Ultimate Well-founded and Stable Semantics for Logic Programs with Aggregates

Marc Denecker [a] Nikolay Pelov [b] Maurice Bruynooghe [b]

[a] Université Libre de Bruxelles, Bld du Triomphe CP 212, 1050 Brussels, Belgium
[b] K.U.Leuven, Celestijnenlaan 200A, 3001 Heverlee, Belgium

In [3] we investigate the problem of defining a well-founded and stable semantics for programs with aggregates. Our work is based on *Approximation Theory* [1] which is a general algebraic framework for approximating non-monotone operators on a complete lattice L by *approximating operators* on the bi-lattice L^2. The theory identifies basic properties of the approximating operators and gives a method to construct a stable operator and their associated fixpoints - Kripke-Kleene (KK), well-founded (WF), and set of stable fixpoints. The authors then show that the different fixpoints enjoy similar properties and relationships as the corresponding semantics in standard logic programming. For example the WF fixpoint is more precise than the KK fixpoint and also approximates every stable fixpoint. Moreover, stable fixpoints are always minimal fixpoints.

In a more recent paper [2] the authors study the concept of *precision* of an approximating operator. They show that more precise approximating operators have more precise KK and WF fixpoints and a larger number of stable fixpoints. Moreover, there exists a most-precise approximating operator which they call the *ultimate approximation*. In contrast with other approximating operators which have to be defined explicitly, the construction of the ultimate approximating operator is entirely algebraic and depends only on the original non-monotone operator. This makes it particularly well-suited for our purposes because defining a T_P operator for programs with aggregates is relatively straightforward. Another advantage of the ultimate approximation is that in cases where T_P is monotone the ultimate well-founded model will be 2-valued and will coincide with the least fixpoint of T_P. This is not the case for the standard well-founded semantics. For example in the standard well-founded model of the program:

$$p \leftarrow \neg p.$$
$$p \leftarrow p.$$

p is undefined while the associated T_P operator is monotone and p is true in the ultimate well-founded model.

One disadvantage of using the ultimate semantics is that it has a higher computational cost even for programs without aggregates. The complexity goes one level higher in the polynomial hierarchy to Δ_2^p for the well-founded model and to Σ_2^p for a stable model which is also complete for this class [2]. Fortunately, by adding aggregates the complexity does not increase further.

To give an example of a logic program with aggregates we consider the problem of computing the length of the shortest path between two nodes in a directed graph. The graph is represented with a set of $edge(X, Y, W)$ facts meaning that there is an edge between X and Y with weight W. The program consists of a single rule:

$$
\begin{aligned}
sp(X, Y, L) \leftarrow \textsc{min}(\{C \mid\ & edge(X, Y, C) \vee \\
& \exists Z, C_1, C_2.\ (sp(X, Z, C_1) \wedge edge(Z, Y, C_2) \wedge \\
& \qquad C = C_1 + C_2)\}, L).
\end{aligned}
$$

This program shows the syntax which we use. First of all, we allow arbitrary first-order formulas in the bodies of rules and in the conditions of set expressions. Unlike most of the previous work on aggregates, we use an explicit notation for set expressions $\{X \mid \varphi[X]\}$ denoting the set of all elements d for which the condition $\varphi[d]$ is true. Aggregate functions are written in a form of atoms like $\textsc{f}(s, t)$ where F is an aggregate function, s is a set expression, and t is a term. It will evaluate to true if the aggregate function $\textsc{f}$ applied to the set denoted by s is equal to the interpretation of the term t.

Having defined a 2-valued semantics for formulas containing aggregate atoms it is easy to extend the definition of the 2-valued van Emden-Kowalski immediate consequence operator T_P for programs with aggregates. Constructing the ultimate approximating operator of T_P we obtain an ultimate well-founded and ultimate stable semantics for programs with aggregates. The two semantics have all the properties which we discussed earlier. In addition for aggregate stratified programs the ultimate well-founded model will be 2-valued and will coincide with the unique ultimate stable model.

Our current research is focused on defining a less precise approximating operator for programs with aggregates which extends the standard well-founded and stable semantics for programs without aggregates. Although the complexity of this operator will still be the same as the the complexity of the ultimate approximation, if we limit our language to some specific aggregate functions we can stay in the complexity classes of the standard semantics - P for the well-founded model and NP for the stable models.

References

[1] M. Denecker, V. Marek, and M. Truszczyński. Approximating operators, stable operators, well-founded fixpoints and applications in non-monotonic reasoning. In J. Minker, editor, *Logic-based Artificial Intelligence*, pages 127–144. Kluwer Academic Publishers, 2000.

[2] M. Denecker, V. Marek, and M. Truszczyńsky. Ultimate approximations in nonmonotonic knowledge representation systems. In, *KR'02*, pages 177–188. Morgan Kaufmann, 2002.

[3] M. Denecker, N. Pelov, and M. Bruynooghe. Well-founded and stable model semantics for logic programs with aggregates. In P. Codognet, editor, *ICLP'01*, volume 2237 of *LNCS*, pages 212–226. Springer-Verlag, 2001.