An alternative paradigm for Answer Set Programming

Seemran Mishra[1], Jorge Fandinno[1], Javier Romero[1], Torsten Schaub[1], and Abhaya Nayak[2]

[1] University of Potsdam, Germany
[2] Macquarie University, Australia

An alternative logic programming paradigm is proposed in this work. The main idea is develop a structure for logic programs and use limited answer set programming language constructs for expressing the logic program.

The syntactical structure of a program in this programming paradigm is a four-valued tuple consisting of a set of facts, followed by a set of choice rules with empty bodies, a set definite rules and finally a set of integrity constraints in the respective order. Since the choice rules are expressed as disjunction of literals in the rule head, we use an extended version of answer set semantics for programs as proposed by Lifschitz, Tang and Turner [1]. This extension permits nested expressions formed from literals arbitrarily using negation as failure, conjunction and disjunction in the bodies and heads of rules.

Subsequently we try to translate programs made up of various language constructs not included in this programming paradigm such as normal rules with negative literals in the bodies, choice rules with bodies etc. to this programming paradigm to retain the expressiveness. This can be done by extending the language with auxiliary atoms. Finally, we try to establish various types of equivalence such as answer-set equivalence and strong equivalence between the original program and the translated version. For example, we prove that a normal rule with negative body literals of the form:

$$a_0 \leftarrow a_1, ..., a_m, not\, a_{m+1}, \dots, not\, a_n \tag{1}$$

can be expressed in this programming paradigm equivalently as:

$$\begin{array}{lll} & a_0 \leftarrow a_1, \dots, a_m, \bar{a}_{m+1}, \dots, \bar{a}_n & \\ \{\bar{a}_{m+1}\} & \dots & \{\bar{a}_n\} \\ \leftarrow a_{m+1}, \bar{a}_{m+1} & \dots & \leftarrow a_n, \bar{a}_n \\ \leftarrow not\, a_{m+1}, not\, \bar{a}_{m+1} & \dots & \leftarrow not\, a_n, not\, \bar{a}_n \end{array}$$

where A is a set of ground atoms and $a_i \in A$, $\bar{a_i} \in A$, $0 \leq m \leq n$ and $0 \leq i \leq n$

This programming paradigm lets us express a logic program in a modular form and will aid module by module computation of the stable models. This can improve the performance by limiting the search space. Properties of programs in this programming paradigm can be analyzed, for example, how the stable models depend on the structure of the program. Also the limited language construct used here can be extended to develop a branching mechanism

for computation of stable models. This mechanism makes the computation of stable models mostly deterministic with the non-determinism only arising from branching due to choice rules.

References

1. Lifschitz, V., Tang, L. R., Turner, H. (1999). Nested expressions in logic programs. Annals of Mathematics and Artificial Intelligence, 25(3-4), 369-389.