Answer Set Programming with External Sources

Christoph Redl[1]

1 **Institute of Information Systems, TU Vienna**
Karlsplatz 13, 1040 Vienna, Austria

1 Introduction and Problem Description

1.1 The Answer-Set Programming Paradigm

In recent years, the *Answer Set Programming* (ASP) paradigm has emerged as an important approach for declarative problem solving Problems are represented in terms of nonmonotonic logic programs, such that the models of the latter encode the solutions of a problem at hand. The most widely used notions of models in this context are *stable models* [11] and the generalized notion of *answer sets* for a (possible disjunctive) logic program [12]. One of the main reasons for the increasing popularity of ASP is the availability of sophisticated solvers for the respective languages, including DLV [14] and and clasp [10].

Formally, a propositional *answer set program* is a set of rules r of form

$$a_1 \vee \cdots \vee a_k \leftarrow b_1, \ldots, b_m, not\ b_{m+1}, \ldots, not\ b_n \tag{1}$$

where *not* denotes default-negation, and $a_i, 1 \leq i \leq k$, $b_j, 1 \leq j \leq n$ are literals, i.e., propositional atoms p or strongly (classically) negated atoms $\neg p$. Intuitively, a rule is satisfied by an interpretation, i.e. set of classical literals, I, iff either $b_i \notin I$ for some $1 \leq i \leq m$, $b_i \notin I$ for some $m+1 \leq i \leq n$, or $a_i \in I$ for some $1 \leq i \leq k$. A set of literals is an answer set of a program P, iff it is subset-minimal and satisfies all rules of P.

1.2 External Sources

The rise of the World Wide Web and distributed systems fostered the development of formalisms that are geared towards modularity and integrating multiple data sources. Various extensions of ASP that allow to access information in external sources different from logic programs have been proposed, e.g., the DLV^{DB} system [20] and HEX-*programs* [5]. The latter accommodate a universal bidirectional interface for arbitrary sources of external computation through the notion of an external atom and are in the focus of this PhD project. For example, an external atom of the form $\&synonym[thes\_file, car](X)$ might be used to access an external thesaurus "*thes_file*", and to retrieve all terms X in it that are synonyms of "*car*" (i.e., the atom evaluates to true). Using such external atoms, whose semantics is abstractly modeled by an input-output relationship, one can easily access different kinds of information resources and reason about them in a single program.

HEX-programs have been successfully used in various application domains; e.g., in the Semantic Web. A solver for HEX-programs is dlvhex [4], which implements a compilation approach for evaluation. First external atoms are replaced by ordinary atoms, and their truth value is guessed nondeterministically by disjunctive rules. The resulting program is then evaluated by the use of existing reasoners for answer set programs. In a postprocessing step, the guess for the external atoms is validated, i.e., the result is filtered to keep only those answer sets which are stable also under external sources. However, the performance of the reasoner is unsatisfactory. This is caused by the fact that the actual model building is hidden in the ASP solver employed at the backend. The development of scalable genuine algorithms is therefore the overall goal of the PhD project.

2 Background and Overview of the Existing Literature

The following subsections summarize approaches for three related topics which will play a role for the proposed project.

2.1 Model Finding for Propositional Ordinary Answer Set Programs

Model finding strategies for ASP programs can be classified in two major groups. The first one consists of algorithms that *reduce* the problem to another host logic, for which one can apply specialized SAT solvers. Approaches of the second kind search directly for models and are called *genuine* algorithms.

Genuine algorithms (as e.g. implemented by DLV) essentially boil down to an intelligent (restricted) enumeration of truth assignments. Deterministic consequences of the rules under partial truth assignments are computed in order to set the truth values of further atoms. If the assignment is still partial after this step, the value of an yet undefined atom is guessed in the style of DPLL algorithms for SAT, and again deterministic consequences are determined, etc.; in case the guess fails, the computation backtracks and the alternative value is considered. In contrast to `DLV`, the CLASP system [10] employs a conflict-driven method corresponding to conflict driven SAT solvers [15]. The distinguishing feature is *learning*: Whenever a conflict emerges, the literals which were initially responsible for the conflict are determined and recorded to prevent the reasoner from running into the same conflict again.

2.2 Intelligent Grounding

Non-ground answer set programs are like propositional programs but the atoms in a rule (1) are of the form $p(t_1, \ldots, t_n)$, where p is a (first-order) predicate and the t_i are terms (usually function-free) in a first order language. The semantics of such a program P is defined in terms of its *grounding*, which consists of all possible ground instances of the rules in P.

Most current ASP solvers (including `DLV` and CLASP) step to the grounding of a program before the actual model finding algorithms are started. In contrast, *lazy grounding*, as used for instance in GASP [16] and the ASPeRiX solver [13], is an alternative to pregrounding. Rules are only grounded when the body is satisfied, and consequently it prevents the grounding of rules which are potentially unnecessary.

2.3 External Domains

Clingo (gringo+CLASP) provides an interface which allows the user to call Lua functions (`http://www.lua.org`), a lightweight and embeddable scripting language, at certain points during evaluation, e.g., before grounding, after a model has been found, and after termination [9]. While communication between the reasoner and external scripts is possible, it is constrained to happen between specific evaluation phases and is not tightly coupled to and interleaved with model building, in contrast to HEX-programs.

DLV-EX and DLV-Complex are ASP systems extending DLV by external predicates and complex values like lists and sets. This allows to use sources of computation that are defined outside the logic program [1], which is especially useful for functions that are difficult or not efficiently expressible by rules (e.g. string manipulation functions).

3 Goal of the Research

I now describe the expected main outcomes of the PhD project.

3.1 Algorithms

One of the main results are newly developed efficient, native model-finding algorithms for ASP programs with external source access, which push the frontier of applicability. This allows to evaluate programs with external calls of reasonable size efficiently, relative to the intrinsic complexity.

3.2 Scalable Computation

The novel algorithms will relieve the current evaluation bottleneck of external accesses, and will contribute to position ASP programs with external access better as a formalism for declarative problem solving in real-world application settings. For programs of benign structure, I expect an exponential speedup in certain situations. In the general case, I aim at computation times comparable to ordinary ASP solvers, modulo complexity of external sources.

3.3 Prototype Implementation

The theoretical contributions are then realized in the dlvhex reasoner. The main practical outcome will be a prototype implementation of the developed evaluation algorithms as a proof of concept. The implementation will be evaluated using existing and newly developed benchmarks, and will be made publicly available through a website.

4 Current Status of the Research

At this point of the project, a state-of-the-art conflict-driven ASP solver was integrated into our prototype system dlvhex and extended by *additional learning techniques* to capture the semantics of external atoms. Empirical experiments show already a significant speedup compared to the traditional evaluation method, which is based on a translation to ordinary ASP programs. Results were published in an accepted ICLP 2012 paper [3].

A current topic is the development of techniques for *minimality checking* of answer set candidates in order to extend the algorithms to problems on the second level of the polynomial hierarchy. While the minimality check is polynomial for ordinary disjunction-free ASP programs, it becomes intractable in presence of disjunctions or nonmonotonic external sources. Hence, the development of efficient algorithms is not straightforward. While the traditional evaluation algorithm for HEX does the check explicitly, i.e., it directly searches for smaller models, I am currently working on a new algorithm based on *unfounded sets* [8]. Experiments show that this approach is superior to explicit minimality checking, and that optimization techniques for search space pruning are applicable. The approach is under preparation for being submitted as a conference paper.

Another current issue concerns *syntactic criteria* which allow for a simpler evaluation. The idea is that for programs of a certain structure, tailored and more efficient algorithms may employed.

5 Preliminary Results Accomplished

After finishing my master's studies in July 2010 (*Computational Intelligence*) and October 2010 (*Medical Computer Science*), I started my PhD studies in October 2010 and got a position as a research assistant at the Institute of Information Systems at TU Vienna, while

the research project at our institute (*Evaluation of ASP Programs with External Source Access*), which I am currently involved in, started in January 2012 and is planned for 3 years.

In the first year my PhD studies I have published the main results of my two master's theses in three conference papers. The first thesis in my studies of *Computational Intelligence* is on the development of a belief merging framework, called MELD, for the HEX-program solver dlvhex [17]. More specifically, the integration of heterogeneous data sources is abstractly described by tree-shaped *merging plans*, where the data sources appear in the leaf nodes and different *merging operators* in the inner nodes [19].

The MELD system is based on an extension of HEX-programs which allows for *program nesting*. That is, a HEX-program can contain calls to other programs and reason about their answer sets. Hence, answer sets are turned into accessible objects. Besides its application in the belief merging framework, program nesting has turned out to be useful also for the implementation of user-defined aggregation functions, which need to reason over multiple answer sets of subprograms. The subsystem for the evaluation of nested programs was described in another publication [7].

My second master's thesis in my studies of *Medical Computer Science* deals with a particular application of the belief merging framework MELD in medical and biological applications [18]. In particular, I developed *concrete merging operators* which are useful for the integration of *decision diagrams*, which are frequently used in medicine and related sciences to describe decision processes (e.g. for DNA classification). More results of this thesis were published in [6].

Besides writing the mentioned publications, I participated in writing the project proposal for our project to the Austrian science fond during the first year of my PhD studies. The overall goal of the project is the development of new evaluation algorithms for HEX-programs. This turned out to be necessary in order to make HEX-programs practically useful, as I discovered during my work on my master's theses that the scalability of the current algorithms is limited. The project was approved in September 2011 and started in January 2012.

In the first phase of the project, the ASP solver clasp was integrated into dlvhex as new backend, replacing DLV. This allows for a much tighter coupling of the solver with external source evaluation, exploiting clasp's extensible SMT interface, and is the foundation for the further development. At this point of the project I have developed *external behavior learning* (EBL) as a new learning strategy besides classical conflict-driven learning as used by ordinary ASP solvers. While conflict-driven learning gains new knowledge from conflict situations, EBL learns from evaluations of external sources. This knowledge can be used in the further search to exclude model candidates beforehand if they are not compliant with the external sources. The learned knowledge about external sources is itself represented as a nogood clause and recorded in the solver. As the algorithm proceeds, this knowledge is used to guide the search.

The prototype implementation of *external behavior learning* (EBL) shows already positive effects wrt. performance. Depending on the program structure and the external atoms involved, I could observe an up to exponential speedup in some cases, e.g., for some programs with DL-atoms; DL-atoms allow for querying description logic knowledge bases from the logic program by the use of external atoms. Also for string manipulation functions, which are conceptually simple but important from a practical point of view, I could observe very promising improvements by EBL and exploiting functionality. Details of EBL are described in an accepted ICLP 2012 and TPLP paper [3].

6 Open Issues and Expected Achievements

The current translation-based HEX-evaluation algorithm does not scale well to real-world applications because of the high number of generated model candidates. The situation is comparable to the evolution of the answer set semantics as a programming paradigm: at the very beginning, the lack of efficient ASP solvers prevented its use in practice. But since efficient algorithms and ASP systems have become available, ASP has gained momentum in a range of applications, from science over humanities to industrial use. The main goal of the PhD project is significantly better scalability. It is expected that an exponential speedup is achievable in some cases, whereas a prediction in the average case is difficult to make.

One step towards this goal was the development of external behavior learning (EBL). In the next step, the basic idea shall be refined. Additionally to deriving knowledge from known properties of external sources, it shall also be possible to write *customized learning functions* for specific sources. The idea is that the provider of an external source often has a better insight to the behavior of the source, which can be used to help the reasoner excluding model candidates early. For this purpose, a user-friendly language shall be developed, which is an open issue.

Another important open issue, which was disregarded until now, concerns the minimality check of model candidates. While I have recently developed a minimality checking algorithm based on unfounded sets [8] (instead of the explicit minimality check, which was used until now), it is still realized as a post-check. That is, the algorithm first constructs an answer set candidate, and then filters out those candidates which contain unfounded sets. An open issue is *interleaving* the unfounded set search with the main search for answer sets. The idea is that one can in some cases already identify unfounded sets when the interpretation is still partial. Then it makes no sense to further complete the interpretation, but it is more reasonable to immediately backtrack.

Case-based reasoning (CBR) is solving of current computational problems by the use of relevant aspects of past problem solutions [2]. This topic seems to be a related to HEX-program evaluation using EBL. An open issue is therefore also the investigation to what extend CBR techniques can be adopted for our purposes.

7 Bibliographical references

References

1 F. Calimeri, S. Cozza, and G. Ianni. External Sources of Knowledge and Value Invention in Logic Programming. *Annals of Mathematics and Artificial Intelligence*, 50(3–4):333–361, Aug. 2007.

2 R. L. de Mántaras and E. Plaza. Case-based reasoning: An overview. *AI Communications Journal*, 10(1):21–29, 1997.

3 T. Eiter, M. Fink, T. Krennwallner, and C. Redl. Conflict-driven ASP solving with external sources. *Theory and Practice of Logic Programming: Special Issue ICLP*, 2012. To appear.

4 T. Eiter, G. Ianni, R. Schindlauer, and H. Tompits. dlvhex: A Prover for Semantic-Web Reasoning under the Answer-Set Semantics. In *Proceedings of the ICLP'06 Workshop on Applications of Logic Programming in the Semantic Web and Semantic Web Services (ALPSWS2006)*, pages 33–39. CEUR WS, 2006.

5 T. Eiter, G. Ianni, R. Schindlauer, and H. Tompits. Effective Integration of Declarative Rules with External Evaluations for Semantic-Web Reasoning. In *Proceedings of the 3rd*

European Conference on Semantic Web (ESWC 2006), volume 4011 of *LNCS*, pages 273–287. Springer, 2006.

6 T. Eiter, T. Krennwallner, and C. Redl. Declarative merging of and reasoning about decision diagrams. In A. D. Palù, A. Dovier, and A. Formisano, editors, *Workshop on Constraint Based Methods for Bioinformatics (WCB 2011), Perugia, Italy, September 12, 2011*, pages 3–15. Dipartimento di Matematica e Informatica, Universita degli Studi di Perugia, September 2011.

7 T. Eiter, T. Krennwallner, and C. Redl. Nested hex-programs. *CoRR*, abs/1108.5626, 2011.

8 W. Faber. Unfounded sets for disjunctive logic programs with arbitrary aggregates. In *In Logic Programming and Nonmonotonic Reasoning, 8th International Conference (LPNMR'05), 2005*, pages 40–52. Springer Verlag, 2005.

9 M. Gebser, B. Kaufmann, R. Kaminski, M. Ostrowski, T. Schaub, and M. Schneider. Potassco: The Potsdam Answer Set Solving Collection. *AI Commun.*, 24(2):107–124, 2011.

10 M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub. *Clasp* : A conflict-driven answer set solver. In C. Baral, G. Brewka, and J. S. Schlipf, editors, *LPNMR*, volume 4483 of *Lecture Notes in Computer Science*, pages 260–265. Springer, 2007.

11 M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. In R. Kowalski and K. Bowen, editors, *Logic Programming: Proceedings of the 5th International Conference and Symposium*, pages 1070–1080. MIT Press, 1988.

12 M. Gelfond and V. Lifschitz. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing*, 9(3–4):365–386, 1991.

13 C. Lefèvre and P. Nicolas. The First Version of a New ASP Solver: ASPeRiX. In E. Erdem, F. Lin, and T. Schaub, editors, *Proceedings of the 10th International Conference on Logic Programming an Nonmonotonic Reasoning (LPNMR 2009), Potsdam, Germany, September 14-18, 2009*, volume 5753 of *LNCS*, pages 522–527. Springer, 2009.

14 N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello. The DLV System for Knowledge Representation and Reasoning. *ACM Transactions on Computational Logic (TOCL)*, 7(3), July 2006.

15 D. G. Mitchell. A SAT solver primer. *EATCS Bulletin (The Logic in Computer Science Column)*, 85:112–133, February 2005.

16 A. Palù, A. Dovier, E. Pontelli, and G. Rossi. Answer set programming with constraints using lazy grounding. In P. Hill and D. S. Warren, editors, *Proceedings of the 25th International Conference on Logic Programming (ICLP 2009)*, volume 5649 of *LNCS*, pages 115–129. Springer, 2009.

17 C. Redl. Development of a belief merging framewerk for dlvhex. Master's thesis, Vienna University of Technology, Institute of Information Systems, Knowledge-Based Systems Group, A-1040 Vienna, Karlsplatz 13, July 2010.

18 C. Redl. Merging of biomedical decision diagrams. Master's thesis, Vienna University of Technology, Institute of Information Systems, Knowledge-Based Systems Group, A-1040 Vienna, Karlsplatz 13, October 2010.

19 C. Redl, T. Eiter, and T. Krennwallner. Declarative Belief Set Merging using Merging Plans. In R. Rocha and J. Launchbury, editors, *13th International Symposium on Practical Aspects of Declarative Languages (PADL'11), Austin, Texas, U.S.A., January 24-25, 2011*, volume 6539 of *LNCS*, pages 99–114. Springer, January 2011.

20 G. Terracina, N. Leone, V. Lio, and C. Panetta. Experimenting with recursive queries in database and logic programming systems. *CoRR*, abs/0704.3157, 2007.