Axiomatization of Aggregates in Answer Set Programming

J. Fandinno Z. Hansen Y. Lierler

UNO

January 2022

Table of Contents

Table of Contents

Motivation: Formal Verification of Programs With Aggregates

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

Example: Paths in a graph

$$cost(a, b, 3). \quad cost(b, c, 7). \quad cost(c, a, 1).$$
$$path(a, b). \quad path(b, c). \quad path(c, a).$$
$$expensive \text{ :- } \#\mathtt{sum}\{C, X, Y : path(X, Y), cost(X, Y, C)\} \geq 5.$$

Motivation: Formal Verification of Non-ground Programs

Grounding

Grounding replaces variables with constants from the program signature.

$$p(X) \text{ :- } q(X, Y).$$

might be replaced by rules

$$p(1) \text{ :- } q(1, 1).$$
$$p(1) \text{ :- } q(1, 2).$$
$$p(2) \text{ :- } q(2, 1).$$
$$\ldots$$

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

Context

Defining Aggregate Semantics

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

$\widehat{\mathtt{sum}}(\Delta)$

We wish to express that $\mathtt{sum}$ is a function on a set of tuples:
$\widehat{\mathtt{sum}}(\Delta)$ is the numeral corresponding to the sum of the weights of all tuples in Δ, if Δ contains finitely many tuples with non-zero weights; and 0 otherwise
Example: $\widehat{\mathtt{sum}}(\{\langle 2, a\rangle, \langle 3, b\rangle, \langle c, d\rangle\}) = 5$

Overview

1. 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
2. Define the semantics of these logic programs in terms of a many-sorted generalization of the **SM operator**
3. Replace some restrictions on standard models with equivalent **axiomatizations** in many-sorted SOL
4. Show that for programs with finite aggregates, the second-order SM characterization can be represented with a **first-order characterization**
5. Demonstrate that our semantics coincide with that of Clingo

Table of Contents

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

$$\#\mathtt{op}\{E\} \prec u$$

Rules have the form

$$Head \;\text{:-}\; B_1, \ldots, B_n,$$

Example

$$s(X) \text{ :- } q(X), \#\mathtt{sum}\{Y : r(X, Y, Z)\} \geq 1.$$
$$t \text{ :- } \#\mathtt{sum}\{Y, Z : r(X, Y, Z)\} \geq 1.$$
$$q(a).\ q(b).\ q(c).$$

Translation τ^*

- Atomic formulas are translated as themselves;
- An aggregate atom A of form $\#\mathtt{sum}\{E\} \prec u$ is translated

$$sum(set_{|E/\mathbf{X}|}(\mathbf{X})) \prec u$$

  where $set_{|E/\mathbf{X}|}$ is a function symbol that takes as many arguments of the program sort as there are variables in $\mathbf{X}$ (the global variables in the aggregate rule);
- Literals of the form *not* A become $\neg\tau^* A$;
- Literals of the form *not not* A become $\neg\neg\tau^* A$;
- Rules are translated to the universal closure across global variables of the following:

$$\tau^* B_1 \wedge \cdots \wedge \tau^* B_n \rightarrow \tau^*\, Head,$$

Example

$e1 = Y : r(X, Y, Z)/X$
$e2 = Y, Z : r(X, Y, Z)$

$$q(X) \wedge sum(set_{e1}(X)) \geq 1 \;\rightarrow\; s(X)$$
$$sum(set_{e2}) \geq 1 \;\rightarrow\; t$$
$$q(a) \quad q(b) \quad q(c)$$
$$r(a, 1, a) \quad r(b, -1, a) \quad r(b, 1, a) \quad r(b, 1, b) \quad r(c, 0, a)$$

Where where $e1$ and $e2$ are the names for aggregate symbols $Y : r(X, Y, Z)/X$ and $Y, Z : r(X, Y, Z)$

Table of Contents

Many-Sorted SM Operator

The SM operator transforms a first-order formula into a second-order one. If $\boldsymbol{u}$ and $\boldsymbol{p}$ are tuples of predicate constants, then by $\mathrm{SM}_{\mathbf{p}}[F]$ we denote the second-order formula

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

Many-Sorted SM

We generalize the unsorted definition of the SM operator to the many-sorted setting by mandating that arities respect sort information

Stable Models and Agg-Interpretations

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

1. the domain $|I|^{s_{prg}}$ is the set containing all ground program terms;
2. I interprets each ground program term as itself;
3. universe $|I|^{s_{set}}$ is the set of all sets of non-empty tuples that can be formed with elements from $|I|^{s_{prg}}$;
4. for each aggregate symbol $E/\mathbf{X}$, $set_{|E/\mathbf{X}|}(\mathbf{x})^I$ 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 $\widehat{\mathtt{sum}}(t^I_{set})$;

Stable Models

We say that an agg-interpretation I is a **stable model** of program Π if it satisfies the second-order sentence $\mathrm{SM}_{\mathbf{p}}[\tau^*\Pi]$ where $\mathbf{p}$ is the list of all predicate symbols in Π

$\mathrm{SM}[\tau^*\Pi] \wedge \Lambda$

Scratch Paper

$\langle a \rangle$
$\langle b \rangle$
$\ldots$
$\langle 1 \rangle$
$\ldots$
$\langle 1, a \rangle$
$\ldots$

Agg-Interpretation Example

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 $e1$ 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:

$$set_{e1}(a)^I = \{(1)\} \qquad sum(set_{e1}(a))^I = 1$$
$$set_{e1}(b)^I = \{(-1),(1)\} sum(set_{e1}(b))^I = 0$$

Table of Contents

Removing Conditions 4 & 5

Recall that:

- Condition 4 defines the behavior of the $set_{|E/\mathbf{X}|}$ function symbols
- Condition 5 defines the behavior of the $\mathtt{sum}$ 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. $\mathcal{P}(|I|^{s_{tuple}}) = |I|^{s_{set}}$

Standard Interpretations

A many-sorted interpretation I is considered standard if:

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

Axiom Example

Characterizing *sum*

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

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

Thus, *sum* is formalized:

$$\forall S\,(ZeroWeight(S) \rightarrow sum(S) = \overline{0}) \qquad (1)$$

$$\forall S\,(FiniteWeight(S) \rightarrow FiniteSum(S)) \qquad (2)$$

$$\forall S\,(\neg FiniteWeight(S) \rightarrow sum(S) = \overline{0}) \qquad (3)$$

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

Table of Contents

Sets With Finite Weight

Second-order Characterization

To capture the behavior of $\widehat{\mathtt{sum}}$, 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:

$$\forall \mathbf{X}\ T\big(T \in \mathit{set}_{|E/\mathbf{X}|}(\mathbf{X}) \leftrightarrow \exists \mathbf{Y}\ \big(T = \mathit{tuple}_k(t_1, \ldots, t_k) \wedge l_1 \wedge \cdots \wedge l_m\big)\big)$$

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

$$\forall S T S'\big(\mathit{rem}(S, T) = S' \leftrightarrow \forall T'\big(T' \in S' \leftrightarrow (T' \in S \wedge T' \neq T)\big)\big)$$

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

$$\forall N X_2 \ldots X_k\, \mathit{weight}(\mathit{tuple}_k(N, X_2, \ldots, X_k)) = N\big)$$
$$\forall X_1 X_2 \ldots X_k \big((\neg \exists N\, X_1 = N) \rightarrow \mathit{weight}(\mathit{tuple}_k(X_1, X_2, \ldots, X_k)) = 0\big).$$

Sum

Expression $\mathit{FiniteWeight}(t_{set})$ stands for the second-order formula

$$\exists f\left(\mathit{InjectiveWeight}(f, t_{set}) \wedge \exists N\ \mathit{ImageWeight}(f, t_{set}, 0, N)\right)$$

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

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

Thus, `sum` is formalized:

$$\forall S\left(\mathit{ZeroWeight}(S) \rightarrow \mathit{sum}(S) = \overline{0}\right) \tag{4}$$

$$\forall S\left(\mathit{FiniteWeight}(S) \rightarrow \mathit{FiniteSum}(S)\right) \tag{5}$$

$$\forall S\left(\neg \mathit{FiniteWeight}(S) \rightarrow \mathit{sum}(S) = \overline{0}\right) \tag{6}$$

First-Order Characterization

Finite Aggregates

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

First-Order Axioms

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

$$\forall S\,(FiniteWeight(S) \to FiniteSum(S))$$
$$\forall S\,(\neg FiniteWeight(S) \to sum(S) = \overline{0})$$

with first-order sentence

$$\forall \mathbf{X}\ S\big(Subset(S, set_{|E/\mathbf{X}|}(\mathbf{X})) \to FiniteSum(S)\big)$$