Tools and Techniques for Easing the Application of Answer Set Programming

Davide Fuscà[1]

Department of Mathematics and Computer Science, University of Calabria, Italy
fusca@mat.unical.it

Abstract. Answer Set Programming (ASP) is a well-established declarative problem solving paradigm; it features high expressiveness and the ability to deal with incomplete knowledge, so it became widely used in AI and it is now recognized as a powerful tool for knowledge representation and reasoning (KRR).
Thanks to the expressive language and the availability of diverse robust systems, Answer Set Programming has recently gained popularity and has been applied fruitfully to a wide range of domains. This made clear the need for proper tools and interoperability mechanisms that ease the development of ASP-based applications. Also, the spreading of ASP from a strictly theoretical ambit to more practical scenarios requires additional features for easing the interoperability and integration with other software; furthermore, improving the performance of actual ASP system is crucial for allowing the use of the potential of ASP in new practical contexts.
The contribution of our work aims at addressing such challenges; we introduce new tools and techniques for fostering the application of ASP. In particular, we present EMBASP: a framework for the integration of ASP in external systems for general applications, over different platforms and ASP reasoners. Furthermore, we define proper means for handling external computations in ASP programs, and implement a proper framework for explicit calls to Python scripts via external atoms into the ASP grounder *I-DLV*. Eventually, we work at improving the ASP computation, and present *I-DLV+MS*, a new ASP system that integrates *I-DLV* with an automatic solver selector for inductively choose the best solver.

1 Introduction

Answer Set Programming (ASP) is a purely declarative formalism for knowledge representation and reasoning developed in the field of logic programming and nonmonotonic reasoning. Unlike the traditional programming languages, ASP allows representation of a given computational problem by means of a logic program specifying a description of the desired solution. The problem is encoded using logic rules, allowing for both disjunctions in rule heads and nonmonotonic negation in the body, such that its solution can be computed as models, called *answer sets*; hence, an answer set solver can be used in order to actually find such solutions [12].

Unlike formalisms like *Prolog* that have strong procedural elements, the answer set semantics are fully declarative, therefore neither the order of rules nor the order of the literals affects the result and program termination. The answer set semantics is an extension of the *stable model semantics* [9] that has been enriched and generalized. Such work on the language definition has been carried out by the scientific community, and several extensions have been studied and proposed over the years until the ASP-Core-2 [2] standard language became the official language of the ASP Competition series.

After more than twenty years of scientific research, the theoretical properties of ASP are well understood, as witnessed by the availability of a number of robust and efficient systems, including *DLV* [11], *wasp* [1], and the Potassco suite featuring *clingo*, *clasp* and *gringo* [6, 7]. The availability of such systems enabled ASP to be employed in many different domains for practical applications and the development of industrial and enterprise applications [3, 5]. Notably, this spreading of ASP from a strictly theoretical ambit to more practical aspects make clear the need for proper tools and interoperability mechanisms that ease the development of ASP-based applications. Also, the increasing employment of ASP in many different domains requires additional features for easing the interoperability and integration with other software and accommodating external source of computation and value invention within ASP programs.

The "traditional" approach to the evaluation of ASP programs relies on a grounding module (*grounder*), that generates a propositional theory semantically equivalent to the input program, coupled with a subsequent module (*solver*) that applies propositional techniques for generating its answer sets. Many ASP tools are focused in one of the two processes, due to the complexity of implementing a *monolithic* full ASP system. However, monolithic systems offer more control over the entire process enabling new features for improving the performance due to the coupling of the grounding and solving system. Moreover, the current ASP solvers feature several different optimization techniques, thus causing them to outperform each other, depending on the domain at hand. This is due to many reasons, such as different data structures, input simplifications and heuristics that might work better or worse, depending on the specific domain. Therefore, one might think of obtaining consistently good performance over different problems by means of proper machine learning techniques that inductively choose the "best" solver according to input features that increase the performance of the actual ASP systems.

Our work has the goal of proposing solutions for properly addressing several challenges arising from the practical application of ASP in real-world domains. In particular, the main contributions of the work are summarised in the following sections.

2 Integrating ASP Systems into external applications

EMBASP is a framework for the integration of ASP in external systems for general applications, along with ready-made specializations to different platforms and ASP reasoners. The framework features explicit mechanisms for two-way translations between strings recognisable by ASP solvers and objects in the programming language of choice, thus giving the developer the possibility to work separately on ASP-based modules and on the applications that make use of them. In order to illustrate the use of the framework, we implement an actual Java implementation and several specialized libraries for the state-of-the-art ASP systems, on mobile and desktop platforms, respectively, showing some applications developed that prove the effectiveness of the framework.

The framework, documentation, an application showcase and further details are freely available online [4].

3 Easing Interoperability of ASP Systems

In order to facilitate the integration of ASP with external systems we extend the ASP language with the capability of handling external computations with explicit calls to Python scripts via external atoms, and with interoperability mechanisms for the connection with relational and graph databases via explicit directives for importing/exporting data. Similar features have been already proposed in the literature; however, we proposed them mainly for two reasons. First, we wanted to enrich *I-DLV*, the new grounding module of the ASP system *DLV*[1], with such capabilites; furthermore, we wanted to guarantee optimal performance in all scenarios, even at the cost of lowering the expressivity of the extensions. It is worth noting that *I-DLV* has been conceived as a flexible tool for experimenting with ASP and its applications and as a system explicitly adapt for encompassing extensions and new features.

4 Automating Solver Selection

The performance of current ASP systems can be defined as good enough for different real-world applications. However, they feature several different optimization techniques, which cause systems to outperform each other depending on the domain at hand. The capability to enjoy good performance over various problems domains has already been studied by other communities, by means of proper strategies of *algorithm selection* [15] This approach consists of building machine learning techniques to inductively choose the "best" solver on the basis of some input program characteristics, or *features.* As far as ASP is concerned, some interesting works in this respect have already been carried out in [14].

I-DLV+MS is a new ASP system that integrates *I-DLV* with an automatic solver selector: machine-learning techniques are applied to inductively choose the best solver among a set of available ones, depending on the inherent features of the instantiation produced by *I-DLV*. We define a specific set of features, and

[1] It is worth noting that we were actively involved in the *I-DLV* project since the very beginning, being part of the core team during our PhD program

then carry out an experimental analysis for computing them over the instantiations obtained from the instances of benchmarks submitted to the 6th ASP competition. Furthermore, we test *I-DLV+MS* performance both against the state-of-the-art ASP systems and the best-established multi-engine ASP system ME-ASP, proving that *I-DLV+MS*, even though still at a prototypical stage, already shows good performance.

Notably, *I-DLV+MS* participated in the latest ($7th$) ASP competition [8], winning in the regular track, category SP (i.e., one processor allowed).

Moreover, a preliminary summary about *I-DLV+MS* will be presented at the 24th RCRA International Workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion (RCRA 2017), workshop of the 16th Conference of the Italian Association for Artificial Intelligence (AI*IA 2017).

5 Future Works

As a future work we would like to test the EMBASP framework over different platforms and solvers, and properly evaluate performances. Although the framework has been initially conceived for fostering the usage of ASP, its abstract core makes it also adaptable to other declarative knowledge representation formalisms; indeed, we introduce a proper extension supporting the *PDDL* planning language in [4], far beyond logic formalisms similar to ASP.

Moreover, we plan to increase the functionalities of *I-DLV* system related to the language extensions for easing the interoperability with the external system. More in detail, additional native directives for interoperating with external data will be added, along with means for a tighter integration with the ASP solver *wasp* in the new ASP system *DLV2*.

Notwithstanding the good performance, *I-DLV+MS* is still in a prototypical phase. As future work, we aim to test additional supervised learning method and also several frameworks for automatic algorithm configuration, like Autofolio [13] or Auto-WEKA [10]. We also plan to significantly extend experiments over additional domains and analyze possible over-fittings of the model and try different splits of the dataset for the train and test set among the available problems. Moreover, we aim to both include additional ASP solvers with different parameterizations, and explore more features for improving the classification capabilities and achieve better overall performance.

In addition, we are studying the possibility of taking advantage from machine-learning techniques for improving performance of ASP grounding engines; in particular, we plan to develop a built-in automatic algorithm selector within the *I-DLV* system (which *I-DLV+MS* is based on), thus opening up the possibility to dynamically adapt all the optimization strategies to the problem at hand.

References

1. Alviano, M., Dodaro, C., Leone, N., Ricca, F.: Advances in WASP. In: Logic Programming and Nonmonotonic Reasoning - 13th International Conference, LPNMR 2015, Lexington, KY, USA, September 27-30, 2015. Proceedings. pp. 40–54 (2015), http://dx.doi.org/10.1007/978-3-319-23264-5_5
2. Calimeri, F., Faber, W., Gebser, M., Ianni, G., Kaminski, R., Krennwallner, T., Leone, N., Ricca, F., Schaub, T.: Asp-core-2: Input language format (2012), https://www.mat.unical.it/aspcomp2013/files/ASP-CORE-2.03b.pdf
3. Calimeri, F., Ricca, F.: On the application of the answer set programming system dlv in industry: a report from the field. Book Reviews 2013(03) (2013)
4. EMBASP, https://www.mat.unical.it/calimeri/projects/embasp/
5. Erdem, E., Gelfond, M., Leone, N.: Applications of answer set programming 37, 53 (10 2016)
6. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: Clingo= asp+ control: Preliminary report. arXiv preprint arXiv:1405.3694 (2014)
7. Gebser, M., Kaminski, R., König, A., Schaub, T.: Advances in *gringo* series 3. In: Logic Programming and Nonmonotonic Reasoning - 11th International Conference, LPNMR 2011, Vancouver, Canada, May 16-19, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6645, pp. 345–351. Springer (2011), http://dx.doi.org/10.1007/978-3-642-20895-9_39
8. Gebser, M., Maratea, M., Ricca, F.: The design of the seventh answer set programming competition. In: Balduccini, M., Janhunen, T. (eds.) Logic Programming and Nonmonotonic Reasoning - 14th International Conference, LPNMR 2017, Espoo, Finland, July 3-6, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10377, pp. 3–9. Springer (2017), https://doi.org/10.1007/978-3-319-61660-5_1
9. Gelfond, M., Lifschitz, V.: The Stable Model Semantics for Logic Programming. In: Logic Programming, Proceedings of the Fifth International Conference and Symposium, Seattle, WA, Aug 15-19, 1988 (2 Volumes). pp. 1070–1080. MIT Press, Cambridge, Mass. (1988)
10. Kotthoff, L., Thornton, C., Hoos, H.H., Hutter, F., Leyton-Brown, K.: Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka. Journal of Machine Learning Research 17, 1–5 (2016)
11. Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The dlv system for knowledge representation and reasoning. ACM Transactions on Computational Logic (TOCL) 7(3), 499–562 (2006)
12. Lifschitz, V.: Answer Set Planning. In: Schreye, D.D. (ed.) Proceedings of the 16th International Conference on Logic Programming (ICLP'99). pp. 23–37. The MIT Press, Las Cruces, New Mexico, USA (Nov 1999)
13. Lindauer, M., Hoos, H.H., Hutter, F., Schaub, T.: Autofolio: An automatically configured algorithm selector. Journal of Artificial Intelligence Research 53, 745–778 (2015)
14. Maratea, M., Pulina, L., Ricca, F.: A multi-engine approach to answer-set programming. TPLP 14(6), 841–868 (2014), https://doi.org/10.1017/S1471068413000094
15. Rice, J.R.: The algorithm selection problem. Advances in Computers 15, 65–118 (1976), https://doi.org/10.1016/S0065-2458(08)60520-3