbf{y_r}$, such that for some assignment of the form $\mathbf{xy_r} \longrightarrow \mathbf{ab_r}$,*

1. *$\mathcal{A} \models Body(r)[\mathbf{xy_r}/\mathbf{ab_r}]$;*
2. *$(Pos(r) \setminus PosAgg(r))[\mathbf{xy_r}/\mathbf{ab_r}] \cap S = \emptyset$;*
3. *For all aggregate atom $\delta \in PosAgg(r)$ of the form (1),*[7]

$$\text{OP}\langle \mathbf{c_v}[1] : \mathcal{A} \models Bd(\delta)[\alpha],\ Pos(\delta)[\alpha] \cap S = \emptyset \rangle \preceq N,$$

where α is the assignment of the form $\mathbf{xy_r wv} \longrightarrow \mathbf{ab_r c_w c_v}$.

By Definition 4, a set $S \subseteq [\mathcal{P}_{int}(\Pi)]^{\mathcal{A}}$ is *not* externally supported if for all $P(\mathbf{a}) \in S$, rule $Body(r) \rightarrow P(\mathbf{x}) \in \Pi$ with local variables $\mathbf{y_r}$, and assignments $\mathbf{xy_r} \longrightarrow \mathbf{ab_r}$ such that $\mathcal{A} \models Body(r)[\mathbf{xy_r}/\mathbf{ab_r}]$ (i.e. a "support" for $P(\mathbf{a})$), there exists some atom $\beta \in Pos(r)$ such that either

1. $\beta = Q(\mathbf{y})$, where $Q \in \mathcal{P}_{int}(\Pi)$ and $Q(\mathbf{b}) \in S$ that is a further elaboration $\beta[\mathbf{xy_r}/\mathbf{ab_r}]$, or
2. β is a aggregate atom δ of the form (1) and

$$\text{OP}\langle \mathbf{c_v}[1] : \mathcal{A} \models Bd(\delta)[\alpha],\ Pos(\delta)[\alpha] \cap S = \emptyset \rangle \preceq N \tag{16}$$

does not hold, where α is the assignment $\mathbf{xy_r wv} \longrightarrow \mathbf{ab_r c_w c_v}$.

Lemma 1 *Let Π be a program with aggregate atoms of form (1) under the restrictions in Definition 2, and $\mathcal{A}$ a finite $\tau(\Pi)$-structure. Then, $\mathcal{A}$ is a stable model of Π iff $\mathcal{A} \models \widehat{\Pi}$ and every $S \subseteq [\mathcal{P}_{int}(\Pi)]^{\mathcal{A}}$ is externally supported.*

Proof: (sketch) "$\Rightarrow$:" Since $\mathcal{A} \models SM(\Pi)$, we have $\mathcal{A} \models \widehat{\Pi}$. Now assume there exists a set $S \subseteq [\mathcal{P}_{int}(\Pi)]^{\mathcal{A}}$ which is not externally supported. Let σ_U denote the set of predicate symbols $\{U_i \mid 1 \leq i \leq n\}$. We construct a structure $\mathcal{U}$ of $\tau(\Pi) \cup \sigma_U$ as follows:

- $c^{\mathcal{U}} = c^{\mathcal{A}}$ for each constant $c \in \tau(\Pi)$;
- $P^{\mathcal{U}} = P^{\mathcal{A}}$ for each predicate $P \in \tau(\Pi)$;
- $U_i^{\mathcal{U}} = P_i^{\mathcal{A}} \setminus \{\mathbf{a} \mid P_i(\mathbf{a}) \in S\}$ for $1 \leq i \leq n$.

Clearly, $\mathcal{U} \models \mathbf{U} < \mathcal{P}_{int}(\Pi)$. Since $\mathcal{A} \models SM(\Pi)$, there exists a rule r such that $\mathcal{U} \not\models r^*$. Then, $\mathcal{U} \models Body(r)^*$ and $\mathcal{U} \not\models Head(r)^*$. It follows that $Head(r) \in S$. Then, any body supporting $Head(r)$ must be dependent on S, i.e., there is an atom in S which is used to satisfy the body. However, this atom is not in $\mathcal{U}$ according to our construction. Hence, $\mathcal{U} \not\models Body(r)^*$, a contradiction. This shows that every S is externally supported.

"$\Leftarrow$:" Since $\mathcal{A} \models \widehat{\Pi}$, it suffices to show $\mathcal{A} \models \neg\exists \mathbf{U}(\mathbf{U} < \mathcal{P}_{int}(\Pi) \wedge \bigwedge_{r \in \Pi} r^*)$. We prove this by contradiction. Suppose there exists such a set $\mathbf{U}$ of interpretations. Let $\mathcal{U}$ be the structure obtained from $\mathcal{A}$ by doubling every interpretation of $P \in P_{int}(\Pi)$ with the interpretation of corresponding $U \in \mathbf{U}$. Then, $\mathcal{U} \models \bigwedge_{r \in \Pi} r^*$. Let $S = \{P(\mathbf{a}) \mid \mathbf{a} \in P^{\mathcal{A}} \backslash U^{\mathcal{U}}\}$. Firstly, S is not empty as $\mathbf{U} < P_{int}(\Pi)$ holds. Since S is externally supported, there exists r such that $Head(r) \in S$ and $Body(r)$ is satisfied by some atoms irrelevant to S. According to our construction, $\mathcal{U} \models Body(r)^*$ but $\mathcal{U} \not\models Head(r)^*$. It follows that $\mathcal{U} \not\models r^*$, a contradiction. This completes our proof. □

Now we prove our main theorem.

Proof of Theorem 2: (sketch) "$\Rightarrow$:" By Lemma 1, for every set $S \subseteq [\mathcal{P}_{int}(\Pi)]^{\mathcal{A}}$, S is externally supported. We rank the grounded atoms in $[\mathcal{P}_{int}(\Pi)]^{\mathcal{A}}$ as follows. At each step, we select $P(a) \in [\mathcal{P}_{int}(\Pi)]^{\mathcal{A}}$ such that there is a grounded rule satisfied by $\mathcal{A}$, whose head is $P(\mathbf{a})$, and whose positive body is satisfied by the extensional grounded atoms and the intentional grounded atoms already ranked. This ranking will range over all grounded atoms in $[\mathcal{P}_{int}(\Pi)]^{\mathcal{A}}$. Otherwise, let S be the set of grounded atoms not ranked. Since S is externally supported, there exists a grounded rule r satisfied by $\mathcal{A}$, whose head is in S and whose positive body is satisfied by $[\mathcal{P}_{int}(\Pi)]^{\mathcal{A}} \backslash S$ together with the extensional atoms. However, according to our ranking criterion, $Head(r)$ can be ranked, a contradiction. Based on this ranking, we expand $\mathcal{A}$ to $\mathcal{A}'$ of the signature $\tau(\Pi) \cup \sigma_{\leq}$ such that

$$\leq_{PQ}^{\mathcal{A}'} = \{\mathbf{ab} \mid P(\mathbf{a}) \text{ is ranked ahead of } Q(\mathbf{b})\},$$

where $P, Q \in \mathcal{P}_{int}(\Pi)$. Then, it can be shown that $\mathcal{A}'$ is a model of $OC(\Pi)$.

"$\Leftarrow$:" Since $\mathcal{A}' \models OC(\Pi)$, $\mathcal{A}' \models \widehat{\Pi}$. Therefore, the restriction of $\mathcal{A}'$ on $\tau(\Pi)$ is a model of $\widehat{\Pi}$ since $\widehat{\Pi}$ mentions no comparison predicates $\leq_{PQ}$. Hence by Lemma 1, it is enough to show that every set $S \subseteq [\mathcal{P}_{int}(\Pi)]^{\mathcal{A}'}$ is externally supported. On the contrary, assume there exist such a set $S \subseteq [\mathcal{P}_{int}(\Pi)]^{\mathcal{A}'}$ that is non-externally supported. Since $\mathcal{A}'$ is finite and all the aggregates are (anti)monotone, we can obtain a total (linear) order $\leq^S$ on S that is consistent with the interpretations of the comparison predicates by using (9), i.e. we use the fact that every ground atom $P(\mathbf{a}) \in S$ is supported by some grounded rule and associated comparison relations. Then, since the total order on S imply that S is externally supported (since consistent with the comparison relations), we get a contradiction. □

It is important to mention that although our work only

[7] In the following, $\mathbf{c_v}[1]$ denotes the first component (or position) of $\mathbf{c_v}$.

considers a certain subclass of aggregates, it is indeed powerful as it covers the types of aggregates used in most applications. As far as we have checked, almost all aggregates used in benchmark normal programs (Calimeri et al. 2011) belong to our class. According to Theorem 2, those programs can all be captured by their ordered completions.

In addition, our proof technique for Theorem 2, e.g. the concept of external support, is significantly different from the original proof for ordered completion in (Asuncion et al. 2012). We believe this will shed new insights in first-order ASP (with aggregates) as well.

Ordered Completion in FOL

Theorem 2 shows that normal answer set programs with a large variety of aggregates can be captured by their ordered completions. However, the ordered completion defined in Definition 3 is not exactly in first-order logic as it contains aggregate atoms. A natural question arises whether this can be further translated in classical first-order logic. We answer it positively in this section. That is, we show that programs with aggregates functions restricted in Definition 2 can be expressed in classical first-order logic with theory on numbers.

In fact, what we need to do is to translate an aggregate atom of form (1) into classical first-order logic. For the aggregate functions MIN and MAX, this is relatively simple.

Definition 5 *Let δ be an aggregate atom of the form (1) where* OP $\in \{\text{MIN}, \text{MAX}\}$*, and $Lt(\delta)$ be as in $Bd(\delta)$ but where $Lt(\delta)$ can mention the comparison atoms. Then by δ^{FO}, we denote the following formula*[8]

$$\exists \mathbf{v}\mathbf{w} Lt(\delta) \wedge \Phi_{\mathbb{Z}} \wedge \tag{17}$$

$$\forall v_1 \{\, \forall v_2 (\exists \mathbf{w} Lt(\delta)[\mathbf{v}^1/v_1] \wedge \exists \mathbf{w} Lt(\delta)[\mathbf{v}^1/v_2] \wedge v_1 \neq v_2 \rightarrow v_1 \odot v_2\,) \rightarrow v_1 \preceq N\,\}, \tag{18}$$

where $\odot$ is $<$ if OP $=$ MIN *and $>$ if* OP $=$ MAX*, and $\Phi_{\mathbb{Z}} = \forall \mathbf{v}\mathbf{w}(Lt(\delta) \rightarrow (\mathbf{v}^1 \leq 0 \vee \mathbf{v}^1 \geq 0))$, i.e. states that the (satisfying) $\mathbf{v}^1$ are integers.*

However, for aggregate atoms where OP $\in \{\text{CARD}, \text{SUM}, \text{PROD}\}$, it is more complicated. Here, for space reasons, we only present the translation for aggregate atoms of the form (1) where OP $=$ SUM and $\preceq = \leq$. The others can be done in a similar manner.

Definition 6 *Let δ be the aggregate atom*

$$\text{SUM}\langle \mathbf{v} : \exists \mathbf{w} Lt(\delta)\,\rangle \leq N.$$

By δ^{FO}, we denote the following formula[9]

$$\exists \mathbf{v}\mathbf{w} Lt(\delta) \rightarrow \exists \mathbf{v_1}\mathbf{w_1} n_1 \ldots \mathbf{v_N}\mathbf{w_N} n_N \tag{19}$$

$$\{ \bigwedge_{1 \leq i \leq N} Lt(\delta)[\mathbf{v}\mathbf{w}/\mathbf{v_i}\mathbf{w_i}] \wedge \bigwedge_{1 \leq i \leq N} (n_i = \mathbf{v_i}[1] \vee n_i = 0) \tag{20}$$

$$\wedge \bigwedge_{1 \leq i < j \leq N} (\mathbf{v_i}\mathbf{w_i} = \mathbf{v_j}\mathbf{w_j} \rightarrow (n_i = 0 \vee n_j = 0)) \tag{21}$$

$$\wedge \forall \mathbf{v}\mathbf{w}(Lt(\delta) \wedge (\mathbf{v}[1] > 0) \rightarrow \bigvee_{1 \leq i \leq N} (\mathbf{v}\mathbf{w} = \mathbf{v_i}\mathbf{w_i} \wedge \mathbf{v}[1] = n_i))\} \tag{22}$$

$$\wedge \forall \mathbf{v}\mathbf{w}(Lt(\delta) \rightarrow \mathbf{v}[1] \geq 0) \wedge (n_1 + \cdots + n_N \leq N). \tag{23}$$

Although Definition 6 seems a little complicated, the underlying idea is quite simple. We use n_i to simulate $\mathbf{v_i}[1]$ but only counting non-duplicated occurrences of $\mathbf{v_i}$. $\exists \mathbf{v}\mathbf{w} Lt(\delta)$ in formula (19) is to cover the case where SUM is defined on $\emptyset$. The $\exists \mathbf{v_1}\mathbf{w_1} n_1 \ldots \mathbf{v_N}\mathbf{w_N} n_N$ part in formula (19) is for our counting. Formula (22) ensures that every satisfiable instance of $\mathbf{v}\mathbf{w}$ is considered in our counting. Formula (20) forces n_i to be either $\mathbf{v_i}[1]$ (count this number) or 0 (not count it). Formula (21) means that for duplicated satisfiable instance $\mathbf{v}\mathbf{w}$, we only count at most once. Finally, formula (23) ensures that the (satisfying) $\mathbf{v}[1]$'s are in $\mathbb{Z}^+$ (i.e. must satisfy $\mathbf{v}[1] \geq 0$) and that the counting result is less than or equal to N.

It should be noted that the CARD aggregates are already well known to be translatable into FOL via the $\exists^{=N} F(\mathbf{x})$ formulas (e.g. see (Lee, Lifschitz, and Palla 2008)).

Theorem 3 *For a program Π with aggregate atoms of form (1) under the restrictions in Definition 2. Let $OC(\Pi)^{FO}$ be the formula obtained from $OC(\Pi)$ by replacing every aggregate atom δ with δ^{FO}. Then, a finite structure $\mathcal{A}$ is a stable model of Π iff $\mathcal{A}$ can be expanded into a model of $OC(\Pi)^{FO}$.*

Theorem 3 enables us to use a first-order theorem prover or a SMT solver to compute the stable models of a program (with aggregates). This is our main motivation.

Related Work and Discussions

Aggregates are extensively studied in the literature (Dao-Tran et al. 2009; Bartholomew, Lee, and Meng 2011; Faber, Leone, and Pfeifer 2011; Ferraris 2011; Son and Pontelli 2007). Although the syntactic form of aggregates is usually presented in a first-order language, its semantics is normally defined propositionally via grounding. There are several major approaches: Ferraris' semantics (Ferraris 2011), the FLP semantics (Faber, Leone, and Pfeifer 2011) (later extended for arbitrary formulas by Truszczyński (2010)), and the SP semantics (Son and Pontelli 2007).

The Ferraris' semantics and the FLP semantics (and its extension to arbitrary formulas by Truszczyński) are extended into first-order case (Ferraris, Lee, and Lifschitz 2011; Bartholomew, Lee, and Meng 2011). Our work captures the first one, also called the stable model semantics,

[8] In the following, $\mathbf{v}^1$ denotes the first component (or position) of $\mathbf{v}$ such that for a variable v and formula F, $F[\mathbf{v}^1/v]$ denotes the formula obtained from F by replacing every occurrence of the variable in the first position of $\mathbf{v}$ by v.

[9] For the following, $\mathbf{v_i}[1]$ and $\mathbf{v}[1]$ denotes the first components (or positions) of $\mathbf{v_i}$ and $\mathbf{v}$ respectively.

in first-order logic. Certainly, it is interesting to consider whether the FLP semantics can be captured in first-order logic as well. We leave this to our future work. Nevertheless, it is worth mentioning that if the aggregate atoms only occurs in the positive bodies of rules and the bodies of these aggregates contain no negative atoms (this is actually the case in most benchmark programs), these semantics coincide.

Bartholomew et al. (2011) studied the aggregate semantics in first-order case via translating into second-order logic. This work is slightly different from ours. Syntactically, the former considers aggregate atoms as arbitrary formula while we only consider a special form, i.e. form (1). Semantically, the former combines the theory of first-order atoms and aggregate atoms into a unified one, while the latter defines them separately in the sense that the theory of aggregates is regarded as a background theory. The main reason is for simplicity. It can be shown that they essentially coincide when restricted into the specific aggregate forms.

Our main results can also be extended for programs with other useful building blocks in ASP, e.g. choice rules and constraints, in the same way as in (Asuncion et al. 2012), However, for clarity and simplicity, we omit this in the paper.

Conclusion

In this paper, we showed that normal answer set programs with a large variety of aggregates (covering the aggregates used in most benchmark programs) can be captured by their ordered completions (see Theorem 2), which can be further translated into classical first-order logic (see Theorem 3). This work enables us to implement a new direction of ASP solver by firstly translating a program to its ordered completion, and then working on finding a model of this first-order sentence using, e.g. SAT or SMT solvers. We leave this promising task to our future work.

Acknowledgement

This publication was made possible by the support of an NPRP grant (NPRP 09-079-1-013) from the Qatar National Research Fund (QNRF). The statements made herein are solely the responsibility of the authors.

References

Asuncion, V.; Lin, F.; Zhang, Y.; and Zhou, Y. 2012. Ordered completion for first-order logic programs on finite structures. *Artificial Intelligence* 177-179:1–24.

Baral, C. 2003. *Knowledge representation, reasoning and declarative problem solving*. Cambridge University Press.

Bartholomew, M.; Lee, J.; and Meng, Y. 2011. First-order extension of the flp stable model semantics via modified circumscription. In *IJCAI-2011*, 724–730.

Calimeri, F.; Ianni, G.; Ricca, F.; Alviano, M.; Bria, A.; Catalano, G.; Cozza, S.; Faber, W.; Febbraro, O.; Leone, N.; Manna, M.; Martello, A.; Panetta, C.; Perri, S.; Reale, K.; Santoro, M. C.; Sirianni, M.; Terracina, G.; and Veltri, P. 2011. The third answer set programming competition: Preliminary report of the system competition track. In *LPNMR*, 388–403.

Chen, Y.; Lin, F.; Wang, Y.; and Zhang, M. 2006. First-order loop formulas for normal logic programs. In *KR-2006*, 298–307.

Clark, K. L. 1978. Negation as failure. In *Logics and Databases*, 293–322.

Dao-Tran, M.; Eiter, T.; Fink, M.; and Krennwallner, T. 2009. Modular nonmonotonic logic programming revisited. In *ICLP*, 145–159.

Faber, W.; Leone, N.; and Pfeifer, G. 2003. Aggregate functions in dlv. In *Answer Set Programming: Advances in Theory and Implementation*, 274–288.

Faber, W.; Leone, N.; and Pfeifer, G. 2011. Recursive aggregates in disjunctive logic programs: Semantics and complexity. *Artificial Intelligence* 175(1):278–298.

Ferraris, P.; Lee, J.; and Lifschitz, V. 2011. Stable models and circumscription. *Artificial Intelligence* 175(1):236–263.

Ferraris, P. 2011. Logic programs with propositional connectives and aggregates. *ACM Transactions on Computational Logic* 12(4):25.

Gebser, M.; Kaminski, R.; Kaufmann, B.; and Schaub, T. 2009. On the implementation of weigh constraints in conflict-driven ASP solvers. In *ICLP'09*, volume 5649, 250–264.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Proceedings of International Logic Programming Conference and Symposium*, 1070–1080.

Lee, J., and Meng, Y. 2009. On the reductive semantics of aggregates in answer set programming. In *LPNMR-2009*, 182–195.

Lee, J.; Lifschitz, V.; and Palla, R. 2008. A reductive semantics for counting and choice in answer set programming. In *AAAI-2008*, 472–479.

Lin, F., and Zhou, Y. 2011. From answer set logic programming to circumscription via logic of GK. *Artif. Intell.* 175(1):264–277.

Niemelä, I. 1999. Logic programs with stable model semantics as a constraint programming paradigm. *Ann. Math. and AI* 25(3-4):241–273.

Nieuwenhuis, R.; Oliveras, A.; and Tinelli, C. 1999. Solving SAT and SAT modulo theories: from an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(t). *Journal of the ACM (JACM)* 53(6):937–977.

Pontelli, E.; Son, T. C.; and Elkabani, I. 2004. A treatment of aggregates in ASP (system description). In *LPNMR-2004*, 356–360.

Son, T. C., and Pontelli, E. 2007. A constructive semantic characterization of aggregates in answer set programming. *TPLP* 7(3):355–375.

Truszczyński, M. 2010. Reducts of propositional theories, satisfiability relations, and generalizations of semantics of logic programs. *Artificial Intelligence* 174(16-17):1285–1306.