Towards Parallelising Extension Construction for Serialisable Semantics in Abstract Argumentation

Lars Bengel, **Matthias Thimm**
Artificial Intelligence Group, University of Hagen, Germany
{lars.bengel, matthias.thimm}@fernuni-hagen.de

Abstract

We consider the recently proposed notion of *serialisability* of semantics for abstract argumentation frameworks. This notion describes a method for the serialised non-deterministic construction of extensions through iterative addition of non-empty minimal admissible sets. Depending on the semantics, the task of enumerating all extensions for an argumentation framework can be computationally complex. Serialisability provides a natural way of parallelising the construction of extensions for most admissible-based semantics. In this work, we investigate the feasibility of using the serialisable construction scheme for a more efficient enumeration of extensions on the example of the recently introduced *unchallenged* semantics and provide an experimental evaluation.

1 Introduction

The formalism of *abstract argumentation* (Dung 1995) has received a lot of attention in the literature and has become an important tool for knowledge representation and reasoning. An *abstract argumentation framework* (AF) consists of a set of arguments and an attack relation representing conflicts between arguments. One of the main objectives in abstract argumentation is computing acceptable sets of arguments, also called extensions. For that purpose, there exist many different semantics that impose varying constraints on the acceptable extensions (Baroni, Caminada, and Giacomin 2018). For a lot of these semantics, tasks like enumerating all extensions are intractable (Dvorák and Dunne 2018; Kröll, Pichler, and Woltran 2017). Thus, there is an increased interest in algorithms for solving these tasks, see for example the ICCMA competition (Bistarelli et al. 2020) for comparing argumentation solvers.

However, only few approaches utilise parallel computing to solve argumentation problems. There are, for example, SAT-based solvers, like *μ-toksia* (Niskanen and Järvisalo 2020), which employ parallelisation on the level of the SAT solver. In (Cerutti et al. 2015), an algorithm for enumerating extensions based on the decomposition of an AF into its strongly connected components (SCCs) has been proposed. On the other hand, (Doutre, Lafages, and Lagasquie-Schiex 2019) presents a clustering-based algorithm for enumerating complete, preferred or stable extensions. The algorithm first divides the AF into clusters via a spectral clustering method and computes the extensions of each cluster in parallel and finally assembles the extensions of the whole AF.

In this work, we consider the concept of *serialisability*, a non-deterministic construction scheme for admissible-based semantics (Thimm 2022; Bengel and Thimm 2022; Blümel and Thimm 2022). In this construction scheme, an extension is represented by a *serialisation sequence* consisting of *initial sets*, i. e., minimal non-empty admissible sets, of the respective reducts of the AF. This concept also enables the definition of novel semantics, such as the recently introduced unchallenged semantics (Thimm 2022). Constructing unchallenged extensions has been shown to be more computationally complex than many other admissible-based semantics (Bengel and Thimm 2022).

The contributions of this work are as follows. In Section 2 we introduce the theoretical background on AFs and serialisability. We provide SAT-encodings for determining initial sets in Section 3 and analyse the distribution of initial sets in the ICCMA'19 dataset (Section 4). Our results show that there are generally very few initial sets in these AFs, which means that exploiting the structure of serialisation sequences may be computationally beneficially for constructing extensions, in particular due to a small and structured search space. Subsequently, we propose an algorithm for enumerating unchallenged extensions in Section 5. Our algorithm first computes the initial sets of an AF and then, in parallel, recursively continues the search by the induced reducts. If no initial sets exist for a reduct, the serialisation sequence is complete and represents an extension. In contrast to previous approaches our algorithm does not require dividing the AF into clusters or SCCs or assembling the extensions in the end. Instead, the parallelisation follows directly from the nature of the non-deterministic construction scheme of serialisability. In Section 6 we present first results of an experimental evaluation of our algorithm. These results show that parallelisation does indeed provide a significant improvement in running time compared to sequential approaches for enumerating unchallenged extensions. We conclude and highlight potential future work in Section 7.

2 Preliminaries

Let $\mathfrak{A}$ denote a universal set of arguments. An *abstract argumentation framework* (AF) is a tuple $\mathsf{F} = (\mathsf{A}, \mathsf{R})$ where $\mathsf{A} \subseteq \mathfrak{A}$ is a finite set of arguments and R is a relation $\mathsf{R} \subseteq$

$\mathsf{A} \times \mathsf{A}$ (Dung 1995). Let $\mathfrak{AF}$ denote the set of all abstract argumentation frameworks. For two arguments $a, b \in \mathsf{A}$, the relation $a\mathsf{R}b$ means that argument a attacks argument b. For a set $X \subseteq \mathsf{A}$, we denote by $\mathsf{F}|_X = (X, \mathsf{R} \cap (X \times X))$ the projection of F on X. For a set $S \subseteq \mathsf{A}$ we define

$$S^+_{\mathsf{F}} = \{a \in \mathsf{A} \mid \exists b \in S : b\mathsf{R}a\} \quad S^-_{\mathsf{F}} = \{a \in \mathsf{A} \mid \exists b \in S : a\mathsf{R}b\}$$

If S is a singleton set, we omit brackets for readability, i. e., we write a^-_{F} (a^+_{F}) instead of $\{a\}^-_{\mathsf{F}}$ ($\{a\}^+_{\mathsf{F}}$). For two sets S and S' we write $S\mathsf{R}S'$ iff $S' \cap S^+_{\mathsf{F}} \neq \emptyset$.

For $\mathsf{F} = (\mathsf{A}, \mathsf{R})$ and $S \subseteq \mathsf{A}$, the S-reduct F^S is defined via $\mathsf{F}^S = \mathsf{F}|_{A \setminus (S \cup S^+)}$ (Baumann, Brewka, and Ulbricht 2020).

We say that a set $S \subseteq \mathsf{A}$ is *conflict-free* if for all $a, b \in S$ it is not the case that $a\mathsf{R}b$. A set S *defends* an argument $b \in \mathsf{A}$ if for all a with $a\mathsf{R}b$ there is $c \in S$ with $c\mathsf{R}a$. A conflict-free set S is called *admissible* if S defends all $a \in S$. Let $\mathsf{adm}(\mathsf{F})$ denote the set of admissible sets of F.

Non-empty minimal admissible sets have been coined *initial sets* by Xu and Cayrol (2016).

Definition 1. For $\mathsf{F} = (\mathsf{A}, \mathsf{R})$, a set $S \subseteq \mathsf{A}$ with $S \neq \emptyset$ is called an *initial set* if S is admissible and there is no admissible $S' \subsetneq S$ with $S' \neq \emptyset$. $\mathsf{IS}(\mathsf{F})$ denotes the set of initial sets of F.

We can also differentiate between three types of initial sets (Thimm 2022).

Definition 2. For $\mathsf{F} = (\mathsf{A}, \mathsf{R})$ and $S \in \mathsf{IS}(\mathsf{F})$, we say that

1. S is *unattacked* iff $S^- = \emptyset$,
2. S is *unchallenged* iff $S^- \neq \emptyset$ and there is no $S' \in \mathsf{IS}(\mathsf{F})$ with $S'\mathsf{R}S$,
3. S is *challenged* iff there is $S' \in \mathsf{IS}(\mathsf{F})$ with $S'\mathsf{R}S$.

In the following, we will denote with $\mathsf{IS}^{\not\leftarrow}(\mathsf{F})$, $\mathsf{IS}^{\not\leftrightarrow}(\mathsf{F})$, and $\mathsf{IS}^{\leftrightarrow}(\mathsf{F})$ the set of unattacked, unchallenged, and challenged initial sets, respectively. Based on the notions of initial sets and the reduct, we can now introduce the concept of serialisability (Thimm 2022; Blümel and Thimm 2022).

Definition 3. A *serialisation sequence* for $\mathsf{F} = (\mathsf{A}, \mathsf{R})$ is a sequence $\mathscr{S} = (S_1, \ldots, S_n)$ with $S_1 \in \mathsf{IS}(\mathsf{F})$ and for each $2 \leq i \leq n$ we have that $S_i \in \mathsf{IS}(\mathsf{F}^{S_1 \cup \cdots \cup S_{i-1}})$.

As shown in (Blümel and Thimm 2022), a serialisation sequence $(S_1, \ldots, S_n)$ induces an admissible set $E = S_1 \cup \cdots \cup S_n$ and for every admissible set there is at least one such sequence. Recalling the distinction between initial sets from Definition 2, we can further restrict the serialisation sequences to characterise existing admissibility-based semantics as well as define entirely new semantics. One such semantics is the *unchallenged* semantics introduced in (Thimm 2022).

Definition 4. Let $\mathsf{F} = (\mathsf{A}, \mathsf{R})$ be an AF and $E \subseteq \mathsf{A}$. We have that $E \in \mathsf{uc}(\mathsf{F})$ if and only if there is a serialisation sequence $(S_1, \ldots, S_n)$ with $E = S_1 \cup \cdots \cup S_n$ and for all S_i it holds that $S_i \in \mathsf{IS}^{\not\leftarrow}(\mathsf{F}^{S_1 \cup \cdots \cup S_{i-1}}) \cup \mathsf{IS}^{\not\leftrightarrow}(\mathsf{F}^{S_1 \cup \cdots \cup S_{i-1}})$ and it holds that $\mathsf{IS}^{\not\leftarrow}(\mathsf{F}^{S_1 \cup \cdots \cup S_n}) \cup \mathsf{IS}^{\not\leftrightarrow}(\mathsf{F}^{S_1 \cup \cdots \cup S_n}) = \emptyset$.

In other words, the unchallenged semantics essentially amounts to exhaustively adding unattacked and unchallenged initial sets of the F and its respective reducts. In

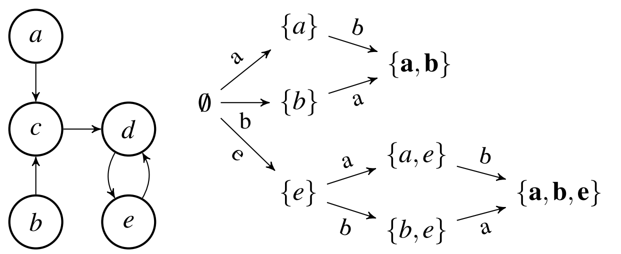

Figure 1: The AF F_1 (left) and a graph representing all serialisation sequences for the unchallenged extensions of F_1 (right).

(Bengel and Thimm 2022) it has been shown that reasoning tasks related to the unchallenged semantics—i. e., deciding whether a given argument is contained in one or all unchallenged extensions—are on the second level of the polynomial hierarchy, thus unchallenged extensions cannot be determined by a simple SAT encoding.

Example 1. Consider the AF F_1 in Figure 1. The graph on the right shows all serialisation sequences for F_1. A sequence is a path that starts at the the node $\emptyset$ and ends at one of the nodes highlighted in bold, i. e., the two unchallenged extensions: $\{a, b\}$ and $\{a, b, e\}$. The corresponding serialisation sequence is then the edge labels, e.g., (a, b) and (b, a) which both lead to the same unchallenged extension $\{a, b\}$.

3 SAT-Encoding for Initial Sets

In the following, we will introduce SAT-encodings for determining initial sets in AFs. The developed encodings are based on standard SAT-encodings for AFs by (Besnard, Doutre, and Herzig 2014) and adapted from the encodings developed by (Niskanen and Järvisalo 2020). First, we introduce some notation. Acceptance and rejection of an argument a are represented by the variables I_a and O_a. For any model, we say that if I_a is *true*, then the argument a is accepted in the corresponding extension, and if I_a is *false*, a cannot be accepted. Similarly, if O_a is *true* the argument a is rejected by the extension, otherwise it is not. Note, that while only one of I_a and O_a can be *true*, both can be set to *false*, meaning the argument is neither accepted nor rejected directly by the respective extension.

The encoding for initial sets consists of four subproblems. The first encoding in Equation 1 models the basic acceptance and rejection of arguments through the attacks in the AF and consists of three parts. The first part models that an argument cannot be accepted and rejected at the same time. Furthermore, the second part ensures that if an attacker b of an argument a is accepted, then a must be rejected. The third part constitutes that if an argument a is rejected, one of its attackers must be accepted.

$$\Psi_{rej} = \bigwedge_{a \in \mathsf{A}} (\neg O_a \vee \neg I_a) \wedge \bigwedge_{a \in \mathsf{A}} \bigwedge_{b \in a^-} (O_a \vee \neg I_b) \wedge \bigwedge_{a \in \mathsf{A}} (\neg O_a \vee \bigvee_{c \in a^-} I_c) \quad (1)$$

$$\Psi_{cf} = \bigwedge_{a \in \mathsf{A}} \bigwedge_{b \in a^-} \neg I_a \vee \neg I_b \tag{2}$$

$$\Psi_{adm} = \Psi_{rej} \wedge \Psi_{cf} \wedge \bigwedge_{a \in \mathsf{A}} \bigwedge_{b \in a^-} \neg I_a \vee O_b \tag{3}$$

$$\Psi_{non-empty,adm} = \Psi_{adm} \wedge \bigvee_{a \in \mathsf{A}} I_a \tag{4}$$

The conflict-freeness property of initial sets is modelled by Equation 2, i. e., for every pair of arguments, if there is an attack between them, then one of these arguments has to be not accepted. Combining these encodings together with the notion of defense, we obtain Equation 3 to encode admissible sets of an AF. Finally, the nonemptiness of initial sets is modelled in the second part of Equation 4.

The models of the encoding in Equation 4 will then represent non-empty admissible sets. In order to obtain initial sets, we also need to ensure the minimality. In the sense that, if we find a model S of the encoding, we ensure there is no model S' with $|\{a \in \mathsf{A} | S'(I_a)\}| < |\{a \in \mathsf{A} | S(I_a)\}|$, i. e., there exists no model for the encoding where only a subset of the arguments is accepted. If we find a model S, represented by the extension E, we add assumptions $\neg I_a$ for each argument $a \in \mathsf{A} \setminus E$ and an assumption $\bigvee_{a \in E} \neg I_a$. This blocks the solver from accepting any argument not accepted by E and also enforces that at least one accepted argument of E is rejected. If there exists model E' for this modified encoding, then E is not minimal and we continue searching with E'. If there is no model, E is minimal and an initial set of the AF.

4 Initial Sets in the ICCMA'19 Dataset

To gain an understanding of how initial sets appear in typical AFs, we consider the ICCMA'19 dataset (Bistarelli et al. 2020). We presume that a smaller number of initial sets in an AF is advantageous for an algorithm that relies on them. The reason is that a smaller number of initial sets essentially equated to fewer possibilities for the serialisation sequences and thus less paths for the algorithm to traverse.

The ICCMA'19 dataset consists of 326 AFs ranging from 4 to 10,000 arguments. For those AFs, we are interested in the number of initial sets and how the three different types of initial sets are distributed.

Figure 2 shows the frequency distribution and a boxplot for the number of initial sets. For readability, three outliers with 552, 680 and 1972 initial sets are excluded from the figure. For the whole dataset, there is an average number of 33.6 initial sets per AF and a median of 14 initial sets. There are 2 AFs with no initial sets, i. e., they consist only of arguments in odd cycles and contain no non-empty extension for all admissible-based semantics. We observe that 90% of the AFs have less than 64 initial sets and 95% of them have 95 or less initial sets. Overall 66.6% of initial sets are challenged, 26.1% are unattacked and only 7.3% are unchallenged.

The results show that most AFs have only few initial sets, which makes a serialisability-based approach for computing extensions seem interesting, especially for difficult cases, such as the unchallenged semantics. This also allows us to naturally utilise parallel computing to speed up the enumeration of extensions.

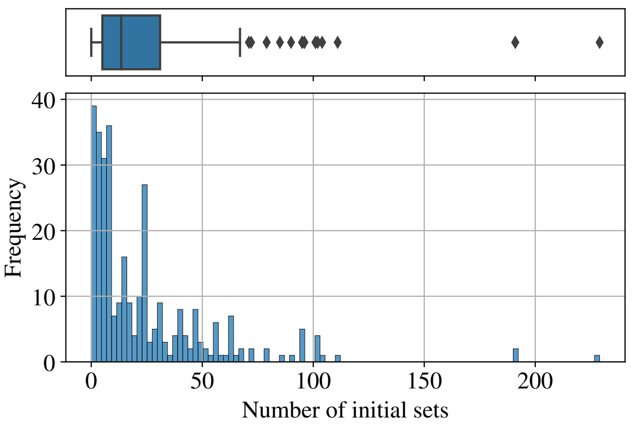

Figure 2: Frequency distribution and boxplot for the number of initial sets for the AFs in the ICCMA'19 dataset.

5 A Serialisability-based Approach to Reasoning with Unchallenged Semantics

We now propose an algorithm for enumerating unchallenged extensions based on the previously described SAT encodings and the serialisable construction scheme. In particular, we introduce a recursive algorithm that constructs all serialisation sequences for an AF while computing the initial sets at each step via the SAT encodings. Essentially, this amounts to traversing all paths in a graph, like the one depicted in Figure 1, starting from the empty set and ending in some leaf node which represent the unchallenged extensions.

Given an AF F, we initialise the search procedure by calling the function `ee_unchallenged(F)` from Algorithm 1 which starts the recursion of Algorithm 2 with F and the empty set. The recursive procedure of Algorithm 2 then works as follows:

1. We start with creating the SAT-encodings for initial sets introduced in Section 3.
2. Afterwards, we compute all initial sets of F. We compute all initial sets first, so that we can then decide for each initial set if it is challenged or not.
3. If all initial sets of the current AF are challenged, the serialisation sequence is valid and finished and the current state *ext* is an unchallenged extension of the original F.
4. We then iterate over all initial and ignore challenged initial sets. For each unattacked or unchallenged initial sets S we recursively continue with the S-reduct of F and $ext \cup S$ as the new current status of the extension construction.

Once the recursion is finished, we will have all unchallenged extensions of F. Theorem 1 follows directly from the definition of Algorithm 1.

Algorithm 1 The function `ee_unchallenged()` that starts the recursive enumeration of unchallenged extensions.

```
1: input argumentation framework $\mathsf{F} = (\mathsf{A}, \mathsf{R})$
2: ee_unchallenged(F,∅)
```

Algorithm 2 The function `ee_unchallenged()` for recursively enumerating the unchallenged extensions in AFs.

```
 1: input argumentation framework $\mathsf{F} = (\mathsf{A}, \mathsf{R})$, set of arguments $ext$
 2: $solver = \mathtt{initial\_encoding}(\mathsf{F})$
 3: $initialSets = solver.getModels()$
 4: if $existOnlyChallengedInitialSets()$ then
 5:     show $ext$
 6:     return
 7: end if
 8: for $S \in initialSets$ do
 9:     if $S \in \mathsf{IS}^{\leftrightarrow}(\mathsf{F})$ then
10:         continue
11:     end if
12:     $\mathtt{ee\_unchallenged}(\mathsf{F}^S, ext \cup S)$
13: end for
```

Theorem 1. *Algorithm 1 is sound and complete for the task of enumerating all unchallenged extensions.*

It should be noted, that since the same unchallenged extension can be constructed by taking the initial sets in a different order (see Figure 1), the above algorithm might produce the same extension multiple times. We prevent this problem by simply checking (in line 12 of Algorithm 2) if the current state of the extension $ext \cup S$ has already been checked before and in that case we skip the recursive call. That ensures every extension is only computed once.

An important aspect of Algorithm 1 is that we can parallelise the recursive calls $\mathtt{ee\_unchallenged}(\mathsf{F}^S, ext \cup S)$, which allows us to traverse multiple serialisation sequences in parallel only limited by the number of CPU cores.

In general, this procedure can be used for any serialisable semantics. For other semantics, it may also be simplified, e. g., we do not need to differentiate between different types of initial sets when considering other extensions such as those for admissible or preferred semantics.

6 Experiments

To investigate the feasibility of the parallel approach to enumerating extensions, we compare two approaches: (1) `serial-naive` a naive serialisability-based approach and (2) `serial-para` an implementation of Algorithm 2, where all recursive calls are done in parallel. Both implementations[1] are done in C++ and `serial-para` relies on the *CryptoMiniSatSolver* (Soos, Nohl, and Castelluccia 2009). The experiments where run via the *probo2* benchmark tool[2], with a timeout of 10 minutes, on a machine with 2x 3.4 GHz Intel Xeon E5-2643v3 CPU with a total of 12 cores and 24 threads as well as 192 GB memory.

The first results of the evaluation on the ICCMA'19 dataset are shown in Figure 3. Note the logarithmic scale of the y-axis. The figure shows a cactus plot comparing the runtimes of both versions. While `serial-naive` was only able to solve 69 of the 326 instances, `serial-para`

[1]https://github.com/aig-hagen/serialisability-solver
[2]https://github.com/aig-hagen/probo2

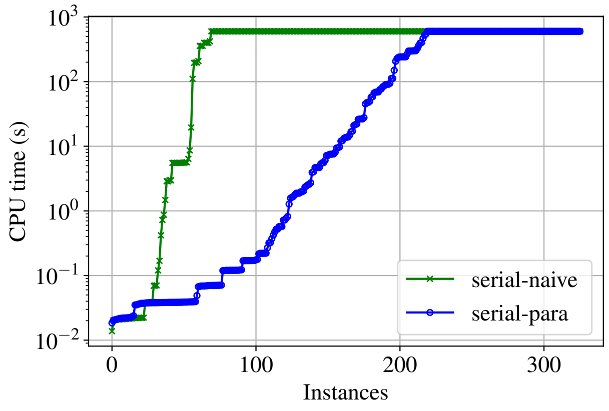

Figure 3: Cactus plot with the results of the experiment for enumerating unchallenged extensions for the ICCMA'19 dataset.

was able to solve 219 instances in total. On average, the parallel approach improves total runtime by $53,7\%$ compared to the naive approach. In general, we can observe that the `serial-para` approach was faster on almost all instances of the ICCMA'19 dataset. Only for some of the easier instances, we can see cases where `serial-naive` is faster, most likely due to the computational overhead of parallelisation. However, on the larger and more difficult instances `serial-para` consistently and significantly outperforms the naive approach.

Overall, the first results show that parallelisation does indeed improve the runtime compared to the naive approach and it seems especially interesting to further explore the parallel approach for larger and more difficult instances.

7 Discussion and Future Work

We considered the notion of serialisability and proposed an algorithm for enumerating unchallenged extensions based on this construction scheme. The algorithm can be parallelised naturally without requiring any additional computations on the AF. Our first experimental results show a significant improvement in runtime, especially for the more difficult instances of the ICCMA'19 dataset.

For future work, we plan to further refine the serialisable extension construction. That includes algorithmic optimisations, but also adapting the approach to other admissible-based semantics. An interesting application are also the more difficult reasoning problems, like skeptical preferred reasoning. Here, we could use the serialisable construction scheme in combination with some heuristics to guide the search for a counterexample to disprove skeptical acceptance of an argument. We also intent to provide a more thorough evaluation of our approach, including a comparison to other existing approaches (Doutre, Lafages, and Lagasquie-Schiex 2019; Bistarelli et al. 2020) for tasks related to admissible-based semantics.

Acknowledgements

The research reported here was partially supported by the Deutsche Forschungsgemeinschaft (grant 375588274).

References

Baroni, P.; Caminada, M.; and Giacomin, M. 2018. Abstract argumentation frameworks and their semantics. In Baroni, P.; Gabbay, D.; Giacomin, M.; and van der Torre, L., eds., *Handbook of Formal Argumentation*. College Publications. 159–236.

Baumann, R.; Brewka, G.; and Ulbricht, M. 2020. Revisiting the foundations of abstract argumentation–semantics based on weak admissibility and weak defense. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 2742–2749.

Bengel, L., and Thimm, M. 2022. Serialisable semantics for abstract argumentation. In *Computational Models of Argument*. IOS Press. 80–91.

Besnard, P.; Doutre, S.; and Herzig, A. 2014. Encoding argument graphs in logic. In *Information Processing and Management of Uncertainty in Knowledge-Based Systems: 15th International Conference, IPMU 2014, Montpellier, France, July 15-19, 2014, Proceedings, Part II 15*, 345–354. Springer.

Bistarelli, S.; Kotthoff, L.; Santini, F.; and Taticchi, C. 2020. A first overview of iccma'19. In *AI^3 @ AI* IA*, 90–102.

Blümel, L., and Thimm, M. 2022. A ranking semantics for abstract argumentation based on serialisability. *Computational Models of Argument: Proceedings of COMMA 2022* 353:104.

Cerutti, F.; Tachmazidis, I.; Vallati, M.; Batsakis, S.; Giacomin, M.; and Antoniou, G. 2015. Exploiting parallelism for hard problems in abstract argumentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29.

Doutre, S.; Lafages, M.; and Lagasquie-Schiex, M.-C. 2019. A distributed and clustering-based algorithm for the enumeration problem in abstract argumentation. In *PRIMA 2019: Principles and Practice of Multi-Agent Systems: 22nd International Conference, Turin, Italy, October 28–31, 2019, Proceedings 22*, 87–105. Springer.

Dung, P. M. 1995. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games. *Artificial Intelligence* 77(2):321–358.

Dvorák, W., and Dunne, P. E. 2018. Computational problems in formal argumentation and their complexity. In Baroni, P.; Gabbay, D.; Giacomin, M.; and van der Torre, L., eds., *Handbook of formal argumentation*, volume 2. College Publications. chapter 14, 631–687.

Kröll, M.; Pichler, R.; and Woltran, S. 2017. On the complexity of enumerating the extensions of abstract argumentation frameworks. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17)*.

Niskanen, A., and Järvisalo, M. 2020. μ-toksia: An efficient abstract argumentation reasoner. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, volume 17, 800–804.

Soos, M.; Nohl, K.; and Castelluccia, C. 2009. Extending sat solvers to cryptographic problems. In *Theory and Applications of Satisfiability Testing-SAT 2009: 12th International Conference, SAT 2009, Swansea, UK, June 30-July 3, 2009. Proceedings 12*, 244–257. Springer.

Thimm, M. 2022. Revisiting initial sets in abstract argumentation. Submitted for publication, https://arxiv.org/abs/2204.09985.

Xu, Y., and Cayrol, C. 2016. Initial sets in abstract argumentation frameworks. In *Proceedings of the 1st Chinese Conference on Logic and Argumentation (CLAR'16)*, volume 1811, 72–85.