ASP for Abduction in Natural Language Understanding made more efficient using External Propagators

Peter Schüller[1], Carmine Dodaro[2], and Francesco Ricca[2]

[1] Computer Engineering Department, Faculty of Engineering
Marmara University, Turkey
peter.schuller@marmara.edu.tr
[2] Department of Mathematics and Computer Science
University of Calabria, Italy
{dodaro,ricca}@mat.unical.it

Abstract. Answer Set Programming (ASP) is a powerful paradigm for knowledge representation and reasoning. Several tasks in Natural Language Understanding (NLU) have been or have the potential to be modeled in ASP. Among these, abduction under various optimality conditions has been recently implemented in ASP. Experiments revealed that pure ASP is not effective enough because the complete instantiation of some critical constraints is not scalable. The recent extension of ASP solvers with external propagators may provide means for avoiding the instantiation bottleneck, and thus can help to obtain more efficient implementations of abduction in NLU via ASP. We conducted preliminary experiments with ASP solver interfaces for external propagators. The results look promising and provide directions for the development of full-fledged extensions of ASP solvers with non-ground constraints.

Keywords: ASP, Propagators, Abduction, NLU

ASP for Abduction in Natural Language Understanding

Abduction is a popular formalism for NLU, and we here consider a benchmark for abduction under preference relations of cardinality minimality, coherence [7], and Weighted Abduction [6]. For example given the text "Mary lost her father. She is depressed." using appropriate background knowledge and reasoning formalism we can obtain the interpretation of the sentence that Mary is depressed *because* of the death of her father.

Several tasks in Natural Language Understanding (NLU) have been or have the potential to be modeled effectively using logic programming techniques [2, 4, 8]. Answer Set Programming (ASP) [3] is a powerful paradigm for knowledge representation developed in the area of logic programming that is naturally suited as a computational means for the realization of abduction under preferences. Indeed, ASP formulations for the above NLU tasks were described in [8]. However, the prevalent evaluation strategy adopted by state of the art ASP systems, which

is carried out by performing in a row grounding (i.e., variable elimination) and solving (i.e., search for the answer sets of a propositional program), resulted to be not effective in large instances. This is due to the grounding blow-up caused by a small set of constraints.

In this work we study initial experiments on an extension of the ASP solver WASP [1] suitable to overcome this problem. Indeed, WASP has been extended with an API that allows a user to provide the solver with external Python programs extending the main solving procedure. In particular, we experimented with the API features for checking answer sets for constraint violations and adding propositional constraints lazily. In this way we circumvent the critical issue of grounding some constraints of the ASP programs modeling NLU tasks. Our solution works in the presence of optimization including the usage of unsatisfiable-core optimization, which is not possible in the Python API of the CLINGO solver [5].

Experimenting with external propagators

Preliminary Results. Table 1 shows preliminary experiments with the WASP solver on the Bwd-A encoding for first order Horn abduction from [8]. We show accumulated results for 50 natural language understanding instances from [7] for objective functions cardinality minimality, coherence [7], and Weighted Abduction [6]. We compare two evaluation methods: *Constraint* instantiates all constraints during the initial grounding step and sends them to the solver, while *Propagator* omits a significant portion of constraints (those related to transitivity) from the initial grounding and instantiates them lazily in the propagator whenever a transitivity violation is detected in an answer set candidate.

We observe that for all objective functions, there are out-of-memory conditions for 6 instances (maximum memory was 5 GB) while memory is not exhausted with propagators, and average memory usage is significantly lower with propagators (1.7 GB vs. around 150 MB). For cardinality minimality, the average time to find the optimal solution decreases sharply from 76 sec to 8 sec and we find optimal solutions for all instances. For coherence we can solve more instances optimally however the required time increases from 64 sec to 103 sec on average and 4 instances reach the timeout (600 sec). For Weighted Abduction, which represents the most complex optimization criterion, we solve fewer instances (37) compared with using pre-instantiated constraints (44 instances).

Propagators can clearly be used to trade space for time, and in some cases we decrease both space and time usage. For the complex Weighted Abduction objective functions, we can observe in the *Odc* column that many more invalid answer sets (2067) were rejected by the propagators compared with cardinality minimality (70) or coherence (751).

Ongoing and Future Work. Our current prototype implementation only checks when a full answer set candidate has been found, while most violated constraints could also be detected based on a partial interpretations. Thus we are implementing a propagator that can work on partial interpretations. We also plan to experiment with the optimal frequency of propagation, which is known

Objective Function	Method	MO #	TO #	OPT #	T sec	M MB	Odc #
Cardinality Minimality	Constraint	6	0	44	76	1715	0
	Propagator	0	0	50	8	119	70
Coherence	Constraint	6	0	44	64	1723	0
	Propagator	0	4	46	103	131	751
Weighted Abduction	Constraint	6	0	44	66	1731	0
	Propagator	0	13	37	229	141	2067

Table 1. Experimental Results: MO/TO indicates number of instances were memory/time was exhausted, OPT the number of optimally solved instances, T/M indicates average time and memory usage, and *Odc* shows number of times an answer set was invalidated and a new clause was learned, i.e., a constraint was lazily instantiated.

to play a role in similar implementations for robotics planning. Moreover, our prototype is able to learn only a single constraint per invalidated answer set, however one answer set might contain several violations of not instantiated constraints. Adding all these at once might guide the solver much better to find an optimal solution that violates no constraints.

Acknowledgements. This work has been supported by Scientific and Technological Research Council of Turkey (TUBITAK) Grant 114E777 and by MISE under project "PIUCultura", N. F/020016/01-02/X27.

References

1. Alviano, M., Dodaro, C., Leone, N., Ricca, F.: Advances in WASP. In: International Conference on Logic Programming and Non-monotonic Reasoning. pp. 40–54 (2015)
2. Balduccini, M., Baral, C., Lierler, Y.: Knowledge representation and question answering. In: Handbook of Knowledge Representation, Foundations of Artificial Intelligence, vol. 3, pp. 779–819. Elsevier (2008)
3. Brewka, G., Eiter, T., Truszczynski, M.: Answer set programming at a glance. Communications of the ACM 54(12) (2011)
4. Christiansen, H.: Constraint programming for context comprehension. In: Brézillon, P., Gonzalez, A.J. (eds.) Context in Computing - A Cross-Disciplinary Approach for Modeling the Real World, pp. 401–418. Springer (2014)
5. Gebser, M., Kaminski, R., Obermeier, P., Schaub, T.: Ricochet Robots Reloaded: A Case-Study in Multi-shot ASP Solving. In: Advances in Knowledge Representation, Logic Programming, and Abstract Argumentation, pp. 17–32. Springer (2015)
6. Hobbs, J.R., Stickel, M., Martin, P., Edwards, D.: Interpretation as Abduction. Artificial Intelligence 63(1-2), 69–142 (1993)
7. Ng, H.T., Mooney, R.J.: Abductive Plan Recognition and Diagnosis: A Comprehensive Empirical Evaluation. In: Knowledge Representation and Reasoning. pp. 499–508 (1992)
8. Schüller, P.: Modeling Variations of First-Order Horn Abduction in Answer Set Programming. Fundamenta Informaticae (2016), to appear, arXiv:1512.08899 [cs.AI]