ASPARTIX-V: Utilizing Improved ASP Encodings

Alessandro Ronca[1], Johannes Peter Wallner[2], and Stefan Woltran[3]

[1] La Sapienza, University of Rome
[2] University of Helsinki, Department of Computer Science, HIIT
[3] Vienna University of Technology, Institute of Information Systems

Abstract. ASPARTIX-V is a novel system for solving reasoning tasks for argumentation frameworks under preferred semantics. Similarly to its predecessor, it calls a state-of-the-art ASP solver with ASP encodings tuned for performance. We describe the system architecture, the main components, and how to obtain ASPARTIX-V.

1 System Architecture

ASPARTIX-V (Answer Set Programming Argumentation Reasoning Tool – Vienna) takes as input an argumentation framework (AF) [1] in `apx` format [2]. Together with an answer-set programming (ASP) encoding for preferred semantics, the answer-sets are in a 1-to-1 correspondence with the preferred extensions of the given AF. Utilizing capabilities of modern ASP solvers, we can straightforwardly augment this workflow (see Fig. 1) to support the desired reasoning tasks. ASP solvers themselves offer enumeration of all answer-sets and returning an arbitrary one. For query-based reasoning, we add a single ASP constraint stating that the queried argument has to be outside the preferred extension (skeptical reasoning). Unsatisfiability of the resulting program means that the queried argument is skeptically accepted.

The underlying ASP solver is *clingo 4.4* [5, 6]. We in particular make use of the conditional literal [7, 4] language extension offered by clingo.

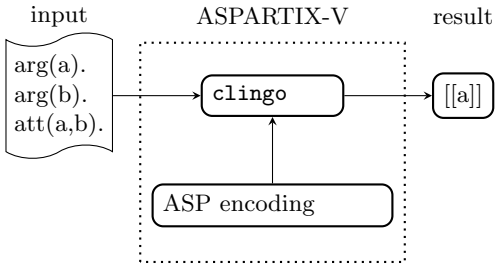

Fig. 1. ASPARTIX-V system architecture

2 Novel ASP Encodings

The predecessor of ASPARTIX-V is ASPARTIX [2]. In both systems, preferred semantics is encoded as a disjunctive logic program. While the latter makes heavy use of so-called loop constructs in ASP, our new system is able to do without and uses conditional literals for enhancing performance. Intuitively, conditional literals allow to use, e.g., a dynamic head in a disjunctive rule that contains a literal iff its condition is true. The loop constructs can be avoided by alternative characterizations of preferred semantics.

We briefly sketch some of the main ideas of our novel encoding for preferred semantics. For both ASPARTIX and ASPARTIX-V the so-called saturation technique (which originates from the complexity analysis of disjunctive logic programs [3]) is employed. Intuitively, in the saturation technique encodings for preferred semantics we make a first "guess" for a set of arguments in the framework, and then verify if this set is admissible. To verify if this set is also subset maximal admissible, we perform a second guess and verify if this second guess is an admissible set that is a superset of the first guess. Usage of default negation within the saturation technique for the second guess is restricted, and thus in ASPARTIX a loop-style encoding checks if the second guess is admissible. Roughly, a loop construct in ASP checks a certain property for the least element in a set, and then "iteratively" for each (immediate) successor. If the property holds for the greatest element, it holds for all elements. In our encoding, the second guess is constructed using a disjunctive rule with a dynamic head. We illustrate the main idea in Listing 1.1.

Listing 1.1. Rule with dynamic head

```
witness(Z) : att(Z,Y) ← witness(X), att(Y,X).
```

The atoms with the predicate **witness** correspond to the second guess. After making sure that the second guess is non-empty (via another rule), this disjunctive rule with conditional literals "adds" for attacked witnesses other witnesses that defend them. The idea is to keep this second guess small to overcome computational overhead. Additional rules then verify if the witness set represents an admissible set that may be combined with the first guess to result in a larger admissible set. If this is the case, the first guess does not represent a preferred extension.

2.1 Supported Reasoning Tasks

ASPARTIX-V supports the following reasoning tasks:

- skeptical acceptance under preferred semantics,
- returning a single preferred extension, and
- enumerating all preferred extensions.

3 Competition Specific Settings

For conforming to the ICCMA 2015 specifications, we utilized a (slightly modified) bash script from the competition that in particular takes care of the correct output formating. After initial explorative performance testing, we decided to use clingo in its default setting, i.e., not using non-standard heuristical settings.

4 Web Access and License

The novel encodings are available from `http://www.dbai.tuwien.ac.at/proj/argumentation/systempage/` under "encodings for clingo using conditional disjunction". For running the encodings, clingo 4.4 is required, which can be downloaded from `http://potassco.sourceforge.net`. Notice that clingo 4.4 is published under the GNU public license (version 3).

Acknowledgements

This work has been funded by the Austrian Science Fund (FWF) through projects I1102 and Y698, and by Academy of Finland through grants 251170 COIN and 284591.

References

1. Dung, P.M.: On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games. Artificial Intelligence 77(2), 321–358 (1995)
2. Egly, U., Gaggl, S.A., Woltran, S.: Answer-set programming encodings for argumentation frameworks. Argument & Computation 1(2), 147–177 (2010)
3. Eiter, T., Gottlob, G.: On the computational cost of disjunctive logic programming: Propositional case. Ann. Math. Artif. Intell. 15(3-4), 289–323 (1995)
4. Gebser, M., Kaminski, R., Kaufmann, B., Lindauer, M., Ostrowski, M., Romero, J., Schaub, T., Thiele, S.: Potassco User Guide, second edn. (2015), `http://potassco.sourceforge.net`
5. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: *Clingo* = ASP + control: Preliminary report. In: Leuschel and Schrijvers [6], Theory and Practice of Logic Programming, Online Supplement
6. Leuschel, M., Schrijvers, T. (eds.): Technical Communications of the Thirtieth International Conference on Logic Programming (ICLP'14), vol. 14(4-5) (2014), Theory and Practice of Logic Programming, Online Supplement
7. Syrjänen, T.: Logic Programs and Cardinality Constraints: Theory and Practice. Ph.D. thesis, Aalto University (2009)