Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence

Explaining Answer Set Programming in Argumentative Terms

Claudia Schulz
claudia.schulz@imperial.ac.uk
Department of Computing
Imperial College London
London SW7 2AZ, UK

Introduction

Argumentation Theory and Answer Set Programming (ASP) are two prominent theories in the field of knowledge representation and non-monotonic reasoning, where Argumentation Theory stands for a variety of approaches following similar ideas. The main difference between Argumentation Theory and ASP is that the former focusses on representing knowledge and reasoning about it in a way that resembles human reasoning, neglecting the efficiency of the reasoning procedure, whereas the latter is concerned with the efficient computation of solutions to a reasoning problem, resulting in a less human-understandable process.

In recent years, ASP has been frequently applied for the computation of reasoning problems represented in argumentation-theoretical-terms and has been found an efficient method for determining solutions to problems in Argumentation Theory (Egly, Gaggl, and Woltran 2010; Wakaki and Nitta 2008). My research is concerned with the opposite direction, i.e. with applying Argumentation Theory to ASP in order to explain the solutions to an ASP reasoning problem in a more human-understandable way. Developing such an explanation method also involves to investigate both the exact relationship between different approaches in Argumentation Theory in order to find the most suitable one for explanations and their connection with ASP, in particular with respect to their semantics.

Background

In *Answer Set Programming* (ASP) (Gelfond and Lifschitz 1991) knowledge is represented as a *logic program*, that is in terms of rules containing defeasible elements called negation-as-failure (NAF) literals, which are assumed to be true as long as the contrary cannot be proven. The semantics of ASP are defined declaratively as a fixpoint, yielding sets of accepted literals called *answer sets*, intuitively a set of all non-contradictory conclusions drawn from the logic program. Such an answer set does not provide any further information as to why a literal is or is not part of it. This can be a shortcoming, especially when using ASP in applications such as decision support systems or agent planning, where it is important to understand why a decision was made or why a potential decision was not reached.

In *Argumentation Theory* two types of approaches can be distinguished: Abstract approaches such as *Abstract Argumentation* (AA) (Dung 1995) represent knowledge in terms of arguments along with a conflict relation between them. Different semantics exist, which are all determined based on the conflicts between arguments, yielding sets of accepted arguments called *extensions*. On the other hand, structured approaches such as *Assumption-Based Argumentation* (ABA) (Bondarenko et al. 1997; Toni 2014) or *ASPIC+* (Prakken 2010) represent knowledge in the form of rules along with contrary relations between elements of the rules. Especially the representation of knowledge in ABA, where rules contain defeasible elements called *assumptions*, is very close to the ASP representation as a logic program. In structured approaches arguments can be constructed from the rules and conflict relations between the arguments can be inferred from the contrary relations. The semantics can then be expressed in terms of extensions as in the case of abstract approaches. Thus, an extension of a structured approach provides further information as to why an argument is part of it, namely in terms of the conflict relation with other arguments and the internal structure of the argument.

Past Work - ABA-Based Answer Set Justification

Since reasoning problems in structured approaches of Argumentation Theory are represented in a very similar way to ASP problems, I investigated the relationship between the solutions of a problem computed by ASP and the solutions of the same problem determined using Argumentation Theory. It turned out that consistent answer sets in ASP correspond to the *stable extensions* of both ASPIC+ (Schulz, Sergot, and Toni 2013) and ABA (Schulz and Toni 2014), in particular for every literal l in an answer set there exists a (corresponding) argument with conclusion l in the corresponding stable extension and vice versa. Based on this correspondence, I developed a method called *Argumentation-Based Answer Set Justification* (Schulz, Sergot, and Toni 2013) for explaining why a literal is or is not part of an answer set in terms of the structure and relation information of the corresponding argument in ASPIC+. This explanation approach was improved by using ABA instead of ASPIC+, yielding *ABA-Based Answer Set Justification* (Schulz and Toni 2013; 2014).

In order to explain why a literal l is in an answer set using ABA-Based Answer Set Justification, a corresponding argument A of l has to be found, i.e. an argument with conclusion l in the corresponding stable extension. For this argument A an *Attack Tree* is constructed that expresses which arguments attack A, how these attacking arguments are further attacked and so on. An Attack Tree provides a first explanation as to why l is part of the answer set. However, the explanation is in terms of arguments rather than literals, and experts in ASP might prefer an explanation in terms of literals. Thus, ABA-Based Answer Set Justification provides a second kind of explanation, called *LABAS Justification*, in terms of support and conflict relations between literals, extracted from the Attack Tree. The support relation between literals is extracted from the internal structure of arguments in the tree, in particular the sets of assumptions and facts used to construct an argument, whereas the conflict relation between literals is derived from the attacks between arguments in the tree. Both Attack Trees and LABAS Justifications can be displayed as graphs and can also be used to explain why a literal is not part of an answer set. The only existing approach for justifying literals with respect to an answer set are off-line justifications (Pontelli, Son, and Elkhatib 2009), which are conceptually different from LABAS Justifications and which use the well-founded model semantics for the construction.

Future Work - Justifying Inconsistency

ABA-Based Answer Set Justification can only be applied if the logic program is consistent, i.e. if it has meaningful answer sets. However, when formalising a reasoning problem using ASP mistakes can easily be made, leading to an *inconsistent* logic program which has no answer sets or whose answer set is not meaningful (i.e. the set of all literals occurring in the logic program). In this case, there is a necessity to explain why the logic program is inconsistent. Different *debugging* approaches for ASP have been developed which identify different types of inconsistencies (Gebser et al. 2008; Oetsch, Pührer, and Tompits 2010; 2011). However, all of them lack the ability to automatically identify the cause of inconsistency if a logic program has no answer set because a literal l depends (in an odd-length loop) on the NAF literal $not\ l$. Identifying this as well as other causes of inconsistency and to explain them is part of my ongoing research.

As a first step I am currently trying to identify which types of mistakes a logic program contains by checking whether or not it has an answer set and whether or not it has a 3-valued stable model (Przymusinski 1991). These answer sets and 3-valued stable models can then be used to construct different kinds of justifications for the different mistakes by using ABA. The justifications are based on my result (Schulz and Toni 2015) that complete extensions in ABA correspond to 3-valued stable models of a logic program and that semi-stable extensions correspond to L-stable models (Eiter, Leone, and Sacc 1997). So far, the ideas of how to construct justifications for the different types of mistakes are preliminary. By January 2015 I expect to have fully worked out the details of how to justify certain types of mistakes.

During the remaining time of my PhD, I am planning to turn from theoretical investigations to more practical work. On the one hand, I will implement the justification approaches for consistent and inconsistent logic programs. On the other hand, I am planning to apply the justification methods to real data, in particular to medical or legal decision making problems in order to judge its impact in applications.

References

Bondarenko, A.; Dung, P.; Kowalski, R.; and Toni, F. 1997. An abstract, argumentation-theoretic approach to default reasoning. *AI* 93(1-2):63 – 101.

Dung, P. M. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *AI* 77(2):321–357.

Egly, U.; Gaggl, S. A.; and Woltran, S. 2010. Answer-set programming encodings for argumentation frameworks. *Argument & Computation* 1(2):147–177.

Eiter, T.; Leone, N.; and Sacc, D. 1997. On the partial semantics for disjunctive deductive databases. *Annals of Mathematics and AI* 19(1-2):59–96.

Gebser, M.; Pührer, J.; Schaub, T.; and Tompits, H. 2008. A meta-programming technique for debugging answer-set programs. In *AAA'08I*.

Gelfond, M., and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Computing* 9:365–385.

Oetsch, J.; Pührer, J.; and Tompits, H. 2010. Catching the ouroboros: On debugging non-ground answer-set programs. *TPLP* 10(4-6):513–529.

Oetsch, J.; Pührer, J.; and Tompits, H. 2011. Stepping through an answer-set program. In *LPNMR'11*.

Pontelli, E.; Son, T. C.; and Elkhatib, O. 2009. Justifications for logic programs under answer set semantics. *TPLP* 9(1):1–56.

Prakken, H. 2010. An abstract framework for argumentation with structured arguments. *Argument & Computation* 1(2):93–124.

Przymusinski, T. 1991. Stable semantics for disjunctive programs. *New Generation Computing*.

Schulz, C., and Toni, F. 2013. Aba-based answer set justification. *TPLP, On-line Supplement* 13(4-5).

Schulz, C., and Toni, F. 2014. Justifying answer sets using argumentation. *TPLP* To appear.

Schulz, C., and Toni, F. 2015. Logic programming in assumption-based argumentation revisited – semantics and graphical representation. In *AAAI'15*.

Schulz, C.; Sergot, M.; and Toni, F. 2013. Argumentation-based answer set justification. In *Commonsense'13*.

Toni, F. 2014. A tutorial on assumption-based argumentation. *Argument & Computation* 5(1):89–117.

Wakaki, T., and Nitta, K. 2008. Computing argumentation semantics in answer set programming. In *New Frontiers in AI, JSAI'08*.