Demonstrating PyArg 2.0

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

We demonstrate the latest release of PyArg, an open-source Python package of implementation algorithms with a web interface. PyArg provides various argumentation-based functionalities, including evaluation and visualisation of abstract argumentation frameworks, ASPIC$^+$ argumentation theories and assumption-based argumentation frameworks; explanation algorithms; multiple generators; a learning environment; implementations of theoretical papers and a showcase of a practical application.

Keywords

argumentation, implementation, visualisation, education

1. Introduction

PyArg is an open-source software implementation in Python that provides practical algorithms for theoretical problems in various argumentation formalisms and makes (potential) applications of these algorithms visible in a web interface. 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. Depending on the users' goals, they can (A) validate and extend the open-source implementations of argumentation algorithms on GitHub; (B) install the Python package in one line and use it as a dependency in other Python projects; and/or (C) explore PyArg's functionalities in the web interface. PyArg was first presented in [1]. A new version, with multiple new functionalities, was proposed in [2]. In this paper, we give a brief overview of PyArg's functionalities from the front-end and from the back-end.

2. Related work

For an extended overview of software related to computational argumentation, we refer to [3]. The implementations that are most similar to PyArg are Tweety [4], the Online Argument Structures Tool (TOAST) [5], Gorgias Cloud [6] and NEXAS [7]. Finally, many algorithms for a limited set of argumentation-related problems have been submitted to the ICCMA competition[1].

7th Workshop on Advances in Argumentation in Artificial Intelligence
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]http://argumentationcompetition.org/

3. Back-end functionalities

In this section, we describe the functionalities that can be used from the back-end, by cloning the Python code from GitHub[2] or installing it from PyPI[3] using `pip install python-argumentation`. Specific usage examples can be found on the documentation website.[4]

The Python package PyArg currently supports abstract argumentation [8], ASPIC$^+$ [9] and assumption-based argumentation (ABA) [10, 11]: it provides algorithms for *evaluating* argumentation settings in different semantics [12]. In addition, it has functionality for *explaining* the (non-)acceptance of arguments and formulas in abstract argumentation frameworks (AFs) and ASPIC$^+$ argumentation theories [13, 14]. Furthermore, PyArg provides algorithms for *dynamic* argumentation problems. In particular, the package contains an implementation of the approximation algorithm for the stability problem from [15], as well as an inexact but efficient algorithm for estimating relevance [16] based on the labels from the aforementioned stability algorithm. Finally, PyArg provides algorithms for *realisability* in abstract argumentation [17] and ABA [18], that is: given a semantics and a set of extensions, is there an (assumption-based) AF that, for the given semantics, has exactly these extensions?

In addition, PyArg provides several *generators*. For generating ASPIC$^+$ argumentation systems, PyArg uses the "layered" generator from [15, Section 4.2.5]. For abstract AFs, PyArg provides a basic random generator. Furthermore, PyArg provides various *importers and exporters* to convert argumentation settings to various formats.

4. Web interface

In order to demonstrate how PyArg's algorithms can be applied in various settings, we provide a web interface.[5] In this section, we describe each of its five pages.

On the **generator** pages, users can generate an ASPIC$^+$ argumentation system or abstract AF, parameterised by specific settings.

For **visualisation**, the user can choose between abstract argumentation [8], ASPIC$^+$ [9] and ABA [10, 11]. As a first step, the user either chooses a predefined argumentation setting or gives a specification of a new one. For the abstract AFs, users can provide arguments and the attacks between them; in the ASPIC$^+$ setting users can provide axioms, ordinary premises with their preferences, strict rules, defeasible rules with their preferences and a choice in how to derive an ordering from these preferences; and in the ABA setting, users provide atoms, rules, assumptions and contraries. Given this input, the corresponding AF is visualised as a graph. Next, this input is evaluated based on a large variety of extension-based semantics [12]. The extensions of a given semantics are presented as buttons; by clicking on a button, the corresponding extension is visualised by coloring the graph – see Figure 1. PyArg features both a regular mode (in which arguments in the extension are coloured green, while other arguments are yellow or red) and a colourblind-friendly mode that uses an adapted colour

[2] https://github.com/DaphneOdekerken/PyArg
[3] https://pypi.org/project/python-argumentation/
[4] https://daphneodekerken.github.io/PyArg/
[5] https://pyarg.npai.science.uu.nl/

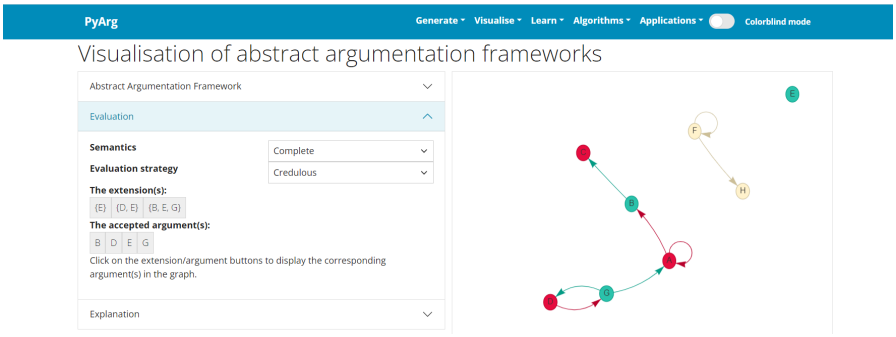

Figure 1: The visualisation page for abstract AFs.

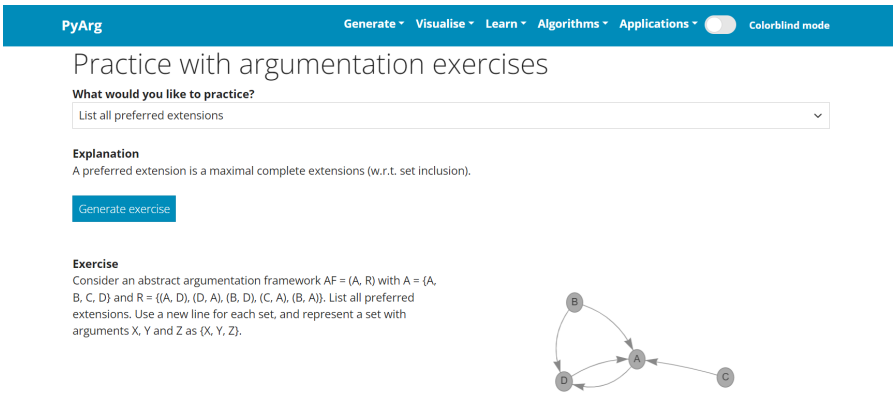

Figure 2: One of the learning pages, where a user can practice identifying preferred extensions.

palette. For abstract argumentation and ASPIC$^+$, the user can additionally request explanations for (non-)accepted arguments and formulas [13, 14].

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. As the learner starts an exercise, PyArg generates a random abstract AF using its generators (see Figure 2). The learner then gives the extensions, and PyArg uses its semantics algorithms for validating the learner's solutions.

The **algorithms** pages showcase research on realisability, see Figure 3. The user can input sets of extensions in the text field. PyArg then computes and displays properties of these extensions [17]. Depending on these properties and the semantics, it may be possible to generate a canonical AF with exactly these semantics. If that is the case, the corresponding semantics button becomes active and the user can generate the canonical AF.

The chat interface (Figure 4) is an **application** for the algorithms for stability and relevance in inquiry dialogue. First, the user chooses an ASPIC$^+$ argumentation system, 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.

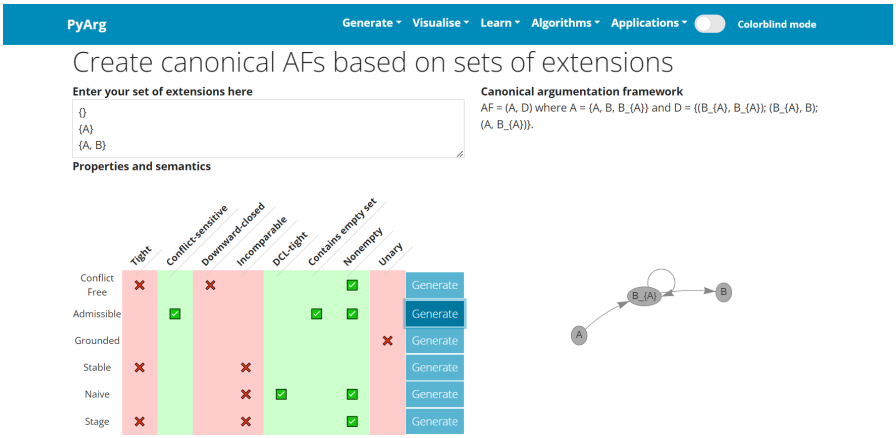

Figure 3: One of the algorithms pages showcasing research on realisability.

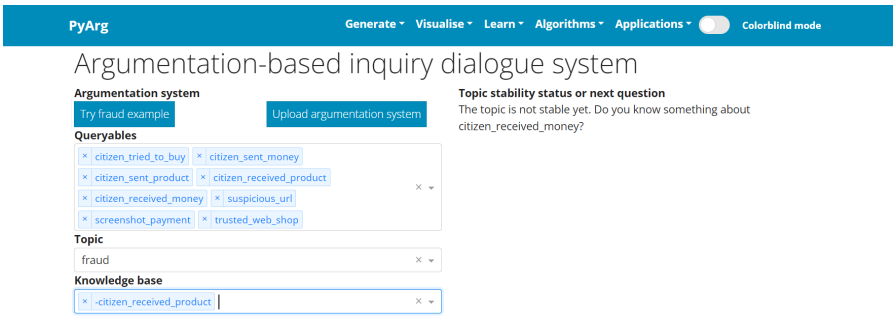

Figure 4: The chat interface on the applications page.

5. 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. We are currently working on improving the back-end performance by calling more efficient algorithms, collaborating with authors of existing solvers. In addition, we aim to add support for more formalisms. On a final note, we are open to collaborations and suggestions for additional functionalities, algorithms or other feedback, which we hope to incorporate in future releases of PyArg.

Acknowledgments

The authors 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] A. Borg, D. Odekerken, PyArg for solving and explaining argumentation in Python: Demonstration, in: Proceedings of (COMMA-22), 2022, pp. 349–350.
[2] D. Odekerken, A. Borg, M. Berthold, Accessible algorithms for applied argumentation, in: Proceedings of (Arg&App-23), 2023, pp. 92–98.
[3] 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.
[4] M. Thimm, The formal argumentation libraries of Tweety, in: Proceedings of (TAFA-17), Springer, 2018, pp. 137–142.
[5] M. Snaith, C. Reed, TOAST: Online ASPIC+ implementation, in: Proceedings of (COMMA-12), IOS Press, 2012, pp. 509–510.
[6] N. I. Spanoudakis, G. Gligoris, A. Koumi, A. C. Kakas, Explainable argumentation as a service, Journal of Web Semantics (2023) 100772.
[7] 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.
[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] A. Bondarenko, F. Toni, R. A. Kowalski, An assumption-based framework for non-monotonic reasoning, in: Proceedings of (LPNMR-93), 1993, pp. 171–189.
[11] 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.
[12] P. Baroni, M. Caminada, M. Giacomin, An introduction to argumentation semantics, The Knowledge Engineering Review 26 (2011) 365–410.
[13] A. Borg, F. Bex, A basic framework for explanations in argumentation, IEEE Intelligent Systems 36 (2021) 25–35.
[14] A. Borg, F. Bex, Necessary and sufficient explanations for argumentation-based conclusions, in: Proceedings of (ECSQARU-21), Springer, 2021, pp. 45–58.
[15] D. Odekerken, F. Bex, A. Borg, B. Testerink, Approximating stability for applied argument-based inquiry, Intelligent Systems with Applications 16 (2022) 200110.
[16] 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.
[17] P. E. Dunne, W. Dvořák, T. Linsbichler, S. Woltran, Characteristics of multiple viewpoints in abstract argumentation, Artificial Intelligence 228 (2015) 153–178.
[18] M. Berthold, A. Rapberger, M. Ulbricht, On the expressive power of assumption-based argumentation, in: Proceedings of (JELIA-23), 2023.