Argumentation benchmarks from the wild

Daphne Odekerken
National Police Lab AI, Netherlands Police
Department of Information and Computing Sciences, Utrecht University
d.odekerken@uu.nl

***Abstract*—We present a benchmark generator for abstract argumentation frameworks (AFs) and assumption-based argumentation frameworks (ABAFs). These AFs and ABAFs are derived from ASPIC$^+$ argumentation theories that are similar in structure to the argumentation theories used in a real-life application at the Dutch police.**

***Index Terms*—Benchmark generator, applications, abstract argumentation, assumption-based argumentation**

I. Background

One of the applications of computational argumentation in law enforcement is a system for the intake of police reports on online trade fraud [1]. This system advices citizens whether or not they should submit the report to the police. The advice is obtained automatically by reasoning with argumentation theories in the ASPIC$^+$ formalism. These argumentation theories contain a set of defeasible rules, which model the definition of online trade fraud, and a knowledge base with evidence provided by the citizen.

The set of rules in the argumentation theories has a layered structure: there is one central high-level rule, which gives a general definition of fraud, in combination with rules that further specify the elements of this definition and rules for exceptions. There are 43 rules in total. More details are given in Table 3 of [1].

In the future, we would like to develop applications for more general topics in the law enforcement domain than just online trade fraud. This requires algorithms that can efficiently reason with larger ASPIC$^+$ argumentation theories. We expect that the rule sets for such domains have a similar layered structure. Therefore we developed a script for randomly generating layered argumentation theories of a specified size.

II. Properties of the generated argumentation theories

The generator uses PyArg [2] to randomly generate ASPIC$^+$ argumentation theories with a given number of literals. Each literal is assigned a layer, which informally is the largest number of rule applications to reach this literal from another literal. The notion of literal layers is illustrated in Figure 1. For example, the literal l_1 is on Layer 0, as there is no rule for l_1. Given that the only rule for l_2 is $l_1 \Rightarrow l_2$, the literal l_2 is on Layer 1. Even though l_3 can be derived from $\neg l_0, \neg l_2 \Rightarrow l_3$ (and both $\neg l_0$ and $\neg l_2$ are on Layer 0), it is on Layer 2 since there is a rule $l_2 \Rightarrow l_3$ and l_2 is on Layer 1.

The literal layer distribution is selected to have $\frac{2}{3} \cdot |\mathcal{L}|$ literals with layer 0, each one-tenth of the literals for Layers 1, 2 and 3 and the remaining literals in Layer 4. This distribution is similar to the distribution of the fraud intake application.

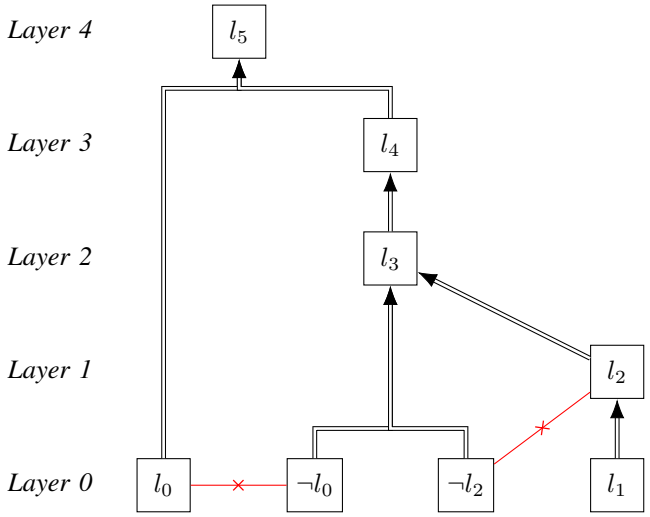

Fig. 1. Sample of a generated rule set, illustrating the layers of literals. Defeasible rules are represented by double-lined arrows and contradicting literals by a red line with a cross in the middle. For readability, not all literals and rules of the argumentation theory are visualised.

After assigning a layer to each literal, the defeasible rules are generated in such a way that they establish this distribution of layers. Just like in the application for fraud intake, there are no strict rules and no ordinary premises. The number of rules is 1.5 times the number of literals. Each rule has at least 1 and at least 5 antecedents.

Contrariness is based on classical negation. Each literal in the language has only its negation as its contradictory. Finally, a consistent set of axioms is randomly selected from the literals with Layer 0.

These steps result in ASPIC$^+$ argumentation theories that are similar in structure to the application at the Dutch police. In order to convert these into instances for ICCMA, we next convert these into AFs as well as ABAFs.

III. Conversion into AFs

Any ASPIC$^+$ argumentation theory can be converted into an abstract argumentation framework by listing all arguments inferred by the argumentation theory (following [3] Definition 5) and identifying all attacks between those arguments. For converting the generated ASPIC$^+$ argumentation theories to AFs we provide a script based on PyArg [2]. Note that

all attacks are rebuttals, because the argumentation theory does not contain ordinary premises (which are required for undermining attacks) or rules contradicting rule applications (which are required for undercutting attacks). These AFs are typically small in size compared to the input argumentation theory. An explanation for this is that many rules do not participate in any argument, as some of their antecedents are not axioms, nor conclusions of any other rule-based argument. This is also in line with the aforementioned application in fraud intake, where it depends on the evidence provided by the citizen which rules apply.

IV. Conversion into ABAFs

Alternatively, the ASPIC$^+$ argumentation theory can be converted into an ABAF. For this conversion we provide a script that implements the method from Definition 24 of [4]. It replaces every defeasible rule from the ASPIC$^+$ argumentation theory by two rules in the ABAF and adds an assumption that this defeasible rule holds. For instance, the ASPIC$^+$ rule $r_0 : l_0, l_4 \Rightarrow l_5$ is replaced by the ABA rules $r_0\_holds, l_0, l_4 \rightarrow l_5$ and $\neg l_5 \rightarrow \neg r_0\_holds$. Furthermore, the ABAF contains the assumption r_0_holds.

If the ASPIC$^+$ framework had $|\mathcal{L}|$ literals, then the ABAF has $4 \cdot |\mathcal{L}|$ atoms because the ASPIC$^+$ framework with $|\mathcal{L}|$ literals is created with $1.5 \cdot |\mathcal{L}|$ rules and the conversion to ABA introduces two new atoms for every ASPIC$^+$ rule. If the ASPIC$^+$ framework had $|\mathcal{R}|$ rules, then the ABA framework has $2 \cdot |\mathcal{R}|$ rules. Finally, note that the ABA framework is flat: no assumption is the consequent of a rule.

V. Usage instructions

The Python package, containing the aforementioned scripts for generating ASPIC$^+$ argumentation theories and converting them to AFs and ABAFs, can be found on GitHub.[1] The README of this repository contains usage instructions.

VI. Discussion

We developed the generator in such a way that it creates ASPIC$^+$ argumentation theories that are similar in structure to the argumentation theory in an actual application at the Dutch police. In addition, we developed scripts for converting these argumentation theories to AFs and ABAFs, which can be used for testing solvers in the ICCMA competition.

Given the small size of the AFs, we expect that solving the problems considered in ICCMA (i.e., identifying extensions and accepted arguments under a given semantics) for these instances does not require state-of-the-art solvers. However, it should be noted that the application in fraud intake required solving the more challenging argumentation problems of identifying stability [1] and relevance [5], to find out whether (and which) additional knowledge could still change the acceptance of specific literals. It would be interesting to consider these dynamic argumentation problems in future ICCMA editions.

[1] https://github.com/DaphneOdekerken/LayeredGeneratorICCMA

References

[1] D. Odekerken, F. Bex, A. Borg, and B. Testerink, "Approximating stability for applied argument-base inquiry," Intelligent Systems with Applications, Vol. 16, pp 200110, 2022.

[2] D. Odekerken, A. Borg, and M. Berthold, "Accessible algorithms for applied argumentation," Proc. of Arg&App, pp. 92–98, 2023.

[3] S. Modgil, and H. Prakken, "A general account of argumentation with preferences," Artificial Intelligence, Vol. 195, pp 361–397, 2013.

[4] J. Heyninck, and C. Strasser, "Relations between assumption-based approaches in nonmonotonic logic and formal argumentation," Proc. of NMR, pp 65–75, 2016.

[5] D. Odekerken, T. Lehtonen, A. Borg, J.P. Wallner, and M. Järvisalo, "Argumentative reasoning in ASPIC+ under incomplete information," Proc. of KR, pp 531–541, 2023.