Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence

Backdoors to Tractability of Answer-Set Programming

Johannes Klaus Fichte*
Vienna University of Technology, Austria
fichte@kr.tuwien.ac.at

Thesis Background

Reasoning is "the power of the mind to think, understand, and form judgments by a process of logic" (McKean 2001). In the late 1950s and 1960s researchers came up with the idea to express human knowledge with mathematical logic and to create a machine that can derive conclusions in the real world (McCarthy et al. 1955). In the early 1970s a logic programming system called Prolog was invented by Colmerauer and Roussel (1993) which describes a problem by means of facts and rules that form a logic program. Classical logic programming is characterized by the fact that we do reasoning in a closed world and whenever we add new facts or rules to the program this will never invalidate any conclusions, so-called *monotonic reasoning*. From the early 1980s the research has moved closer to real world reasoning which is somewhat different. A main characteristic is that we derive conclusion because we have no evidence for the contrary (*reasoning by default*). Since additional information may retract conclusions, such reasoning is called *nonmonotonic* (Brewka, Niemelä, and Truszczyński 2008).

In the late 1980s a new semantics (*stable model semantics*) for reasoning by default was proposed by Gelfond and Lifschitz (1988) and implemented into the new programming paradigm called *answer-set programming* (ASP) (Marek and Truszczynski 1999). Various ASP solvers have consequently been designed. Nonmonotonic reasoning has come of age, since modern ASP solvers work efficiently on industrial instances (Gebser et al. 2012). Unfortunately, the main computational problems of ASP (such as deciding whether a program has a solution, or if a certain atom is contained in at least one or in all solutions) are of high worst-case complexity and are located at the second level of the Polynomial Hierarchy (Eiter and Gottlob 1995).

In the 1990s and 2000s research on the complexity of problems in the field of nonmonotonic reasoning has mainly focused on classical computational complexity where we measure the amount of a resource (e.g., time or space) in a function of the input. Unfavorably, we ignore the structural nature of our input instances. Thus problems from an industrial context seem to be harder in theory than they are in practice. An alternative approach is to measure the complexity in an additional parameter which depends arbitrarily on the input. The underlying theory is called *parameterized complexity theory* (Downey and Fellows 1999). Its fundamental concept of *fixed-parameter tractability* relaxes classical polynomial-time tractability in such a way that all nonpolynomial parts depend only on the size of the parameter and not on the size of the input. This explains gaps between classical theory and practice if the parameter is comparatively small. In recent years this has become a very active research area.

A tool that offers a parameter for parameterized complexity analysis is *backdoors* which are small sets of atoms that represent "clever reasoning shortcuts" through the search space. Exemplarily, we consider the Boolean satisfiability problem where backdoors originate from (Williams, Gomes, and Selman 2003). The problem is to decide whether a Boolean formula is satisfiable or not. We take a class $\mathcal{C}$ of formulas where membership and satisfiability are decidable in polynomial time e.g., Horn formulas. A set X of variables of a formula φ is a *$\mathcal{C}$-backdoor* if all formulas, that can be obtained from φ by instantiating the variables in X, yield simplified formulas which belong to the class $\mathcal{C}$. Once we have found a backdoor X, the formula φ can be evaluated by considering all $2^{|X|}$ truth assignments to the variables in X.

Research Question and Contribution

The practical results of ASP indicate that classical complexity theory is insufficient as a theoretical framework to explain why ASP solvers work fast on industrial applications. Complexity analysis by means of parameterized complexity theory seems to be promising, because we think that the reason for the gap between theory and practice is the presence of a "hidden structure" in real-world instances. The application of parameterized complexity theory to ASP would give a crucial understanding of how solver heuristics work. This profound understanding can be used to improve the decision heuristics of modern solvers and yields new efficient algorithms for decision problems in the nonmonotonic setting.

My research aims to explain the gap between theoretical upper bounds and the effort to solve real-world instances. I will further develop by means of parameterized complexity exact algorithms which work efficiently for real-world instances when parameterized by the size of a backdoor. This seems to be very reasonable as backdoors have suc-

*Research supported by ERC (COMPLEX REASON 239962).

cessfully been used to analyze the performance of SAT algorithms (Williams, Gomes, and Selman 2003) and ASP and SAT are closely related (Lin and Zhao 2003). I will utilize backdoors to explain differences and improve common algorithms.

The main research questions read as follows:

- Can we adapt the backdoor framework to ASP?
- Can we develop new efficient algorithms for the main computational problems of ASP when the problems are parameterized by the size of a backdoor?
- Can we determine small backdoors efficiently?
- Will the algorithms be of practical use, at least for certain classes of instances, and hence might fit into a portfolio-based solver?
- Will we be able to improve heuristics in modern solvers in particular the decision strategy?

The backdoor approach offers a parameter to analyze parameterized complexity of reasoning problems in the nonmonotonic setting. Treewidth (Jakl, Pichler, and Woltran 2009) has already been considered in literature and I am aware that other parameters may be applicable. I consider backdoors as orthogonal to treewidth. Considering the area of SAT, backdoors did not yield a considerable speed-up in current state of the art SAT solvers until now, but they improved the understanding of modern techniques in SAT solvers. The same might happen in the context of ASP.

Backdoors for ASP

In (Fichte and Szeider 2012) we have explained how the concept of backdoors can be adapted to propositional ASP. A *backdoor* of a program is a set of atoms such that any instantiation of the atoms yields a simplified program that belongs to a *target class* of programs. In constrast to SAT backdoors where a base class only needs to be polynomial-time decidable, backdoors for ASP need more. Therefore, we introduced the new notion of an enumerable class. Moreover, we need a two-step approach (generate an answer-set candidate and check the candidate) to utilize a backdoor. We have established that the main computational problems of ASP are tractable when the decision problem is tractable for the considered target class and we additionally bound the size of a smallest backdoor by a constant. We have utilized recent advances of parameterized complexity to detect small backdoors. The target classes considered by us are based on notions of acyclicity where various types of cycles (good and bad cycles) are excluded from graph representations of programs. Furthermore, we have studied the class of Horn programs. I have generalized some results in (Fichte 2012). We have investigated backdoors to the intractable target classes of normal and tight programs to the problems *brave reasoning* and *cautious reasoning* which ask whether a given atom is contained in at least one or in all answer sets, respectively. We have transformed both problems into SAT such that the combinatorial explosion, which is expected when transforming problems from the second level of the Polynomial Hierarchy to the first level, is confined by the size k of a backdoor to a normal program, while the running time is polynomial in the input size n and the order of the polynomial is independent of k. Moreover, we have shown that such a transformation is not possible if we consider backdoors with respect to tightness instead of normality (Fichte and Szeider 2013).

Future Work

Currently, my supervisor and I prepare work that additionally contains considerations on preprocessing (kernelization) and a more precise approach to the evaluation of backdoors (dynamic backdoors). It will also consist of various experiments where I determine the size of a smallest backdoor for various ASP instances. Subsequently, I will focus on experimental work in particular on the influence of backdoors to the heuristics of ASP solvers (Gebser et.al. 2013) and an implementation of the translation of brave and cautious reasoning into ASP (Gebser, Kaminski, and Schaub 2011) or SAT.

References

Brewka, G.; Niemelä, I.; and Truszczyński, M. 2008. Nonmonotonic reasoning. *Foundations of Artificial Intelligence*. 3:239–284.

Colmerauer, A., and Roussel, P. 1993. The birth of prolog. In *HOPL Preprints*.

Downey, R.G.; and Fellows, M.R. 1999. *Monographs in CS*.

Eiter, T., and Gottlob, G. 1995. On the computational cost of disjunctive logic programming: Propositional case. *Ann. Math. Artif. Intell.* 15(3–4).

Fichte, J. K., and Szeider, S. 2012. Backdoors to tractable answer-set programming. Extended and updated version of a paper that appeared in the Proc. of *IJCAI'11*. arXiv:1104.2788.

Fichte, J. K., and Szeider, S. 2013. Backdoors to normality for disjunctive logic programs. *AAAI'13*. To appear.

Fichte, J. K. 2012. The good, the bad, and the odd: Cycles in answer-set programs. In *LNCS* 7415. 78–90.

Gebser, M.; Kaminski, R.; Kaufmann, B.; and Schaub, T. 2012. *Answer Set Solving in Practice*.

Gebser, M.; Kaminski, R.; and Schaub, T. 2011. Complex optimization in answer set programming. *Th. Pract. Log. Prog.* 11(4-5).

Gebser, M.; Kaufmann, B.; Otero, R.; Romero, J.; Schaub, T.; Wanko, P. 2013. Domain-specific Heuristics in Answer Set Programming. AAAI'13. To appear.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. ICLP/SLP'88.

Jakl, M.; Pichler, R.; and Woltran, S. 2009. Answer-set programming with bounded treewidth. IJCAI'09.

Lin, F., and Zhao, J. 2003. On tight logic programs and yet another translation from normal logic programs to propositional logic. IJCAI'03.

Marek, V. W., and Truszczynski, M. 1999. Stable models and an alternative logic programming paradigm. *The Logic Programming Paradigm: a 25-Year Perspective*.

McCarthy, J.; Minsky, M. L.; Rochester, N.; and Shannon, C. 1955. A proposal for the dartmouth summer research project on artificial intelligence.

McKean, E. 2001. *The New Oxford American Dictionary*. Oxford University Press.

Williams, R.; Gomes, C.; and Selman, B. 2003. Backdoors to typical case complexity. IJCAI'03.