# Accessible Algorithms for Applied Argumentation

Daphne Odekerken[1,2], AnneMarie Borg[1] and Matti Berthold[3]

[1]*Department of Information and Computing Sciences, Utrecht University, The Netherlands*
[2]*National Police Lab AI, Netherlands Police, The Netherlands*
[3]*ScaDS.AI Dresden/Leipzig, Universität Leipzig, Germany*

Abstract
Computational argumentation is a promising research area, yet there is a gap between theoretical contributions and practical applications. Bridging this gap could potentially raise interest in this topic even more. We argue that one part of the bridge could be an open-source package of implementations of argumentation algorithms, visualised in a web interface. Therefore we present a new release of PyArg, providing various new argumentation-based functionalities – including multiple generators, a learning environment, implementations of theoretical papers and a showcase of a practical application – in a new interface with improved accessibility.

**Keywords**
argumentation, implementation, visualisation, education

## 1. Introduction

Computational argumentation is a promising interdisciplinary research area, with applications in, e.g. the legal, medical and e-government domain [1]. Thanks to its natural connection with human cognition, its flexibility and its dialectic nature, argumentation seems to be particularly suitable as a logical foundation for human-centered artificial intelligence (AI) [2].

The construction of argumentation-based AI systems that are applicable to real-world use cases requires not only theoretical developments, but also effective solving of problems related to argumentation. Although there is a significant amount of work on theoretical aspects of computational argumentation, which include various argumentation formalisms, semantics and their properties, as well as a growing body of research on solvers that are sufficiently efficient to be applied in practice[1], we argue that the connection between these two areas could (and should) be strengthened. This is based on our observation that real-life applications use a different formalism than those mainly studied in the community [3] or require efficient algorithms for yet unexplored problems [4].

In addition, we hypothesise that the application of argumentation-based AI is more cumbersome for non-experts than, for example, the application of machine-learning based AI, which is

*Arg&App 2023: International Workshop on Argumentation and Applications, September 2023, Rhodes, Greece*
d.odekerken@uu.nl (D. Odekerken); a.borg@uu.nl (A. Borg); berthold@informatik.uni-leipzig.de (M. Berthold)
https://webspace.science.uu.nl/~3827887/ (D. Odekerken); https://annemarieborg.nl/ (A. Borg);
https://www.informatik.uni-leipzig.de/~berthold/ (M. Berthold)
0000-0003-0285-0706 (D. Odekerken); 0000-0002-7204-6046 (A. Borg); 0009-0006-9231-5115 (M. Berthold)

[1]In particular by submissions to the International Competition on Computational Models of Argumentation (ICCMA): http://argumentationcompetition.org/

more convenient thanks to the availability of numerous software packages (such as Scikit-learn[2] and Tensorflow[3]) as well as user-friendly interfaces to try this AI.[4] This makes argumentation less accessible to students, software developers and companies searching for AI-based solutions to domain-specific problems.

In order to improve both the connection within theoretical and practical work within the computational argumentation community and the connection with stakeholders outside the community, it would be helpful to have open-source software, paired with an accessible web interface. In this paper, we therefore present a new release of PyArg.

PyArg is an open-source software implementation in Python that not only provides practical algorithms for theoretical problems in various argumentation formalisms, but also makes (potential) applications of these algorithms visible in a user interface that is accessible from the internet without installation. PyArg is intended to be a software solution for researchers within the argumentation community, students who may become part of it, as well as stakeholders outside the community, thanks to the following features:

1. Open-source implementations of argumentation algorithms on GitHub can be validated and extended by community members (https://github.com/DaphneOdekerken/PyArg);
2. The Python package can be installed using `pip install python-argumentation` and is therefore directly usable for software developers;
3. The web interface on https://pyarg.npai.science.uu.nl/ makes argumentation more accessible to those who wish to learn more about argumentation and serves as a platform for showcasing applications of argumentation to stakeholders outside the community.

The release presented here is a major update compared to the preliminary version earlier presented in [5]. In the following sections, we describe PyArg's new functionalities.

## 2. Support for more formalisms

Research on computational argumentation ranges over an ever-growing number of formalisms and extensions of formalisms. Whereas the initial version of PyArg [5] only supported abstract argumentation frameworks and ASPIC$^+$, in this iteration we added support for assumption-based argumentation (ABA) [6, 7]. In particular, PyArg can now (1) compute the extensions of a given ABA framework under the prominent semantics and visualise them in the instantiated abstract argumentation framework; and (2) verify if a specific set of extensions can be realised under a given semantics (see Section 3.2).

## 3. Algorithms for argumentation problems

The new PyArg version still contains all algorithms presented in [5], including various implementations of formalisms and algorithms in both abstract argumentation [8] and ASPIC$^+$ [9]. It provides algorithms for evaluating argumentation settings in different semantics [10], as well

[2]https://scikit-learn.org/
[3]https://www.tensorflow.org/
[4]e.g. https://freegpt.cc/

as explaining the (non-)acceptance of arguments and formulas in various ways [11, 12]. In the new version, we also provide algorithms for dynamic argumentation problems as well as for the construction of canonical representations.

### 3.1. Dynamic argumentation problems

Many problems in argumentation assume that all information required to decide upon, for example, the acceptance of an argument is present. This is however not always a realistic assumption in applied settings. Therefore, recently the problems of stability [4] and relevance [13] have been introduced for various formalisms, including ASPIC$^+$. Informally, the stability status of a "topic" argument or formula represents the impossibility that its acceptability status may change in view of additional, yet uncertain information. For topics that are not stable, it is interesting to study relevance, where only information that can change the stability status of the topic is relevant. PyArg provides an implementation of the approximation algorithm for stability from [4], as well as an inexact but efficient algorithm for estimating relevance based on the labels from the aforementioned stability algorithm.

### 3.2. Canonical argumentation frameworks

The work by Dunne et al. [14] studies the *realisability* in abstract argumentation: given a semantics $\sigma$ and a set of extensions $\mathbb{S}$, is there an argumentation framework $F$ realising $\mathbb{S}$ under $\sigma$, i.e., such that $\sigma(F) = \mathbb{S}$ – and which characteristics determine whether such an argumentation framework exists? Similarly, realisability can be characterised for ABAFs [15]. PyArg now provides the algorithms to determine whether a given set extensions satisfies these characteristic properties and generates "canonical" argumentation frameworks or ABA frameworks, when they exist; see Figure 1.

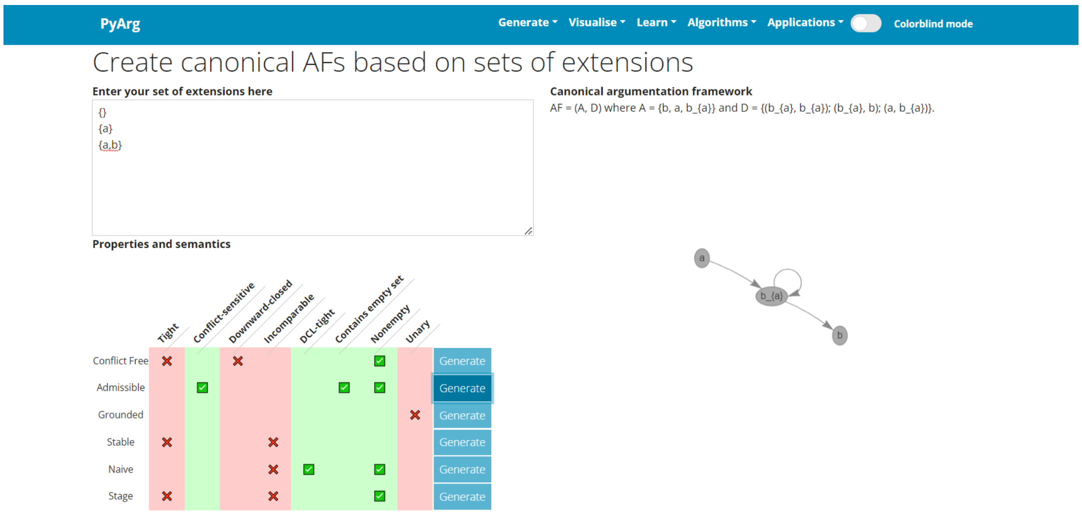
## 4. Practical features

### 4.1. Generators

There are various situations in which it is useful to randomly generate an argumentation setting. One example would be for testing new algorithms: a large part of the data sets used in the ICCMA competition to assess runtime of algorithms is based on randomly generated argumentation frameworks. A second example is related to education: in order to assess a student's understanding of, e.g., specific argumentation semantics, it is convenient to have a generator for automatically creating new exercises.

In order to address this demand, PyArg provides several generators. For generating ASPIC$^+$ argumentation systems, PyArg uses the "layered" generator from [4, Section 4.2.5]. In addition, PyArg provides a basic random generator for abstract argumentation frameworks.

The generators can be found in the source code; in addition, the web interface provides functionality for generating a single argumentation framework or system, parameterised by the values given in input fields. The resulting argumentation setting can then be downloaded for further use within or outside PyArg.



Create canonical AFs based on sets of extensions

**Enter your set of extensions here**

{}
{a}
{a,b}

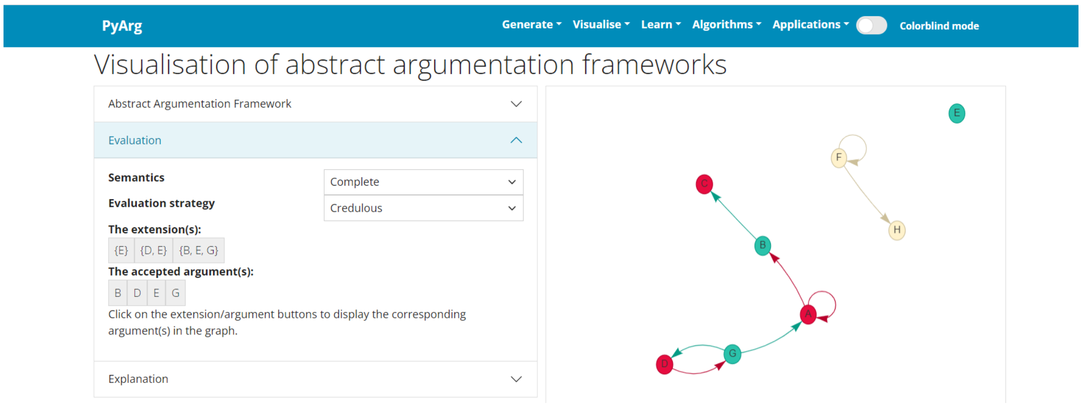
**Canonical argumentation framework**

AF = (A, D) where A = {b, a, b_{a}} and D = {(b_{a}, b_{a}); (b_{a}, b); (a, b_{a})}.

**Properties and semantics**

| | Tight | Conflict-sensitive | Downward-closed | Incomparable | DCL-tight | Contains empty set | Nonempty | Unary | |
|---|---|---|---|---|---|---|---|---|---|
| Conflict Free | ✗ | | ✗ | | | | ✓ | | Generate |
| Admissible | | ✓ | | | | ✓ | ✓ | | Generate |
| Grounded | | | | | | | | ✗ | Generate |
| Stable | ✗ | | | ✗ | | | | | Generate |
| Naive | | | | ✗ | ✓ | | ✓ | | Generate |
| Stage | ✗ | | | ✗ | | | ✓ | | Generate |

**Figure 1:** Given an extension set provided by the user, PyArg computes certain properties and generates a canonical argumentation framework when possible.



Visualisation of abstract argumentation frameworks

Abstract Argumentation Framework

Evaluation

**Semantics** Complete

**Evaluation strategy** Credulous

**The extension(s):**

{E} {D, E} {B, E, G}

**The accepted argument(s):**

B D E G

Click on the extension/argument buttons to display the corresponding argument(s) in the graph.

Explanation

**Figure 2:** Improved visualisation of the abstract argumentation environment.

## 4.2. Improved visualisation

Compared to the previous version, PyArg's user interface has been made more user-friendly and accessible. A screenshot of the new visualisation for abstract argumentation frameworks is shown in Figure 2. PyArg now features both a *regular* mode (in which accepted arguments are coloured green, while other arguments are yellow or red) and a *colourblind-friendly* mode that uses an adapted colour palette.

### 4.3. Importers and exporters

PyArg provides various importers and exporters to convert argumentation settings to various formats. For abstract argumentation frameworks, it is possible to read from and write to all formats used in recent ICCMA competitions, that is: the ASPARTIX format (.APX), the Trivial Graph Format (.TGF) and the input format used for ICCMA 2023 (.ICCMA23). In addition, for both abstract argumentation frameworks and ASPIC$^+$ argumentation systems, there are importers and exporters to and from JSON.

## 5. Applications of algorithms in the web interface

In order to demonstrate how PyArg's algorithms can be applied in various settings, we provide two use cases in the web interface: a learning environment and a chat interface.

### 5.1. Learning environment

The learning environment is intended for anyone interested in learning computational argumentation. In this functionality in the web interface, a learner can choose between various exercises. The current PyArg version features three exercises: identifying grounded, complete and preferred extensions in abstract argumentation. As the learner starts an exercise, PyArg generates a random abstract argumentation framework using its generators. The learner then gives the extensions, and PyArg uses its semantics algorithms for validating the learner's solutions.

### 5.2. Chat interface

The chat interface showcases an application for the algorithms for stability and relevance in inquiy dialogue. First, the user chooses an ASPIC$^+$ argumentation system (which can be randomly generated, hand-made or the predefined fraud example), a set of queryables (e.g. formulas that can be asked in a dialogue), a topic formula for the chat and an initial knowledge base. PyArg then uses the stability algorithm to find out if it makes sense to ask for more information - if so, it uses the relevance algorithm for identifying relevant questions.

## 6. Related work

For an extended overview of software related to computational argumentation, we refer to [16]. In this section, we relate to implementations that are most similar to PyArg. Tweety [17] is a comprehensive collection of Java libraries that includes algorithms for both abstract and structured approaches to argumentation. It is in fact more comprehensive than PyArg, but does not have a visualisation option. The Online Argument Structures Tool (TOAST) [18] does provide a visualisation for ASPIC$^+$, but the source code is not openly available. Gorgias Cloud [19] is a recent system that is similar to PyArg, but is based on the Gorgias argumentation system. NEXAS [20] is an alternative approach to visualising extensions of abstract argumentation frameworks, which, compared to PyArg, is more aimed towards (large) frameworks with many extensions. Finally, many algorithms for argumentation-related problems have been submitted to the ICCMA competition. However, these are mostly focused on fast implementation of limited problems, mainly in the context of abstract argumentation.

## 7. Conclusion and future work

PyArg combines algorithms for argumentation problems with a web interface, aiming to improve the connections between theoretical and practical work on argumentation on the one hand, and inside and outside the community on the other hand.

The contributions of this version of PyArg are mainly focused on the front-end. In the back-end, the algorithms for computing the semantics are not state-of-the-art. This becomes noticeable if the visualisation is tested on hard (large) instances, as no output will appear on the screen within a reasonable amount of time. In a next version of PyArg, we plan to improve this performance aspect by calling more efficient solvers in the back-end. In addition, we aim to add support more formalisms, such as abstract dialectical frameworks [21], abstract frameworks with collective attacks [22] and abstract frameworks with support for claims [23], and to implement their intertranslations [24].

On a final note, we hope that the functionalities presented in this paper are just the beginning, as are the plans for future work mentioned above. We are open to any suggestions for additional functionalities, algorithms or other feedback, which we hope to incorporate in future releases of PyArg. Hopefully, this leads to an increase of interest and understanding of computational argumentation, both within and outside the community, eventually resulting in more responsible applications of artificial intelligence.

## Acknowledgments

The authors have not been sorted alphabetically. They acknowledge the financial support by the Federal Ministry of Education and Research of Germany and by the Sächsische Staatsministerium für Wissenschaft Kultur und Tourismus in the program Center of Excellence for AI-research "Center for Scalable Data Analytics and Artificial Intelligence Dresden/Leipzig", project identification number: ScaDS.AI.

## References

[1] K. Atkinson, P. Baroni, M. Giacomin, A. Hunter, H. Prakken, C. Reed, G. Simari, M. Thimm, S. Villata, Towards artificial argumentation, AI magazine 38 (2017) 25–36.

[2] E. Dietz, A. Kakas, L. Michael, Argumentation: A calculus for human-centric AI, Frontiers in Artificial Intelligence 5 (2022).

[3] A. C. Kakas, P. Moraitis, N. I. Spanoudakis, Gorgias: Applying argumentation, Argument & Computation 10 (2019) 55–81.

[4] D. Odekerken, F. Bex, A. Borg, B. Testerink, Approximating stability for applied argument-based inquiry, Intelligent Systems with Applications 16 (2022) 200110.

[5] A. Borg, D. Odekerken, PyArg for solving and explaining argumentation in Python: Demonstration, in: Proceedings of (COMMA-22), 2022, pp. 349–350.

[6] A. Bondarenko, F. Toni, R. A. Kowalski, An assumption-based framework for non-monotonic reasoning, in: Proceedings of (LPNMR-93), 1993, pp. 171–189.
[7] K. Čyras, X. Fan, C. Schulz, F. Toni, Assumption-based argumentation: Disputes, explanations, preferences, in: Handbook of Formal Argumentation, volume 1, 2018, pp. 365–408.
[8] P. M. Dung, On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games, Artificial Intelligence 77 (1995) 321–357.
[9] H. Prakken, An abstract framework for argumentation with structured arguments, Argument & Computation 1 (2010) 93–124.
[10] P. Baroni, M. Caminada, M. Giacomin, An introduction to argumentation semantics, The Knowledge Engineering Review 26 (2011) 365–410.
[11] A. Borg, F. Bex, A basic framework for explanations in argumentation, IEEE Intelligent Systems 36 (2021) 25–35.
[12] A. Borg, F. Bex, Necessary and sufficient explanations for argumentation-based conclusions, in: Proceedings of (ECSQARU-21), Springer, 2021, pp. 45–58.
[13] D. Odekerken, T. Lehtonen, A. Borg, J. P. Wallner, M. Järvisalo, Argumentative reasoning in ASPIC+ under incomplete information, in: Proceedings of (KR-23), 2023, pp. 531–541.
[14] P. E. Dunne, W. Dvořák, T. Linsbichler, S. Woltran, Characteristics of multiple viewpoints in abstract argumentation, Artificial Intelligence 228 (2015) 153–178.
[15] M. Berthold, A. Rapberger, M. Ulbricht, On the expressive power of assumption-based argumentation, in: Proceedings of (JELIA-23), 2023.
[16] F. Cerutti, S. A. Gaggl, M. Thimm, J. Wallner, Foundations of implementations for formal argumentation, IfCoLog Journal of Logics and their Applications 4 (2017) 2623–2705.
[17] M. Thimm, The formal argumentation libraries of Tweety, in: Proceedings of (TAFA-17), Springer, 2018, pp. 137–142.
[18] M. Snaith, C. Reed, TOAST: Online ASPIC+ implementation, in: Proceedings of (COMMA-12), IOS Press, 2012, pp. 509–510.
[19] N. I. Spanoudakis, G. Gligoris, A. Koumi, A. C. Kakas, Explainable argumentation as a service, Journal of Web Semantics (2023) 100772.
[20] R. Dachselt, S. Gaggl, M. Krötzsch, J. Méndez, D. Rusovac, M. Yang, Nexas: A visual tool for navigating and exploring argumentation solution spaces, in: Proceedings of (COMMA-22), volume 353, IOS Press, 2022, pp. 116–127.
[21] G. Brewka, H. Strass, S. Ellmauthaler, J. P. Wallner, S. Woltran, Abstract dialectical frameworks revisited, in: Proceedings of (IJCAI-13), 2013, pp. 803–809.
[22] S. H. Nielsen, S. Parsons, A generalization of dung's abstract framework for argumentation: Arguing with sets of attacking arguments, in: Proceedings of (ArgMAS-2006), Springer, 2006, pp. 54–73. doi:10.1007/978-3-540-75526-5_4.
[23] W. Dvorák, A. Rapberger, S. Woltran, Argumentation semantics under a claim-centric view: Properties, expressiveness and relation to setafs, in: Proceedings of (KR-20), 2020, pp. 341–350. doi:10.24963/kr.2020/35.
[24] M. König, A. Rapberger, M. Ulbricht, Just a matter of perspective, in: Proceedings of (COMMA-22), 2022, pp. 212–223. doi:10.3233/FAIA220154.