GG-Sts: Argumentation Solver from Aalto University in ICCMA 2017

Shahab Tasharrofi Tomi Janhunen

Abstract

This paper describes the argumentation solver GG-Sts [10] from Aalto University which is submitted to 2017 Argumentation Competition (ICCMA 2017 [1]). GG-Sts is a solver on top of Sat-to-Sat [7] which is developed based on the ideas from Bogaerts et. al. [3]. The main importance of GG-Sts is that it is obtained automatically from a declarative description of the different argumentation semantics.

1 Introduction

Abstract argumentation frameworks (AFs) [2] are simple and abstract systems to deal with contentious information and draw conclusions from it. In AFs, we are not interested in the actual content of arguments; this information is abstracted away. Formally, an *abstract argumentation framework* Θ is a directed graph (A, R) in which the nodes A represent arguments and the arcs in R form the attack relation between arguments. We say that argument *a attacks* b if $(a, b) \in R$. Given such a graphical representation (A, R) of an abstract argumentation instance, many different semantics are proposed to capture the intended solutions to that instance. Such semantics are defined in terms of *extensions*, i.e., subsets E of arguments A that satisfy certain criteria. We refer interested readers to [2, 12, 6] for a more detailed description of different argumentation semantics.

While the prominence of abstract argumentation frameworks (AFs) necessitates the development of well-performing solvers for them, the existence of several semantics with varying computational and/or conceptual complexity has slowed the process of developing AF solvers that use state-of-the-art techniques. To address this issue, Bogaerts et al. [3] proposed to use a declarative approach to the development of new solvers for knowledge representation systems. They proposed to use second-order logic to describe the semantics of a new logical fragment and to convert such a second-order specification to an instance of Sat-to-Sat [7, 5, 4], our Beyond-SAT solver.

While the approach described in [3] provides a simple way to obtain new solvers, due to its dependence on answer set solving technology for the grounding phase, it requires its input to be in a reified form. For example, while Bogaerts et. al. [3] could easily obtain solvers for new logical fragments such

as equilibrium semantics [9], the input to those solvers could not take the form of a propositional formula as the syntax of equilibrium logic dictates. Instead, one would have to convert a propositional formula to its meta-representation by using facts such as "conj(i, j, k)" which, informally, means that subformula number i is the conjunction of subformulas j and k.

This paper describes a solver for abstract argumentation framework that is obtained automatically using an extension of the principle set forth by Bogaerts et. al. [3], i.e., relying on a declarative approach towards solver development. That is, using a yet-unpublished framework which can now translate a declarative specification of the *syntax*, the *semantics*, and the *grounding process* of a logical fragment to a solver, we now present our new and automatically-generated AF solver: GG-Sts [10].

Source codes for GG-Sts can be downloaded from the following URL: `https://research.ics.aalto.fi/software/sat/gg-sts/`.

2 Technical Details

This section describes the technical details behind GG-Sts. We first note that the name GG-Sts is meant to denote that the final solver is obtained by combining two technological parts: (1) Abbreviation GG that stands for *Grounder Generator*, our new and yet-unpublished framework for automatic construction of a declaratively specified grounding process; and, (2) postfix Sts which abbreviates Sat-to-Sat, our in-house nested SAT solver at Aalto University. The rest of this section briefly describes each of these two parts.

2.1 Pre-processing and Translation for Abstract Argumentation Graphs

This section briefly describes the first stage of our GG-Sts argumentation solver that takes an argumentation graph, a chosen semantics and a task to perform and generates a Sat-to-Sat instance. As discussed in Section 1, this part is automatically generated from a declarative specification of the grounding process and contains the following three segments:

1. **Grammar:** A context-free grammar that describes the input format and accepts both the TGF and APX formats. In order to distinguish between these two formats, we assume that the instance file is preceded by one of the keywords TGF or APX. Our call shell scripts guarantees that the right preceding keyword is forwarded to our declaratively specified pre-processing tool. Moreover, since the translation steps depend on the semantics argumentation and the task, the input grammar is extended to include them at the end of the input file.

2. **Bound Computation:** While the context-free grammar above declaratively specifies the input format to be expected in the argumentation competition, we use an extension of relational algebra to declaratively

specify what needs to be computed during the translation process. At this stage, we have only hard-coded datalog definitions for bound computation according to lifted unit propagation procedure as proposed by Vaezipoor et. al. [11].

3. **Generating Output:** The final step in our translation is to generate a Sat-to-Sat encoding of the original argumentation problem given the bounds computed in the previous section. At the current step this part is done only for our in-house nested SAT solver: Sat-to-Sat. However, in the future, we plan to add support for other solver backends such as QBF and/or answer set solvers.

2.2 Solving Abstract Argumentation Instances

In the second stage of GG-Sts, we depend on our state-of-the-art nested SAT solver Sat-to-Sat to find a solution to the original argumentation problem. Sat-to-Sat is a conflict driven clause learning solver that goes beyond NP using nested SAT calls which was recently introduced in [7] to solve $\exists\forall$-QBFproblems. It was then extended in [5, 3] to arbitrary QBF problems. Essentially, this framework is based on lazy clause generation [8] where clauses are obtained from calls to other SAT solvers.

The main difference between Sat-to-Sat and a naive nesting of SAT solvers is that Sat-to-Sat uses the concept of under-approximating formulas to deal with non-exact interpretations. That is, Sat-to-Sat removes the requirement by naive nested SAT solving that the inner solver can only be called after the outer solver has assigned all its variables. Instead, in Sat-to-Sat, the inner solver can be called at anytime during the reasoning of the outer solver leading to a more efficient solving process.

References

[1] ICCMA 2017, 2nd international competition on computational models of argumentation. Accessed: 2017-04-01.

[2] Pietro Baroni, Martin Caminada, and Massimiliano Giacomin. An introduction to argumentation semantics. *Knowledge Eng. Review*, 26(4):365–410, 2011.

[3] Bart Bogaerts, Tomi Janhunen, and Shahab Tasharrofi. Declarative solver development: Case studies. In Chitta Baral, James P. Delgrande, and Frank Wolter, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifteenth International Conference, KR 2016, Cape Town, South Africa, April 25-29, 2016.*, pages 74–83. AAAI Press, 2016.

[4] Bart Bogaerts, Tomi Janhunen, and Shahab Tasharrofi. Sat-to-sat in qbfeval 2016. In Florian Lonsing and Martina Seidl, editors, *Proceedings of the*

4th International Workshop on Quantified Boolean Formulas (QBF 2016) co-located with 19th International Conference on Theory and Applications of Satisfiability Testing (SAT 2016), Bordeaux, France, July 4, 2016., volume 1719 of *CEUR Workshop Proceedings*, pages 63–70. CEUR-WS.org, 2016.

[5] Bart Bogaerts, Tomi Janhunen, and Shahab Tasharrofi. Solving QBF instances with nested SAT solvers. In Adnan Darwiche, editor, *Beyond NP, Papers from the 2016 AAAI Workshop, Phoenix, Arizona, USA, February 12, 2016.*, volume WS-16-05 of *AAAI Workshops*. AAAI Press, 2016.

[6] Martin W. A. Caminada, Walter Alexandre Carnielli, and Paul E. Dunne. Semi-stable semantics. *J. Log. Comput.*, 22(5):1207–1254, 2012.

[7] Tomi Janhunen, Shahab Tasharrofi, and Eugenia Ternovska. Sat-to-sat: Declarative extension of SAT solvers with new propagators. In Dale Schuurmans and Michael P. Wellman, editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 978–984. AAAI Press, 2016.

[8] Olga Ohrimenko, Peter J. Stuckey, and Michael Codish. Propagation via lazy clause generation. *Constraints*, 14(3):357–391, 2009.

[9] David Pearce. Equilibrium logic: An extension of answer set programming for nonmonotonic reasoning. In *WLP*, page 17, 2000.

[10] Shahab Tasharrofi and Tomi Janhunen. GG-STS Argumentation Solver. `https://research.ics.aalto.fi/software/sat/gg-sts/`, 2017. [Online; accessed 22-May-2017].

[11] Pashootan Vaezipoor, David G. Mitchell, and Maarten Mariën. Lifted unit propagation for effective grounding. *CoRR*, abs/1109.1317, 2011.

[12] Bart Verheij. Two approaches to dialectical argumentation: Admissible sets and argumentation stages. In *Proceedings of FAPR*, pages 357–368, 1996.