Computational Models of Argument
F. Toni et al. (Eds.)

doi:10.3233/FAIA220172

CPrAA – A Checker for Probabilistic Abstract Argumentation

Nikolai KÄFER [1]
Technische Universität Dresden, Germany

Keywords. probabilistic semantics, reasoning problems, constraint solving

1. Introduction

Classical *argumentation semantics* [1] determine which arguments of a given argumentation framework (AF) are considered to be jointly acceptable in light of the AF's attack relation. A collection of such compatible arguments is called an extension; thus semantics can be seen as functions that associate AFs with sets of extensions. Recently, abstract argumentation has been lifted into probabilistic settings to allow modelling of uncertainty, beliefs, and other quantitative aspects, giving rise to various *probabilistic argumentation semantics*. These include semantics working on the marginal probabilities of single arguments [2], notions to capture admissibility and complete semantics in the probabilistic setting [3], and direct liftings on the level of extensions of classical semantics [4].

For a given AF, a probabilistic semantics induces a family of probability distributions over the AF's extensions. Notably, while there is only a finite number of possible extensions for finite argumentation graphs, the space of distributions over extensions is infinite. As a result, reasoning problems for probabilistic semantics come with additional challenges of representing and enumerating solutions.

CPrAA[2] is a Python tool developed in the context of [3] that is capable of solving common reasoning problems in the probabilistic setting. Given an AF as input and selecting one or multiple probabilistic semantics from [2,3,4], the tool can

- compute distributions satisfying the semantics' constraints, or get an assertion that no such distribution exists,
- check credulous and skeptical acceptance of single arguments in the AF under a given threshold (see [3]), i.e., verify whether the marginal probability of the argument exceeds the threshold for at least one, or, respectively, all distributions,
- find a distribution under which the marginal probability of a given argument is maximal (or minimal),
- enumerate infinite solution spaces by finding the distributions at corners of solution polytopes,
- utilize a number of labelling schemes [5] to yield the finite set of all labellings corresponding to the (potentially infinitely many) distributions.

[1] E-mail: nikolai.kaefer@tu-dresden.de, ORCID: https://orcid.org/0000-0002-0645-1078
[2] https://perspicuous-computing.science/cpraa/

2. Tool Architecture

Based on the input AF, the selected semantics, and the chosen task, CPrAA generates a set of constraints on distributions for the AF. Subsequently, the constraints are passed to an appropriate backend solver, with two general classes of solvers available.

First, linear solvers (via CVXOPT [6]) are applicable for semantics where all induced constraints are linear, which is the case for the majority of semantics taken from the literature (see [3] for details). They allow convex optimization tasks like the minimization or maximization of marginal probabilities mentioned above. Linear constraints further imply that the solution space forms a convex polytope. By enumerating the distributions found at the corners of the polytope, one yields an explicit representation of the solution space: all distributions within this space arise as convex combination of the corner distributions.

Second, SMT solvers like Z3 [7] cover all probabilistic semantics characterized by polynomial constraints. This includes not only all semantics from [2,3,4], but also their respective complement semantics, that is, a semantics inducing exactly those distributions not induced by the original semantics. Further, they allow enforcement of additional context-specific constraints by using the standard SMT-LIB format [8], e.g., to specify the conditional probability for an argument to be accepted given that certain other arguments are not accepted. In addition to Z3, all SMT solvers available via the pySMT interface [9] can be used as backend.

The tool comes with a rich command line interface for direct interaction and can also be included as a Python library in other projects. Due to the constraint-based design, extending CPrAA with new probabilistic semantics for AFs is straightforward.

Acknowledgments. I'd like to thank Christel Baier, Martin Diller, Clemens Dubslaff, Sarah Gaggl, and Holger Hermanns for the collaboration that sparked the tool's development. This work was funded by DFG grant 389792660 as part of TRR 248 – CPEC.

References

[1] Dung PM. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games. Artificial Intelligence. 1995;77(2):321-58.

[2] Hunter A, Thimm M. Probabilistic Reasoning with Abstract Argumentation Frameworks. Journal of Artificial Intelligence Research. 2017;59:565-611.

[3] Käfer N, Baier C, Diller M, Dubslaff C, Gaggl SA, Hermanns H. Admissibility in Probabilistic Argumentation. Journal of Artificial Intelligence Research. 2022;74.

[4] Thimm M, Baroni P, Giacomin M, Vicig P. Probabilities on Extensions in Abstract Argumentation. In: Proceedings of TAFA 2017. vol. 10757 of LNCS. Springer; 2017. p. 102-19.

[5] Käfer N. A General Framework for Probabilistic Abstract Argumentation; 2020. Master's thesis. Saarland University.

[6] Andersen M, Dahl J, Vandenberghe L. CVXOPT – Python Software for Convex Optimization; 2014. https://cvxopt.org/.

[7] de Moura L, Bjørner N. Z3: An Efficient SMT Solver. In: Tools and Algorithms for the Construction and Analysis of Systems. vol. 4963 of Lecture Notes in Computer Science. Springer; 2008. p. 337-40.

[8] Barrett C, Fontaine P, Tinelli C. The Satisfiability Modulo Theories Library (SMT-LIB); 2016. http://smt-lib.org.

[9] Gario M, Micheli A. PySMT: a solver-agnostic library for fast prototyping of SMT-based algorithms; 2015. SMT Workshop 2015.