Fantastic Argumentation Tools And Where To Find Them

Michele Persiani*, Esteban Guerrero, Andreas Brännström, Kaan Kilic and Timotheus Kampik

Umeå University, Umeå, Sweden

Abstract

This paper presents a survey of software for computational argumentation, focusing on tools and libraries that can be re-used for building applications. The (so far preliminary) survey is intended as a continuously updated resource that researchers and potential practitioners can refer to when selecting tools for their argumentation use cases. The survey results indicate that while a plethora of argumentation tools exist, covering various formal argumentation approaches, the software engineering maturity of the tool ecosystem is low. For example, while many tools are openly available, few are properly documented and covered by automated tests. We argue that many of these shortcomings can be straightforwardly addressed by following simple engineering practices if community consensus regarding basic requirements for software artifacts can be established, as is the case in software engineering-oriented communities.

Keywords

Computational Argumentation, Systems for Formal Argumentation, Tools for Formal Argumentation

1. Introduction

The field of computational argumentation has evolved from a primarily theory-oriented domain to a vibrant research area that increasingly produces applied research results[1]. Still, one can arguably assess applications of computational argumentation as *nascent*, e.g., in contrast to defeasible reasoning with rules engines, or temporal reasoning, both of which are employed at scale in large-scale real-world software systems [1, 2]. In order to further advance applications of computational argumentation, an important prerequisite is a vibrant ecosystem of tools and libraries for computational argumentation. Without a mature ecosystem, practitioners would have to implement their argumentation-based software systems from scratch, which *i)* increases adoption hurdles and *ii)* would leave the impression that the time for serious adoption has not yet come. Over the past decade, the argumentation community has put substantial

SAFA'24: Fifth International Workshop on Systems and Algorithms for Formal Argumentation, September 17, 2024, Hagen, Germany

*Corresponding author.

michelep@cs.umu.se (M. Persiani); esteban@cs.umu.se (E. Guerrero); andreasb@cs.umu.se (A. Brännström); kaank@cs.umu.se (K. Kilic); tkampik@cs.umu.se (T. Kampik)

https://people.cs.umu.se/~tkampik/ (T. Kampik)

0000-0001-5993-3292 (M. Persiani); 0000-0002-6035-800X (E. Guerrero); 0000-0001-9379-4281 (A. Brännström); 0009-0005-4903-8610 (K. Kilic); 0000-0002-6458-2252 (T. Kampik)

[1] This is, for example, evidenced by the recently launched workshop on Argumentation and Applications: https://argapp-workshop.github.io/.

effort into the design, implementation, and analysis of software tools, e.g., in the context of the bi-annual workshop on Systems and Algorithms for Formal Argumentation (SAFA)[2] and the (also bi-annual) International Competition on Computational Models of Argumentation (ICCMA)[3].

However, an improper handling of research code that is somewhat common in the broader computer science community can lead to issues related to reproducibility, because as the code gets older (and stale), it becomes increasingly difficult to properly execute it and to reuse it in new projects. Methodologies to reduce this effect come from research in software engineering, whose community has already adopted principles from the Open Science movement [3], promoting transparency, openness, and reproducibility in scientific works. These principles are not only applied to the research and papers themselves, but also to the products of such papers, which in contemporary research practice often is code [4]. Indeed, in other communities, such as the Multi-Agent Systems (MAS) community, efforts to organize and assess tools through the lens of open science and software engineering principles are emerging [5, 6][4].

It becomes therefore important to view research on computational argumentation tools through the lens of open science principles, for the sake of maintaining a healthy ecosystem of tools that are reusable, reproducible, and correct; with this broader goal in mind, this paper provides an initial survey of tools from some of the venues of the argumentation community that explicitly encourage the presentation of such tools. As such, the objectives of this paper are two-fold:

1. To provide an overview of re-usable tools & libraries for formal argumentation and their properties to facilitate software selection and re-usability.
2. Critically assess the state-of-the-art, identify shortcomings, and propose action items that the community can potentially take up to increase the maturity of its software ecosystem.

This paper is structured as follows. Section 2 introduces the survey methodology, in particular, its scope (*Where do we search?*) and our criteria for inclusion (*What do we search for?*) and assessment (*What are the properties we are interested in?*). We report the results in Section 3. Finally, Section 4 provides a discussion of the status quo given the survey results, and outlines future work, both with respect to the survey and the formal argumentation tool ecosystem.

2. Survey Methodology

2.1. Survey Scope

Our survey focuses on *formal* argumentation *tools* such as re-usable reasoners and solvers, as well as tools for analysis, visualization and manipulation of argumentation graphs. While relevant for the broader scope of research in argumentation, tools supporting other types of argumentation, such as natural language argumentation or argument mining, are not considered. This means that i) the software library must be grounded in *formal* argumentation, i.e., we do not consider tools that provide "informal" argumentation capabilities that are not connected to formal abstractions. ii) the software implementation must be re-usable by developers who want

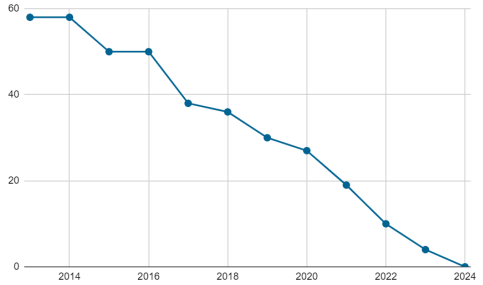
[2] http://safa2024.argumentationcompetition.org/index.html
[3] http://argumentationcompetition.org/
[4] The name of our paper is derived from the initiative *Fantastic MASs and Where to Find Them* [5].

to employ formal argumentation approaches in their applications, for example distributed as a library or independent application. We systematically survey a few selected venues in which we expected a high density of such tools among the overall publications, namely venues of the argumentation community that explicitly invite tool presentations and evaluations: i) The International Competition on Computational Models of Argumentation (ICCMA)[5]; ii) The demonstration track of the International Conference On Computational Models of Argument (COMMA)[6]; iii) The International Workshop on Systems and Algorithms for Formal Argumentation (SAFA)[7]. Let us highlight that the aim of the survey is to elicit an initial sample of argumentation tools from which a first, if only partial, overview of the state-of-the-art can be compiled. This overview can then serve as a starting point for a living document that provides a continuously updated overview of tools, as a resource for the argumentation community and in particular for the growing number of researchers that have an interest in applications of computational argumentation. We argue that for this purpose, surveying the above-mentioned selected venues is sufficient, as many relevant tools can be expected to have an associated publication in one of these venues. For example, even if a group of researchers publishes the main paper introducing a tool in a less specialized venue, they may showcase it at COMMA or evaluate it in the course of ICCMA.

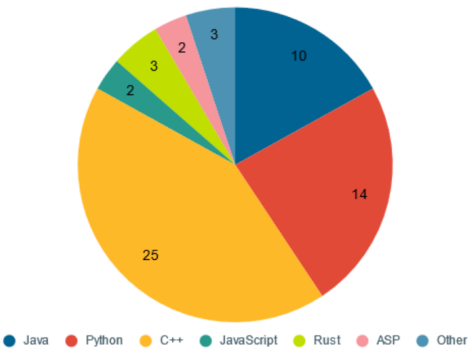
2.2. Evaluation Criteria

The objectives of our survey are to both provide an overview of the general properties of the tools, *i.e.* what tools exist and what/how they can be used; and to assess the software engineering maturity of the ecosystem. To cover the former objective, we document: i) the name of the tool; ii) venue were the tool was presented; iii) a brief description; iv) a list of formal argumentation approaches (such as: *abstract argumentation* or *structured argumentation*) that the tool covers; v) for solvers, which approach was used (e.g. ASP or ASPIC semantics); vi) the *type* of tool, such as *solver*, *visualizer*, or *utility*; vii) the URL of a website describing or hosting the tool or its source code; viii) the programming languages that the tool has been developed in or for; ix) the paper (reference) that describes the tool; x) whether the tool is part of a larger application. To cover the latter objective we assess whether: i) the tool is open-source; ii) the tool is available under an open-source license; iii) documentation for the tool is available; iv) tests are available for the tool; v) the tests are run on a continuous integration server; vi) it is possible to open issues and contribute code; vii) eventual dependencies from other software; viii) the last update of the tool, if this can be traced. We therefore gathered 17 features for each tool, 9 for general description and 8 concerning software engineering principles. In reporting solvers for the ICCMA, we acknowledge that dates for the "last update" of some tools' source code were obtained from the URL reported in the ICCMA tool description, however, some tools were moved to other code repositories, university, or personal Web pages. In this sense, we tried to track the last update of the main source code in the currently available repository.

[5] http://argumentationcompetition.org/
[6] https://comma.csc.liv.ac.uk/, limited to COMMA 2016 and later; due to earlier proceedings being pay-walled.
[7] https://safa2016.argumentationcompetition.org/, 1st edition.



(a) Number of tools that have been updated a given year or later.



(b) Distribution of programming languages

3. Results: Fantastic Argumentation Tools

3.1. General Findings

Our search identified 59 different software tools related to argumentation. Different versions of the same tool were removed and only the last version for each tool was kept. The overview of all tools is provided in Tables 1, 2 and 3, and additional information is available online at https://people.cs.umu.se/~tkampik/argtools/. In the tables only 12 features are shown for space and formatting reasons.

The vast majority were argumentation solvers (n=52), followed by general utility libraries (n=6) and visualizers of argumentation graphs (n=5), only one tool for benchmarking has been identified (n=1). Utilities are tools for specific tasks like annotating arguments, getting specific meanings from arguments, and tools used for comparing different argumentation systems. Some tools perform multiple functions, for example, reasoner suites usually embed together a solver and several accessory functions. Concerning programming languages, a plurality of the tools is implemented in C++ (n=25). Other common languages are Python (n=14) and Java (n=10), followed by a long tail of less common languages and polyglot or language-wise ambiguous implementations (Figure 1b).

3.2. Ecosystem Maturity

Our review shows that most of the source code of the tools, namely 47 (80%), is openly available. Moreover, we found that most of these argumentation tools are released without any type of code license (n=33), but still, many argumentation tools are released under an open source license, such as GPL or MIT (where the latter is more friendly towards re-use in proprietary, closed-source contexts). Six tools had other mixed types of licenses.

Only 16 tools (27%) have source documentation or a technical description of the tools. In general, *internal documentation* (*e.g.*, API, in-line or architectural documentation) of the software is considered a good practice for most software developers [7], however, our review found that

there is a lack of code documentation impacting the *reusability* and maintainability of the tools.

Automatic tests in software engineering are suites of pre-scripted test cases that are executed using software tools to validate the functionality and performance of software applications [8]. These tests enhance the reliability of the software by ensuring consistent test coverage and identifying defects early in the development lifecycle. We found in our review 10 available tests (*w.r.t.*, 49 not available) for the argumentation tools with released code. In this sense, automated testing for argumentation tools contributes to the efficiency of the development process by reducing manual effort and facilitating continuous integration and delivery practices. Moreover, we found that only one tool (ASPARTIX) has been set up for continuous integration. In the same context, *bug* or *issue tracking* refers to the systematic process of identifying, documenting, and managing errors or flaws within a software application to ensure they are addressed and resolved, usually performed by specific tools. We found that 3 tools (5%) have bug tracker available, suggesting an impact on the overall quality and reliability of most argumentation tools. An overview of the maintenance status of the tools is provided in Figure 1a: as far as we can see given publicly available information, only a few of the surveyed tools have been recently updated.

4. Discussion: Towards More Fantastic Argumentation Tools

The maturity of the computational argumentation ecosystem can be more qualitatively assessed by examining the development and availability of tools and libraries (*e.g.,* types of source code licenses), the community's efforts in standardizing and sharing resources, and the integration of argumentation systems into practical applications. Our review suggests at least three conclusions about the ecosystem maturity:

- **Development of tools and libraries**. As our review suggests, the argumentation software engineering ecosystem has gradually matured, with an acceptable corpus of argumentation solvers (n=52), and visualization tools (n=6), indicating a gradual shift from theoretical research to practical, usable systems.
- **Community efforts**. The argumentation community has been active in organizing conferences such as COMMA (with a demonstration track), workshops like SAFA, and competitions like ICCMA, which foster the development and comparison of argumentation systems. Moreover, as our results indicate, there is a trend to utilize permissive licenses (w.r.t. GPL and MIT) to solvers and tools, contributing back to the open-source community practices.
- **Integration into practical applications**. While well-established engineering practices in computational argumentation are still emerging compared to fields related to data science, the efforts to create a vibrant ecosystem are evident. The regular organization of SAFA and ICCMA demonstrates a commitment to advancing the field.

In conclusion, our survey results indicate that while a plethora of computational argumentation tools exist, most of them are either not reusable at all or are risky or difficult to re-use due to a lack of proper licensing, documentation, or tests. Generally speaking, the ecosystem of argumentation tools is diverse and still maturing; here, an exception is the regular performance evaluation of some tools as part of ICCMA. A stronger software engineering practice is needed when creating and maintaining tools that are as easily reusable as possible. Ideally, application

developers can leverage existing tools, by either importing them as libraries or accessing the application programming interface of an "externally" running tool. However, this becomes difficult if tools are too narrowly specialized, poorly documented, or not re-usable at all. As such, it is important that we start seeing tools as an independent product of research (in argumentation) rather than only a side-product of answering entirely different research questions. One relevant point to mention is that while the survey is preliminary in its scope and hence at least to some extent incomplete, given our focus on key argumentation venues, we assume that our conclusions based on the obtained results are broadly correct. Still, we intend to further extend the survey and to maintain the corresponding web page at https://people.cs.umu.se/~tkampik/argtools/ as a living document.

Some threats to validity exist that an extension of the presented work could address in the future. First, the survey covers different venues and in particular all argumentation competitions, in which the same or very similar solvers appear in slightly different versions. Here, we have not (yet) executed a careful assessment that leads to distinct counts. Yet more importantly, all information about the surveyed tools was obtained by reviewing the scientific literature and online resources pointed to by the relevant papers. The creators of the tools may possess additional and more precise information that we could not obtain or elicit. As such, while we expect the broader conclusions to maintain their validity through time, on a tool-level some properties may be incorrect; also, given the selective scope of the survey, additional argumentation tools that the survey does not cover are expected to exist. Having gathered these results, we hope this survey to be useful to researchers who are in the process of developing software for computational argumentation and are searching for some previously developed tool to either leverage or study, or in search for guidelines when developing new tools. Equally importantly, let us claim that the results provided in this survey highlight the need for further action when it comes to the continuous advancement of the argumentation tools ecosystem in terms of re-usability and, more broadly, software engineering maturity. Ideally, the initial survey effort presented in this paper can be understood as the start of a continuous process of assessing the status quo of the argumentation tooling ecosystem at a given point, as well as over time.

References

[1] S. Liang, P. Fodor, H. Wan, M. Kifer, Openrulebench: an analysis of the performance of rule engines, in: Proceedings of the 18th International Conference on World Wide Web, WWW '09, Association for Computing Machinery, New York, NY, USA, 2009, p. 601–610. URL: https://doi.org/10.1145/1526709.1526790. doi:10.1145/1526709.1526790.

[2] M. Bravo, N. Diegues, J. Zeng, P. Romano, L. E. Rodrigues, On the use of clocks to enforce consistency in the cloud., IEEE Data Eng. Bull. 38 (2015) 18–31.

[3] A. K. Das, Unesco recommendation on open science: An upcoming milestone in global science, Science Diplomacy (2021) 39.

[4] H. Thimbleby, Improving science that uses code, The Computer Journal (2023).

[5] D. Briola, A. Ferrando, V. Mascardi, Fantastic mass and where to find them: First results and lesson learned, in: A. Ciortea, M. Dastani, J. Luo (Eds.), Engineering

Multi-Agent Systems - 11th International Workshop, EMAS 2023, London, UK, May 29-30, 2023, Revised Selected Papers, volume 14378 of *Lecture Notes in Computer Science*, Springer, 2023, pp. 233–252. URL: https://doi.org/10.1007/978-3-031-48539-8_16. doi:10.1007/978-3-031-48539-8_16.

[6] A. Rago, O. Cocarascu, F. Toni, Argumentation-based recommendations: Fantastic explanations and how to find them, in: Proceedings of the 27th International Joint Conference on Artificial Intelligence, IJCAI'18, AAAI Press, 2018, p. 1949–1955.

[7] E. Aghajani, C. Nagy, M. Linares-Vásquez, L. Moreno, G. Bavota, M. Lanza, D. C. Shepherd, Software documentation: the practitioners' perspective, in: ICSE '20: Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering, Association for Computing Machinery, New York, NY, USA, 2020, pp. 590–601. doi:10.1145/3377811.3380405.

[8] S. Anand, E. K. Burke, T. Y. Chen, J. Clark, M. B. Cohen, W. Grieskamp, M. Harman, M. J. Harrold, P. McMinn, A. Bertolino, J. Jenny Li, H. Zhu, An orchestrated survey of methodologies for automated software test case generation, Journal of Systems and Software 86 (2013) 1978–2001. doi:10.1016/j.jss.2013.02.061.

[9] A. Borg, D. Odekerken, et al., Pyarg for solving and explaining rgumentation in python, Computational Models of Argument-Proceedings of COMMA 2022 353 (2022) 349–350.

[10] S. Ellmauthaler, S. A. Gaggl, D. Rusovac, J. P. Wallner, Adf-bdd: An adf solver based on binary decision diagrams., in: COMMA, 2022, pp. 355–356.

[11] C. Baier, M. Diller, C. Dubslaff, S. A. Gaggl, H. Hermanns, N. Käfer, Admissibility in Probabilistic Argumentation, in: Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, 2021, pp. 87–98. URL: https://doi.org/10.24963/kr.2021/9. doi:10.24963/kr.2021/9.

[12] J. Klein, M. Thimm, probo2: A benchmark framework for argumentation solvers., in: COMMA, 2022, pp. 363–364.

[13] M. Lenz, R. Bergmann, User-centric argument mining with arguemapper and arguebuf, in: Computational Models of Argument, IOS Press, 2022, pp. 367–368.

[14] N. Potyka, Attractor-a java library for gradual bipolar argumentation., in: COMMA, 2022, pp. 369–370.

[15] H. Prakken, et al., deliberate–online argumentation with collaborative filtering, Computational Models of Argument: Proceedings of COMMA 2020 326 (2020) 453.

[16] M. Caminada, S. Uebis, An implementation of argument-based discussion using aspic, in: Computational Models of Argument, IOS Press, 2020, pp. 455–456.

[17] R. Duthie, J. Lawrence, C. Reed, J. Visser, D. Zografistou, Navigating arguments and hypotheses at scale, in: 8th International Conference on Computational Models of Argument, COMMA 2020, IOS Press, 2020, pp. 459–460.

[18] S. Wells, The open argumentation platform (oapl), in: Computational Models of Argument, IOS Press, 2020, pp. 475–476.

[19] P. Baroni, S. Borsato, A. Rago, F. Toni, et al., The" games of argumentation" web platform., in: COMMA, 2018, pp. 447–448.

[20] R. Booth, M. Caminada, B. Marshall, Disco: A web-based implementation of discussion games for grounded and preferred semantics, in: Computational Models of Argument, IOS Press, 2018, pp. 453–454.

[21] D. Neugebauer, Dabasco: Generating af, adf, and aspic+ instances from real-world discussions, in: Computational Models of Argument, IOS Press, 2018, pp. 469–470.
[22] F. Cerutti, M. Giacomin, M. Vallati, et al., Generating structured argumentation frameworks: Afbenchgen2., in: COMMA, 2016, pp. 467–468.
[23] F. Brons, Labsat-solver: Utilizing caminada's labelling approach as a boolean satisfiability problem, System Descriptions of the First International Competition on Computational Models of Argumentation (ICCMA'15) (2015) 1.
[24] B. van Gijzel, Dungell: A reference implementation of dung's argumentation frameworks in haskell, Thimm and Villata (2015) (2015) 15–18.
[25] B. Liao, L. Jin, R. C. Koons, Dynamics of argumentation systems: A division-based method, Artificial Intelligence 175 (2011) 1790–1814.
[26] O. Rodrigues, Gris: Computing traditional argumentation semantics through numerical iterations., Thimm and Villata (2015) (2015) 37–40.
[27] K. Sprotte, Asgl: Argumentation semantics in gecode and lisp, System Descriptions of the First International Competition on Computational Models of Argumentation (ICCMA'15) (2015) 41–44.
[28] S. Groza, A. Groza, Prograph: towards enacting bipartite graphs for abstract argumentation frameworks, System Descriptions of the First International Competition on Computational Models of Argumentation (ICCMA'15) (2015) 29–32.
[29] T. F. Gordon, Carneades iccma: A straightforward implementation of a solver for abstract argumentation in the go programming, System Descriptions of the First International Competition on Computational Models of Argumentation (ICCMA'15) (2015) 54.
[30] W. Faber, M. Vallati, F. Cerutti, M. Giacomin, Solving set optimization problems by cardinality optimization with an application to argumentation (2016).
[31] E. Hadjisoteriou, M. Georgiou, Assa: Computing stable extensions with matrices, Thimm and Villata (2015) (2015) 62–65.
[32] F. Pu, G. Luo, Z. Jiang, Encoding argumentation semantics by boolean algebra, IEICE TRANSACTIONS on Information and Systems 100 (2017) 838–848.
[33] W. Faber, M. Vallati, F. Cerutti, M. Giacomin, et al., Enumerating preferred extensions using asp domain heuristics: The asprmin solver., in: COMMA, 2018, pp. 459–460.
[34] W. Dvorák, M. Järvisalo, J. P. Wallner, S. Woltran, Cegartix: A sat-based argumentation system, in: Pragmatics of SAT Workshop (POS), 2012.
[35] M. Vallati, F. Cerutti, M. Giacomin, Chimærarg: Configuring static portfolios for solving argumentation problems (2017).
[36] F. Cerutti, M. Giacomin, M. Vallati, et al., Argsemsat: Solving argumentation problems using sat., COMMA 14 (2014) 455–456.
[37] S. Nofal, K. Atkinson, P. E. Dunne, Looking-ahead in backtracking algorithms for abstract argumentation, International Journal of Approximate Reasoning 78 (2016) 265–282.
[38] S. Tasharrofi, T. Janhunen, Gg-sts: Argumentation solver from aalto university in iccma 2017, 2017.
[39] S. Ellmauthaler, H. Strass, The diamond system for computing with abstract dialectical frameworks., in: Comma, 2014, pp. 233–240.
[40] N. Geilen, M. Thimm, Heureka: a general heuristic backtracking solver for abstract argumentation, in: Theory and Applications of Formal Argumentation: 4th International

Workshop, TAFA 2017, Melbourne, VIC, Australia, August 19-20, 2017, Revised Selected Papers 4, Springer, 2018, pp. 143–149.
[41] M. Thimm, Dredd-a heuristics-guided backtracking solver with information propagation for abstract argumentation, The Third International Competition on Computational Models of Argumentation (2019).
[42] J.-M. Lagniez, E. Lonca, J.-G. Mailly, Coquiaas v3. 0 iccma 2019 solver description, System descriptions of the Third International Competition on Computational Models of Argumentation (ICCMA'19) (2019).
[43] L. Malmqvist, Yonas: An experimental neural argumentation solver, International Competition on Computational Models of Argumentation (ICCMA) 295 (2019).
[44] A. Previti, M. arvisalo, Argpref: A sat-with-preferences approach to ideal semantics, The 3rd International Competition on Computational Models of Argumentation (ICCMA'19) (2019).
[45] S. Bistarelli, L. Kotthoff, F. Santini, C. Taticchi, et al., A first overview of iccma'19., in: AI^3@ AI* IA, 2020, pp. 90–102.
[46] W. Dvořák, M. König, A. Rapberger, J. P. Wallner, S. Woltran, Aspartix-v-a solver for argumentation tasks using asp, in: 14th Workshop on Answer Set Programming and Other Computing Paradigms, 2021, pp. 1–12.
[47] J.-M. Lagniez, E. Lonca, J.-G. Mailly, J. Rossit, Design and results of iccma 2021, arXiv preprint arXiv:2109.08884 (2021).
[48] M. Heinrich, The matrixx solver for argumentation frameworks, arXiv preprint arXiv:2109.14732 (2021).
[49] M. Alviano, The pyglaf argumentation reasoner (iccma2021), arXiv preprint arXiv:2109.03162 (2021).
[50] J.-M. Lagniez, E. Lonca, J.-G. Mailly, J. Rossit, Crusti g2io, a benchmark generator based on a inner/outer template, ICCMA 2023 (2023) 35.
[51] M. Thimm, F. Cerutti, M. Vallati, Fudge v3. 2.8, ICCMA 2023 171 (2007) 25.
[52] A. Niskanen, M. Järvisalo, μ-toksia in iccma 2023, ICCMA 2023 (2023) 31.
[53] S. Declercq, Q. J. Capellini, C. Yang, J. Delobelle, J.-G. Mailly, Portsat: A portfolio of sat solvers for abstract argumentation, ICCMA 2023 (2023) 32.
[54] J. Delobelle, J.-G. Mailly, J. Rossit, Revisiting approximate reasoning based on grounded semantics, in: European Conference on Symbolic and Quantitative Approaches with Uncertainty, Springer, 2023, pp. 71–83.
[55] J. Delobelle, J.-G. Mailly, J. Rossit, Aripoter-degrees: Argumentation approximate reasoning using in/out degrees of arguments, ICCMA 2023 (2023) 10.
[56] M. Thimm, Fargo-limited v1. 1.1, ICCMA 2023 (2023) 21.
[57] M. Thimm, Harper++: Using grounded semantics for approximate reasoning in abstract argumentation, The Fourth International Competition on Computational Models of Argumentation (ICCMA'21) (2021).
[58] P. Besnard, S. Doutre, D. Longin, et al., Sesame-a system for specifying semantics in abstract argumentation, in: 1st International Workshop on Systems and Algorithms for Formal Argumentation (SAFA 2016) co-located with the 6th International Conference on Computational Models of Argument: COMMA 2016, volume 1672, 2016, pp. 40–51.
[59] W. Dvorák, A. Greßler, S. Woltran, Evaluating setafs via answer-set programming, in:

SAFA@ COMMA, 2018, pp. 10–21.
[60] M. Thimm, T. Rienstra, Approximate reasoning with aspic+ by argument sampling., SAFA@ COMMA 2672 (2020) 22–33.
[61] T. Kampik, D. Gabbay, Towards diarg: An argumentation-based dialogue reasoning engine, in: SAFA 2020: The Third International Workshop on Systems and Algorithms for Formal Argumentation, Online, September 8, 2020, Technical University of Aachen, 2020, pp. 14–21.
[62] R. Baumann, M. Heinrich, A python script for abstract dialectical frameworks., in: SAFA@ COMMA, 2020, pp. 74–79.
[63] M. Diller, S. A. Gaggl, P. Gorczyca, Strategies in flexible dispute derivations for assumption-based argumentation., in: SAFA@ COMMA, 2022, pp. 59–72.

Table 1
Computational argumentation tools, their basic properties, and their software engineering maturity. Here and henceforth, the following legend applies. OS: Open Source; D: Documented; T: Tested; CI: Continuous Integration; IT: Issue Tracker; EE: Empirical Evaluation; LU: Last Update; ✓denotes existence, whereas X denotes absence of availability.

Name	Type	Language(s)	Reference	OS	Licensed	D	T	CI	IT	EE	LU
PyArg	Solver	Python	[9]	X	X	X	X	X	X	X	X
ADF-BDD	Solver	Rust	[10]	✓	✓	✓	X	X	X	✓	2024
CPrAA	Solver	Python	[11]	✓	X	✓	X	X	X	X	2022
probo2	Benchmark	Python	[12]	✓	✓	✓	X	X	X	X	2024
ArgueMapper and Arguebuf	Annotation	Typescript, Python	[13]	✓	✓	✓	X	X	X	X	2024
Attractor	Solver	Java	[14]	✓	X	X	X	X	X	X	2022
deliberate	Visualization	X	[15]	X	X	X	X	X	X	X	X
ABDA	Solver	Python	[16]	✓	X	✓	X	X	X	X	2021
ArgNav	Visualization	Python	[17]	✓	✓	X	X	X	X	X	2021
OAPL	Utility, Visualization, Solver	Python	[18]	✓	✓	✓	X	X	X	X	2024
Game of Argumentation	Solver, Visualization	NA	[19]	X	X	X	X	X	X	X	X
DISCO	Solver, Visualization	JavaScript	[20]	✓	✓	✓	✓	X	X	X	2017
DABASCO	Utility	Python	[21]	✓	✓	✓	X	X	X	X	2021
AFBenchGen2	Utility	JavaScript	[22]	✓	X	X	X	X	X	X	2014
LabSAT-Solver	Solver	Java	[23]	✓	✓	X	X	X	X	✓	2022
Dungell	Solver	Haskell	[24]	✓	✓	X	X	X	X	✓	2015
ZJU-ARG	Solver	Java	[25]	X	X	X	X	X	X	✓	X

Table 2
Continued: computational argumentation tools, their basic properties, and their software engineering maturity.

Name	Type	Language(s)	Reference	OS	Licensed	D	T	CI	IT	EE	LU
GRIS	Solver	C++	[26]	X	✓	✓	X	X	X	✓	X
ASGL	Solver	LISP, CLOS, C++	[27]	X	X	X	✓	X	X	✓	2019
ProGraph	Solver	Prolog	[28]	✓	X	X	X	X	X	✓	2015
Carneades	Solver	Go	[29]	✓	X	X	X	X	X	✓	2001
prefMaxSAT	Solver	C++	[30]	✓	✓	X	X	X	X	✓	2015
ASSA	Solver	Java	[31]	X	X	X	X	X	X	✓	X
argmat-clpb	Solver	C++, Prolog	[32]	✓	✓	X	X	X	X	✓	2017
argmat-dvisat	Solver	C++	[32]	X	X	X	X	X	X	✓	X
argmat-mpg	Solver	C++	[32]	X	X	X	X	X	X	✓	X
argmat-sat	Solver	C++	[32]	X	X	X	X	X	X	✓	X
ASPrMin	Solver	C++	[33]	✓	✓	X	X	X	X	✓	2017
cegartix	Solver	C++	[34]	✓	✓	✓	X	X	X	✓	2017
Chimærarg	Solver	C++	[35]	✓	✓	X	X	X	X	✓	2017
ArgSemSAT	Solver	C++	[36]	✓	✓	X	✓	X	X	✓	2020
ArgTools	Solver	C++	[37]	✓	✓	X	X	X	X	✓	2019
gg-sts	Solver	C++	[38]	✓	X	X	X	X	X	✓	2017
goDIAMOND	Solver	GO	[39]	✓	X	✓	✓	X	✓	✓	2015
heureka	Solver	C++	[40]	✓	✓	X	X	X	X	✓	2017
DREDD	Solver	C++	[41]	✓	✓	X	X	X	X	✓	2022
CoQuiAAS v3.0	Solver	C++	[42]	✓	X	X	X	X	X	✓	2019
Yonas	Solver	Python	[43]	✓	X	X	X	X	X	✓	2019
Argpref	Solver	Python	[44]	✓	X	X	X	X	X	✓	2019

Table 3
Continued: computational argumentation tools, their basic properties, and their software engineering maturity.

Name	Type	Language(s)	Reference	OS	Licensed	D	T	CI	IT	EE	LU
Mace4/Prover9	Solver	C++	[45]	✓	X	X	X	X	X	✓	2019
ASPARTIX-V21	Solver	ASP	[46]	✓	X	X	X	X	X	✓	2021
ConArg	Solver	C++	[47]	X	X	X	X	X	X	✓	2021
MatrixX	Solver	Python	[48]	✓	X	X	X	X	X	✓	2021
pyglaf '21	Solver	Python	[49]	✓	X	X	X	X	X	✓	2021
Crustabri	Solver	Rust, C++	[50]	✓	✓	X	X	X	X	✓	2022
FUDGE '23	Solver	C++	[51]	✓	✓	X	X	X	X	✓	2023
μ-toksia '23	Solver	C++	[52]	✓	X	X	X	X	X	✓	2023
PORTSAT	Solver	Rust, C++	[53]	✓	X	X	X	X	X	✓	2023
AFGCNv2	Solver	Python	[54]	✓	X	X	X	X	X	✓	2023
ARIPOTER-Degrees	Solver	Java	[55]	✓	X	X	X	X	X	✓	2023
ARIPOTER-HCAT	Solver	Java	[55]	✓	X	X	X	X	X	✓	2023
fargo-limited	Solver	C++	[56]	✓	✓	X	X	X	X	✓	2022
HARPER++	Solver	C++	[57]	✓	✓	X	X	X	X	✓	2022
SESAME	Specifying semantics	Java	[58]	X	X	✓	X	X	X	X	2017
SETAFs via ASP	Solver	ASP	[59]	✓	X	✓	X	X	X	X	2018
AR with ASPIC+	Solver	Java	[60]	X	X	X	X	X	X	✓	X
DiArg	Dialogue Reasoner	Java	[61]	✓	✓	✓	✓	X	✓	X	2024
Python for ADFs	Solver	Python	[62]	✓	✓	X	✓	X	X	✓	2020
flexABle	Solver	Prolog	[63]	✓	X	✓	✓	X	X	✓	2023