Axiomatization of Aggregates in Answer Set Programming

J. Fandinno    Z. Hansen    Y. Lierler

UNO

January 2022
Table of Contents

1 Background

2 Logic Programs to Many-Sorted FOL

3 Semantics of Logic Programs With Aggregates

4 Axiomatization of Aggregates

5 First-Order Characterization
Table of Contents

1 Background

2 Logic Programs to Many-Sorted FOL

3 Semantics of Logic Programs With Aggregates

4 Axiomatization of Aggregates

5 First-Order Characterization
Motivation: Formal Verification of Programs With Aggregates

- Aggregates are widely used ASP constructs
- They intuitively represent functions on sets

Example: Paths in a graph

\[
\begin{align*}
\text{cost}(a, b, 3). & \quad \text{cost}(b, c, 7). & \quad \text{cost}(c, a, 1). \\
\text{path}(a, b). & \quad \text{path}(b, c). & \quad \text{path}(c, a). \\
\text{expensive} & \leftarrow \sum\{ C, X, Y : \text{path}(X, Y), \text{cost}(X, Y, C) \} \geq 5.
\end{align*}
\]
Grounding

Grounding replaces variables with constants from the program signature.

\[ p(X) :- q(X, Y). \]

might be replaced by rules

\[ p(1) :- q(1, 1). \]
\[ p(1) :- q(1, 2). \]
\[ p(2) :- q(2, 1). \]
\[ \ldots \]
Motivation: Formal Verification of Non-ground Programs

Disadvantages of grounding

1. Reasoning about the two-step ground and solve procedure is cumbersome
2. Inseparability of problem class and instance

Automatic Verification

1. First-order theorem provers can help verify the adherence of a first-order theory to a specification
2. We would like to translate ASP programs with aggregates into first-order theories
Defining Aggregate Semantics

- The semantics of aggregates are traditionally captured via grounding.
- Our goal is to characterize aggregates using the language of classical logic.

\[ \text{sum}(\Delta) \]

We wish to express that \( \text{sum} \) is a function on a set of tuples: 
\( \text{sum}(\Delta) \) is the numeral corresponding to the sum of the weights of all tuples in \( \Delta \), if \( \Delta \) contains finitely many tuples with non-zero weights; and 0 otherwise. 
Example: 
\[ \text{sum}(\{\langle 2, a \rangle, \langle 3, b \rangle, \langle c, d \rangle \}) = 5 \]
Define a **syntactic transformation** from logic programs with aggregates into a theory in many-sorted first-order logic with some meta-logical restrictions on “standard” interpretations

Define the semantics of these logic programs in terms of a many-sorted generalization of the **SM operator**

Replace some restrictions on standard models with equivalent **axiomatizations** in many-sorted SOL

Show that for programs with finite aggregates, the second-order SM characterization can be represented with a **first-order** characterization

Demonstrate that our semantics coincide with that of Clingo
| Chapter | Title |
|---------|-------|
| 1       | Background |
| 2       | Logic Programs to Many-Sorted FOL |
| 3       | Semantics of Logic Programs With Aggregates |
| 4       | Axiomatization of Aggregates |
| 5       | First-Order Characterization |
Program Syntax

We consider programs of a typical ASP syntax.
An aggregate element has the form
\[ t_1, \ldots, t_k : l_1, \ldots, l_m \]
An aggregate atom has the form
\[ \#op\{ E \} \prec u \]
Rules have the form
\[ Head :- B_1, \ldots, B_n, \]

Example

\[ s(X) :- q(X), \#sum\{ Y : r(X, Y, Z) \} \geq 1. \]
\[ t :- \#sum\{ Y, Z : r(X, Y, Z) \} \geq 1. \]
\[ q(a). \ q(b). \ q(c). \]
Translation $\tau^*$

- Atomic formulas are translated as themselves;
- An aggregate atom $A$ of form $\#\sum\{E\} \prec u$ is translated as:
  \[
  \text{sum}(\text{set}_{E/X}(X)) \prec u
  \]
  where $\text{set}_{E/X}$ is a function symbol that takes as many arguments of the program sort as there are variables in $X$ (the global variables in the aggregate rule);
- Literals of the form $\text{not } A$ become $\neg \tau^* A$;
- Literals of the form $\text{not not } A$ become $\neg \neg \tau^* A$;
- Rules are translated to the universal closure across global variables of the following:
  \[
  \tau^* B_1 \land \cdots \land \tau^* B_n \rightarrow \tau^* \text{Head},
  \]
Example

\[ e_1 = Y : r(X, Y, Z)/X \]
\[ e_2 = Y, Z : r(X, Y, Z) \]

\[
q(X) \land \sum(\text{set}_{e_1}(X)) \geq 1 \rightarrow s(X) \\
\sum(\text{set}_{e_2}) \geq 1 \rightarrow t \\
q(a) \quad q(b) \quad q(c) \\
r(a, 1, a) \quad r(b, -1, a) \quad r(b, 1, a) \quad r(b, 1, b) \quad r(c, 0, a)
\]

Where where \( e_1 \) and \( e_2 \) are the names for aggregate symbols
\[ Y : r(X, Y, Z)/X \] and \( Y, Z : r(X, Y, Z) \)
Table of Contents

1 Background

2 Logic Programs to Many-Sorted FOL

3 Semantics of Logic Programs With Aggregates

4 Axiomatization of Aggregates

5 First-Order Characterization
The SM operator transforms a first-order formula into a second-order one. If \( u \) and \( p \) are tuples of predicate constants, then by \( SM_p[F] \) we denote the second-order formula

\[
F \land \neg \exists u ((u < p) \land F^*(u))
\]
Stable Models and Agg-Interpretations

As a preliminary step, we restrict our attention to **agg-interpretations**:

1. the domain \(|I|^{sprg}\) is the set containing all ground program terms;
2. \(I\) interprets each ground program term as itself;
3. universe \(|I|^{set}\) is the set of all sets of non-empty tuples that can be formed with elements from \(|I|^{sprg}\);
4. for each aggregate symbol \(E/X\), \(set|E/X|(x)^I\) is the set of all tuples of ground program terms that satisfy the list of literals from the corresponding aggregate element;
5. \(sum(t_{set})^I\) is \(\widehat{sum}(t_{set})^I\);

**Stable Models**

We say that an agg-interpretation \(I\) is a **stable model** of program \(\Pi\) if it satisfies the second-order sentence \(SM_p[\tau^*\Pi]\) where \(p\) is the list of all predicate symbols in \(\Pi\)

\[ SM[\tau^*\Pi] \land \Lambda \]
\( \langle a \rangle \)
\( \langle b \rangle \)
\( \ldots \)
\( \langle 1 \rangle \)
\( \ldots \)
\( \langle 1, a \rangle \)
\( \ldots \)
Many-sorted first-order formulas

\[ q(X) \land sum(set_{e1}(X)) \geq 1 \rightarrow s(X) \]
\[ q(a) \quad q(b) \quad q(c) \]
\[ r(a, 1, a) \quad r(b, -1, a) \quad r(b, 1, a) \quad r(b, 1, b) \quad r(c, 0, a) \]

Where where \( e_1 \) is the name of aggregate symbol \( Y : r(X, Y, Z)/X \) and

\[ q^I = \{a, b, c\} \]
\[ r^I = \{(a, 1, a), (b, -1, a), (b, 1, a), (b, 1, b), (c, 0, a)\}. \]

Consequently:

\[ set_{e1}(a)^I = \{(1)\} \quad sum(set_{e1}(a))^I = 1 \]
\[ set_{e1}(b)^I = \{(-1), (1)\} sum(set_{e1}(b))^I = 0 \]
Table of Contents

1 Background

2 Logic Programs to Many-Sorted FOL

3 Semantics of Logic Programs With Aggregates

4 Axiomatization of Aggregates

5 First-Order Characterization
Removing Conditions 4 & 5

Recall that:

- Condition 4 defines the behavior of the $\text{set}_{E/X}$ function symbols
- Condition 5 defines the behavior of the $\text{sum}$ function symbol

We can refine these conditions with the following:

- Extend our program signature to include tuples and integers
- Make assumptions about the form of the tuple and set universes
- Define the behavior of tuple construction, addition and set membership

This results in more assumptions, but they are more thoroughly studied in arithmetic and set theory. $P(|I|_{\text{tuple}}) = |I|_{\text{set}}$
Standard Interpretations

A many-sorted interpretation $I$ is considered standard if:

1. the domain $|I|^{spg}$ is the set containing all ground program terms;
2. $I$ interprets each ground program term as itself;
3. universe $|I|^{set}$ is the set of all sets of non-empty tuples that can be formed with elements from $|I|^{spg}$;
4. the domain $|I|^{int}$ is the set of all numerals;
5. $I$ interprets $\overline{m} + \overline{n}$ as $\overline{m + n}$,
6. universe $|I|^{tuple}$ is the set of all tuples of form $\langle d_1, \ldots, d_m \rangle$ with $m \geq 1$ and each $d_i \in |I|^{spg}$;
7. $I$ interprets each tuple term of form $\text{tuple}_k(t_1, \ldots, t_k)$ as the tuple $\langle t^I_1, \ldots, t^I_k \rangle$.
8. $I$ interprets object constant $\overline{\emptyset}$ as the empty set $\emptyset$;
9. $I$ satisfies $t_1 \in t_2$ iff tuple $t^I_1$ belongs to set $t^I_2$. 
Characterizing \textit{sum}

\textit{FiniteSum}(t_{set}) \text{ stands for the formula:}

\[
\forall T \left( T \in t_{set} \rightarrow \text{sum}(t_{set}) = \text{sum}(\text{rem}(t_{set}, T)) + \text{weight}(T) \right)
\]

Thus, \textit{sum} is formalized:

\[
\forall S \left( \text{ZeroWeight}(S) \rightarrow \text{sum}(S) = 0 \right) \quad (1)
\]
\[
\forall S \left( \text{FiniteWeight}(S) \rightarrow \text{FiniteSum}(S) \right) \quad (2)
\]
\[
\forall S \left( \neg \text{FiniteWeight}(S) \rightarrow \text{sum}(S) = 0 \right) \quad (3)
\]

Adding axioms restricts satisfying interpretations to exactly those that satisfy the meta-logical conditions 4 and 5.
Sets With Finite Weight

**Second-order Characterization**

To capture the behavior of \( \sum \), we need second-order axioms (omitted) to express that a set has finitely many tuples with non-zero weight.

**First-order Characterization**

- In the case of finite aggregates, we can replace these second-order sentences with axioms in many-sorted first-order logic.
- Tight programs can be represented in first-order logic instead of using the SM operator.
- The key result is that standard interpretations satisfying all axioms and first-order translations of *tight programs with finite aggregates* are stable models.
Conclusion

**Contribution**

A nonground characterization of aggregate semantics that coincides with the ASP-Core-2 standard and the solver Clingo

**Limitations**

Our semantics coincides with that of Clingo only when there exists no positive recursion through aggregates

**Future Work**

- Anthem translates certain ASP programs to first-order theories
- Utilizes Vampire to automatically verify ASP programs
- We hope to extend Anthem to programs with aggregates
Set Formation, Set Minus, and Weight

We can associate each aggregate element of form (10) with a unique set:

\[ \forall X \ T \in \text{set}_{E/X}(X) \leftrightarrow \exists Y \ (T = \text{tuple}_k(t_1, \ldots, t_k) \land l_1 \land \cdots \land l_m) \]

where \( Y \) is the list of all the variables occurring in \( E \) that are not in \( X \). Similarly, the notion of set minus can be captured:

\[ \forall S T S' (\text{rem}(S, T) = S' \leftrightarrow \forall T' (T' \in S' \leftrightarrow (T' \in S \land T' \neq T'))) \]

Finally, the weight of a tuple is the integer weight of its first element:

\[ \forall N X_2 \ldots X_k \ \text{weight}(\text{tuple}_k(N, X_2, \ldots, X_k)) = N \]

\[ \forall X_1 X_2 \ldots X_k ((\neg \exists N \ X_1 = N) \rightarrow \text{weight}(\text{tuple}_k(X_1, X_2, \ldots, X_k)) = 0) \]
Expression $\text{FiniteWeight}(t_{\text{set}})$ stands for the second-order formula

$$\exists f (\text{InjectiveWeight}(f, t_{\text{set}}) \land \exists N \text{ ImageWeight}(f, t_{\text{set}}, 0, N))$$

$\text{FiniteSum}(t_{\text{set}})$ stands for the formula:

$$\forall T \left( T \in t_{\text{set}} \rightarrow \text{sum}(t_{\text{set}}) = \text{sum}(\text{rem}(t_{\text{set}}, T)) + \text{weight}(T) \right)$$

Thus, $\text{sum}$ is formalized:

$$\forall S \left( \text{ZeroWeight}(S) \rightarrow \text{sum}(S) = 0 \right) \quad (4)$$
$$\forall S \left( \text{FiniteWeight}(S) \rightarrow \text{FiniteSum}(S) \right) \quad (5)$$
$$\forall S \left( \neg \text{FiniteWeight}(S) \rightarrow \text{sum}(S) = 0 \right) \quad (6)$$
First-Order Characterization

Finite Aggregates

An interpretation $I$ has finite aggregates if set $\text{set}_{E/X}(x)^I$ is finite for every aggregate symbol $E/X$ and any list $x$ of ground program terms of the same length as $X$.

First-Order Axioms

In the case of finite aggregates, we can replace second-order sentences:

$$\forall S \ (\text{FiniteWeight}(S) \rightarrow \text{FiniteSum}(S))$$
$$\forall S \ (\neg \text{FiniteWeight}(S) \rightarrow \text{sum}(S) = 0)$$

with first-order sentence

$$\forall X \ S \ (\text{Subset}(S, \text{set}_{E/X}(X)) \rightarrow \text{FiniteSum}(S))$$