On Computing the Set of Acceptable Arguments in Abstract Argumentation

Matthias THIMM [a,1], Federico CERUTTI [b] and Mauro VALLATI [c]
[a] *University of Koblenz-Landau, Germany*
[b] *University of Brescia, Italy*
[c] *University of Huddersfield, UK*

Abstract. We investigate the computational problem of determining the set of acceptable arguments in abstract argumentation wrt. credulous and skeptical reasoning under grounded, complete, stable, and preferred semantics. In particular, we investigate the computational complexity of that problem and its verification variant, and develop four SAT-based algorithms for the case of credulous reasoning under complete semantics, two baseline approaches based on iterative acceptability queries and extension enumeration and two optimised algorithms.

Keywords. abstract argumentation, computational complexity, algorithms

1. Introduction

In abstract argumentation [5], an argument a is skeptically (credulously) accepted wrt. some semantics σ, if it belongs to all (at least one) σ-extensions, respectively. Work on algorithms for solving reasoning problems in abstract argumentation—see e. g. the survey [2]—so far focused on deciding acceptability for a single query argument, or determining a single or all σ-extensions. However, the computational problem of directly computing the set of all acceptable arguments (wrt. either credulous or skeptical reasoning) has not been considered yet explicitly in the literature. Of course, this problem can be solved by reducing it to the aforementioned problems. For example, one can determine the set of all credulously accepted arguments by first computing all σ-extensions and then taking their union. In this paper, we ask the question whether this obvious approach is appropriate for the problem and whether other approaches provide superior performance.

Having efficient algorithms for computing the set of credulously or skeptically accepted arguments is of practical importance. Knowing whether specific arguments are not in any possible extensions—the dual problem of credulous acceptance—or knowing whether arguments are skeptically justified is of great service as also discussed in [11]. It allows human analysts to reduce their cognitive burden by consciously deciding whether or not to look more into a specific argument they made in their sense-making process.

[1] Corresponding Author: Matthias Thimm, University of Koblenz-Landau, Germany; E-mail: thimm@uni-koblenz.de.

In this paper, we first have a look at the theoretical complexity of the problem of verifying whether a given set of arguments is exactly the set of acceptable arguments wrt. both credulous and skeptical reasoning and the grounded, complete, stable, and preferred semantics. Our results mirror similar previous results [6] in that, for example, the verification problem for grounded semantics under both credulous and skeptical reasoning is in P, while the verification problem for skeptical reasoning for preferred semantics is DP2-complete (see Section 3 for definitions of the complexity classes). While the proofs of membership follow easily from existing results [6], the hardness proofs require some novel reduction techniques and insights. In addition to this theoretical analysis, we also present four SAT-based algorithms for addressing the question of determining the set of acceptable arguments wrt. credulous reasoning under complete semantics: two baseline approaches based on iterative acceptability queries and extension enumeration and two optimised algorithms. We provide an extensive experimental evaluation of the introduced algorithms considering all benchmarks from the past ICCMA competitions.

2. Preliminaries

An *abstract argumentation framework* AF is a tuple $\mathsf{AF} = (\mathsf{A}, \mathsf{R})$ where A is a set of arguments and R is a relation $\mathsf{R} \subseteq \mathsf{A} \times \mathsf{A}$. For two arguments $a, b \in \mathsf{A}$ the relation $a\mathsf{R}b$ means that argument a attacks argument b. For $a \in \mathsf{A}$ define $a^- = \{b \mid b\mathsf{R}a\}$ and $a^+ = \{b \mid a\mathsf{R}b\}$. We say that a set $S \subseteq \mathsf{A}$ *defends* an argument $b \in \mathsf{A}$ if for all a with $a\mathsf{R}b$ then there is $c \in S$ with $c\mathsf{R}a$.

Semantics are given to abstract argumentation frameworks by means of extensions [5]. An extension E is a set of arguments $E \subseteq \mathsf{A}$ that is intended to represent a coherent point of view on the argumentation modelled by AF. Arguably, the most important property of a semantics is its admissibility. An extension E is called *admissible* if and only if (1) E is *conflict-free*, i. e., there are no arguments $a, b \in E$ with $a\mathsf{R}b$ and (2) E *defends* every $a \in E$, and it is called *complete* (CO) if, additionally, it satisfies (3) if E defends a then $a \in E$.

Different types of classical semantics can be phrased by imposing further constraints. In particular, a complete extension E: is *grounded* (GR) if and only if E is minimal; is *preferred* (PR) if and only if E is maximal; and is *stable* (ST) if and only if $\mathsf{A} = E \cup \{b \mid \exists a \in E : a\mathsf{R}b\}$.

All statements on minimality/maximality are meant to be with respect to set inclusion. Note that the grounded extension is uniquely determined and that stable extensions may not exist [5].

Let $\sigma \in \{\mathsf{CO}, \mathsf{GR}, \mathsf{ST}, \mathsf{PR}\}$ be some semantics and $\mathsf{AF} = (\mathsf{A}, \mathsf{R})$ and abstract argumentation framework. Then, an argument $a \in \mathsf{A}$ is *skeptically accepted* in AF, denoted by $\mathsf{AF} \models^{\mathrm{s}}_{\sigma} a$, if a is contained in *every* σ-extension. An argument $a \in \mathsf{A}$ is *credulously accepted* in AF, denoted by $\mathsf{AF} \models^{\mathrm{c}}_{\sigma} a$, if a is contained in *some* σ-extension. Define $\mathsf{Acc}^s_\sigma(\mathsf{AF}) = \{a \in \mathsf{A} \mid \mathsf{AF} \models^s_\sigma a\}$ and $\mathsf{Acc}^c_\sigma(\mathsf{AF}) = \{a \in \mathsf{A} \mid \mathsf{AF} \models^c_\sigma a\}$ to be the sets of skeptically and credulously accepted arguments in AF, respectively. Observe that $\mathsf{Acc}^s_\sigma(\mathsf{AF}) \subseteq \mathsf{Acc}^c_\sigma(\mathsf{AF})$ for all semantics and abstract argumentation frameworks, except for $\sigma = \mathsf{ST}$ and an argumentation framework AF' that possesses no stable extension. In the latter case $\mathsf{Acc}^s_\sigma(\mathsf{AF}') = \mathsf{A}$ and $\mathsf{Acc}^c_\sigma(\mathsf{AF}') = \emptyset$ by definition.

In the remainder of the paper, we consider the computational problem of determining the sets $\mathsf{Acc}^s_\sigma(\mathsf{AF})$ and $\mathsf{Acc}^c_\sigma(\mathsf{AF})$, respectively. Note that, these exact problems have

not been investigated before, to the best of our knowledge, in terms of computational complexity and algorithms. Previous studies and algorithms either focus on a single acceptability problem, such as deciding whether $\mathsf{AF} \models^{x}_{\sigma} a$ is true for $x \in \{s,c\}$ and some argument $a \in \mathsf{A}$, or computing one or all extensions (as done in the ICCMA series of argumentation competitions[2]).

3. Complexity of Computing the Set of Acceptable Arguments

We assume familiarity with basic concepts of computational complexity and basic complexity classes such as P, NP and coNP, see [9] for an introduction. Recall that every decision problem can be represented as a language L that contains exactly those instances to the problem with answer "yes." A complexity class can then be represented by the languages of those problems it contains. We will make use of the complexity class DP, which is defined as $\mathsf{DP} = \{L_1 \cap L_2 \mid L_1 \in \mathsf{NP}, L_2 \in \mathsf{coNP}\}$. So DP contains those languages that are intersections of a language in NP and a language in coNP. We also need the following class $\mathsf{DP2} = \{L_1 \cap L_2 \mid L_1 \in \mathsf{NP}^{\mathsf{NP}}, L_2 \in \mathsf{coNP}^{\mathsf{NP}}\}$ where $\mathsf{NP}^{\mathsf{NP}}$ is the class of problems that can be solved by a non-deterministic Turing machine in polynomial time that has access to an NP oracle and $\mathsf{coNP}^{\mathsf{NP}}$ is the class of problems where the complement can be solved by a non-deterministic Turing machine in polynomial time that has access to an NP oracle. $\mathsf{NP}^{\mathsf{NP}}$ is also written as Σ_2^P and $\mathsf{coNP}^{\mathsf{NP}}$ as Π_2^P. So $\mathsf{DP2}$ contains those languages that are intersections of a language in Σ_2^P and a language in Π_2^P.

In this section we are interested in the computational complexity of the following decision problem:

$\mathsf{ACC}^{x}_{\sigma}$ **Input**: $\mathsf{AF} = (\mathsf{A}, \mathsf{R})$ and $E \subseteq \mathsf{A}$
Output: TRUE iff $E = \mathsf{Acc}^{x}_{\sigma}(\mathsf{AF})$.

for a semantics σ and $x \in \{s,c\}$.

The proofs of the following results are omitted due to space restrictions but can be found in an online appendix[3]. We start with the tractable problems.

Proposition 1. *ACC^{s}_{GR}, ACC^{c}_{GR}, and ACC^{s}_{CO} are in P.*

Many other problems are DP-complete.

Proposition 2. *ACC^{c}_{CO}, ACC^{c}_{PR}, and ACC^{c}_{ST} are DP-complete.*

Proposition 3. *ACC^{s}_{ST} is DP-complete.*

Skeptical inference with preferred semantics is (unsurprisingly) on the second level of the polynomial hierarchy.

Proposition 4. *ACC^{s}_{PR} is DP2-complete.*

[2] `http://argumentationcompetition.org`
[3] `http://mthimm.de/misc/mtfcmv_accAF_proofs.pdf`

The results from above also allow us to easily provide an upper bound for the computational complexity of the functional problem of determining the set of acceptable arguments. For the following result, recall that $\mathsf{FNP}^{\mathsf{DP}[1]}$ is the complexity class of functional problems that can be solved by a non-deterministic Turing machine running in polynomial time that can call a DP-oracle for a constant number of times. The class $\mathsf{FNP}^{\mathsf{DP2}[1]}$ is defined analogously.

Corollary 5. *Let* AF *be an abstract argumentation framework.*

1. *The problems of computing* ACC^s_{GR}, ACC^c_{GR}, ACC^s_{CO} *are in FP, respectively.*
2. *The problems of computing* ACC^c_{CO}, ACC^c_{PR}, ACC^c_{ST}, ACC^s_{ST} *are in* $FNP^{DP[1]}$, *respectively.*
3. *The problem of computing* ACC^s_{PR} *is in* $FNP^{DP2[1]}$.

4. SAT-based Algorithms for Credulous Reasoning

We will now investigate some algorithms that compute the set ACC^x_σ. Here, we will focus on the case of credulous reasoning under complete semantics (which is equivalent to credulous reasoning under preferred semantics).

We will develop reduction-based algorithms [4,2] and leverage SAT-solving technologies. Our encodings of acceptability problems into SAT are based on the encodings proposed initially in [1] and used in modern SAT-based argumentation solvers, see e. g. [4,3]. Let $\mathsf{AF} = (\mathsf{A}, \mathsf{R})$ be and abstract argumentation framework. For each argument $a \in \mathsf{A}$ we introduce three propositional variables $\mathtt{in}_a, \mathtt{out}_a, \mathtt{undec}_a$ which model the cases that a is in the extension, a is attacked by the extension, a is not in the extension nor attacked by it, respectively. Then define $\Phi_a = (\mathtt{out}_a \Leftrightarrow \bigvee_{b \in a^-} \mathtt{in}_b) \wedge (\mathtt{in}_a \Leftrightarrow \bigwedge_{b \in a^-} \mathtt{out}_b) \wedge (\mathtt{in}_a \vee \mathtt{out}_a \vee \mathtt{undec}_a)$ and $\Psi_{\mathsf{AF}} = \bigwedge_{a \in \mathsf{A}} \Phi_a$. For any propositional formula Φ, let $\mathsf{Mod}(\Phi)$ denote its set of models. For any model ω let $E(\omega) = \{a \mid \omega(\mathtt{in}_a) = \text{TRUE}\}$. Variants of the following observations have been proven in e. g. [1], so we state it without proof.

Proposition 6. *Let* $\mathsf{AF} = (\mathsf{A}, \mathsf{R})$ *be an abstract argumentation framework.*

1. *If* $\omega \in Mod(\Psi_{\mathsf{AF}})$ *then* $E(\omega)$ *is a complete extension of* AF.
2. *If* E *is a complete extension of* AF *then there is* $\omega \in Mod(\Psi_{\mathsf{AF}})$ *with* $E(\omega) = E$.
3. $a \in Acc^c_{CO}(\mathsf{AF})$ *if and only if* $\Psi_{\mathsf{AF}} \wedge \mathtt{in}_a$ *is satisfiable.*

The above observations enable us to use SAT solving technology by encoding abstract argumentation problems into one or a series of SAT problems.[4]

4.1. Iterative Acceptability Queries via SAT

A straightforward algorithm for determining $\mathsf{Acc}^c_{\mathsf{CO}}(\mathsf{AF})$ is to exploit observation 3.) of Proposition 6 and check for each $a \in \mathsf{A}$ whether $\Psi_{\mathsf{AF}} \wedge \mathtt{in}_a$ is satisfiable using some SAT solver. We denote this algorithm IAQ, it is depicted as Algorithm 1. We write $\text{SAT}(\phi)$ for a call to an external SAT solver that evaluates to TRUE if ϕ is satisfiable.

[4]Note that formulas such as Ψ_{AF} can be easily turned in conjunctive normal form, the standard input format for SAT solvers, with only polynomial overhead, so we do not explicitly discuss matters related to this aspect in the following.

Algorithm 1 Algorithm IAQ

Input: $\mathsf{AF} = (\mathsf{A}, \mathsf{R})$
Output: $\mathsf{Acc}^c_{\mathsf{CO}}(\mathsf{AF})$
1: $S = \emptyset$
2: **for** $a \in \mathsf{A}$ **do**
3: **if** $\textsc{Sat}(\Psi_{\mathsf{AF}} \wedge \mathtt{in}_a)$ **then**
4: $S \leftarrow S \cup \{a\}$
5: **return** S

4.2. Exhaustive extension enumeration via SAT

Another straightforward approach is to leverage the fact that SAT solvers usually do not only report on the satisfiability of a given formula but also provide a model as witness. For a model ω let $C(\omega) = \bigvee_{\omega(\alpha)=\textsc{true}} \neg\alpha \vee \bigvee_{\omega(\alpha)=\textsc{false}} \alpha$. One can then enumerate all models of formula ϕ by first retrieving any one model ω, then retrieving a model ω' of $\phi \wedge C(\omega)$, then a model ω'' if $\phi \wedge C(\omega) \wedge C(\omega')$ and so on. It is clear that all models retrieved this way are models of ϕ and that by adding $C(\omega)$ we avoid retrieving the same model on future calls again. At some point, the formula becomes unsatisfiable and we retrieved all models. We can use this strategy to enumerate all complete extensions of an input abstract argumentation framework (using observations 2 and 3 of Proposition 6). The union of these is then the set $\mathsf{Acc}^c_{\mathsf{CO}}(\mathsf{AF})$. We denote this algorithm EEE, it is depicted as Algorithm 2. We write $\textsc{Witness}(\phi)$ for a call to an external SAT solver that evaluates to a model ω of ϕ if ϕ is satisfiable, or FALSE otherwise.

Algorithm 2 Algorithm EEE

Input: $\mathsf{AF} = (\mathsf{A}, \mathsf{R})$
Output: $\mathsf{Acc}^c_{\mathsf{CO}}(\mathsf{AF})$
1: $S = \emptyset$
2: $\Psi \leftarrow \Psi_{\mathsf{AF}}$
3: **while** $\textsc{false} \neq \omega = \textsc{Witness}(\Psi)$ **do**
4: $S \leftarrow S \cup E(\omega)$
5: $\Psi \leftarrow \Psi \wedge C(\omega)$
6: **return** S

4.3. Selective extension enumeration via SAT

We now turn to our proposal of the first non-trivial algorithm for computing $\mathsf{Acc}^c_{\mathsf{CO}}(\mathsf{AF})$. A major drawback of the algorithm EEE is that an abstract argumentation framework may feature an exponential number of complete extensions and many may overlap to a large degree. It may therefore be the case that in many iterations of the main loop in line 3 of Algorithm 2 no new arguments are added to S. In order to address this issue we propose a more selective extension enumeration SEE, implemented in Algorithm 3.

Differently from Algorithm 2, the algorithm SEE constrains the search for further models (line 3) by requiring that at least one argument that has not already been classified as accepted, needs to be included. Indeed, at the first iteration (line 3) the SAT solver will

Algorithm 3 Algorithm SEE

Input: $\mathsf{AF} = (\mathsf{A}, \mathsf{R})$
Output: $\mathsf{Acc}^c_{\mathsf{CO}}(\mathsf{AF})$
1: $S = \emptyset$
2: $D \leftarrow \mathsf{A}$
3: **while** FALSE $\neq \omega = \textsc{Witness}(\Psi_{\mathsf{AF}} \wedge \bigvee_{a \in D} \mathtt{in}_a)$ **do**
4: $\quad S \leftarrow S \cup E(\omega)$
5: $\quad D \leftarrow D \setminus E(\omega)$
6: **return** S

identify a complete extension with at least one `in` argument. The set of `in` arguments in the found extension will then be removed from the set D of *unvisited* arguments (line 5). From the second iteration, the SAT solver will then be forced to identify complete extensions that intersect with the unvisited arguments. It is straightforward to see that this algorithm is sound and complete.

4.4. Selective extension enumeration via MAXSAT

In (unweighted) MAXSAT [8] formulas can be either *hard* or *soft*. The solutions of a MAXSAT problem are determined among all assignments that satisfy all the hard formulas and are those that maximize the number of satisfied soft formulas. We write $\textsc{MaxSat}(S, H)$ (with a set of formulas S and a formula H) for a call to an external MAXSAT solver that evaluates to a model ω that satisfies H and a maximal number of formulas in S. If H is not satisfiable, $\textsc{MaxSat}(S, H)$ evaluates to FALSE. Algorithm 4 shows our final algorithm SEEM.

Algorithm 4 Algorithm SEEM

Input: $\mathsf{AF} = (\mathsf{A}, \mathsf{R})$
Output: $\mathsf{Acc}^c_{\mathsf{CO}}(\mathsf{AF})$
1: $S = \emptyset$
2: $D \leftarrow \mathsf{A}$
3: **while** FALSE $\neq \omega = \textsc{MaxSat}(\{\mathtt{in}_a \mid a \in D\}, \Psi_{\mathsf{AF}})$ **do**
4: $\quad S \leftarrow S \cup E(\omega)$
5: $\quad D \leftarrow D \setminus E(\omega)$
6: **return** S

The algorithm SEEM forces the MAXSAT solver to maximise the set of *unvisited* arguments at each iteration. Once again, it is straightforward to see how this algorithm is sound and complete.

5. Experimental Evaluation

We implemented the presented algorithms in the TWEETYPROJECT[5] and used the Open-WBO MAXSAT solver[6] for all calls of the form $\textsc{Sat}(\cdot)$, $\textsc{Witness}(\cdot)$, and $\textsc{MaxSat}(\cdot, \cdot)$.

[5] `http://tweetyproject.org/r/?r=acc_reasoner`
[6] `http://sat.inesc-id.pt/open-wbo/`

ICCMA'15					
No.	Algorithm	*N*	#TO	RT	PAR10
1	SEE	192	58	19947.35	3728.89
2	EEE	192	72	27061.65	4640.95
3	SEEM	192	79	21247.51	5048.16
4	IAQ	192	149	20285.96	9418.16
ICCMA'17					
No.	Algorithm	*N*	#TO	RT	PAR10
1	SEE	1050	558	60866.95	6435.11
2	SEEM	1050	579	55810.53	6670.29
3	IAQ	1050	742	54504.04	8531.91
4	EEE	1050	791	50607.98	9088.2
ICCMA'19					
No.	Algorithm	*N*	#TO	RT	PAR10
1	SEE	326	81	9775.1	3011.58
2	SEEM	326	82	14574.94	3063.11
3	EEE	326	109	11717.29	4048.21
4	IAQ	326	130	20257.52	4847.42

Table 1. Results of the ICCMA'15, ICCMA'17, and ICCMA'19 benchmark set; *N* is the total number of instances of the benchmark set; #TO gives the number of time-outs/errors of each solver on this benchmark set; RT gives the runtime in seconds on all correctly solved benchmarks; PAR10 gives the average runtime where time-outs count ten times the cutoff-time, i. e., 12,000 seconds.

We ran the experiments on a virtual machine running Ubuntu 18.04 with a 2.9 GHz CPU core and 8GB of RAM. We considered the following sets of benchmarks: ICCMA'15 consisting of 192 abstract argumentation frameworks [10]; ICCMA'17 consisting of 1050 abstract argumentation frameworks [7]; ICCMA'19 consisting of 326 abstract argumentation frameworks.[7]

Each algorithm was given 20 minutes to compute the set of acceptable arguments wrt. credulous reasoning with complete semantics. Algorithms are ranked by the number of unsolved instances (in increasing order). In case of ties, solvers are then ranked by runtime (in increasing order). We also considered the PAR10 (Penalised Average Runtime) score for comparing the performance of algorithms. PAR10 is a metric usually exploited in algorithm configuration techniques, where average runtime is calculated by considering runs that did not solve the problem as ten times the cutoff time. Intuitively,

Table 1 shows the performance of the considered algorithms on benchmarks from ICCMA'15, ICCMA'17, and ICCMA'19. The SEE algorithm is consistently delivering the best performance. Notably, on benchmarks from ICCMA'15 and ICCMA'19, the cumulative runtime of SEE is much lower than those of the other algorithms, despite the largest number of problems solved: SEE solved a larger number of problems in a much shorter amount of CPU-time. On the other end of the spectrum, the IAQ algorithm is generally delivering the worst performance. This comes as no surprise, considering that the IAQ algorithm is the less optimized among the implemented approaches.

The performance of EEE and SEEM algorithms, instead, are more remarkable. On the ICCMA'15 benchmarks the EEE algorithm is outperforming SEEM while on the ICCMA'17 benchmarks it is delivering the worst performance among the considered ap-

[7] The interested reader is referred to `https://www.iccma2019.dmi.unipg.it` for details.

proaches. In particular, the EEE algorithm shows a very limited coverage on instances generated on the basis of the Barabási–Albert model. Since frameworks derived according to the Barabási–Albert model usually have an extremely large number of complete extensions, the number of SAT calls made by the EEE algorithm is even larger than those made by the IAQ where one call per argument is made.

Finally, we observe that the selective extension enumeration implemented by the SEEM does not improve performance; instead it has a detrimental impact on performance, particularly when compared with the SEE approach. This may be due to the fact that the use of MAXSAT results in a more complex problem to be solved.

6. Summary and Conclusion

In this paper, we considered the computational task of computing the set of acceptable arguments in abstract argumentation wrt. credulous and skeptical reasoning and grounded, complete, stable, and preferred semantics. Our study on computational complexity showed that the corresponding decision variants are complete for the DP family of complexity classes, mirroring results for classical problems. We presented four different SAT-based algorithms for computing the set of credulously accepted arguments wrt. complete semantics and our evaluation showed that the two optimised algorithms significantly outperform the baseline algorithms. Future work will focus on extending the experimental evaluation to credulous reasoning with the other investigated semantics, and then investigating the case of skeptical reasoning.

Acknowledgements The research reported here was partially supported by the Deutsche Forschungsgemeinschaft (grant KE 1686/3-1).

References

[1] Philippe Besnard and Sylvie Doutre. Checking the acceptability of a set of arguments. In *Proceedinfs of NMR*, 2004.

[2] Federico Cerutti, Sarah A. Gaggl, Matthias Thimm, and Johannes P. Wallner. Foundations of implementations for formal argumentation. In *Handbook of Formal Argumentation*. 2018.

[3] Federico Cerutti, Massimiliano Giacomin, and Mauro Vallati. How we designed winning algorithms for abstract argumentation and which insight we attained. *Artificial Intelligence*, 276:1 – 40, 2019.

[4] Günther Charwat, Wolfgang Dvořák, Sarah Alice Gaggl, Johannes Peter Wallner, and Stefan Woltran. Methods for solving reasoning problems in abstract argumentation - A survey. *Artificial Intelligence*, 220:28–63, 2015.

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

[6] Wolfgang Dvořák and Paul E. Dunne. Computational problems in formal argumentation and their complexity. In *Handbook of Formal Argumentation*. 2018.

[7] Sarah A. Gaggl, Thomas Linsbichler, Marco Maratea, and Stefan Woltran. Design and results of the second international competition on computational models of argumentation. *Artificial Intelligence*, 278, 2020.

[8] Chu Min Li and Felip Manya. Maxsat, hard and soft constraints. In *Handbook of Satisfiability*. 2009.

[9] Christos Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[10] Matthias Thimm and Serena Villata. The first international competition on computational models of argumentation: Results and analysis. *Artificial Intelligence*, 252:267–294, August 2017.

[11] Mauro Vallati, Federico Cerutti, and Massimiliano Giacomin. Predictive models and abstract argumentation: the case of high-complexity semantics. *The Knowledge Engineering Review*, 34:e6, 2019.