Journal of Artificial Intelligence Research 66 (2019) 503-554 Submitted 07/2018; published 10/2019

Synthesizing Argumentation Frameworks from Examples

Andreas Niskanen ANDREAS.NISKANEN@HELSINKI.FI
Helsinki Institute for Information Technology HIIT,
Department of Computer Science,
University of Helsinki, Finland

Johannes P. Wallner WALLNER@DBAI.TUWIEN.AC.AT
Institute of Logic and Computation,
TU Wien, Austria

Matti Järvisalo MATTI.JARVISALO@HELSINKI.FI
Helsinki Institute for Information Technology HIIT,
Department of Computer Science,
University of Helsinki, Finland

Abstract

Argumentation is today a topical area of artificial intelligence (AI) research. Abstract argumentation, with argumentation frameworks (AFs) as the underlying knowledge representation formalism, is a central viewpoint to argumentation in AI. Indeed, from the perspective of AI and computer science, understanding computational and representational aspects of AFs is key in the study of argumentation.

Realizability of AFs has been recently proposed as a central notion for analyzing the expressive power of AFs under different semantics. In this work, we propose and study the *AF synthesis problem* as a natural extension of realizability, addressing some of the shortcomings arising from the relatively stringent definition of realizability. In particular, realizability gives means of establishing exact conditions on when a given collection of subsets of arguments has an AF with exactly the given collection as its set of extensions under a specific argumentation semantics. However, in various settings within the study of dynamics of argumentation—including revision and aggregation of AFs—non-realizability can naturally occur. To accommodate such settings, our notion of AF synthesis seeks to construct, or synthesize, AFs that are semantically closest to the knowledge at hand even when no AFs exactly representing the knowledge exist. Going beyond defining the AF synthesis problem, we study both theoretical and practical aspects of the problem. In particular, we (i) prove NP-completeness of AF synthesis under several semantics, (ii) study basic properties of the problem in relation to realizability, (iii) develop algorithmic solutions to NP-hard AF synthesis using the constraint optimization paradigms of maximum satisfiability and answer set programming, (iv) empirically evaluate our algorithms on different forms of AF synthesis instances, as well as (v) discuss variants and generalizations of AF synthesis.

1. Introduction

The study of representational and computational aspects of argumentation is a core topic in modern artificial intelligence (AI) research (Bench-Capon & Dunne, 2007; Baroni, Gabbay, Giacomin, & van der Torre, 2018b). A current strong focus of argumentation research is the extension-based setting of abstract argumentation frameworks (AFs) (Dung, 1995) and their generalizations. AFs model conflicts among abstract arguments via directed graphs. Arguments are represented via distinct nodes as abstract entities, hiding their internal logical structure. Information on directly conflicting pairs of arguments is modeled through attacks, represented as directed edges. Semantics of AFs define *extensions* as non-conflicting sets of arguments with sought after properties.

The complexity of reasoning over AFs depends strongly on the prescribed semantics, in addition to the computational task at hand (Dunne & Wooldridge, 2009; Dvořák & Dunne, 2018). From the computational perspective, most work on argumentation focuses on reasoning about extensions of a given AF (Egly, Gaggl, & Woltran, 2010; Bistarelli & Santini, 2011; Cerutti, Dunne, Giacomin, & Vallati, 2014a; Cerutti, Giacomin, & Vallati, 2014b; Dvořák, Järvisalo, Wallner, & Woltran, 2014; Nofal, Atkinson, & Dunne, 2014; Cerutti, Gaggl, Thimm, & Wallner, 2018), addressing, e.g., the problems of extension enumeration or credulous and sceptical acceptance. Recently, the study of dynamic aspects of argumentation has also gained more ground from the computational perspective (Cayrol, de Saint-Cyr, & Lagasquie-Schiex, 2010; Liao, Jin, & Koons, 2011; Dunne, Marquis, & Wooldridge, 2012; Coste-Marquis, Konieczny, Mailly, & Marquis, 2014a, 2015; Niskanen, Wallner, & Järvisalo, 2016a; Wallner, Niskanen, & Järvisalo, 2017; Airiau, Bonzon, Endriss, Maudet, & Rossit, 2017; Dimopoulos, Mailly, & Moraitis, 2018).

In this work we take a computational perspective to the knowledge representation problem of compactly representing extensions as an AF, as an inverse problem to that of computing the extensions of a given AF. Tightly connected to this problem setting is *realizability* of AFs, as proposed by Dunne, Dvořák, Linsbichler, and Woltran (2015) and further studied extensively by various authors (Baumann, Dvořák, Linsbichler, Strass, & Woltran, 2014; Dyrkolbotn, 2014; Dunne et al., 2015; Linsbichler, Spanring, & Woltran, 2015; Pührer, 2015; Linsbichler, Pührer, & Strass, 2016a; Linsbichler, Pührer, & Strass, 2016b). In particular, realizability focuses on establishing exact conditions for when a specific AF semantics allows for exactly representing a given set of extensions as an AF, i.e., given a collection of subsets of the arguments at hand, does there exists an AF the set of extensions of which is exactly the given collection. One central motivation for the study of realizability comes from the analysis of the relationships of central AF semantics (Dunne et al., 2015) in terms of the range of sets of extensions different semantics allow for representing as AFs. Furthermore, realizability has strong connections to the study of argumentation dynamics (Coste-Marquis et al., 2014a; Delobelle, Haret, Konieczny, Mailly, Rossit, & Woltran, 2016; Diller, Haret, Linsbichler, Rümmele, & Woltran, 2018). In particular, realizability is an idealistic property in the revision of AFs (Coste-Marquis et al., 2014a; Baumann & Brewka, 2015; Diller et al., 2018) which refers to a setting where the task is to revise a current AF with new information. However, application of many of the existing revision operators results in a collection E of (desired) extensions, which is not guaranteed to be *realizable* (under many central AF semantics). In this case realizability does not provide a solution to the revision step.

More broadly, while the study of realizability has provided various fundamental insights into AFs, the concept of realizability is quite strict in that a set E of extensions is considered realizable (under a specific AF semantics σ) if and only if there is an AF the σ-extensions of which are *exactly* those in E. Implicitly, this definition hence requires that *all* other sets of arguments *must not* be extensions of the AF of interest. This strictness requires that we have *complete* knowledge of the extensions of interest, and further, in order to actually construct a corresponding AF of interest, relies on the assumption that the set of extensions is *not conflicting* in terms of allowing them to be exactly represented by an AF. From more practical perspectives, the requirement of complete knowledge of the extensions (or, equivalently, of the non-extensions) requires taking into account an exponential number of extensions or non-extensions, which is problematic. Secondly, the definition does not allow for “mistakes” or noise in the process of obtaining the extensions, and also rules out the possibility of dealing with multiple sources of potentially conflicting sets of extensions.

In this work, with a central goal of generalizing the concept of realizability to accommodate settings where non-realizability naturally occurs, we propose and study what we call the *AF synthesis problem*[1]. Specifically, AF synthesis relaxes the notion of realizability to incomplete information, assuming only partial knowledge of extensions and non-extensions as *positive* and *negative examples*. In this generalized setting, we define AF synthesis as the constrained optimization task of finding an AF that optimally represents the given examples in terms of minimizing the costs (defined via the weights of the given examples) incurred from the AF by including a negative example or not including a positive example in the corresponding set of extensions. Thereby our notion of AF synthesis gives a way of going beyond the limitation of exact realizability, allowing for obtaining a “semantically closest” AF also in cases where non-realizability occurs, such as revision of AFs. While non-realizability has been earlier addressed by allowing revision to produce a *collection of AFs* (by considering the union of the extensions of the produced AFs) (Coste-Marquis et al., 2014a), AF synthesis gives means of directly producing a single AF representing the given collection of extensions as well as possible. In a similar vein, AF synthesis also provides a solution to obtaining a single AF for semantical aggregation of extensions arising from multiple sources (Delobelle et al., 2016). Furthermore, as in the case of revision, requiring that the result is one AF faces serious challenges also for current aggregation operators. Finally, we note that AF synthesis and the algorithms developed for the problem in this article can accommodate additional structural constraints on the synthesized AF, allowing thereby to take into account potential partial structural information available at synthesis time.

Beyond precisely defining the AF synthesis problem, with more motivations for the problem discussed in Section 3, our main technical contributions are the following.

- We formally analyze the relationship of AF synthesis and realizability in terms of necessary and sufficient conditions for an AF synthesis instance to be realizable under the important semantic concepts of conflict-free and admissible sets, and under the well-established stable and preferred AF semantics (Section 4).

- We provide complexity results for AF synthesis under conflict-free and admissible sets, as well as under grounded, complete, stable, and preferred semantics. In the general case, it turns out that AF synthesis is most often NP-hard under the considered semantics, and we establish completeness for several of the semantics. We also show that under restrictions on the form of examples (positive or negative) in the input, complexity of AF synthesis drops to polynomial-time under many—but not all—of the semantics; we give polynomial-time algorithms as witnesses for inclusion in P. (Section 5)

- We develop a first constraint-based approach to optimal AF synthesis, by providing declarative encodings for AF synthesis for the NP problem variants in two Boolean optimization paradigms: maximum satisfiability (MaxSAT) and answer set programming (ASP), as well as second-level approaches using both of the paradigms for AF synthesis under the preferred semantics which we conjecture to be Σ_2^P-complete. The choice of these paradigms is strongly

1. Alternatively, one could refer to the problem focused on in this paper as an AF *learning* problem.

motivated by the success of SAT and ASP approaches to other central AF reasoning problems (Egly et al., 2010; Cerutti et al., 2014a, 2014b; Dvořák et al., 2014; Gaggl, Manthey, Ronca, Wallner, & Woltran, 2015). (Section 6)

- We present results from an extensive empirical evaluation of our MaxSAT and ASP approaches to NP-hard AF synthesis, using as benchmarks both instances generated from the benchmark AFs of the ICCMA argumentation solver competition (Thimm, Villata, Cerutti, Oren, Strass, & Vallati, 2016) as well as randomly generated AF synthesis instances (Section 7).
- Extending on the main focus of this work, we also explain how the proposed algorithms can accommodate for further structural constraints on the desired outcomes of AF synthesis, and discuss additional variants and generalizations of AF synthesis from representational and computational complexity perspectives, including how to adapt the problem to allowing for mixtures of different AF semantics, as well as symbolic representations of examples via Boolean formulas (Section 8).

In addition to Section 3, further related work is discussed in Section 9. For readability of the main text, some of the more involved formal proofs are provided in full in Appendix A.

A preliminary version of this work was presented at ECAI 2016, 22nd European Conference on Artificial Intelligence (Niskanen, Wallner, & Järvisalo, 2016b). Beyond the results of the preliminary version, in this article we further provide new complexity results for the complete and preferred semantics, additional declarative encodings based on ASP, and a novel MaxSAT-based algorithm for synthesis under preferred semantics. In addition, we considerably extend discussion on related work, as well as the empirical evaluation by including results on the additional semantics and providing a comparison between different MaxSAT algorithms and ASP.

2. Argumentation Frameworks

We start by briefly recalling argumentation frameworks (Dung, 1995) (see also definitions in Baroni, Caminada, & Giacomin, 2011, 2018a) as the central formalism in abstract argumentation, and the argumentation semantics considered in this work. We focus on finite argumentation frameworks.

Definition 1. *A (finite)* argumentation framework (AF) *is a pair $F = (A, R)$, where A is a finite non-empty set of arguments and $R \subseteq A \times A$ is the attack relation. The pair $(a, b) \in R$ indicates that a attacks b. An argument $a \in A$ is* defended *(in F) by a set $S \subseteq A$ if, for each $b \in A$ such that $(b, a) \in R$, there is a $c \in S$ such that $(c, b) \in R$.*

Example 1. *Let $F = (A, R)$ be an AF with the set of arguments $A = \{a, b, c, d, e\}$ and the attack relation $R = \{(a, b), (b, a), (b, c), (c, d), (d, e), (e, e)\}$. The AF F is represented as a directed graph in Figure 1.*

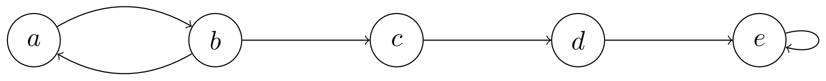

Figure 1: Argumentation framework from Example 1.

Semantics for AFs are defined through functions σ which assign to each AF $F = (A, R)$ a set $\sigma(F) \subseteq 2^A$ of extensions. We consider for σ the functions cf, stb, adm, com, prf, and grd, which stand for conflict-free, stable, admissible, complete, preferred, and grounded, respectively.[2] To formally define the semantics, we recall two standard auxiliary concepts.

Definition 2. *Given an AF $F = (A, R)$, the* characteristic function $\mathcal{F}_F : 2^A \to 2^A$ *of F is $\mathcal{F}_F(S) = \{x \in A \mid x \text{ is defended by } S\}$. Moreover, for a set $S \subseteq A$, the* range of S *is $S^+_R = S \cup \{x \in A \mid (y, x) \in R, y \in S\}$.*

The semantics considered in this work can now be defined as follows.

Definition 3. *Let $F = (A, R)$ be an AF. A set $S \subseteq A$ is* conflict-free (in F) *if there are no $a, b \in S$ such that $(a, b) \in R$. We denote the collection of conflict-free sets of F by $\mathit{cf}(F)$. For a conflict-free set $S \in \mathit{cf}(F)$, it holds that*

- *$S \in \mathit{stb}(F)$ iff $S^+_R = A$;*
- *$S \in \mathit{adm}(F)$ iff $S \subseteq \mathcal{F}_F(S)$;*
- *$S \in \mathit{com}(F)$ iff $S = \mathcal{F}_F(S)$;*
- *$S \in \mathit{grd}(F)$ iff $S \in \mathit{com}(F)$ and there is no $S' \in \mathit{com}(F)$ with $S' \subset S$; and*
- *$S \in \mathit{prf}(F)$ iff $S \in \mathit{adm}(F)$ and there is no $S' \in \mathit{adm}(F)$ with $S \subset S'$.*

If $S \in \sigma(F)$, then S is called a σ-extension, i.e., an extension under the semantics σ.

For any AF F, the subset relations $\mathit{cf}(F) \supseteq \mathit{adm}(F) \supseteq \mathit{com}(F) \supseteq \mathit{prf}(F) \supseteq \mathit{stb}(F)$ hold, which follows from the previous definition. The set-inclusions can be also seen in the following example.

Example 2. *Consider the AF F in Example 1 on page 506. For the semantics σ considered, the σ-extensions of F are enumerated in the following table.*

σ	$\sigma(F)$
cf	$\{\emptyset, \{a\}, \{b\}, \{c\}, \{d\}, \{a, c\}, \{a, d\}, \{b, d\}\}$
stb	$\{\{b, d\}\}$
adm	$\{\emptyset, \{a\}, \{b\}, \{a, c\}, \{b, d\}\}$
com	$\{\emptyset, \{a, c\}, \{b, d\}\}$
grd	$\{\emptyset\}$
prf	$\{\{a, c\}, \{b, d\}\}$

3. The Argumentation Framework Synthesis Problem

In this section we formally introduce the AF synthesis problem and discuss its motivations and connections to other notions in formal argumentation.

2. For uniformity, we will throughout the article also refer to conflict-free and admissible sets as an AF semantics.

For a given set of (possibly weighted) examples representing semantical information, we focus on the task of synthesizing an AF that has minimum cost over the examples not satisfied. Towards a formal definition, we assume a given non-empty set of arguments A from which we are to construct an AF. An example is a pair of a set of arguments and a positive integer representing the example's weight.

Definition 4. *Let A be a set of arguments. An example $e = (S, w)$ is a pair with $S \subseteq A$ and a positive integer $w > 0$.*

We denote the set S of arguments of an example $e = (S, w)$ by S_e and the weight w by w_e. For a set E of examples, we define $\mathbb{S}_E = \{S_e \mid e \in E\}$ as a shorthand for the set of all sets of arguments occurring in E.

An instance of the AF synthesis problem is defined by a set of arguments, positive and negative examples, and a semantics.

Definition 5. *An instance of the AF synthesis problem is a quadruple $P = (A, E^+, E^-, \sigma)$, with a non-empty set A of arguments, two sets of examples, E^+ and E^-, that we call positive and negative examples, respectively, and semantics σ, with $(\mathbb{S}_{E^+} \cup \mathbb{S}_{E^-}) \subseteq 2^A$.*

An AF F satisfies a positive example e if $S_e \in \sigma(F)$; similarly, F satisfies a negative example if $S_e \notin \sigma(F)$. For a given AF F, the associated cost w.r.t. P, denoted by $cost(P, F)$, is the sum of weights of examples not satisfied by F. Formally, $cost(P, F)$ is

$$\sum_{e \in E^+} w_e \cdot I(S_e \notin \sigma(F)) + \sum_{e \in E^-} w_e \cdot I(S_e \in \sigma(F)),$$

where $I(\cdot)$ is the indicator function that returns 1 if the property (membership in a set) is satisfied, and otherwise 0.[3] The task in AF synthesis is to find an AF of minimum cost over all AFs.

AF Synthesis
INPUT: $P = (A, E^+, E^-, \sigma)$ such that $(\mathbb{S}_{E^+} \cup \mathbb{S}_{E^-}) \subseteq 2^A$
TASK: Find an AF F^* with

$$F^* \in \underset{F=(A,R^*)}{\arg\min}\,(cost(P, F)).$$

Before discussing the underlying motivations of AF synthesis, we provide a concrete syntactic example of the problem definition.

Example 3. *Consider the set of positive examples $E^+ = \{(\{a, b\}, 1), (\{a, c\}, 1), (\{b, c\}, 5)\}$ and the set of negative examples $E^- = \{(\{a\}, 1), (\{a, b, c\}, 5)\}$. We illustrate these examples in Figure 2. Here we see that the positive examples together claim that each pair of arguments of A is a σ-extension. The negative examples claim that the whole set A is not a σ-extension and that the singleton set $\{a\}$ is likewise not a σ-extension. Let $P_{cf} = (A, E^+, E^-, cf)$ with $A = \{a, b, c\}$ be an AF synthesis instance under conflict-free semantics. An optimal solution AF $F_{cf} = (A, R_{cf})$ with*

3. While we focus here on the sum operation for the cost function, we acknowledge that other ways of combining costs—in terms of different operators as well as taking a qualitative view—would also be interesting to consider. We also note that the costs on examples can be used to enforce qualitative preferences over examples.

positive E^+	negative E^-
$e_1 = (\{a, b\}, 1)$	$e_4 = (\{a\}, 1)$
$e_2 = (\{a, c\}, 1)$	$e_5 = (\{a, b, c\}, 5)$
$e_3 = (\{b, c\}, 5)$	

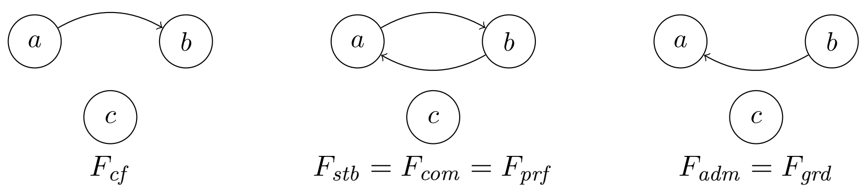

Figure 2: AF synthesis example with optimal solution AFs.

$cost(P_{cf}, F_{cf}) = 2$ is given by $R_{cf} = \{(a, b)\}$. This AF F_{cf} does not satisfy the positive example $(\{a, b\}, 1)$ and the negative example $(\{a\}, 1)$.

Regarding admissible semantics, let $P_{adm} = (A, E^+, E^-, adm)$. In this case the AF $F_{adm} = (A, R_{adm})$ with $R_{adm} = \{(b, a)\}$ is an optimal solution. Except for the positive examples $(\{a, b\}, 1)$ and $(\{a, c\}, 1)$, all other examples are satisfied by F_{adm} for P_{adm}. Thus $cost(P_{adm}, F_{adm}) = 2$. In addition, let $P_{grd} = (A, E^+, E^-, grd)$. Note that only one positive example can be satisfied since the grounded extension is unique. Now $F_{grd} = F_{adm}$ is an optimal solution to P_{grd}, since $grd(F_{grd}) = \{\{b, c\}\}$ and $cost(P_{grd}, F_{grd}) = 2$. The positive examples with the smallest weights, i.e., $(\{a, b\}, 1)$ and $(\{a, c\}, 1)$, are not satisfied.

Now consider stable semantics. Let $P_{stb} = (A, E^+, E^-, stb)$. An optimal solution AF to P_{stb} is given by $F_{stb} = (A, R_{stb})$ with $R_{stb} = \{(a, b), (b, a)\}$. Here $stb(F_{stb}) = \{\{a, c\}, \{b, c\}\}$ and $cost(P_{stb}, F_{stb}) = 1$. Similarly, for complete semantics, let $P_{com} = (A, E^+, E^-, com)$. Now $F_{com} = F_{stb}$ is an optimal solution also to P_{com}, with $com(F_{com}) = \{\{a, c\}, \{b, c\}, \{c\}\}$ and $cost(P_{com}, F_{com}) = 1$.

Finally, for preferred semantics, let $P_{prf} = (A, E^+, E^-, prf)$. Again, $F_{prf} = F_{stb}$ is an optimal solution AF to P_{prf}, with $prf(F_{prf}) = \{\{a, c\}, \{b, c\}\}$ and $cost(P_{prf}, F_{prf}) = 1$. In all these cases, only the positive example $(\{a, b\}, 1)$ is not satisfied.

We now turn to discussing motivations underlying the formal definition of AF synthesis. Formally, AF synthesis generalizes a key concept of the notion of realizability (Baumann et al., 2014; Dyrkolbotn, 2014; Dunne et al., 2015; Linsbichler et al., 2015), as first proposed and studied in the context of AFs (Dunne et al., 2015), and more recently in the context of abstract dialectical frameworks (ADFs) (Pührer, 2015; Linsbichler et al., 2016a, 2016b; Brewka, Ellmauthaler, Strass, Wallner, & Woltran, 2018). Realizability addresses the question of whether a given set of sets (of arguments) $\mathbb{S}$ can be realized by an AF, i.e., whether *there is* an AF F with $\sigma(F) = \mathbb{S}$ for a semantics σ.

Comparing the notions of realizability and AF synthesis, the strict requirement of realizability that a given set of sets of argument must be exactly equal to the semantics of an AF is relaxed by AF synthesis, by allowing for partial specification via positive and negative examples and sets that are neither. Realizability, in its general form, does not presume a set of arguments A as is assumed in AF synthesis. However, under standard semantics an instance of realizability, represented by a set of sets $\mathbb{S}$, can be translated to an instance of AF synthesis by specifying $\mathbb{S}$ as positive examples, every other set as negative, and a "sufficiently" large set A of arguments. Then, a 0-cost solution of AF synthesis implies realizability, and vice versa. In Section 4, we formally investigate this connection.

Overall, the line of work on realizability has fundamental motivations through the study of dynamic aspects of argumentation. As argued e.g. by Dunne et al. (2015), realizability provides a handle to understanding the representational limits of argumentation frameworks in terms of different semantics.

As a concrete scenario, realizability is an idealistic property in the revision of AFs (Coste-Marquis et al., 2014a; Coste-Marquis, Konieczny, Mailly, & Marquis, 2014b; Baumann & Brewka, 2015; Diller et al., 2018). An AF revision operator revises a current AF F with new information (with the new information, e.g., expressed as a formula). Formally, many revision operators define a set $\mathbb{S}$ of extensions, under a semantics σ, as the output of revision. In such a setting, the revision process can be hindered by the fact that in general there may not exist an AF F' such that $\sigma(F') = \mathbb{S}$. The notion of realizability focuses on the question of the existence of such an F', but leaves the question of what can be done in case F' does not exist unsettled. While this issue has been previously suggested to be circumvented e.g. by allowing revision to establish a *collection* of AFs with the joint property that the union of their extensions provides the desired outcome (Coste-Marquis et al., 2014a), our notion of AF synthesis provides an alternative perspective for resolving this issue, with the guarantee of resulting in a single AF. This can be argued to be preferable to a collection of AFs, and on the other hand, the single AF obtained via AF synthesis represents intuitively an aggregation of the (in case the revision step is not realizable, conflicting) knowledge represented by a collection of AFs. As such, AF synthesis also provides an approach to semantical aggregation of extensions arising from multiple sources, i.e., multiple AFs. Indeed, synthesis from multiple sources can be seen as a method of aggregating AFs semantically. While many aggregation operators for AFs exist (Bodanza, Tohmé, & Auday, 2017), most rely on aggregating the graph structures of AFs (e.g., Coste-Marquis, Devred, Konieczny, Lagasquie-Schiex, & Marquis, 2007; Tohmé, Bodanza, & Simari, 2008; Delobelle, Konieczny, & Vesic, 2015) and only few operators aggregate AFs w.r.t. AF semantics (Delobelle et al., 2016). We illustrate semantic aggregation of AFs via AF synthesis next.

Example 4. *Consider three AFs, F_1, F_2, and F_3, shown in Figure 3. Collecting all non-empty admissible sets for each AF as a positive example, and each non-admissible set as a negative example (excluding the empty set), results in 21 examples, shown as well in that figure. An optimal solution AF F' to the AF synthesis instance defined in this manner is shown on the right of that figure, together with the satisfied examples (in bold). The cost of AF F' is five.*

AF synthesis can also be viewed as providing a solution to the initial step, or a "starting point", for a revision process, for obtaining an initial AF representing knowledge represented in terms of a given set of (potentially conflicting) claimed extensions (examples). Via the relaxation of the requirement of exact knowledge of the extensions to be realized, AF synthesis connects to settings in which knowledge is not fully specified. To this end, AF synthesis could also be applicable in e.g. opponent modeling (Oren & Norman, 2009; Rienstra, Thimm, & Oren, 2013; Black, Coles, & Bernardini, 2014; Black & Hunter, 2015; Murphy, Black, & Luck, 2016) assuming that evidence (uttered by agents) revealing partial information on the internal knowledge of an opponent would come in terms of (potentially pairwise conflicting) individually conflict-free sets, i.e., what we call examples in the context of AF synthesis.

positive E_1^+	positive E_2^+	positive E_3^+
$e_1 = (\{a\}, 1)$	$\boldsymbol{e_8 = (\{c\}, 1)}$	$\boldsymbol{e_{15} = (\{c\}, 1)}$
$\boldsymbol{e_2 = (\{a, c\}, 1)}$	$e_9 = (\{b\}, 1)$	$\boldsymbol{e_{16} = (\{a, c\}, 1)}$
negative E_1^-	$e_{10} = (\{a, b\}, 1)$	negative E_3^-
$\boldsymbol{e_3 = (\{b\}, 1)}$	negative E_2^-	$\boldsymbol{e_{17} = (\{a\}, 1)}$
$e_4 = (\{c\}, 1)$	$\boldsymbol{e_{11} = (\{a\}, 1)}$	$\boldsymbol{e_{18} = (\{b\}, 1)}$
$\boldsymbol{e_5 = (\{a, b\}, 1)}$	$e_{12} = (\{a, c\}, 1)$	$\boldsymbol{e_{19} = (\{a, b\}, 1)}$
$\boldsymbol{e_6 = (\{b, c\}, 1)}$	$\boldsymbol{e_{13} = (\{b, c\}, 1)}$	$\boldsymbol{e_{20} = (\{b, c\}, 1)}$
$\boldsymbol{e_7 = (\{a, b, c\}, 1)}$	$\boldsymbol{e_{14} = (\{a, b, c\}, 1)}$	$\boldsymbol{e_{21} = (\{a, b, c\}, 1)}$

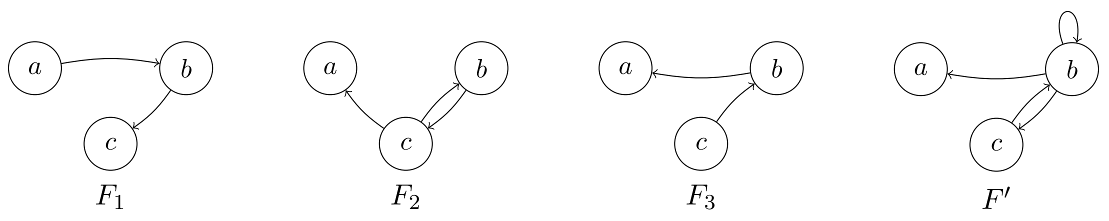

Figure 3: Example AF synthesis instance generated from (non-)admissible sets of three AFs together with an optimal solution AF F'.

Regarding a potential question about defining AF synthesis in the setting of abstract arguments rather than logic-based arguments whose structure could potentially be used to derive information on the attack structure, we note that one cannot in general assume that the structure of arguments is fully accessible (or that it even exists) (Hunter & Williams, 2012; Mailly, 2016; Dung & Thang, 2018). For example, arguments often have no internal structure in experimental medicine, were the aim is to uncover the underlying rules through experiments (Hunter & Williams, 2012; Dung & Thang, 2018). Furthermore, as argued by e.g. Hunter (2007), real arguments are often "approximate arguments", or enthymemes (Black & Hunter, 2012; Hosseini, Modgil, & Rodrigues, 2014; Mailly, 2016), i.e., arguments may only be partially specified. Hence, an attack structure between the arguments can be prone to errors, which is reflected in the AF synthesis problem that provides a way to complete the attack structure by optimizing the placement of attacks based on examples. Furthermore, as detailed later on in Section 8, our algorithms also allow for enforcing structural constraints on the desired outcomes of AF synthesis, such as hard constraints on the (certain) attacks.

Finally, we note that while weighted examples are covered by our formal definition of AF synthesis, this definition naturally also allows for disregarding weights, i.e., considering unit-weighted examples. However, by allowing for weights on examples, we wish to enable addressing settings where the reliability or relative importance of individual examples may be desirable and handled by assigning varying weights on examples. As one potential application scenario of weights, in a revision process it may be desirable to enforce a strong preference on accommodating a new example as an actual extension in the result of the revision step. Such a preference can be stated by assigning larger weights on the new (or, more generally, more recent) examples.

4. Properties of the AF Synthesis Problem

Before proceeding with a detailed complexity analysis of AF synthesis, we discuss underlying properties of the problem. In particular, we investigate the existence of 0-cost solutions for the AF synthesis problem by relating the problem with realizability results of Dunne et al. (2015). In contrast to the AF synthesis problem, Dunne et al. (2015) consider the problem of a given unweighted set $\mathbb{S}$ of sets of arguments, and ask whether there is an AF F s.t. $\mathbb{S} = \sigma(F)$. In words, in the setting of realizability, the given set exactly specifies which sets have to be σ-extensions and which must not be σ-extensions. Further, Dunne et al. (2015) do not consider weights attached to examples, and the set of arguments A is not specified and may contain more arguments than occurring in $\mathbb{S}$. Restricting the set of arguments to only arguments occurring in $\mathbb{S}$ is studied in Baumann et al. (2014), Linsbichler et al. (2015), and Baumann, Dvořák, Linsbichler, Spanring, Strass, and Woltran (2016).

We make use of and generalize the notions proposed by Dunne et al. (2015) by specifying conditions under which 0-cost solutions exist as well as properties 0-cost solutions satisfy. We first focus on the conflict-free semantics. We utilize the following concept adapted from Dunne et al. (2015, Definitions 6 and 7), defining a consequence operator that states which sets must be conflict-free if we assume a given set of sets $\mathbb{S}$ to be conflict-free in an AF. Let $\mathit{ImpliedCF}(\mathbb{S}) = \{X \mid a, b \in X \text{ implies } \exists S \in \mathbb{S} \text{ with } \{a, b\} \subseteq S\}$. Intuitively, if each set in $\mathbb{S}$ is conflict-free, and each pair of arguments in a set X is contained in one set of $\mathbb{S}$, then X is conflict-free as well. Note that a and b in this definition need not be distinct ($\{a, b\}$ is equal to $\{a\}$ if $a = b$). Further, $\emptyset$ is in $\mathit{ImpliedCF}(\mathbb{S})$ for any $\mathbb{S}$.

Lemma 1. *Let $F = (A, R)$ be an AF and $\mathbb{S} \subseteq 2^A$. If $\mathbb{S} \subseteq \mathit{cf}(F)$, then $\mathit{ImpliedCF}(\mathbb{S}) \subseteq \mathit{cf}(F)$.*

Proof. Assume that $\mathbb{S} \subseteq \mathit{cf}(F)$. Let $S \in \mathit{ImpliedCF}(\mathbb{S})$. By definition it follows that for each $a, b \in S$ we have $\exists S' \in \mathbb{S}$ with $\{a, b\} \subseteq S'$. If $\mathbb{S} \subseteq \mathit{cf}(F)$ then $S' \in \mathit{cf}(F)$ and thus $\{a, b\} \in \mathit{cf}(F)$. This implies that $S \in \mathit{cf}(F)$ (each pair in S is conflict-free). □

Example 5. *Continuing from Example 3, consider $\mathbb{S}_{E^+} = \{\{a, b\}, \{a, c\}, \{b, c\}\}$. If each element of $\mathbb{S}_{E^+}$ is conflict-free in an AF F ($\mathbb{S}_{E^+} \subseteq \mathit{cf}(F)$), then, e.g., $\{a, b, c\} \in \mathit{cf}(F)$, since there cannot be an attack between any of these three arguments. In particular, we have $\mathit{ImpliedCF}(\mathbb{S}_{E^+}) = \mathbb{S}_{E^+} \cup \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b, c\}\}$. This directly shows for P_{cf} from Example 3 that there is no solution AF to P_{cf} of cost 0. In fact, there is no AF satisfying both the positive example $(\{a, b\}, 1)$ and the negative example $(\{a\}, 1)$ under the conflict-free semantics. Also, there is no AF satisfying all three positive examples and negative example $(\{a, b, c\}, 5)$ under the conflict-free semantics.*

Equipped with the preceding lemma, we give a necessary and sufficient condition for 0-cost solutions for AF synthesis under the conflict-free semantics.

Proposition 2. *Let $P = (A, E^+, E^-, \mathit{cf})$ be an instance of AF synthesis. There is a solution AF F to P with $\mathit{cost}(P, F) = 0$ if and only if $\mathit{ImpliedCF}(\mathbb{S}_{E^+}) \cap \mathbb{S}_{E^-} = \emptyset$.*

Proof. For the "only-if" direction, assume that an AF F is an optimal solution to P of cost 0, i.e., $\mathit{cost}(P, F) = 0$. It follows that $\mathbb{S}_{E^+} \subseteq \mathit{cf}(F)$. By Lemma 1 we have $\mathit{ImpliedCF}(\mathbb{S}_{E^+}) \subseteq \mathit{cf}(F)$. Thus $\mathit{ImpliedCF}(\mathbb{S}_{E^+}) \cap \mathbb{S}_{E^-} = \emptyset$, since $\mathit{cf}(F) \cap \mathbb{S}_{E^-} = \emptyset$. For the "if" direction, assume that $\mathit{ImpliedCF}(\mathbb{S}_{E^+}) \cap \mathbb{S}_{E^-} = \emptyset$. By Lemma 17, proven in Appendix A, it follows that $\mathit{ImpliedCF}(\mathbb{S}_{E^+})$ is tight, downward closed, and contains $\emptyset$. Due to Dunne et al. (2015, Proposition 5) it immediately follows that there exists an AF $F = (A', R')$ s.t. $\mathit{cf}(F) = \mathit{ImpliedCF}(\mathbb{S}_{E^+})$.

Since $\mathit{ImpliedCF}(\mathbb{S}_{E^+}) \cap \mathbb{S}_{E^-} = \emptyset$, it follows that $\mathit{cost}(P, F) = 0$. In Dunne et al. (2015, Proposition 5) the set A' is specified as all arguments occurring in $\mathit{ImpliedCF}(\mathbb{S}_{E^+})$. If A contains more arguments, we construct $F = (A, R)$ by extending R' with self-attacks for each $A \setminus A'$. □

Intuitively, to synthesize an AF F that has $\mathbb{S}_{E^+}$ as its conflict-free sets, $\mathit{ImpliedCF}(\mathbb{S}_{E^+})$ need to be conflict-free, too. Moreover, using results from Dunne et al. (2015), one can show that there is an AF F with $\mathit{cf}(F) = \mathit{ImpliedCF}(\mathbb{S}_{E^+})$. Furthermore, if no negative example e claims that a set of $\mathit{ImpliedCF}(\mathbb{S}_{E^+})$ should not be conflict-free, i.e., $S_e \notin \mathit{ImpliedCF}(\mathbb{S}_{E^+})$, then F has cost 0.

Now consider admissible sets. Similarly as for conflict-free sets, we define the following consequence operator. For a set of sets $\mathbb{S}$, let

$$\mathit{ImpliedADM}(\mathbb{S}) = \{X \mid X = \bigcup_{S \in \mathbb{S}'} S,\ \mathbb{S}' \subseteq \mathbb{S},\ X \in \mathit{ImpliedCF}(\mathbb{S})\}.$$

Briefly put, if we assume $\mathbb{S}$ to be a collection of admissible sets, then each union of sets in $\mathbb{S}$ that is conflict-free, i.e., in $\mathit{ImpliedCF}(\mathbb{S})$, is also an admissible set. By this definition, $\emptyset \in \mathit{ImpliedADM}(\mathbb{S})$ for any $\mathbb{S}$.

Lemma 3. *Let $F = (A, R)$ be an AF and $\mathbb{S} \subseteq 2^A$. If $\mathbb{S} \subseteq \mathit{adm}(F)$, then $\mathit{ImpliedADM}(\mathbb{S}) \subseteq \mathit{adm}(F)$.*

Proof. Assume that $\mathbb{S} \subseteq \mathit{adm}(F)$ and let $S \in \mathit{ImpliedADM}(\mathbb{S})$. Then $S \in \mathit{ImpliedCF}(\mathbb{S})$ and thus $S \in \mathit{cf}(F)$ (Lemma 1). Finally, every union of admissible sets which is conflict-free is again an admissible set; see, e.g., Caminada (2007, Lemma 1). □

Example 6. *Consider $\mathbb{S}_{E^+} = \{\{a, b\}, \{a, c\}, \{b, c\}\}$ from Example 3. Then $\mathit{ImpliedADM}(\mathbb{S}_{E^+}) = \mathbb{S}_{E^+} \cup \{\{a, b, c\}, \emptyset\}$. Regarding the set $\mathbb{S}' = \{\{b, c\}\}$ which is the set of positive examples satisfied by F_{adm}, we have $\mathit{ImpliedADM}(\mathbb{S}') = \mathbb{S}' \cup \{\emptyset\}$.*

Consider $P'_{adm} = (\{a, b, c\}, E_1^+, E_1^-, \mathit{adm})$ with $E_1^+ = \{(\{a, c\}, 1), (\{b, c\}, 1)\}$ and $E_1^- = \{(\{a\}, 1)\}$. We have $\mathit{ImpliedADM}(\mathbb{S}_{E_1^+}) = \mathbb{S}_{E_1^+} \cup \{\emptyset\}$ and $\mathit{ImpliedADM}(\mathbb{S}_{E_1^+}) \cap \mathbb{S}_{E_1^-} = \emptyset$. Unlike for the conflict-free semantics, this condition for the admissible semantics does not imply the existence of a 0-cost solution AF for P'_{adm}. In fact, a 0-cost solution AF does not exist for P'_{adm}. To see this, consider possible attacks in a candidate for a solution AF. If we assume 0-cost, it follows that all three examples are satisfied by a hypothetical solution AF. In particular $\{a\}$ is not admissible, and $\{a, c\}$ and $\{b, c\}$ are admissible. Since there are no further arguments, it holds that the only two possible attacks are (a, b) and (b, a) (since $\{a, c\}$ and $\{b, c\}$ are conflict-free). Since $\{a\}$ is conflict-free but not admissible, it follows that (b, a) is present in a hypothetical 0-cost solution AF (no attack onto $\{a\}$ directly implies admissibility of this set). Since $\{a, c\}$ is admissible, it holds that either a or c attacks b. Because $\{b, c\}$ is conflict-free, it must hold that (a, b) is present. Since no more attacks are possible, the only candidate solution AF contains the two attacks (a, b) and (b, a). This contradicts satisfaction of negative example $\{a\}$, which is admissible in this hypothetical AF.

However, a 0-cost solution is possible if the set of arguments A includes more arguments. For instance, take $F' = (\{a, b, c, d\}, \{(b, a), (a, b), (c, d), (d, a), (d, d)\})$. We have $\mathit{adm}(F') = \{\emptyset, \{b\}, \{c\}, \{b, c\}, \{a, c\}\}$ and, for $P''_{adm} = (\{a, b, c, d\}, E_1^+, E_1^-, \mathit{adm})$, it holds that F' is an *optimal 0-cost solution AF to P''_{adm}, i.e., $cost(P''_{adm}, F') = 0$. Such "auxiliary" arguments, i.e., arguments not present in the examples, are not always required for 0-cost solutions under the admissible semantics. For instance, given two positive examples $(\{a, c\}, 1)$ and $(\{b, c\}, 1)$, and negative example $(\{a, b, c\}, 1)$, one can synthesize a 0-cost AF with a mutual attack between a and b.*

Similarly as for conflict-free semantics, each 0-cost solution under the admissible semantics implies that for no negative examples e we have $S_e \in \mathit{ImpliedADM}(\mathbb{S}_{E^+})$. For existence of an AF F with $\mathit{ImpliedADM}(\mathbb{S}_{E^+}) = adm(F)$, we make use of results from Dunne et al. (2015) which requires auxiliary arguments, i.e., arguments not present in $\mathbb{S}_{E^+}$. We use the abstract function $\mathit{AuxArgs}(adm, \mathbb{S}_{E^+})$ that returns the number of auxiliary arguments needed to construct F as specified in Dunne et al. (2015, Definitions 13 and 14).[4]

Proposition 4. *Let $P = (A, E^+, E^-, adm)$ be an instance of the AF synthesis problem. Consider the following conditions.*

1. *$\mathit{ImpliedADM}(\mathbb{S}_{E^+}) \cap \mathbb{S}_{E^-} = \emptyset$.*
2. *$|A \setminus (\bigcup_{S \in \mathbb{S}_{E^+}} S)| > \mathit{AuxArgs}(adm, \mathbb{S}_{E^+})$.*

If there is a solution AF F to P with $cost(P, F) = 0$, then condition 1 holds. If both conditions 1 and 2 hold, then there is a solution AF F to P with $cost(P, F) = 0$.

Proof. For the first claim, assume that there is an AF F with $cost(P, F) = 0$. Then $\mathbb{S}_{E^+} \subseteq adm(F)$ and thus $\mathit{ImpliedADM}(\mathbb{S}_{E^+}) \subseteq adm(F)$ by Lemma 3. For the second claim, assume that $\mathit{ImpliedADM}(\mathbb{S}_{E^+}) \cap \mathbb{S}_{E^-} = \emptyset$ and condition 2 holds. By Lemma 18, which we prove in Appendix A, $\mathit{ImpliedADM}(\mathbb{S}_{E^+})$ is conflict-sensitive and contains $\emptyset$. By Dunne et al. (2015, Proposition 8) there is an AF $F' = (A', R')$ s.t. $A' \subseteq A$ and $adm(F') = \mathit{ImpliedADM}(\mathbb{S}_{E^+})$. Define $F = (A, R)$ by extending R' to have self-attacks for each argument in $A \setminus A'$. It follows that $adm(F) = \mathit{ImpliedADM}(\mathbb{S}_{E^+})$. Assuming conditions 1-2, we have $cost(P, F) = 0$. $\square$

We move on to the stable semantics, under which the picture is more complex. Existence of a 0-cost solution for an AF synthesis instance implies that the set of positive examples $\mathbb{S}_{E^+}$ is $\subseteq$-incomparable, does not include $\emptyset$, is disjoint from the negative sets $\mathbb{S}_{E^-}$, and no positive set $S \in \mathbb{S}_{E^+}$ is a proper subset of an implied conflict-free set in $\mathit{ImpliedCF}(\mathbb{S}_{E^+})$. These conditions are quite intuitive, since, e.g., a violation of the last condition violates the fact that stable extensions attack all arguments outside the set.

These conditions imply the existence of 0-cost solutions if a certain number of auxiliary arguments, i.e., arguments not present in $\mathbb{S}_{E^+}$, is available in A. For achieving this result, we use again results from Dunne et al. (2015), providing a construction in this case that utilizes such auxiliary arguments to synthesize the AF. We provide here a rough bound for auxiliary arguments from Dunne et al. (2015, Definition 12). More concretely, we use the function $\mathit{AuxArgs}(stb, \mathbb{S}_{E^+})$ that is equal to the maximum number of stable extensions for any AF with $|\mathbb{S}_{E^+}|$ many arguments (for more details, see Baumann & Strass, 2013, Theorem 1). A proof of Proposition 5 is provided in Appendix A.

4. While this function gives a noticeably large upper bound on the number of sufficiently many auxiliary arguments, we observe in Section 7 empirically that the number of auxiliary arguments needed for obtaining a 0-cost solution can depend drastically on the considered semantics.

Proposition 5. *Let $P = (A, E^+, E^-, stb)$ be an instance of the AF synthesis problem. Consider the following conditions.*

1. *$\forall S \in \mathbb{S}_{E^+}$ we have $S \not\subset S'$ for all $S' \in \mathit{ImpliedCF}(\mathbb{S}_{E^+})$.*
2. *$\emptyset \notin \mathbb{S}_{E^+}$.*
3. *$\mathbb{S}_{E^+} \cap \mathbb{S}_{E^-} = \emptyset$.*
4. *$|A \setminus (\bigcup_{S \in \mathbb{S}_{E^+}} S)| > \mathit{AuxArgs}(stb, \mathbb{S}_{E^+})$.*

If there is a solution AF F to P with $cost(P, F) = 0$, then conditions 1-3 hold. If conditions 1-4 hold, then there is a solution AF F to P with $cost(P, F) = 0$.

Interestingly, the negative examples play a relatively minor role in 0-cost solutions under the stable semantics (see condition 3 of Proposition 5). In contrast to conflict-free or admissible sets, existence of stable extensions does not directly imply the existence of further stable extensions for an unrestricted set of arguments A. This observation also follows from formal results in Dunne et al. (2015, Lemma 2, and Proposition 1 and 7).

Example 7. *The AF F_{stb} from Example 3 has $stb(F_{stb}) = \{\{a, c\}, \{b, c\}\} = \mathbb{S}'$. Conditions 1-3 from Proposition 5 hold for $\mathbb{S}'$. One can synthesize an AF, e.g., F_{stb}, as a 0-cost solution to $P'_{stb} = (\{a, b, c\}, \{(\{a, c\}, 1), (\{b, c\}, 1)\}, \emptyset, stb)$.*

We move on to the preferred semantics. Preferred semantics, in the context of 0-cost solutions for AF synthesis, is similar to stable semantics, and conditions for existence of 0-cost solutions can be defined similarly as well, again based on Dunne et al. (2015). For readability we exclude in the following proposition the special case with an empty set of positive examples. The reason for this is to exclude a special handling in the following proposition for the case of no positive examples and negative examples covering all subsets of the arguments (then, trivially by definition, no 0-cost solutions exist). We later show that an optimal solution to an AF synthesis instance under preferred semantics with no positive examples is straightforward to obtain (see Proposition 7). A proof of the following proposition is provided in Appendix A.

Proposition 6. *Let $P = (A, E^+, E^-, prf)$ be an instance of the AF synthesis problem with E^+ non-empty. Consider the following conditions.*

1. *$\forall S \in \mathbb{S}_{E^+}$ we have $S \not\subset S'$ for all $S' \in \mathit{ImpliedADM}(\mathbb{S}_{E^+})$.*
2. *$\mathbb{S}_{E^+} \cap \mathbb{S}_{E^-} = \emptyset$.*
3. *$|A \setminus (\bigcup_{S \in \mathbb{S}_{E^+}} S)| > \mathit{AuxArgs}(adm, \mathbb{S}_{E^+})$.*

If there is a solution AF F to P with $cost(P, F) = 0$, then conditions 1-2 hold. If conditions 1-3 hold, then there is a solution AF F to P with $cost(P, F) = 0$.

As for stable semantics, presence of preferred extensions does not directly imply the existence of further preferred extensions (Dunne et al., 2015, Lemma 4, and Proposition 2 and 9). That is, in case one knows that the extensions in $\mathbb{S}$ are preferred extensions—but not necessarily all of the preferred extensions—of some AF, then there is always an AF F that has $prf(F) = \mathbb{S}$. This suggests that,

positive E^+		negative E^-
$e_1 = (\{a, b, c\}, 2)$	$e_5 = (\{a', b, c\}, 2)$	$e_8 = (\{a', b', c'\}, 1)$
$e_2 = (\{a, b, c'\}, 2)$	$e_6 = (\{a', b, c'\}, 2)$	
$e_3 = (\{a, b', c\}, 2)$	$e_7 = (\{a', b', c\}, 2)$	
$e_4 = (\{a, b', c'\}, 2)$		

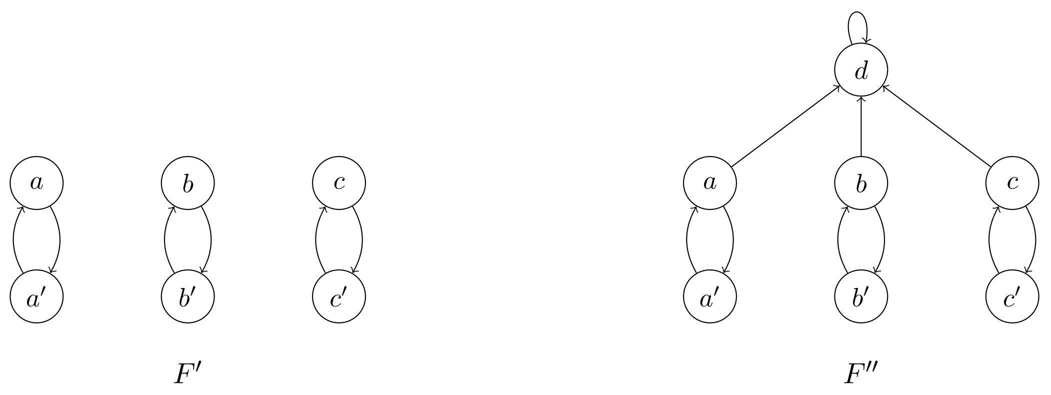

Figure 4: Effect of restrictions on the set of arguments (example from Baumann et al., 2014, 2016)

without knowledge of the AF, there is no direct way of deciding whether a given set of preferred extensions contains all preferred extensions of an AF.

However, with (some) knowledge of the given AF, one can infer from a given set of preferred extensions that there have to be more preferred extensions in the AF. An example is given in Baumann et al. (2014, 2016), where so-called compact realizability is investigated. In particular, these works study the problem of whether there exists an AF $F = (A, R)$ for given set of sets $\mathbb{S}$ s.t. $\sigma(F) = \mathbb{S}$ and $A = \bigcup_{S \in \mathbb{S}} S$ (only arguments from the sets may be used). We formalize this example in the context of AF synthesis next, to illustrate this point.

Example 8 (adapted from Baumann et al., 2014, 2016)**.** *Consider AF synthesis instance $P = (A, E^+, E^-, stb)$. Let $A = \{a, b, c, a', b', c'\}$ and the examples as shown in Figure 4. The unique optimal solution AF F' to P, which satisfies all positive examples but not the negative example e_8 is shown in Figure 4. Any AF $F = (A, R)$ that satisfies all of the positive examples E^+ has to have the same attack structure as F'. Intuitively, if the positive examples are to be satisfied, there can only be attacks between a and a', b and b', and c and c'. Then, for positive examples to be stable, mutual attacks between a and a', b and b', and c and c' have to be present. In this case, also $\{a', b', c'\}$ is stable. Note that $\mathbb{S}_{E^+}$ satisfies conditions 1-3 of Proposition 5 but not condition 4. On the other hand, for the AF synthesis instance to $P' = (A', E^+, E^-, stb)$ with $A' = A \cup \{d\}$, the AF F'' shown in Figure 4 is a 0-cost solution AF to P'. In this case, a single additional argument, not occurring in the positive examples, is enough to enforce $\{a', b', c'\}$ not being stable. For preferred semantics, the same observation can be made, i.e., considering $P'' = (A, E^+, E^-, prf)$, an AF satisfying the positive examples cannot satisfy the negative example as well.*

In the context of an AF synthesis instance $P = (A, E^+, E^-, \sigma)$, the previous example suggests that, under restrictions on the set A and $\sigma \in \{stb, prf\}$, existence of 0-cost solution AFs to P

Table 1: Complexity of AF synthesis

	no restrictions	$E^+ = \emptyset$	$E^- = \emptyset$
Conflict-free	NP-c	trivial	trivial
Admissible	NP-c	trivial	trivial
Stable	NP-c	trivial	NP-c
Grounded	in P	in P	in P
Complete	in NP	in P	in NP
Preferred	NP-hard, in Σ_2^P	in P	NP-hard, in Σ_2^P

require stronger conditions than $\mathbb{S}_{E^+} \cap \mathbb{S}_{E^-} = \emptyset$ (condition 3 in Proposition 5 and condition 2 in Proposition 6). Or, put differently, there is an interconnection between implied stable or preferred extensions and the set of arguments; no or few arguments, not occurring in positive examples, can imply further extensions, and many arguments outside the positive examples can imply less further extensions. Similarly, under admissible semantics, if the set of arguments is restricted, then more admissible sets might be present than returned by $\mathit{ImpliedADM}$ (this function gives a lower bound on which sets must be admissible).

5. Complexity of AF Synthesis

We continue by analyzing the computational complexity of the AF synthesis problem.

Recall that NP is the complexity class comprising of problems that can be solved via a non-deterministic polynomial time algorithm, and P the class of problems that can be solved with a deterministic polynomial time algorithm. The class coNP is the complementary class of NP, that is, coNP contains all problems that have their complementary problem in NP. A decision problem is the complementary of another decision problem if a "yes" instance of the former is a "no" instance of the latter, and vice versa. We will also consider the class Σ_2^P, which consists of problems that can be solved via a non-deterministic polynomial time algorithm that can access an NP oracle that decides a problem in NP in constant time.

As the main results of this section, we show that AF synthesis is NP-complete in the unrestricted case under the conflict-free, admissible, and stable semantics. Furthermore, we show that while restricting either E^+ or E^- to be empty yields fragments of the problem where a trivial AF solves the problem optimally, NP-completeness persists even for $E^- = \emptyset$ under the stable semantics. The results are summarized in Table 1.

We first outline special cases of AF synthesis in which a trivial solution AF is guaranteed to be optimal. In particular, if no positive examples are present, then the complete digraph $F = (A, 2^A \times 2^A)$ satisfies all negative examples e with $S_e \neq \emptyset$ under the conflict-free, admissible, and stable semantics (F has no stable extensions, and the only conflict-free and admissible set is $\emptyset$). If the set of negative examples E^- is empty, then AF synthesis under the conflict-free and admissible semantics is trivial by constructing the AF $F = (A, \emptyset)$ (every subset of A is conflict-free and admissible).

For the complete and preferred semantics, if $E^+ = \emptyset$, one can construct a solution AF with one single complete and preferred extension, chosen either outside the negative examples, or, if all subsets of arguments are specified as a negative example, among the negative examples with minimum weight. Note that polynomial decidability refers to the input of an AF synthesis instance

$P = (A, E^+, E^-, \sigma)$, which may contain an exponential number of examples (exponential w.r.t. the set of arguments A).

Proposition 7. *An optimal solution AF F^* can be computed in polynomial time for an AF synthesis instance $P = (A, E^+, E^-, \sigma)$ if one of the following conditions holds.*

1. *$\sigma \in \{cf, adm, prf, stb, com\}$ and $E^+ = \emptyset$.*
2. *$\sigma \in \{cf, adm\}$ and $E^- = \emptyset$.*

Proof. If the first condition is met, the AF $F = (A, R)$ with $R = (A \times A)$ satisfies $cf(F) = adm(F) = \{\emptyset\}$ and $stb(F) = \emptyset$. For complete and preferred semantics, a direct polynomial time algorithm suffices to find an optimal solution. One can construct an AF that has a unique complete (preferred, grounded) extension that is equal to a given set, by having self attacks for all arguments outside that set. Thus, one has to choose a set that shall be complete (preferred, grounded) such that overall cost is optimal. Since there are no positive examples, by assumption, one has to make sure that either (i) no negative example is dissatisfied, or (ii) in case $\mathbb{S}_{E^-} = 2^A$ (negative examples cover all subsets of arguments) to choose a negative example to dissatisfy with minimum weight. In both cases, one can find a set of arguments to make complete in polynomial time. First sort the negative examples according to their argument sets (ordering, e.g., the sets lexicographically). Then iterate through all subsets of A, in the same order and check whether the current set is in the set of negative examples, and storing the minimum negative example found so far. If the current set is not part of the negative examples, construct an AF with that set as its single complete extension and terminate. Otherwise, after all subsets of A have been iterated, it follows that $\mathbb{S}_{E^-} = 2^A$. Then terminate by constructing an AF with a single complete extension equal to a negative example with minimum weight. This algorithm iterates (after sorting) up to $|E^-| + 1$ sets, implying polynomial time, overall. The cost of the constructed AF is optimal (either 0-cost or, if $\mathbb{S}_{E^-} = 2^A$, with cost of a negative example with minimum weight; if several examples have the same argument set they can be merged to a single example with sum of their weights). If the second condition is met, the AF $F' = (A, \emptyset)$ satisfies $cf(F') = adm(F') = 2^A$. $\square$

The above polynomial-time algorithms are trivial (i.e., a trivial solution solves the problem) except for the case $E^+ = \emptyset$ and $\sigma \in \{com, prf\}$, which we demonstrate via a simple example.

Example 9. *Consider an AF synthesis instance $P = (A, \emptyset, E^-, \sigma)$ with arguments $A = \{a, b, c\}$, the set of negative examples $E^- = \{(\{a\}, 1), (\{a, b, c\}, 5)\}$, and $\sigma \in \{com, prf\}$. Since $\mathbb{S}_{E^-} \neq 2^A$, we pick $\{b\} \notin \mathbb{S}_{E^-}$ and add a self-attack to all arguments not in $\{b\}$, i.e., $R = \{(a, a), (c, c)\}$. This results in the AF $F = (A, R)$ with the single σ-extension $\{b\}$. The cost of F is zero. The set $\{b\}$ can be found in polynomial time: iterate through the given set E^- and check whether some subset of A is not part of E^-.*

We now turn our attention to the NP-hard cases of the AF synthesis problem under the conflict-free, admissible, and stable semantics. Formally, the decision problem corresponding to AF synthesis consists of an AF synthesis instance $P = (A, E^+, E^-, \sigma)$ and an integer $k \geq 0$, and asks whether there is an AF $F = (A, R)$ with $cost(P, F) \leq k$. In all of the following hardness proofs we construct an AF synthesis instance that is polynomial in size w.r.t. the problem instance we reduce from (for instance, the set of examples E^+ and E^- are polynomial in size in regards to the set of arguments A).

positive examples	
satisfied	not satisfied
$(\{x^T, x^F\}, 3)$	$(\{x, x^T\}, 1)$
$(\{x, x^F\}, 1)$	$(\{y, y^F\}, 1)$
$(\{y^T, y^F\}, 3)$	
$(\{y, y^T\}, 1)$	
$(\{x, y\}, 3)$	
$(\{x, y^T\}, 3)$	
$(\{x, y^F\}, 3)$	
$(\{x^T, y\}, 3)$	
$(\{x^T, y^T\}, 3)$	
$(\{x^T, y^F\}, 3)$	
$(\{x^F, y\}, 3)$	
$(\{x^F, y^T\}, 3)$	
$(\{x^F, y^F\}, 3)$	
negative examples	
$(\{x, x^T, x^F\}, 3)$	
$(\{y, y^T, y^F\}, 3)$	
$(\{x, x^T, y, y^F\}, 3)$	

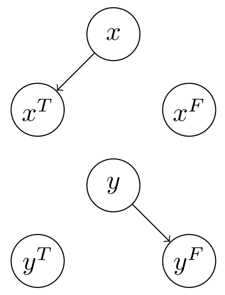

Figure 5: Illustration of reduction in proof of Proposition 8 for formula $(x \lor \neg y)$.

Intuitively, the main source of NP-hardness of the AF synthesis problem for the considered semantics lies in finding an optimal subset of examples from which to synthesize an AF. We start with the conflict-free semantics and prove NP-hardness by a reduction from the Boolean satisfiability problem. For intuition on the reduction, “choosing” a truth assignment can be simulated by a set of positive and negative examples. Assume that x is a Boolean variable, then constructing three arguments, x, x^T, and x^F, and positive examples containing the sets $\{x, x^T\}$, $\{x, x^F\}$, and $\{x^T, x^F\}$, and a negative example $\{x, x^T, x^F\}$ “forces” a truth assignment on x if the positive example $\{x^T, x^F\}$ and the negative $\{x, x^T, x^F\}$ are attached with sufficiently high weights (“high” to be made formal below). For the upcoming reduction, we illustrate this in Figure 5. To see why these examples imply an assignment, first consider the negative example. This example implies that there must be an attack between any of the three arguments. Since $\{x^T, x^F\}$ has a high weight, such an attack does not occur between the arguments x^T and x^F. Thus, there is a conflict in the sets $\{x, x^T\}$ or $\{x, x^F\}$. If an attack is synthesized between x and x^T, then a corresponding truth value assignment would assign x to true, and, otherwise, an attack is synthesized between x and x^F, implying an assignment of x to false in this simulation. Ensuring that not both sets are conflicting (assigning x to both true and false), and incorporation of satisfaction of a given Boolean formula can be achieved via further examples and setting an appropriate k. For instance, satisfaction of clauses of a Boolean formula is simulated by ensuring that attacks have to be present via negative examples.

Proposition 8. *AF synthesis is* NP*-complete under the conflict-free semantics.*

Proof. For an AF synthesis instance $P = (A, E^+, E^-, cf)$ membership in NP follows from a guess of an AF $F = (A, R)$, since $cost(P, F)$ can be computed in polynomial time.

For hardness, we provide a reduction from the satisfiability problem of conjunctive normal form (CNF) Boolean formulas. Let ϕ be a propositional formula in 3-CNF over variables $X = \{x_1, \ldots, x_n\}$, $|X| = n$ and set of clauses C. We define the sets

$$\begin{aligned}
E^+ = &\{(\{x_i, x_i^T\}, 1) \mid x_i \in X\} \cup & (1)\\
&\{(\{x_i, x_i^F\}, 1) \mid x_i \in X\} \cup & (2)\\
&\{(\{x_i^T, x_i^F\}, n+1) \mid x_i \in X\} \cup & (3)\\
&\{(\{y, z\}, n+1) \mid x_i, x_j \in X, i \neq j, y \in \{x_i, x_i^T, x_i^F\}, z \in \{x_j, x_j^T, x_j^F\}\}, & (4)\\
E^- = &\{(\{x_i, x_i^T, x_i^F\}, n+1) \mid x_i \in X\} \cup & (5)\\
&\{(\{x_i, x_i^T \mid x_i \in c\} \cup \{x_i, x_i^F \mid \neg x_i \in c\}, n+1) \mid c \in C\}, & (6)
\end{aligned}$$

and further the set of arguments as $A = \{x_i, x_i^T, x_i^F \mid x_i \in X\}$. Now let $P = (A, E^+, E^-, cf)$ be the constructed instance for AF synthesis with bound $k = n$. This AF synthesis instance can be computed in polynomial time and is of polynomial size w.r.t. the formula ϕ. Intuitively, one cannot satisfy all examples of forms (1), (2), (3), and (5) simultaneously, and due to the chosen weights and cost limit, one has to violate either (1) or (2) for a given $x_i \in X$, thus "choosing" a truth assignment over X (true if and only if an attack between x_i and x_i^T is synthesized). We now claim that there exists an AF $F = (A, R)$ with $cost(P, F) \leq n$ iff ϕ is satisfiable.

"Only-if" direction: Assume that $F = (A, R)$ has $cost(P, F) \leq n$. Then all examples with weight $n + 1$ are satisfied by F. It is immediate that for each $x_i \in X$ we have either $\{x_i, x_i^T\} \in cf(F)$ or $\{x_i, x_i^F\} \in cf(F)$ but not both. If both are conflict-free, then, since $\{x_i^T, x_i^F\}$ is conflict-free (weight is higher than n), it holds that $\{x_i, x_i^T, x_i^F\}$ is conflict-free, as well. This, however, contradicts that the cost of F is lower than $n + 1$. If neither $\{x_i, x_i^T\}$ nor $\{x_i, x_i^F\}$ is conflict free, the two examples incur a cost of 2, which would be a contradiction with the cost of F being at most n, since for each x_i at least one of the examples is not satisfied, incurring a cost of at least $n - 1$. This straightforwardly defines a truth assignment $\tau(x_i) = 0$ iff $\{x_i, x_i^T\} \in cf(F)$ and $\tau(x_i) = 1$ otherwise. Suppose τ does not satisfy ϕ. Then there exists a $c \in C$ s.t. $\tau \not\models c$ and τ does not satisfy any literal l in c.

$$\begin{aligned}
&\quad \tau(l) = 0, \forall l \in c\\
&\text{iff } \forall l \in c\\
&\qquad l = x_i \text{ implies } \tau(x_i) = 0 \text{ and}\\
&\qquad l = \neg x_i \text{ implies } \tau(x_i) = 1\\
&\text{iff } \forall l \in c\\
&\qquad l = x_i \text{ implies } \{x_i, x_i^T\} \in cf(F) \text{ and}\\
&\qquad l = \neg x_i \text{ implies } \{x_i, x_i^F\} \in cf(F)\\
&\text{iff } \{x_i, x_i^T \mid x_i \in c\} \cup \{x_i, x_i^F \mid \neg x_i \in c\} \in cf(F)\ (*)\\
\text{only-if }& cost(P, F) \geq n + 1
\end{aligned}$$

Conclusion $(*)$ follows from (4): for each $x_i, x_j \in X$ with $x_i \neq x_j$ there are no attacks between sets $\{x_i, x_i^T, x_i^F\}$ and $\{x_j, x_j^T, x_j^F\}$.

"If" direction: Assume that ϕ is satisfiable. Construct an AF $F = (A, R)$ with $R = \{(x_i, x_i^T) \mid \tau(x_i) = 1\} \cup \{(x_i, x_i^F) \mid \tau(x_i) = 0\}$. It is immediate that F satisfies all non-unit-weighted examples except for (6), which follows from a similar consideration as in the only-if direction. Finally, the cost of F is n (exactly one unit-weighted example per Boolean variable is not satisfied). We illustrate a solution AF with cost less than $n + 1$ and the satisfaction of examples for a simple satisfiable formula in Figure 5. □

The reduction we use for establishing NP-hardness under the admissible semantics follows essentially the same idea.

Proposition 9. *AF synthesis is* NP-*complete under the admissible semantics.*

Proof. NP-completeness for admissible semantics follows the same reasoning as the proof of Proposition 8, i.e., the same reasoning for membership in NP can be applied, and the same hardness reduction, with the following changes.

"Only-if" direction: the same unit-weighted examples are mutually exclusive for a solution AF as in the conflict-free case. If both unit-weighted examples for a Boolean variable are satisfied, then the corresponding negative example of form 5 is not satisfied (the set of arguments is the conflict-free union of three admissible sets). Note that any solution AF to the AF synthesis instance with cost at most n may contain attacks only between either x_i and x_i^T or x_i and x_i^F. Further, each of these attacks is bidirectional: in case there is an attack between x_i and x_i^T, then both $\{x_i, x_i^F\}$ and $\{x_i^T, x_i^F\}$ are admissible sets, implying that x_i and x_i^T have a mutual attack. In this case each conflict-free set is admissible, and thus the same reasoning as in the remainder of the only-if direction of the proof of Proposition 8 can be applied for this proof.

For the "if" direction, a solution AF with cost below the bound can be synthesized from a satisfying truth assignment by the same construction as in the proof of Proposition 8, except that each attack has to be bidirectional (see also above for the only-if direction). □

For the stable semantics, we establish NP-completeness as well; however, surprisingly, in contrast to the conflict-free and admissible semantics, AF synthesis under the stable semantics is NP-complete even when E^- is empty. The reduction is technically more involved, which is why we present the proof in Appendix A. Intuitively, presence of attacks can be simulated via arguments outside the set of a positive example, since if the set is stable, it has to attack all arguments outside the set. This is also a reason why hardness holds even if E^- is empty.

Proposition 10. *AF synthesis is* NP-*complete for stable semantics, even if the set of negative examples is empty.*

For preferred semantics, the same hardness proof as for stable semantics can be used to show NP-hardness. We also establish membership in the class Σ_2^P.

Proposition 11. *AF synthesis is in* Σ_2^P *and* NP-*hard for preferred semantics. Hardness holds even if the set of negative examples is empty.*

Finally we observe that AF synthesis under the grounded semantics is computable in polynomial time, since exactly one grounded extension is present in an AF.

Proposition 12. *Let* $P = (A, E^+, E^-, grd)$ *be an instance of the AF synthesis problem. An optimal solution AF* F^* *to* P *can be constructed in polynomial time.*

Proof. Any solution AF F for a given instance $P = (A, E^+, E^-, grd)$ has exactly one grounded extension. For a given set of arguments E, an AF F with $grd(F) = \{E\}$ can be constructed in polynomial time (all arguments outside of E self-attacking). To find one with minimum cost, one can iterate in polynomial time through all candidate AFs.

1. Candidate AF F with its grounded extension E among the positive examples, i.e. $e \in E^+$ with $S_e = E$. The cost of F is equal to the sum of all weights of $E^+ \setminus \{e\}$ plus $w_{e'}$ if there is a negative example e' with $S_{e'} = E$. Iterating over all positive examples yields one with minimum cost from these candidates.

2. Candidate AF F with its grounded extension E not among the positive examples, i.e., $E \notin \mathbb{S}_{E^+}$. The cost of F is is equal to the sum of all weights of E^+ plus $w_{e'}$ if there is a negative example e' with $S_{e'} = E$. If there is a set of arguments neither among the positive examples nor the negative examples, i.e., $2^A \setminus (\mathbb{S}_{E^+} \cup \mathbb{S}_{E^-}) \neq \emptyset$, then an AF with minimum cost, over the candidates, is equal to the sum of all weights of E^+. Otherwise, if $2^A \setminus (\mathbb{S}_{E^+} \cup \mathbb{S}_{E^-}) = \emptyset$, iterating over all negative examples yields one with minimum cost over all the considered candidates.

By considering both alternatives above, a minimum cost AF F for P can be found. Running time is polynomial, since we iterate over examples from the input. $\square$

Example 10. *Consider an AF synthesis instance $P = (A, E^+, E^-, grd)$ with $A = \{a, b, c\}$, $E^+ = \{(\{a, b\}, 1), (\{a, c\}, 1), (\{b, c\}, 5)\}$, and $E^- = \{(\{a\}, 1), (\{a, b, c\}, 5)\}$. We iterate through the candidate AFs. For each positive example $e \in E^+$, we construct an AF by adding a self-attack to all arguments outside S_e. This results in three AFs, and the one with minimum cost is $F = (A, R)$ with $R = \{(a, a)\}$, satisfying the maximum-weight positive example $(\{b, c\}, 5)$. The cost of F is 2, since the two other positive examples are not satisfied, and all negative examples are satisfied. It remains to consider the candidate AF with the grounded extension not among the positive examples. However, the cost of such an AF is at least the sum of the weights of positive examples, which is strictly larger than the cost of F. Hence we have arrived at an optimal solution with F.*

Finally, we note that AF synthesis under the complete semantics is in NP without restrictions on the type of examples. This is witnessed by the polynomial-time (Max)SAT encoding presented in Section 6.

Proposition 13. *AF synthesis is in* NP *for complete semantics.*

However, the question of whether NP-hardness can be established remains at present open. In particular, the NP-hardness reductions presented here do not appear to be directly adaptable to the case of the complete semantics. In the empirical evaluation presented in Section 7, we observe that at least empirically AF synthesis under complete is not easier than for the semantics for which we established NP-completeness.

6. Constraint-Based Synthesis of AFs

We continue by presenting declarative encodings for AF synthesis under several different argumentation semantics. We provide encodings both in answer-set programming (using the built-in optimization constructs) and as maximum satisfiability. The success of ASP and SAT-based approaches to acceptance and enumeration problems over AFs motivates our choice of declarative languages.

6.1 MaxSAT-based AF Synthesis for NP Fragments

We start with presenting MaxSAT encodings of AF synthesis. For background on MaxSAT, recall that for a Boolean variable x, there are two literals, x and $\neg x$. A clause is a disjunction ($\vee$) of literals. A truth assignment τ is a function from variables to true (1) and false (0). Satisfaction is defined as usual. A weighted partial MaxSAT (or simply MaxSAT) instance consists of hard clauses φ_h, soft clauses φ_s, and a weight function w associating to each soft clause $C \in \varphi_s$ a positive weight $w(C)$. An assignment τ is a solution to a MaxSAT instance $(\varphi_h, \varphi_s, w)$ if τ satisfies φ_h. The cost of τ, $c(\tau)$, is the sum of weights of the soft clauses not satisfied by τ. A solution τ to MaxSAT instance φ is optimal if $c(\tau) \leq c(\tau')$ for any solution τ' to φ.

Let $P = (A, E^+, E^-, \sigma)$ be an AF synthesis instance with $A = \{a_1, \dots, a_n\}$ the set of arguments, E^+ the set of positive examples, E^- the set of negative examples, and a semantics $\sigma \in \{\mathit{cf}, \mathit{adm}, \mathit{stb}, \mathit{com}\}$. In order to synthesize an optimal solution AF $F = (A, R)$ for P, we declare propositional variables Ext_e for each $e \in E^+ \cup E^-$, and $r_{a,b}$ for each $a, b \in A$. The intended meaning of these variables is as follows: $\mathrm{Ext}_e = 1$ indicates $S_e \in \sigma(F)$, and $r_{a,b} = 1$ indicates $(a, b) \in R$. The hard clauses are of the form

$$\bigwedge_{e \in E^+} (\mathrm{Ext}_e \rightarrow \varphi_\sigma(S_e)) \wedge \bigwedge_{e \in E^-} (\neg\mathrm{Ext}_e \rightarrow \neg\varphi_\sigma(S_e))$$

where $\varphi_\sigma(S_e)$ encodes the fact that S_e is a σ-extension. In other words, for conflict-free sets we have the conjunction of negative literals

$$\varphi_{\mathit{cf}}(S_e) = \bigwedge_{a,b \in S_e} \neg r_{a,b},$$

stating that no attacks should occur between arguments in the set S_e of the example e. Admissible sets are encoded by the formula

$$\varphi_{\mathit{adm}}(S_e) = \varphi_{\mathit{cf}}(S_e) \wedge \bigwedge_{a \in S_e} \bigwedge_{b \in A \setminus S_e} \left(r_{b,a} \rightarrow \bigvee_{c \in S_e} r_{c,b} \right),$$

that is, S_e is conflict-free, and any attack on an argument in S_e implies a defending attack on the attacking argument, i.e., every argument in S_e is defended by S_e. Likewise, if S_e is a stable extension, it is conflict-free and its range is the whole set of arguments, which we encode as

$$\varphi_{\mathit{stb}}(S_e) = \varphi_{\mathit{cf}}(S_e) \wedge \bigwedge_{a \in A \setminus S_e} \left(\bigvee_{b \in S_e} r_{b,a} \right).$$

Finally, our encoding for the complete semantics is

$$\varphi_{\mathit{com}}(S_e) = \varphi_{\mathit{adm}}(S_e) \wedge \bigwedge_{a \in A \setminus S_e} \left(\bigvee_{b \in A} \left(r_{b,a} \wedge \bigwedge_{c \in S_e} \neg r_{c,b} \right) \right),$$

ensuring that all arguments outside S_e are not defended by S_e.

The soft clauses, on the other hand, encode the objective function of AF synthesis under minimization. For each $e \in E^+$ we have a soft clause Ext_e, and for each $e \in E^-$ a soft clause $\neg\mathrm{Ext}_e$, with corresponding weights. An optimal solution to an AF synthesis instance is directly extracted from an optimal solution τ to the MaxSAT instance by including (a, b) to the attack structure iff $\tau(r_{a,b}) = 1$.

6.2 MaxSAT-based CEGAR for AF Synthesis under Preferred Semantics

We continue by providing a MaxSAT-based algorithm for AF synthesis under the preferred semantics, instantiating a general approach called counterexample-guided abstraction refinement (CEGAR) (Clarke, Grumberg, Jha, Lu, & Veith, 2003; Clarke, Gupta, & Strichman, 2004). The algorithm (see Algorithm 1) starts with an NP-abstraction, which overapproximates the problem at hand and is in this case solved via MaxSAT. We use a SAT solver to check whether the solution to the abstraction is actually a solution to the original problem by asking for a counterexample. If no such counterexample exists, we have found an optimal solution, and otherwise, we refine the NP-abstraction by adding constraints that rule out the given counterexample, and iteratively call MaxSAT on the refined abstraction.

For AF synthesis under the preferred semantics, an NP-abstraction is formed by considering the complete semantics for positive examples, since AF synthesis under complete is solvable via a single MaxSAT call. For negative examples, we can simply encode the property 'not preferred' in NP via a disjunction of 'not admissible' and 'exists admissible superset'. To this end, we introduce fresh Boolean variables Superset_e, interpreting $\text{Superset}_e = 1$ iff S_e has an admissible superset. Now, the soft clauses $\textsc{SoftClauses}(E^+, E^-)$ are almost the same as presented in the previous section for NP-fragments. For each positive example $e \in E^+$ we add a unit soft clause Ext_e, and for each negative $e \in E^-$ a binary soft clause $\neg\text{Ext}_e \vee \text{Superset}_e$, with weights corresponding to the weights of the examples.[5]

Similarly as for the NP fragments, we encode the intended meaning of the Ext_e and the Superset_e variables as hard clauses. In order to accomplish this for the Superset_e variables, for each $a \in A$ and $e \in E^-$ we define a Boolean variable x_a^e with the interpretation $x_a^e = 1$ iff the argument a is included in an admissible superset of S_e. Hence, the hard clauses of the MaxSAT encoding of this abstraction are

$$\begin{aligned}\textsc{Abstraction}(E^+, E^-) = & \bigwedge_{e \in E^+} (\text{Ext}_e \rightarrow \varphi_{com}(S_e)) \wedge \bigwedge_{e \in E^-} (\neg\text{Ext}_e \rightarrow \neg\varphi_{adm}(S_e)) \\ & \wedge \bigwedge_{e \in E^-} (\text{Superset}_e \rightarrow \psi(S_e)),\end{aligned}$$

where $\varphi_{adm}(S_e)$ and $\varphi_{com}(S_e)$ are defined as in the NP fragments, and the formula

$$\psi(S_e) = \bigwedge_{a \in S_e} x_a^e \wedge \bigvee_{a \in A \setminus S_e} x_a^e \wedge \bigwedge_{a,b \in A} (r_{a,b} \rightarrow (\neg x_a^e \vee \neg x_b^e)) \wedge \bigwedge_{a,b \in A} \left((x_a^e \wedge r_{b,a}) \rightarrow \bigvee_{c \in A} (x_c^e \wedge r_{c,b}) \right)$$

expresses an admissible superset of S_e. In addition, we observe that if there exist two positive examples $e, e' \in E^+$ with $S_e \subset S_{e'}$, either e or e' is not satisfied under preferred semantics due to the subset-maximality property of the preferred semantics. Hence we also include the binary hard clauses

$$\textsc{NoSubset}(E^+) = \bigwedge_{e,e' \in E^+, S_e \subset S_{e'}} (\neg\text{Ext}_e \vee \neg\text{Ext}_{e'})$$

to the MaxSAT encoding.

5. We note that one presumably cannot similarly encode 'is preferred' for a positive example, since one would have to consider the admissibility of all possible supersets of the example, unlike for a negative example, the satisfaction of which is implied by a single admissible superset.

Algorithm 1 CEGAR-based AF synthesis on input $(A, E^+, E^-, \sigma = prf)$.

```
1: $\varphi_h \leftarrow \textsc{Abstraction}(E^+, E^-) \wedge \textsc{NoSubset}(E^+)$
2: $\varphi_s \leftarrow \textsc{SoftClauses}(E^+, E^-)$
3: **while** true **do**
4:     $\tau \leftarrow \textsc{MaxSAT}(\varphi_h, \varphi_s)$
5:     $optimal \leftarrow true$
6:     **for** $e \in E^+, e$ is satisfied **do**
7:         $result \leftarrow \text{SAT}(\textsc{Counterexample}(e, \tau))$
8:         **if** $result = satisfiable$ **then**
9:             $\varphi_h \leftarrow \varphi_h \wedge \textsc{Refine}(e, \tau)$
10:            $optimal \leftarrow false$
11:            **break**
12:    **if** $optimal = true$ **then** return $F_\tau$
```

Next, we enter the main loop of the CEGAR algorithm, starting with a MaxSAT call. From an optimal truth assignment τ we construct the candidate solution AF $F_\tau = (A, R_\tau)$ with

$$R_\tau = \{(a, b) \mid a, b \in A, \tau(r_{a,b}) = 1\}.$$

Now we need to check whether F_τ is a valid solution under preferred semantics, which is accomplished by considering all positive examples, since negative examples are taken care of in the MaxSAT call. If a positive example $e \in E^+$ is not satisfied, we know that $S_e \notin com(F_\tau)$, and hence $S_e \notin prf(F_\tau)$. However, if $e \in E^+$ is satisfied ($S_e \in com(F_\tau)$), we still need to check whether $S_e \in prf(F_\tau)$.

Therefore, we loop through all satisfied positive examples e, and check using a SAT solver whether $S_e \in prf(F_\tau)$ by asking it for a counterexample, which in this case is a complete extension which is a strict superset of S_e. For this SAT solver call, complete semantics are encoded following Besnard and Doutre (2004). For each $a \in A$, we define a Boolean variable x_a with the interpretation $x_a = 1$ iff a is included in a complete extension of the candidate AF F_τ. The complete semantics can now be expressed by the Boolean formula

$$\textsc{Extension}(F_\tau, com) = \bigwedge_{(a,b)\in R_\tau} (\neg x_a \vee \neg x_b) \wedge \bigwedge_{(b,a)\in R_\tau} \left(x_a \rightarrow \left(\bigvee_{(c,b)\in R_\tau} x_c \right) \right)$$
$$\wedge \bigwedge_{a\in A} \left(\left(\bigwedge_{(b,a)\in R_\tau} \left(\bigvee_{(c,b)\in R_\tau} x_c \right) \right) \rightarrow x_a \right).$$

The satisfying truth assignments to the formula above correspond exactly to the complete extensions of the AF F_τ. Finally, we need to check whether there is a strict superset of S_e, expressed by

$$\textsc{Superset}(S_e) = \bigwedge_{a\in S_e} x_a \wedge \bigvee_{a\in A\setminus S_e} x_a.$$

Now, if there exists a superset of S_e which is also a complete extension, the formula

$$\textsc{Counterexample}(e, \tau) = \textsc{Extension}(F_\tau, com) \wedge \textsc{Superset}(S_e)$$

is satisfiable, proving that $S_e \notin \mathit{prf}(F_\tau)$. If this is the case, we have not arrived at a valid solution, since e is not satisfied under preferred semantics, although it is satisfied under complete semantics. In this case, we add the refinement clause

$$\text{REFINE}(e,\tau) = \text{Ext}_e \rightarrow \left(\bigvee_{(a,b)\in R_\tau} \neg r_{a,b} \vee \bigvee_{(a,b)\notin R_\tau} r_{a,b} \right),$$

which states that if we want to satisfy the example, we need a different attack structure. When for each satisfied example $e \in E^+$ it holds that $S_e \in \mathit{prf}(F_\tau)$, we have arrived at an optimal solution.

6.3 ASP-Based AF Synthesis

In this section we describe answer-set programming (ASP) (Brewka, Eiter, & Truszczynski, 2011) encodings for the AF synthesis problem. We begin by recalling ASP, but also refer the reader to further introductions on this topic (e.g. Eiter, Ianni, & Krennwallner, 2009; Gebser, Kaminski, Kaufmann, & Schaub, 2012).

We fix a countable set U of *constants*. An atom is an expression $p(t_1,\dots,t_n)$, where p is a predicate of arity $n \geq 0$ and each term t_i is either a variable or an element from U. An atom is *ground* if it is free of variables. BU denotes the set of all ground atoms over U. A (disjunctive) rule r is of the form

$$a_1 \mid \cdots \mid a_n \leftarrow b_1,\dots,b_k,\ \mathit{not}\, b_{k+1},\dots,\ \mathit{not}\, b_m. \tag{7}$$

with $n \geq 0$, $m \geq k \geq 0$, $n+m > 0$, where $a_1,\dots,a_n,b_1,\dots,b_m$ are atoms, and "not" stands for default negation. The head of r is the set $\mathit{head}(r) = \{a_1,\dots,a_n\}$ and the body of r is $\mathit{body}(r) = \{b_1,\dots,b_k,\ \mathit{not}\, b_{k+1},\dots,\ \mathit{not}\, b_m\}$. Furthermore, $\mathit{body}^+(r) = \{b_1,\dots,b_k\}$ and $\mathit{body}^-(r) = \{b_{k+1},\dots,b_m\}$. A rule r is ground if r does not contain variables. A program is a finite set of disjunctive rules. If each rule in a program is ground, we call the program ground.

For any program π, let UP be the set of all constants appearing in π. Define GP as the set of rules $r\sigma$ obtained by applying, to each rule $r \in \pi$, all possible substitutions σ from the variables in r to elements of UP. An interpretation $I \subseteq BU$ satisfies a ground rule r iff $\mathit{head}(r) \cap I \neq \emptyset$ whenever $\mathit{body}^+(r) \subseteq I$ and $\mathit{body}^-(r) \cap I = \emptyset$. Interpretation I satisfies a ground program π, if each $r \in \pi$ is satisfied by I. A non-ground rule r (resp., a program π) is satisfied by an interpretation I iff I satisfies all groundings of r (resp., GP). An interpretation $I \subseteq BU$ is an answer set of π if it is a subset-minimal set satisfying the Gelfond-Lifschitz reduct $\pi^I = \{\mathit{head}(r) \leftarrow \mathit{body}^+(r) \mid I \cap \mathit{body}^-(r) = \emptyset, r \in GP\}$.

In this work we also consider optimization programs (following Calimeri, Faber, Gebser, Ianni, Kaminski, Krennwallner, Leone, Ricca, & Schaub, 2012), in particular programs with weak constraints of the form

$$\Leftarrow b_1,\dots,b_n.[w,t_1,\dots,t_m], \tag{8}$$

with each b_i an atom, each t_j a term, and w an integer. Towards optimal answer sets we define $\mathit{weak}(\pi,I) = \{(w,t_1,\dots,t_m) \mid \Leftarrow b_1,\dots,b_n.[w,t_1,\dots,t_m] \in GP \text{ and } \{b_1,\dots,b_n\} \subseteq I\}$, for an answer set I. In words, in $\mathit{weak}(\pi,I)$ we collect all weak constraints satisfied by I (terms t_i can be used to distinguish different sources with the same weight in the set $\mathit{weak}(\pi,I)$). The cost of I is then $\sum_{(w,t_1,\dots,t_m)\in \mathit{weak}(\pi,I)} w$, i.e., the sum of weights of each satisfied weak constraint. An answer set I is optimal if there is no answer set J that has strictly lower cost than I.

```
% extract which sets are positive (negative)
set(ID,pos) ← pos(ID,X).
set(ID,neg) ← neg(ID,X).
% non−deterministically choose sets to satisfy
choose(ID) ← set(ID,M), not n_choose(ID).
n_choose(ID) ← set(ID,M), not choose(ID).
% weights
 ⇐ n_choose(ID), weight(ID,W). [W,ID]
```

Listing 1: Module π_{base}

```
% non−deterministically guess attack structure
att(X,Y) ← arg(X), arg(Y), not n_att(X,Y).
n_att(X,Y) ← arg(X), arg(Y), not att(X,Y).
```

Listing 2: Module π_{att}

Our ASP encoding of AF synthesis is based on several program modules that we conjoin for the different semantics. We partially base our NP encodings on existing encodings for static problems for AFs from Egly et al. (2010). For preferred semantics, we adapt the recent disjunctive encoding from Gaggl et al. (2015). Let $P = (A, E^+, E^-, \sigma)$ be an AF synthesis instance. We encode arguments A, examples $E^+ = \{e_1, \dots, e_n\}$ and $E^- = \{e_{n+1}, \dots, e_m\}$, weights, and semantics as

$$\begin{aligned}\{\mathbf{arg}(a). \mid a \in A\} \cup \\ \{\mathbf{pos}(i,a). \mid a \in S_{e_i}, e_i \in E^+\} \cup \\ \{\mathbf{neg}(i,a). \mid a \in S_{e_i}, e_i \in E^-\} \cup \\ \{\mathbf{weight}(i,w_{e_i}). \mid e_i \in E^+ \cup E^-\} \cup \\ \{\mathbf{sem}(i,\sigma). \mid e_i \in E^+ \cup E^-\}.\end{aligned}$$

We denote this set of facts by π_P.

In Listing 1 we encode semantics independent derivations. In particular, we first derive via the **set** predicate for each example its polarity (positive or negative). The next two rules encode a typical ASP “guess”, i.e., we can choose to either include an example or not with the predicates **choose** and **n_choose**. Finally for this listing, we encode via the weak constraint the penalty for not including an example, which is equal to the weight of the examples not chosen.

The ASP rules in Listing 2 encode the guess for an attack structure for which we verify whether an example is satisfied or not.

The subsequent ASP rules encode the constraints for each semantics individually. We start with conflict-free sets in Listing 3. The first rule in this listing encodes a so-called ASP constraint, stating that it cannot be the case that we have chosen to include a positive example which is not satisfied in the guessed attack structure: there is an attack between two arguments in the example. Similarly, the next three rules encode this for the negative examples. More concretely, if we have chosen to include a negative example, and we have found a conflict inside the set, we derive $\mathbf{neg\_ex\_ok}(i)$ for the example e_i. The final rule states that it cannot be the case that we have chosen a negative

```
% positive examples
← choose(ID), sem(ID,cf), pos(ID,X), pos(ID,Y), att(X,Y).
% negative examples
n_cf(ID) ← choose(ID), neg(ID,X), neg(ID,Y), att(X,Y), sem(ID,cf).
neg_ex_ok(ID) ← choose(ID), n_cf(ID), sem(ID,cf), set(ID,neg).
← sem(ID,S), not neg_ex_ok(ID), set(ID,neg), choose(ID).
```

Listing 3: Module π_{cf}

```
% positive examples
sem(ID,cf) ← choose(ID), set(ID,pos), sem(ID,adm).
defeated(ID,X) ← choose(ID), pos(ID,Y), sem(ID,adm), att(Y,X).
← choose(ID), att(Y,X), not defeated(ID,Y), sem(ID,adm), pos(ID,X).
% negative examples
sem(ID,cf) ← choose(ID), set(ID,neg), sem(ID,adm).
defeated(ID,X) ← choose(ID), neg(ID,Y), sem(ID,adm), att(Y,X).
n_defended(ID,X) ← choose(ID), att(Y,X), not defeated(ID,Y), sem(ID,adm), neg(ID,X).
neg_ex_ok(ID) ← choose(ID), n_defended(ID,X), sem(ID,adm), neg(ID,X).
```

Listing 4: Module π_{adm}

example and have not found a conflict. In the atom $\mathbf{sem}(ID, S)$ we use a variable for the semantics (S) in order to re-use this rule for other semantics.[6]

For admissible semantics, we encode the requirements for positive and negative examples by conjoining the rules for conflict-free semantics and the rules in Listing 4. First, in both positive and negative examples, we rely on the corresponding requirements stated for conflict-free sets. For a positive example to be satisfied, we require that each attack from outside the set is counterattacked, encoded here via the second and third rule stating that we defeat each argument outside, and that it cannot be the case that an argument inside is attacked by a non-defeated attacker. That is, in the predicate $\mathbf{defeated}$ we gather all arguments attacked by an example we have chosen to satisfy, and the ASP constraint specifies that it cannot be the case that an argument in the example is attacked but not defended. For the negative examples, we derive $\mathbf{neg\_ex\_ok}(i)$, i.e., that the example is satisfied, when either there is a conflict from the conflict-free module or an attacker is not counter-attacked.

The rules for complete semantics (Listing 5) depend on the module for admissible semantics, and, additionally, state the requirement for a set to be (not) complete. A positive example is satisfied only if all defended arguments are included, specified by the second and third rule. For negative examples, analogously, the example is satisfied if a defended argument is not included.

The encoding for stable semantics, shown in Listing 6, depends on the module for conflict-free sets and states that a positive example is satisfied if we attack all outside arguments (defeat all outside arguments). We encode this here by stating that it cannot be the case that an argument

6. We note that the first two rules for the negative examples in Listing 3 can be rewritten to a single rule (deriving $\mathbf{neg\_ex\_ok}$), but when experimenting with this variant no noticeable performance difference was observed; we thus opt for the current slightly more readable encoding. A similar observation was made for the last two rules for Listing 4. Likewise, there is some redundancy, for an ASP solver, in the last rule of Listing 3, which is re-used for all semantics: one can "hardcode" the semantics instead of variable S, but, again, our experiments did not indicate any significant performance difference.

```
% positive examples
sem(ID,adm) ← choose(ID), set(ID,pos), sem(ID,com).
n_defended(ID,X) ← choose(ID), att(Y,X), not defeated(ID,Y), sem(ID,com), not pos(ID,X).
← choose(ID), not pos(ID,X), arg(X), not n_defended(ID,X), sem(ID,com), set(ID,pos).
% negative examples
sem(ID,adm) ← choose(ID), set(ID,neg), sem(ID,com).
n_defended(ID,X) ← choose(ID), att(Y,X), not defeated(ID,Y), sem(ID,com), not neg(ID,X).
neg_ex_ok(ID) ← choose(ID), not neg(ID,X), arg(X), not n_defended(ID,X), sem(ID,com), set(ID,neg).
```

Listing 5: Module π_{com}

```
% positive examples
sem(ID,cf) ← choose(ID), set(ID,pos), sem(ID,stb).
defeated(ID,Y) ← choose(ID), sem(ID,stb), pos(ID,X), att(X,Y).
← choose(ID), sem(ID,stb), set(ID,pos), arg(X), not pos(ID,X), not defeated(ID,X).
% negative examples
sem(ID,cf) ← choose(ID), set(ID,neg), sem(ID,stb).
defeated(ID,Y) ← choose(ID), sem(ID,stb), neg(ID,X), att(X,Y).
neg_ex_ok(ID) ← choose(ID), arg(X), not neg(ID,X), not defeated(ID,X), sem(ID,stb), set(ID,neg).
```

Listing 6: Module π_{stb}

is, for a chosen positive example, neither in the example nor defeated. Analogously, a negative example is satisfied if it is not conflict-free or there exists an argument outside that is non-attacked (not defeated).

For obtaining an ASP encoding of AF synthesis for preferred semantics, we extend a recently proposed encoding of preferred extensions for an AF by Gaggl et al. (2015). While there are other encodings for (enumerating) preferred extensions for a given AF (Egly et al., 2010; Dvorák, Gaggl, Wallner, & Woltran, 2011), in the 2015 argumentation competition (Thimm & Villata, 2017) a comparison between ASP encodings for preferred semantics showed favourable performance for the encoding of Gaggl et al. (2015).

Our encoding is presented as Listing 7. Encoding AF synthesis for the positive examples follows very closely the encoding of Gaggl et al. (2015) with the exceptions that the "guess" of a set of arguments is removed (this part in our encoding of AF synthesis is in the "guess" of an example). For intuition on this encoding, two techniques have been applied in conjunction with each other in this encoding: utilization of disjunctive ASP rules to solve coNP or Σ_2^P problems in ASP and conditional literals. We explain the core part of these techniques, and their usage for our encoding in sequence.

First, the usage of the **spoil** follows the nowadays standard technique for encoding coNP or Σ_2^P problems in ASP with disjunctive rules; see also Eiter et al. (2009). Briefly, with this technique, we here consider a "second" guess that can be a counterexample to the example set being a preferred extension. This is realized in Line 5 and Line 6: we guess an argument outside the set of arguments in the example, and extend this set whenever that argument is attacked by a (possible) defender. The set constructed via the second guess is named "witness", as its purpose is to witness that the example set is (not) a preferred extension. In this way, another admissible set (possibly overlapping

```
 1  % positive examples
 2  sem(ID,adm) ← choose(ID), set(ID,pos), sem(ID,prf).
 3  out(ID,X) ← arg(X), not pos(ID,X), choose(ID), set(ID,pos), sem(ID,prf).
 4  n_trivial(ID) ← arg(X), not pos(ID,X), choose(ID), set(ID,pos), sem(ID,prf).
 5  witness(ID,X) : out(ID,X) ← n_trivial(ID), choose(ID), set(ID,pos), sem(ID,prf).
 6  spoil(ID) | witness(ID,Z) : att(Z,Y) ← witness(ID,X), att(Y,X), choose(ID), set(ID,pos), sem(ID,prf).
 7  spoil(ID) ← witness(ID,X), witness(ID,Y), att(X,Y), choose(ID), set(ID,pos), sem(ID,prf).
 8  spoil(ID) ← pos(ID,X), witness(ID,Y), att(X,Y), choose(ID), set(ID,pos), sem(ID,prf).
 9  witness(ID,X) ← spoil(ID), arg(X), choose(ID), set(ID,pos), sem(ID,prf).
10  ← n_trivial(ID), not spoil(ID), choose(ID), set(ID,pos), sem(ID,prf).
11  % negative examples
12  sem(ID,adm) ← choose(ID), set(ID,neg), sem(ID,prf).
13  supIn(ID,X) ← choose(ID), neg(ID,X), sem(ID,prf).
14  supIn(ID,X) ← choose(ID), not supOut(ID,X), not neg(ID,X), sem(ID,prf), arg(X), set(ID,neg).
15  supOut(ID,X) ← choose(ID), not supIn(ID,X), not neg(ID,X), sem(ID,prf), arg(X), set(ID,neg).
16  conflictingPRF(ID) ← supIn(ID,X), supIn(ID,Y), att(X,Y), choose(ID), set(ID,neg), sem(ID,prf).
17  defeatedPRF(ID,X) ← supIn(ID,Y), att(Y,X), choose(ID), set(ID,neg), sem(ID,prf).
18  n_defendedPRF(ID,X) ← att(Y,X), not defeatedPRF(ID,Y), choose(ID), set(ID,neg), sem(ID,prf).
19  setNotDefendedPRF(ID) ← supIn(ID,X), n_defendedPRF(ID,X), choose(ID), set(ID,neg),
20                              sem(ID,prf).
21  admissibleSetPRF(ID) ← choose(ID), not conflictingPRF(ID), not setNotDefendedPRF(ID),
22                             set(ID,neg), sem(ID,prf).
23  neg_ex_ok(ID) ← supIn(ID,X), not neg(ID,X), admissibleSetPRF(ID), choose(ID), set(ID,neg),
24                      sem(ID,prf).
```

Listing 7: Module π_{prf}

with the first) is constructed. By standard results in abstract argumentation, if two admissible sets are non-conflicting, their union is also admissible. If such a set exists, then the union is a larger admissible set (by Line 5 at least one argument outside the first guess is added), implying that the example set is not a preferred extension.

By the previous encodings, the example set is forced to be admissible, and thus a second guess is a counterexample if it represents an admissible set that is a strict superset of the example set. If no such counterexample exists, then the example set is a preferred extension (by definition). Whenever **spoil** is derived, the second guess does not correspond to a strictly larger admissible set. This is the case when the guessed set (i) is not defended (Line 6), is not conflict-free (Line 7), or is in conflict with the example set (Line 8). If the example set corresponds to a preferred extension (there is no counterexample), then one wants to avoid the situation that one has many answer sets for an individual preferred extension, since many second guesses correspond to non-counterexamples. To ensure that one has one answer set per preferred extension, and not one per preferred extension and non-counterexample, whenever **spoil** is derived, all possible atoms making up counterexamples are derived, as well. This implies that all the non-counterexamples collapse to one answer set (i.e., each counterexample is "saturated" to the full unique set of atoms). This is done in Line 9. In this line of reasoning, disjunctive rules for generating the second guess are required: otherwise deriving all atoms would not yield a valid answer set (a guess constructed by usage of default negation assumes non-derivability of atoms not guessed).

On the other hand, if a counterexample exists (implying that the example set is not preferred), then in one candidate for an answer set $\mathbf{spoil}$ is not derived. Via a simple ASP constraint (Line 10), this candidate is prevented from being an answer set. By subset-minimality of answer sets (w.r.t. the reduct), this one counterexample prevents all other non-counterexamples to lead to an answer set (non-counterexamples are then larger w.r.t. subset inclusion within the reduct). This implies that there is no answer set for an example set that does not correspond to a subset maximal admissible set (thus implementing that preferred extensions are subset-maximal admissible sets). Finally, Line 3 and Line 4 are auxiliary rules: deriving the complement of the example set and deriving an atom indicating non-trivialness (an example set is deemed trivial if it spans the whole set of arguments; then, in case admissibility holds, the set in question is trivially a preferred extension).

The second technique that was previously employed by Gaggl et al. (2015) on "top" of the technique to encode coNP or Σ_2^P problems is the use of the conditional ":", supported by clingo (Gebser, Kaminski, Kaufmann, Ostrowski, Schaub, & Wanko, 2016). Put simply, this conditional is expanded to a list of literals that satisfy the condition (e.g., to a list of disjunctions on the head). Conditional literals are utilized in Line 5 (to guess only arguments outside the example set) and in Line 6 (to expand the disjunction to $\mathbf{spoil}$ and any potential defender of an argument in the guess). For further details, we refer the reader to Gaggl et al. (2015).

For the negative examples, we check, using the already described program modules, whether admissibility of the example is violated. If this is the case, the example is satisfied. If admissibility holds, the example can only be satisfied in case there exists a superset that is admissible (i.e., the admissible set is not a preferred extension). To verify this, we guess a superset via the predicates $\mathbf{supIn}$ and $\mathbf{supOut}$, and check admissibility of the superset via auxiliary predicates $\mathbf{conflictingPRF}$ and $\mathbf{admissibleSetPRF}$.

Summarizing, the programs for the semantics are π_{cf} (conflict-free), $\pi_{cf} \cup \pi_{adm}$ (admissible), $\pi_{cf} \cup \pi_{adm} \cup \pi_{com}$ (complete), $\pi_{cf} \cup \pi_{stb}$ (stable), and $\pi_{cf} \cup \pi_{adm} \cup \pi_{prf}$ (preferred). An optimal answer set for the program for a semantics, when conjoined with $\pi_{base} \cup \pi_{att} \cup \pi_P$, corresponds to an optimal solution to the AF synthesis P.

7. Experiments

Complementing the theoretical results, we continue by detailing and presenting results from an extensive empirical evaluation of the scalability of the MaxSAT and ASP approaches to AF synthesis under various central AF semantics. For the experiments, we considered two distinct ways of constructing AF synthesis instances. In addition to evaluating the relative performance of different solvers and algorithms for AF synthesis, we also empirically study the impact of allowing additional arguments (outside the set of arguments in the given examples) on the cost of optimal solution AFs.

The implementations and benchmarks used in the evaluation are available online at `http://www.cs.helsinki.fi/group/coreo/afsynth/`.

7.1 Benchmarks

For the empirical evaluation, we used two different approaches to construct AF synthesis instances. The first set of benchmarks was generated based on the benchmark AFs used in the ICCMA'15 competition (Thimm et al., 2016) as follows. We selected all AFs among the benchmarks that have at least five stable extensions. The number of arguments in these 17 AFs ranges from 141 to 964. For each AF, we picked uniformly at random 5 positive examples from the set of extensions.

To obtain negative examples, we selected $10, 20, \dots, 150$ subsets of $\bigcup \mathbb{S}_{E^+}$ uniformly at random, using $p_{\text{arg}} = \frac{\sum_{e \in E^+} |S_e|/|E^+|}{|\bigcup \mathbb{S}_{E^+}|}$ as the probability of including an argument in a negative example. For intuition, this choice of p_{arg} makes the sizes of positive and negative examples approximately the same. Letting $A = \bigcup \mathbb{S}_{E^+}$ resulted in instances containing 54 to 370 arguments. Further, we introduced noise by flipping the role of each example—from positive to negative and vice versa—with a fixed probability $p_{\text{noise}} \in \{0, 0.25, 0.5\}$. Weights were associated to each example by picking uniformly at random integers from the interval $[1, 10]$.

The second set of benchmarks was generated using the following random model. We picked $5, 10, \dots, 80$ positive examples from a set of 100 arguments uniformly at random with probability $p^+_{\text{arg}} = 0.25$. Then $|E^-| = 20, 40, \dots, 200$ negative examples were sampled from the set $A = \bigcup \mathbb{S}_{E^+}$, and each argument was included with probability $p^-_{\text{arg}} = \frac{\sum_{e \in E^+} |S_e|/|E^+|}{|\bigcup \mathbb{S}_{E^+}|}$. Again, each example was assigned as weight a random integer from the interval $[1, 10]$. For each choice of parameters, this procedure was repeated 10 times to obtain a representative set of benchmarks. The instances for preferred semantics were generated following the same random model, using $|A| = 20$, $|E^+| = 5, 10, 15, 20$ and $|E^-| = 10, 20, 30, 40, 50$ as parameters.

7.2 Experiment Setup

The experiments were run on 2.83-GHz Intel Xeon E5440 quad-core machines with 32-GB memory and Debian GNU/Linux 8 using a per-instance timeout of 900 seconds. For the experiments, we used a variety of state-of-the-art MaxSAT solvers to test which type of algorithmic approaches perform best for AF synthesis under different semantics: MaxHS (version 2.9.0) (Davies & Bacchus, 2013) (a SAT-IP hybrid solver based on the implicit hitting set paradigm), as well as the SAT-based core-guided solvers Maxino (version k16) (Alviano, Dodaro, & Ricca, 2015), MSCG (version 2014) (Morgado, Ignatiev, & Marques-Silva, 2015), Open-WBO (version 1.3.1) (Martins, Manquinho, & Lynce, 2014), and WPM3 (version 2015.co) (Ansótegui & Gabàs, 2017). As the (both non-disjunctive and disjunctive) ASP solver, we used Clingo (version 5.2.1) (Gebser et al., 2016) as arguably the most efficient current ASP solver. We also experimented with the commercial IBM CPLEX integer programming solver (version 12.6) on the considered AF synthesis instances via direct translation of MaxSAT to ILP (Ansótegui & Gabàs, 2013), but it proved to produce considerably more timeouts than any of the other solvers, and is therefore excluded from this evaluation. For the presumably harder problem of AF synthesis under preferred semantics, we used the SAT-IP hybrid MaxSAT solver LMHS (MaxSAT evaluation 2016 version) (Saikko, Berg, & Järvisalo, 2016) in the implementation of the abstraction solver in the CEGAR algorithm due to its API which enables incremental use of the solver. Furthermore, we used MiniSAT (Eén & Sörensson, 2004) (version 2.2.0) as the SAT solver within the CEGAR approach.

7.3 Results

We overview the results of the evaluation. We begin with the evaluation of the benchmark generation approach in terms of relevance of the instances produced in terms of non-zero solution costs. Then, we present detailed results on the algorithmic and solver performance on solving AF synthesis instances, first on the NP fragments and then on the presumable second-level problem variant under the preferred semantics.

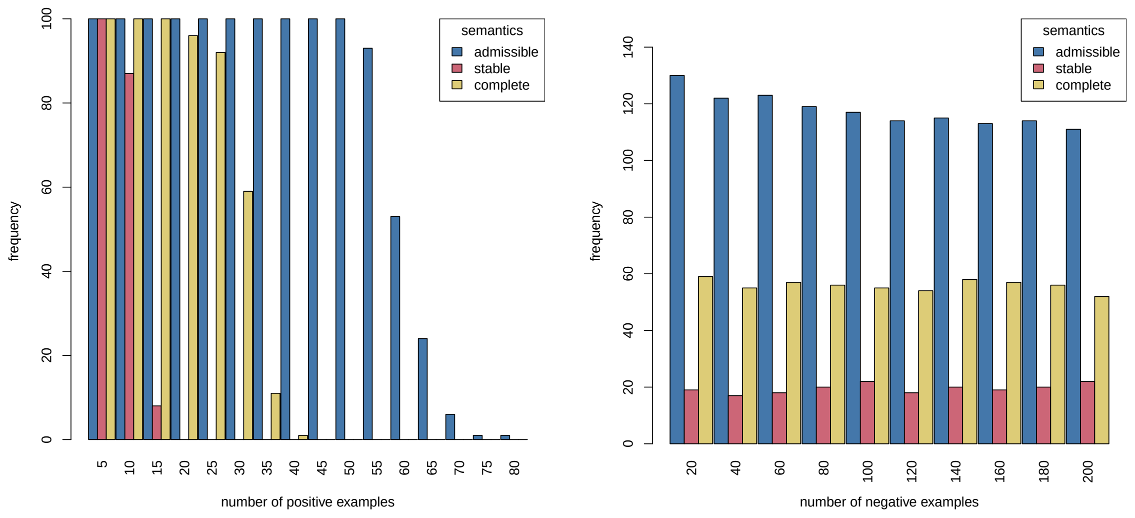

Figure 6: Frequency of 0-cost randomly generated instances with respect to the number of positive (left) and negative (right) examples (over all numbers of negative (left) and positive (right) examples).

7.3.1 OPTIMAL SOLUTION COSTS

For the benchmarks drawn from the random model, we counted the number of 0-cost instances under each semantics (noting that since these are essentially trivial instances, most of them are indeed solved in the timeout limit). In Figure 6 (left) we show the frequency of 0-cost instances for admissible, complete, and stable semantics with respect to the number of positive examples. Increasing this number significantly reduces the frequency of 0-cost instances, which is intuitive: the more positive examples one has, the more conflicting examples can arise. Under stable semantics, 20 positive examples is enough to render all generated instances non-trivial. Under admissible, on the other hand, even at 80 positive examples we still have a few 0-cost instances, although their frequency drops significantly at around 60 examples. The frequency of 0-cost instances under complete semantics lies between admissible and stable, dropping noticeably after 30 positive examples. On the other hand, Figure 6 (right) illustrates the same frequency with respect to the negative examples. Evidently, under all semantics, the number of negative examples does not affect the number of trivial instances as significantly as the number of positive examples.

7.3.2 SOLVER COMPARISON ON NP FRAGMENTS

We now turn the attention to the impact of the MaxSAT solvers used to solve the NP-encodings for AF synthesis under the admissible, complete, and stable semantics. We report on the comparison in terms of median running times as well as so-called "cactus" plots[7], which show the number of

7. The so-called cactus plots are frequently used for comparing the performance of different SAT and MaxSAT solvers in the literature.

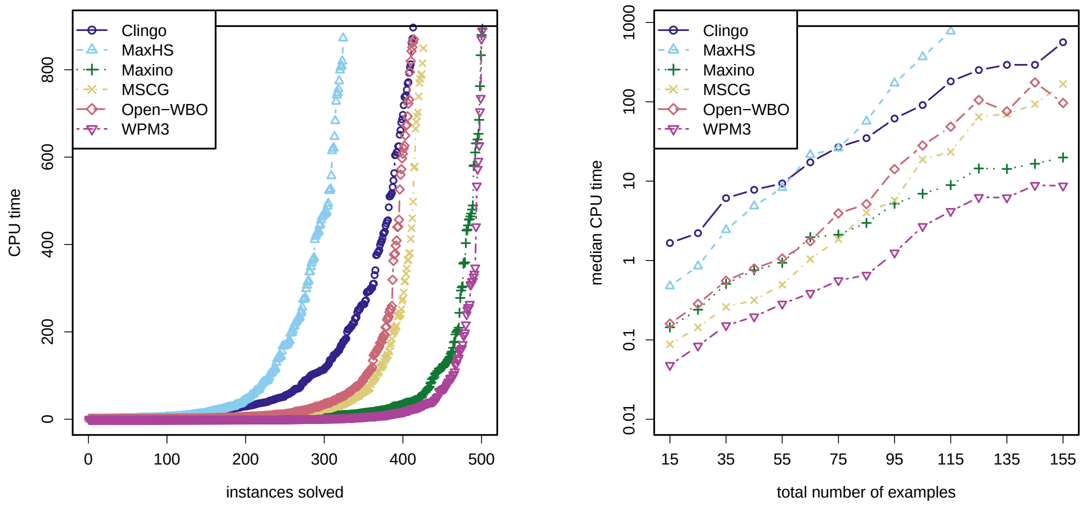

Figure 7: MaxSAT solver comparison on ICCMA instances under the stable semantics.

instances solved by a specific solver (x-axis) under different per-instance time limits (y-axis). The reported median runtimes are over all benchmark instances, while the cactus plots are over instances with non-zero cost to provide a comparison of solver performance on true optimization problems.

We start with the instances generated from ICCMA AFs. Figure 7 summarizes the results for stable semantics. From the cactus plot (left) we observe that the core-guided MaxSAT solvers Maxino and WPM3 clearly solve more of these instances than any other solver considered. The other two core-guided solvers MSCG and Maxino, and the ASP solver Clingo, are essentially on par with respect to the number of instances solved, although Clingo consumes more CPU time on average. The SAT-IP hybrid MaxHS shows weak performance, solving only approximately 300 instances. From the median runtimes (right) we observe that the most efficient solver is indeed WPM3, and additionally, MaxHS surpasses Clingo on smaller instances with less than 75 examples. Finally, we note that under admissible, these instances turned out to be very easy for our approach until running out of memory due to the increasing size of the encoding. Under complete, the size of the encoding becomes at present a bottleneck for scaling up on these benchmarks.

We continue with the second set of benchmarks, generated using the random model. The results for admissible semantics are illustrated in Figure 8. As seen from the left plot, in terms of the number of non-trivial instances solved, the core-guided MaxSAT solvers WPM3, Maxino, and MSCG perform best, with WPM3 solving slightly more instances than the other two. This is also evident from the median runtimes shown on the right. Interestingly, MaxHS and Open-WBO are noticeably slower than the rest of the solvers, although Open-WBO is also core-guided. The ASP solver Clingo performs slightly better than Open-WBO in terms of solving more instances with lower median runtimes for smaller instances.

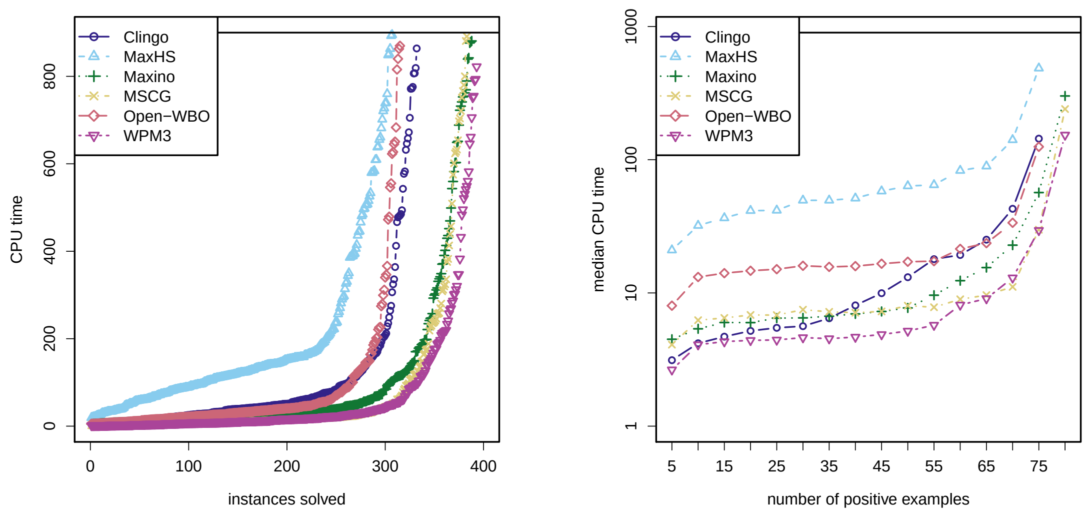

Figure 8: MaxSAT solver comparison on randomly generated instances under the admissible semantics.

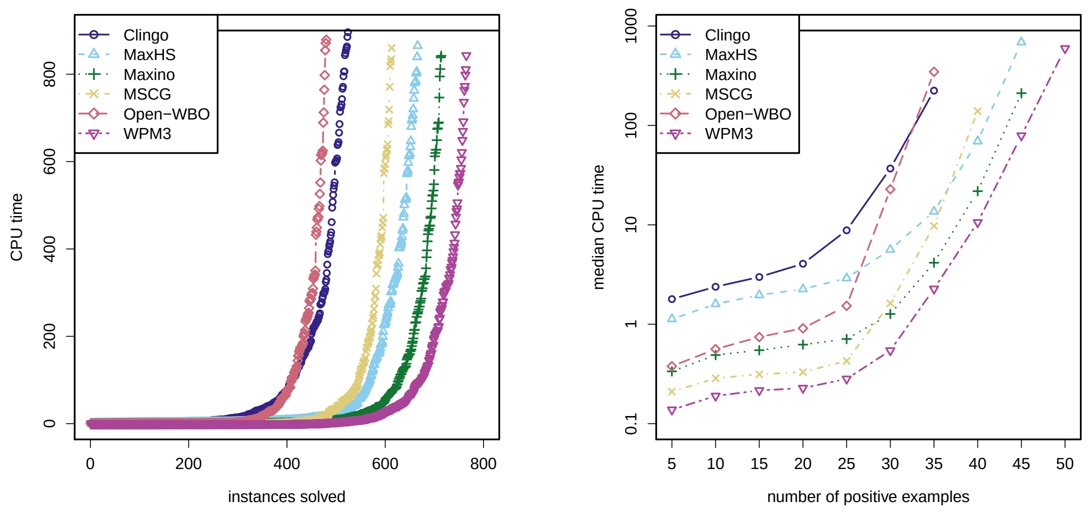

Figure 9: MaxSAT solver comparison on randomly generated instances under the stable semantics.

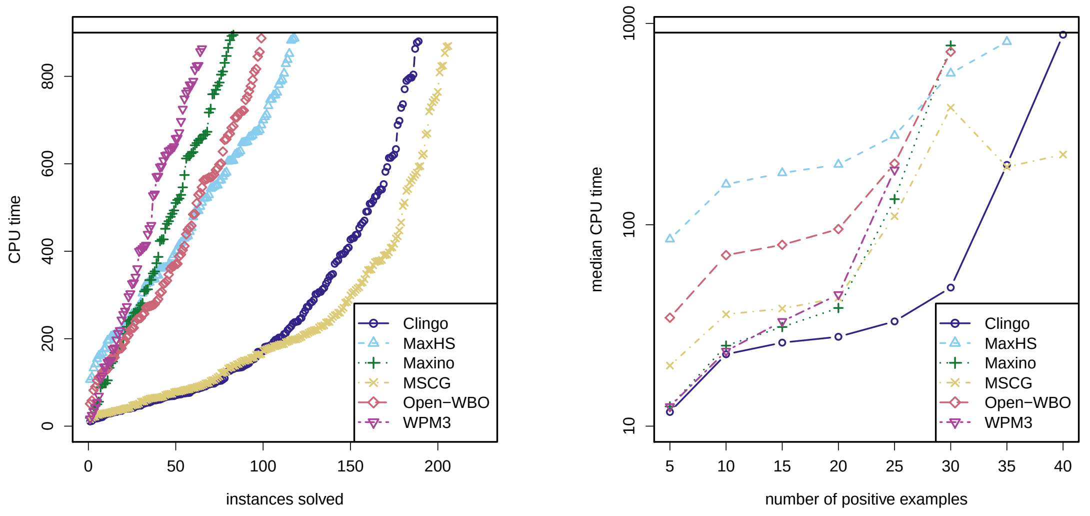

Figure 10: MaxSAT solver comparison on randomly generated instances under the complete semantics.

Regarding the stable semantics, we show the number of instances solved and the median runtimes in Figure 9 on the left and right, respectively. The solvers were unable to solve most of the instances with $|E^+| > 50$, and hence these instances are excluded from this evaluation. We observe that in this case the performance of solvers has a clear ranking from best to worst: WPM3, Maxino, MaxHS, MSCG, Clingo, Open-WBO. However, from the median runtimes we see that MaxHS tends to be slightly slower on smaller instances with less than 40 positive examples. In addition, the performance of Open-WBO drops drastically when the number of positive examples is above 25. In the case of stable semantics, all MaxSAT solvers except Open-WBO clearly outperform the ASP solver Clingo.

Results from the solver comparison under complete semantics are illustrated in Figure 10. All of the solvers produced considerably more timeouts than optimal solutions for $|E^+| > 40$, which is why we do not include these instances in the evaluation. The core-guided MaxSAT solver MSCG solves more nontrivial instances than any other solver, and the ASP solver Clingo is a close second, which is evident from the left plot. We also observe that the other MaxSAT solvers do not perform as well—only the SAT-IP hybrid MaxHS is able to solve over a hundred instances in the timeout limit. The median runtimes on the right reveal that on easier instances with less than 20 positive examples, Maxino and WPM3 are more efficient. An interesting observation is that the median runtime of the top solver MSCG drops at 35 positive examples. We hypothesize that the subpar performance of most MaxSAT solvers on these instances is due to the large size of the MaxSAT encoding for complete semantics.

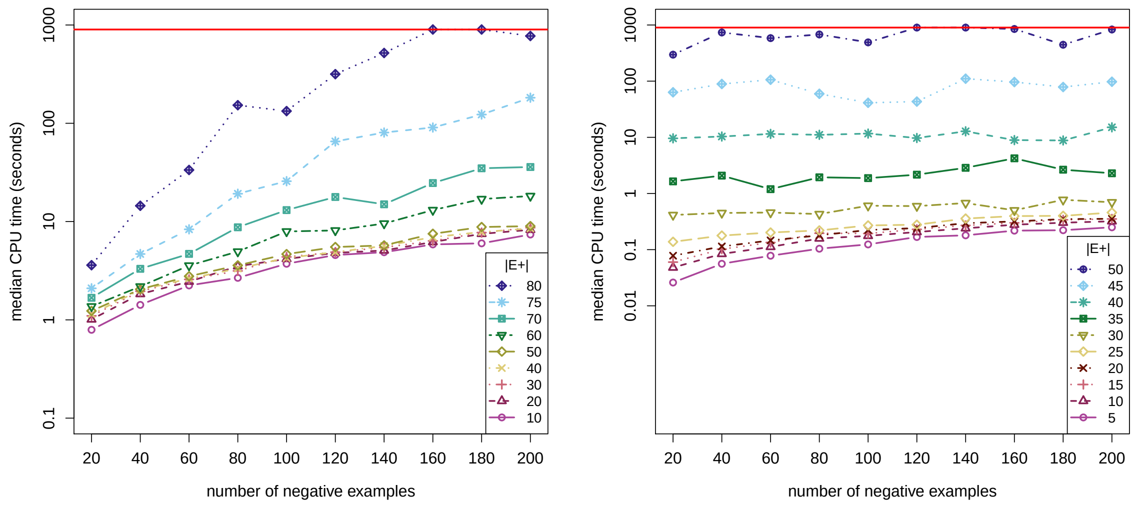

Figure 11: Solver scalability: WPM3 on randomly generated instances under admissible (left) and stable (right) semantics.

7.3.3 SCALABILITY

Scalability results for the randomly generated benchmarks with respect to the number of positive and number of negative examples for the admissible and stable semantics are shown in Figure 11, using the empirically best solver, WPM3. On randomly generated instances under admissible, both $|E^+|$ and $|E^-|$ correlate positively with the runtimes. However, under stable semantics, the number of negative examples does not appear to noticeably affect the runtimes. This is in line with our complexity results—recall that under stable the AF synthesis problem remains NP-complete even without negative examples, unlike under admissible.

Under complete semantics, the empirically best solver turned out to be MSCG. Figure 12 (left) illustrates the scalability with respect to both $|E^+|$ and $|E^-|$. Again, like under admissible, increasing the number of positive examples increases the median runtime of the solver. The same applies to increasing the number of negative examples, with the exception of $|E^-| = 35, 40$, with median runtimes below $|E^-| = 30$. However, we did not notice this kind of behavior with any other solver.

As is evident from the solver comparison, the most efficient solver for ICCMA instances under the stable semantics is WPM3. As seen from Figure 12 (right), every instance can be solved within the timeout limit for up to 115 examples, and the median runtime for 155 examples is under 10 seconds. The noise probability p_{noise} correlates positively with empirical hardness. Recall that the instances were generated by flipping the role of each example. Since we start off with a small number of positive examples and a large number of negative examples, by increasing p_{noise} we are increasing the number of positive examples. We hypothesize this to be the reason for this correlation—recall that AF synthesis under stable remains NP-complete even with no negative examples, and hence positive examples are a source for NP-hardness.

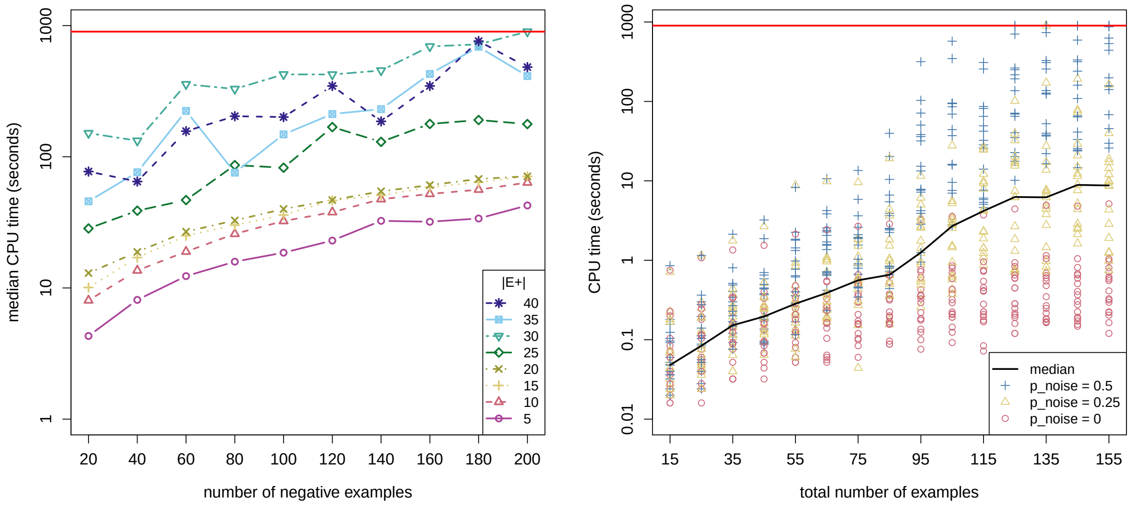

Figure 12: Solver scalability: MSCG on randomly generated instances under complete semantics (left); WPM3 on ICCMA instances under stable semantics (right).

7.3.4 ASP versus CEGAR under Preferred Semantics

For preferred semantics, we compare the performance of the CEGAR algorithm implemented using LMHS as the underlying MaxSAT solver for the base semantics, and Clingo on the disjunctive ASP encoding using the randomly generated instances. The results are shown as a cactus plot in Figure 13. The two approaches show similar performance, with the CEGAR approach being somewhat faster on average, and, on the other hand, Clingo solving one instance more than CEGAR. Although the two approaches are non-trivial, scaling up approaches to AF synthesis under preferred semantics remains currently a challenge: neither of the approach could solve a majority of the benchmark instances even for $|E^+| = 10$ within the enforced per-instance time limit of 900 seconds.

7.3.5 Effect of Additional Arguments on Cost of Optimal Solutions

Finally, we investigate the effect of adding arguments not occurring in the examples on the cost of the instance. For this, we also used the set of benchmarks generated using the random model. Under admissible semantics, even one additional argument in all instances reduces the cost of most instances to zero, as illustrated in Figure 14 (left). With five additional arguments, all examples in all solved instances are satisfied. Under stable semantics, even adding ten additional arguments does not have any effect on the optimal cost for this set of instances, as can be observed from Figure 14 (right). This demonstrates that, at least on the considered benchmark instances, the number of auxiliary arguments required for admissible semantics in order to obtain a 0-cost instance is considerably smaller than for stable semantics. An interesting open question is whether this empirical observa-

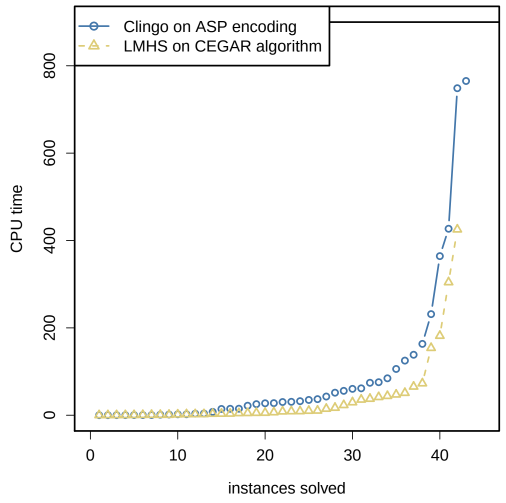

Figure 13: Solver scalability: Clingo and LMHS on randomly generated instances under preferred semantics.

tion could translate into differences between semantics in terms of how many auxiliary arguments are needed to obtain 0-cost solutions.

8. Generalizations and Variants of AF Synthesis

Finally, we explain how the algorithms proposed in this work allow for enforcing additional constraints and preferences on the outcomes of the AF synthesis process, thereby enabling a tighter control on the AFs synthesized. Furthermore, we also shortly discuss further generalizations and variants of AF synthesis, namely, synthesis under multiple semantics and synthesis from symbolically represented examples.

8.1 Enforcing Additional Constraints and Preferences on Outcomes

The declarative methods presented in this work straightforwardly allow for posing additional constraints on the solution AFs of interest, e.g., via MaxSAT we encode such constraints as hard (or soft) clauses. For instance, the presence (resp. absence) of any particular attack (a, b) can be fixed by including the hard clause $(r_{a,b})$ (resp. $(\neg r_{a,b})$). This is particularly useful in the case where a part of the attack structure is certain, for instance obtained by inspecting the logical structure of the arguments, and we do not want to allow arbitrary modifications. Alternatively, one may harden a soft clause (Ext_e) (or $(\neg\text{Ext}_e)$) to ensure that S_e is (or is not) a σ-extension in the solution AF. By hardening all such soft clauses, the AF synthesis problem reduces to realizability, for the given set of arguments. Hence, our methods allow for solving (partial) realizability as well.

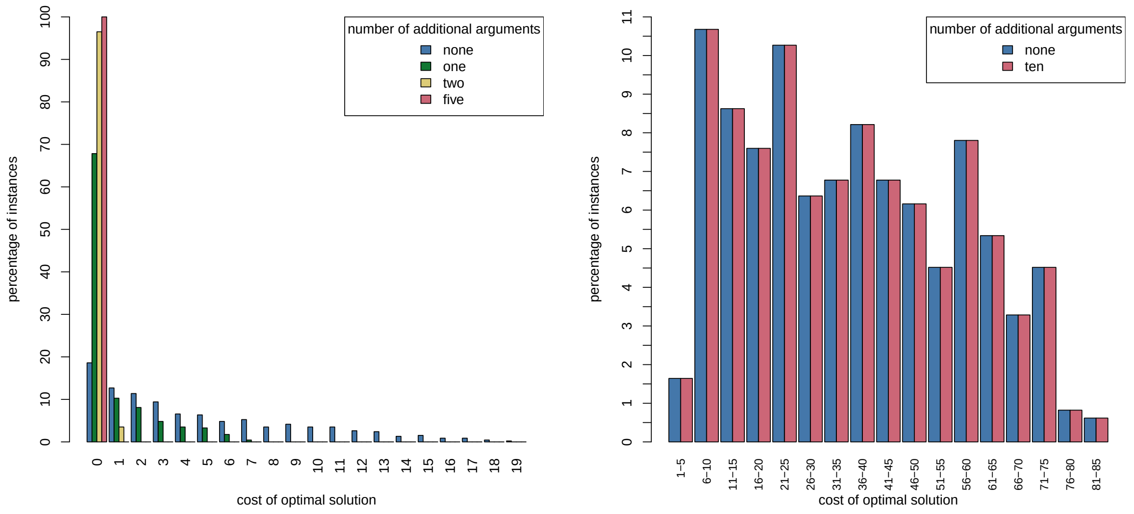

Figure 14: Effect of additional arguments on the cost of the optimal solutions on randomly generated instances under admissible (left) and stable (right) semantics.

Furthermore, one can for instance also synthesize an AF with the minimum number of attacks satisfying the maximum number of examples by adding soft clauses which state that the secondary preference is minimizing the number of attacks in the style of multi-level Boolean optimization (Argelich, Lynce, & Marques-Silva, 2009); in this case, in order to still guarantee that the primary objective of satisfying the maximum number of examples is met, the weights of the examples can be adjusted to be larger than the sum of the weights imposed on adding individual attacks to the solution AFs. As an alternative to the minimum number of attacks, in this way one may also minimize the structural distance to a known AF.

8.2 Covering Multiple Semantics

So far for each AF synthesis problem we required that all examples are given w.r.t. a specific semantics. A natural generalization is to let each example individually be linked to a semantics. Formally, an example $e = (S, w, \sigma)$ is then a triple with a semantics σ, denoted by σ_e. The cost of an AF F is given by

$$\sum_{e \in E^+} w_e \cdot I(S_e \notin \sigma_e(F)) + \sum_{e \in E^-} w_e \cdot I(S_e \in \sigma_e(F)).$$

Example 11. *Consider $E^+ = \{(\{a, c\}, 1, cf), (\{b, c\}, 1, stb)\}$ and $E^- = \{(\{a\}, 1, adm)\}$. This defines a unique 0-cost AF $F = (A, R)$ with $A = \{a, b, c\}$ for the AF synthesis instance with multiple semantics $P = (A, E^+, E^-)$ by $R = \{(b, a)\}$.*

The corresponding decision problem, i.e., for a given AF synthesis problem with multiple semantics $P = (A, E^+, E^-)$, is there an AF F with $cost(P, F) \leq k$ for an integer $k \geq 0$, does not exhibit higher computational complexity among the conflict-free, admissible, complete, and stable semantics.

Corollary 14. *AF synthesis with multiple semantics among the conflict-free, admissible, complete, and stable semantics is* NP-*complete.*

For solving AF synthesis with multiple semantics we can make use of our encodings of Section 6. In particular, we can conjoin the corresponding formulas for the different semantics and share the variables for attacks.

In connection to realizability, we note that Dunne, Spanring, Linsbichler, and Woltran (2016) studied a related problem that asks whether there is an AF that has exactly as its σ-extensions a given set $\mathbb{S}$ and as σ'-extensions a given set $\mathbb{S}'$.

8.3 Symbolic Representation of Examples

For a set A of arguments, there can be in general up to $2^{|A|}$ positive or negative examples. This exponentiality in the input can be restrictive for a large number of examples. Following ideas from Dunne et al. (2015), we note that, instead of explicit representation, examples could also be represented symbolically by encoding them succinctly in Boolean logic. This gives rise to the problem variant of AF synthesis with symbolic representation, with instances of the form $P = (A, \phi^+, \phi^-, \sigma)$, where ϕ^+ and ϕ^- are Boolean formulas. Let $Mod(\phi)$ be the set of models (satisfying assignments) of a Boolean formula ϕ, represented as sets themselves (variables assigned to true). For a given AF F, its associated cost $cost(P, F)$ is

$$\sum_{m \in Mod(\phi^+)} I(m \notin \sigma(F)) + \sum_{m \in Mod(\phi^-)} I(m \in \sigma(F)),$$

that is, unit weight is applied when a model of ϕ^+ is not a σ-extension of F and when a model of ϕ^- is a σ-extension of F.

Lemma 15. *Let ϕ be a Boolean formula, A the vocabulary of ϕ, F an AF, and σ a semantics. It holds that $|Mod(\phi)| = cost(P, F)$ for $P = (A, \phi, \phi, \sigma)$.*

Proof. Any AF F satisfies exactly $|Mod(\phi)|$ examples encoded in the formulas, since if $m \in Mod(\phi)$, then either $m \in \sigma(F)$ or $m \notin \sigma(F)$ (each implies unit weight). If $m \notin Mod(\phi)$, then both $m \in \sigma(F)$ and $m \notin \sigma(F)$ imply no weight. $\square$

Based on this lemma, it follows that determining the cost of a given AF for an AF synthesis instance with symbolic representation is presumably very complex. More concretely, it immediately follows from Lemma 15 that for those semantics σ for which we can in polynomial time verify whether a given set of arguments is a σ-extension, this problem is #P-complete[8]. Thus #P-completeness is established for all of the considered semantics except for preferred. For preferred, this gives a complexity lower bound, i.e., establishes #P-hardness.

8. #P is the class of counting problems where the task is to count the number of accepting paths of a given non-deterministic polynomial-time Turing machine; see Valiant (1979a, 1979b). As a prominent example, counting the number of models of a Boolean formula is #P-complete.

Corollary 16. *Counting the number of examples from a given AF synthesis instance with symbolic representation that are not satisfied by a given AF is* $\#\mathrm{P}$*-complete under the conflict-free, admissible, complete, grounded, and stable semantics, and* $\#\mathrm{P}$*-hard under preferred semantics.*

9. Related Work

As AF synthesis generalizes the notion of realizability, the most closely related work focuses on realizability of AFs (Baumann et al., 2014; Dyrkolbotn, 2014; Dunne et al., 2015; Linsbichler et al., 2015; Baumann, 2018), as already discussed earlier. Initiated by Dunne et al. (2015), realizability considers the question of whether a given set of sets (of arguments) $\mathbb{S}$ can be realized by an AF, i.e., whether an AF F with $\sigma(F) = \mathbb{S}$ for a semantics σ exists. Baumann et al. (2014) and Linsbichler et al. (2015) studied realizability under the restriction that the set of arguments of the constructed AF has to match exactly the set of arguments occurring in the input, i.e., that the set of arguments equals $\bigcup_{S \in \mathbb{S}} S$. Dyrkolbotn (2014) gave a construction for an AF using additional arguments in the three-valued labeling setting under the preferred and semi-stable semantics. We study the problem of synthesizing an AF that optimally matches a given set of examples semantically, even when an exact realization (a 0-cost solution) is not possible. Also, we analyze the complexity of AF synthesis, showing that, in contrast to polynomial-time results for checking realizability (Dunne et al., 2015), AF synthesis is in general NP-hard. To our best knowledge, no previous systems for solving realizability have been empirically evaluated. Recently a declarative encoding in answer set programming (ASP) for realizability was presented but not empirically evaluated (Pührer, 2015; Linsbichler et al., 2016a, 2016b). Our MaxSAT and ASP based implementations for the AF synthesis problem also cover realizability.

The AF synthesis problem connects also with research on change of AFs where the goal is to produce a modified AF. In particular, it shares many similarities to the extension enforcement problem (Baumann & Brewka, 2010; Baumann, 2012; Coste-Marquis et al., 2015; de Saint-Cyr, Bisquert, Cayrol, & Lagasquie-Schiex, 2016; Wallner et al., 2017), where we are given an AF and a subset of its arguments, and asked to modify the attacks in such a way that the subset becomes an extension, minimizing the structural distance to the original attack structure. On the other hand, in AF synthesis, there is no original structure, and instead of a single hard constraint on an extension, we pose multiple (possibly weighted) soft constraints on extensions and non-extensions. Furthermore, AF synthesis connects to AF revision and aggregation (merging) of AFs (Coste-Marquis et al., 2007; Tohmé et al., 2008; Coste-Marquis et al., 2014a, 2014b; Baumann & Brewka, 2015; Delobelle et al., 2015, 2016; Bodanza et al., 2017; Diller et al., 2018). More concretely, AF synthesis can be seen as an approach to extend many proposed operators for AF revision (Coste-Marquis et al., 2014a) and as an approach to aggregation that ensures that the (extended) operators produce a single AF for the revision or aggregation problem at hand. We discuss this relation of AF synthesis to revision and aggregation in Section 3. A different approach that guarantees a single AF in case of revision is to develop new revision operators (Diller et al., 2018). In contrast, AF synthesis allows for using standard operators for revision as a basis, and then applying a synthesis process afterwards, which also is able to handle user-specified weights and constraints (see also Section 8). To our knowledge, for the task of semantical merging of AFs (Delobelle et al., 2016), there are no general operators that guarantee a single AF as output.

In related work that incorporates AF construction from examples, Ontañón, Dellunde, Godo, and Plaza (2012) formally studied a logical characterization of inductive concept learning and AF learning in a multi-agent setting. In contrast to our work, they induce a rule-based theory and construct an AF based on conflicting rules. Riveret and Governatori (2016) studied probabilistic AF (Li, Oren, & Norman, 2011; Hunter, 2014; Riveret, Korkinof, Draief, & Pitt, 2015) learning with non-exact methods; we tackle the exact optimization problem of AF synthesis. Furthermore, Riveret (2016) considered on-the-fly computation of argumentation graphs from so-called statement acceptance labellings under the grounded semantics.

Related to allowing weights on entities such as subsets of arguments (i.e., examples) or attacks, Dunne, Hunter, McBurney, Parsons, and Wooldridge (2011) proposed the notion of weighted argument systems, where weights are assigned on individual attacks in a fixed AF F, an inconsistency budget is enforced on allowing subsets of arguments to be extensions: a *conflicting* subset E of arguments is considered an extension within an inconsistency budget b if by disregarding (or removing) a subset of the attacks in F of total weight at most b makes E an extension of F. In contrast, in AF synthesis, we allow (in case one wants to do so) assigning weights on the examples, instead of inducing a measure of inconsistency for extensions through weighted attacks. On the other hand, the way of inducing weights on subsets of arguments proposed by Dunne et al. (2011) (see therein for additional motivations for this way of inducing weights) could also be viewed as a way of assigning weights on examples e.g. in scenarios with multiple sources for examples. We also note that, as explained in Section 8.1, our algorithms allow for taking into account hard and soft constraints on individual attacks within the AF synthesis setting.

Finally, going beyond argumentation, it should be noted that there is a long line of research of constructing, inducing, or learning logical structures from examples, from Valiant (1984) to, e.g, work for constraint acquisition (Bessiere, Koriche, Lazaar, & O'Sullivan, 2017) as well as inductive logic programming (Bergadano & Gunetti, 1996; Davis & Ramon, 2015).

10. Conclusions

In this work we studied a computational perspective to the central knowledge representation problem in abstract argumentation, namely, that of compactly representing extensions as an argumentation framework. The central notion of realizability of AFs gives a measure of the expressive power of AFs under different semantics. However, in terms of representing sets of extensions via AFs in practice, realizability is a very strict notion: for a collection of sets of arguments to be realizable, it is required that there must be an AF the extensions of which are exactly the given sets of arguments. In other words, the exact set of extensions must be known in advance, and no other extensions are allowed. In this work, we proposed and studied the AF synthesis problem as a natural extension of realizability, addressing some of the shortcomings arising from the relatively stringent definition of realizability. By relaxing realizability in a natural way, AF synthesis allows for accommodating incomplete and noisy information. From the theoretical perspective, we related AF synthesis to realizability, and analyzed the complexity of AF synthesis both in the general case and in restricted settings. Motivated by the NP-hardness of AF synthesis under several AF semantics, we proposed Boolean optimization based algorithmic solutions for the problem, providing MaxSAT and ASP encodings for AF synthesis. We empirically studied this first approach to AF synthesis using state-of-the-art MaxSAT and ASP solvers on different types of AF synthesis instances.

Regarding future aspects, proving sharper bounds of the computational complexity of AF synthesis under complete and preferred, as well as considering other AF semantics, are a natural way to extend this work theoretically. Further, the question of how to extend the analysis presented in Section 4 on conditions for 0-cost solutions to other semantics such as complete remains unaddressed. There is potential for generalizing the very recent (and yet unpublished) analysis on conditions for realizability under complete semantics (Linsbichler, 2018) to analyze AF synthesis; however, this would require a non-trivial extension of the results presented in Section 4. On the algorithmic side, improving the efficiency of the current approach—especially tackling the seemingly hard problem of synthesis under preferred semantics remains a challenge. Investigating synthesis under ranking-based semantics (Bonzon, Delobelle, Konieczny, & Maudet, 2016), where instead of extensions a ranking of arguments is produced, is also a potential direction for future work. Finally, as noted by Thimm and Villata (2017), AF synthesis could be used for generating interesting and hard benchmarks for argumentation solvers e.g. for the central AF acceptance problems.

Acknowledgments

We thank Thomas Linsbichler for insightful discussions on (compact) realizability. This work has been financially supported by Academy of Finland (grants 251170 COIN, 284591, and 312662), Research Funds of the University of Helsinki, Doctoral Programme in Computer Science (DoCS) at the University of Helsinki, and the Austrian Science Fund (FWF): P30168-N31.

Appendix A. Proofs

We provide formal proofs for the results presented in Sections 3 and 5. We start by restating definitions from Dunne et al. (2015), which we need for auxiliary results towards the main proofs.

Definition 6. *(Dunne et al., 2015, Definitions 6, 7, and 8) Let A be a set of arguments and $\mathbb{S} \subseteq 2^A$ a set of sets of arguments. Set $\mathbb{S}$ is*

- *incomparable if $S \not\subseteq S'$ holds for all $S, S' \in \mathbb{S}$ with $S \neq S'$;*
- *tight if for all $S \in \mathbb{S}$ and all $a \in A$ it holds that $S \cup \{a\} \notin \mathbb{S}$ implies that there exists a $b \in S$ s.t. $\{a, b\} \not\subseteq S'$ for all $S' \in \mathbb{S}$;*
- *conflict-sensitive if for each $S, S' \in \mathbb{S}$ s.t. $S \cup S' \notin \mathbb{S}$ it holds that $\exists a, b \in S \cup S'$ s.t. $\{a, b\} \not\subseteq S''$ for all $S'' \in \mathbb{S}$; and*
- *downward closed if $\mathbb{S} = \{S' \mid S \in \mathbb{S}, S' \subseteq S\}$.*

For proving Proposition 2 we use the following lemma.

Lemma 17. *Let $\mathbb{S} \subseteq 2^A$ for a set A. It holds that $\mathit{ImpliedCF}(\mathbb{S})$ (i) contains $\emptyset$, (ii) is downward closed, (iii) is tight, and (iv) $\mathbb{S} \subseteq \mathit{ImpliedCF}(\mathbb{S})$.*

Proof. From the definition it directly follows that $\emptyset$ is contained in $\mathit{ImpliedCF}(\mathbb{S})$ for any $\mathbb{S}$. Let $S \in \mathit{ImpliedCF}(\mathbb{S})$. Then for all $a, b \in S$ it holds that $\exists S' \in \mathit{ImpliedCF}(\mathbb{S})$ s.t. $\{a, b\} \subseteq S'$. It follows that for any $S'' \subseteq S$ it holds that $S'' \in \mathit{ImpliedCF}(\mathbb{S})$. To show (iii), suppose that the set is not tight, i.e., $\exists S \in \mathit{ImpliedCF}(\mathbb{S})$ and $a \in A$ s.t. $S \cup \{a\} \notin \mathit{ImpliedCF}(\mathbb{S})$ and for all $b \in S$ there exists an $S' \in \mathit{ImpliedCF}(\mathbb{S})$ s.t. $\{a, b\} \subseteq S'$. This implies that for all $x, y \in S \cup \{a\}$ we have $\exists S'' \in \mathbb{S}$ s.t. $\{x, y\} \subseteq S''$, and thus $S \cup \{a\} \in \mathit{ImpliedCF}(\mathbb{S})$. This contradicts the assumption that $\mathit{ImpliedCF}(\mathbb{S})$ is not tight. Finally, if $S \in \mathbb{S}$, then by definition $S \in \mathit{ImpliedCF}(\mathbb{S})$ (iv). $\square$

For proving Proposition 4 we utilize the following lemma.

Lemma 18. *Let $\mathbb{S} \subseteq 2^A$ for a set A. It holds that $\mathit{ImpliedADM}(\mathbb{S})$ (i) contains $\emptyset$, (ii) is conflict-sensitive, and (iii) $\mathbb{S} \subseteq \mathit{ImpliedADM}(\mathbb{S})$.*

Proof. Claim (i) follows from the definition. Suppose (ii) does not hold, i.e., there exist $S, S' \in \mathit{ImpliedADM}(\mathbb{S})$ s.t. $S \cup S' \notin \mathit{ImpliedADM}(\mathbb{S})$ and for all $a, b \in S \cup S'$ there exists an $S'' \in \mathit{ImpliedADM}(\mathbb{S})$ with $\{a, b\} \subseteq S''$. It follows that $S \cup S' \in \mathit{ImpliedCF}(\mathbb{S})$ and $S \cup S' \in \mathit{ImpliedADM}(\mathbb{S})$. This contradicts the assumption that $\mathit{ImpliedADM}(\mathbb{S})$ is not conflict-sensitive. Finally, assume $S \in \mathbb{S}$. By Lemma 17, it follows that $S \in \mathit{ImpliedCF}(\mathbb{S})$, and by definition of $\mathit{ImpliedADM}$, also $S \in \mathit{ImpliedADM}(\mathbb{S})$ (S is equal to the union of elements in $\{S\} \subseteq \mathbb{S}$). □

Next, we prove two formal statements regarding the existence of 0-solutions for stable and preferred semantics, namely, Propositions 5 and 6.

Proof of Proposition 5. The claims of the proposition follow directly for the special case with $E^+ = \emptyset$. We proceed with the case that E^+ is non-empty. For the first claim, assume that F is a 0-cost solution to P. Conditions 2 and 3 follow immediately. For condition 1, since $\mathbb{S}_{E^+} \subseteq \mathit{stb}(F)$ it follows that $\mathbb{S}_{E^+} \subseteq \mathit{cf}(F)$ and thus $\mathit{ImpliedCF}(\mathbb{S}_{E^+}) \subseteq \mathit{cf}(F)$. Assuming that condition 1 does not hold directly violates $\mathbb{S}_{E^+} \subseteq \mathit{stb}(F)$ (stable extensions are subset-maximal conflict-free sets). For the second claim, assume that conditions 1-4 hold. Then $\mathbb{S}_{E^+}$ is a subset of the $\subseteq$-maximal elements of $\mathit{ImpliedCF}(\mathbb{S}_{E^+})$ due to $\mathbb{S}_{E^+} \subseteq \mathit{ImpliedCF}(\mathbb{S}_{E^+})$ (Lemma 17), assuming condition 1. By Dunne et al. (2015, Lemma 2) it follows that $\mathbb{S}_{E^+}$ is tight. Further, by Dunne et al. (2015, Proposition 7) and conditions 1-4, it follows that there exists an AF $F' = (A', R')$ with $A' \subseteq A$ s.t. $\mathit{stb}(F') = \mathbb{S}_{E^+}$. Construct $F = (A, R)$ by extending R' to contain self-attacks for each argument in $A \setminus A'$ and attacks from each argument in $\mathbb{S}_{E^+}$ to each $A \setminus A'$. □

Proof of Proposition 6. This proof follows a similar line of reasoning as the proof of Proposition 5. Assume that there is a solution AF F to P with $\mathit{cost}(P, F) = 0$. Condition 2 follows immediately from the existence of a 0-cost solution AF. Suppose condition 1 is violated. Due to the assumption and Lemma 3, it holds that $\mathbb{S}_{E^+} \subseteq \mathit{prf}(F)$, $\mathbb{S}_{E^+} \subseteq \mathit{adm}(F)$, and $\mathit{ImpliedADM}(\mathbb{S}_{E^+}) \subseteq \mathit{adm}(F)$. If, by presumption, there is an $S' \in \mathit{ImpliedADM}(\mathbb{S}_{E^+})$ and $S \in \mathbb{S}_{E^+}$ s.t. $S \subseteq S'$, then $S \notin \mathit{prf}(F)$ and $\mathbb{S}_{E^+} \not\subseteq \mathit{prf}(F)$. This is a contradiction to F being a 0-cost solution AF to P.

Assume that conditions 1-3 hold. We show that there is a solution AF F with $\mathit{cost}(P, F) = 0$. Now we have that $\mathbb{S}_{E^+}$ is a subset of the $\subseteq$-maximal elements in $\mathit{ImpliedADM}(\mathbb{S}_{E^+})$, since $\mathbb{S}_{E^+} \subseteq \mathit{ImpliedADM}(\mathbb{S}_{E^+})$ (Lemma 18) and by assumption of condition 1. Due to Dunne et al. (2015, Lemma 4), it holds that $\mathbb{S}_{E^+}$ is conflict-sensitive. By Dunne et al. (2015, Proposition 9), it follows that there is an AF $F' = (A', R')$ with $\mathit{prf}(F') = \mathbb{S}_{E^+}$ with $A' \subseteq A$. Construct $F = (A, R)$ by extending R' to R via adding self-attacks for each argument in $A \setminus A'$. □

We move on to proofs of complexity results.

Proof of Proposition 10 and Proposition 11. For an AF synthesis instance P with stable semantics, membership follows from a non-deterministic guess of an AF F and computing $\mathit{cost}(P, F)$ (which can be done in polynomial time by checking each example individually).

Given an AF synthesis instance P with preferred semantics, $\mathit{cost}(P, F)$ can be computed as follows. Non-deterministically guess a solution AF F and check for each example, via an NP oracle, whether the example is satisfied or not, i.e., whether the example is a preferred extension in F. Determining whether a given set of arguments is a preferred extension in a given AF is a problem in coNP (follows from results by Dimopoulos & Torres, 1996).

For both semantics, we prove hardness by a reduction from the Boolean satisfiability problem. Let ϕ be a Boolean formula in CNF with vocabulary X, $|X| = n$, and set of clauses C.

$$\begin{aligned} A =&\{x^T, x^F, d_x \mid x \in X\} \cup \{d'_c, d''_c \mid c \in C\} \cup \{d\} && (9)\\ E^+ =&\{(\{d'_c\} \cup \{x^T \mid x \in c\} \cup \{x^F \mid \neg x \in c\}, n+1) \mid c \in C\} \cup && (10)\\ &\{(\{d'_c, d''_c, d\}, n+1) \mid c \in C\} \cup && (11)\\ &\{(\{x^T, x^F, d_x\}, n+1) \mid x \in X\} \cup && (12)\\ &\{(\{x^T, d_x, d\}, 1) \mid x \in X\} \cup && (13)\\ &\{(\{x^F, d_x, d\}, 1) \mid x \in X\} && (14) \end{aligned}$$

Let $P_{stb} = (A, E^+, \emptyset, stb)$, $P_{prf} = (A, E^+, \emptyset, prf)$, with bound $k = n$. These AF synthesis instances can be constructed in polynomial time. The intuition behind this reduction is similar to the proof of Proposition 8: we have to choose between the unit-weighted examples (13,14) that simulate truth assignments, include a positive example that simulates satisfaction of a clause (10), and have certain examples ensuring integrity of this simulation (11,12).

We claim that

1. there exists an AF F s.t. $cost(P_{stb}, F) \leq n$ iff
2. there exists an AF F s.t. $cost(P_{prf}, F) \leq n$ iff
3. ϕ satisfiable.

We now prove that the first statement in the list implies the second, which, in turn, implies the third, which implies the first. Assume that there is an AF F s.t. $cost(P_{stb}, F) \leq n$. Since $stb(F) \subseteq prf(F)$, we have $cost(P_{prf}, F) \leq n$ (recall that we only have positive examples; additional preferred extensions, that are not stable, can only lower the cost).

Assume that there exists an AF F s.t. $cost(P_{prf}, F) \leq n$. It is immediate that all examples with weight $n+1$ are satisfied by F. For each $x \in X$ it holds that exactly one example from $\{(\{x^T, d_x, d\}, 1), (\{x^F, d_x, d\}, 1)\}$ is satisfied. To see this, we first prove that it cannot be the case that both are satisfied. Suppose the contrary, i.e., both are satisfied. Then both $\{x^T, d_x, d\}$ and $\{x^F, d_x, d\}$ are preferred extensions of F. This implies that $S = \{x^T, x^F, d_x, d\}$ is conflict-free (due to $\{x^T, x^F, d_x\}$ being preferred as well due to examples of the form (12)). Further, S is admissible in F, since S is a union of two admissible sets, $S = \{x^T, d_x, d\} \cup \{x^T, x^F, d_x\}$, which is also conflict-free. This directly violates $\{x^T, x^F, d_x\}$ being preferred (S is a proper superset), and the cost of F has to be higher than n. If there exists an $x \in X$ s.t. neither $(\{x^T, d_x, d\}, 1)$ nor $(\{x^F, d_x, d\}, 1)$ is satisfied, then the cost of F is higher than n as well, since for each $x' \in X \setminus \{x\}$ at least one example is not satisfied, contributing at least unit cost to F.

The unit-weight examples satisfied by F define a truth assignment $\tau(x) = 0$ iff $(\{x^T, d_x, d\}, 1)$ is satisfied by F and $\tau(x) = 1$ otherwise. We claim that $\tau \models \phi$. Suppose the contrary. Then $\exists c \in C$

with $\tau \not\models c$ and all literals in c are not satisfied by τ.

$$\begin{aligned}
&\tau \not\models \phi \\
&\text{iff } \exists c \in C, \tau \not\models c \\
&\text{iff } \exists c \in C \text{ s.t.} \\
&\qquad \text{if } x \in c \text{ then } \tau(x) = 0 \text{ and} \\
&\qquad \text{if } \neg x \in c \text{ then } \tau(x) = 1 \\
&\text{iff } \exists c \in C \text{ s.t.} \\
&\qquad \text{if } x \in c \text{ then } \{x^T, d_x, d\} \in \mathit{prf}(F) \text{ and} \\
&\qquad \text{if } \neg x \in c \text{ then } \{x^F, d_x, d\} \in \mathit{prf}(F) \\
&\text{only if } \exists c \in C \text{ s.t. } S' = \{d, d'_c\} \cup \{x^T \mid x \in c\} \cup \{x^F \mid \neg x \in c\} \in \mathit{cf}(F)
\end{aligned}$$

Consider the set $S' \setminus \{d\} = \{d'_c\} \cup \{x^T \mid x \in c\} \cup \{x^F \mid \neg x \in c\}$, which must be preferred in F due to examples of the form (10). The last implication holds, since $\{d'_c, d''_c, d\}$ is also preferred due to examples of the form (11), and, by the previous items in the iff chain, no argument from $\{x^T \mid x \in c\} \cup \{x^F \mid \neg x \in c\}$ can attack d. We now show that S' is admissible in F. By assumption each argument in $S' \setminus \{d\}$ is defended by S'. Since $\{d'_c, d''_c, d\}$ is preferred, and for all $x \in X$ exactly one example from $\{x^T, d_x, d\}$ and $\{x^F, d_x, d\}$ is preferred, it follows that only arguments y^T or y^F, for some $y \in X$, can attack d. Assume (y^T, d) is an attack in F (the claim is symmetric for y^F). Suppose that the attack (d, y^T) is not present in F. Now both $\{y^F, d_y, d\}$ and $\{y^T, y^F, d_y\}$ are preferred in F (the former set is preferred, since either $\{y^T, d_y, d\}$ or $\{y^F, d_y, d\}$ must be preferred). This implies that $\{y^F, d_y, d\}$ does not defend itself against the attack (y^T, d), and thus, cannot be preferred, which is a contradiction. Now we have shown that d defends itself against all attacks (all attacks are, in fact, bidirectional attacks). Since $S' \setminus \{d\}$ is admissible, also S' is admissible, and hence, $S' \setminus \{d\}$ cannot be preferred, violating the corresponding assumption. This means $\tau \models c$.

Assume now that ϕ is satisfiable. Construct AF $F = (A, R)$ with

$$\begin{aligned}
R = &\{(d''_c, d_x), (d_x, d''_c) \mid c \in C, x \in X\} \cup \\
&\{(d''_c, x^T), (d''_c, x^F), (x^T, d''_c), (x^F, d''_c) \mid c \in C, x \in X\} \cup \\
&\{(a, b), (b, a) \mid a \in \{d'_c, d''_c\}, b \in \{d'_{c'}, d''_{c'}\}, c, c' \in C, c \neq c'\} \cup \\
&\{(d_x, d_y), (d_x, z), (z, d_x) \mid x, y \in X, x \neq y, z \in \{y^T, y^F\}\} \cup \\
&\{(d_x, d'_c), (d'_c, d_x) \mid c \in C, x \in X\} \cup \\
&\{(d'_c, x^T), (x^T, d'_c) \mid c \in C, x \in X, x \notin c\} \cup \\
&\{(d'_c, x^F), (x^F, d'_c) \mid c \in C, x \in X, \neg x \notin c\} \cup \\
&\{(x^T, d), (d, x^T) \mid \tau(x) = 1\} \cup \\
&\{(x^F, d), (d, x^F) \mid \tau(x) = 0\}.
\end{aligned}$$

We illustrate this constructed AF in Figure 15 for a simple satisfiable formula and a truth assignment that assigns 1 to x and 0 to y. The corresponding satisfaction of examples, for P_{stb}, is also shown in that figure.

examples	satisfied?
$(\{d'_c, x^T, y^F\}, 3)$	yes
$(\{d'_c, d''_c, d\}, 3)$	yes
$(\{x^T, x^F, d_x\}, 3)$	yes
$(\{y^T, y^F, d_y\}, 3)$	yes
$(\{x^T, d_x, d\}, 1)$	no
$(\{x^F, d_x, d\}, 1)$	yes
$(\{y^T, d_y, d\}, 1)$	yes
$(\{y^T, d_y, d\}, 1)$	no

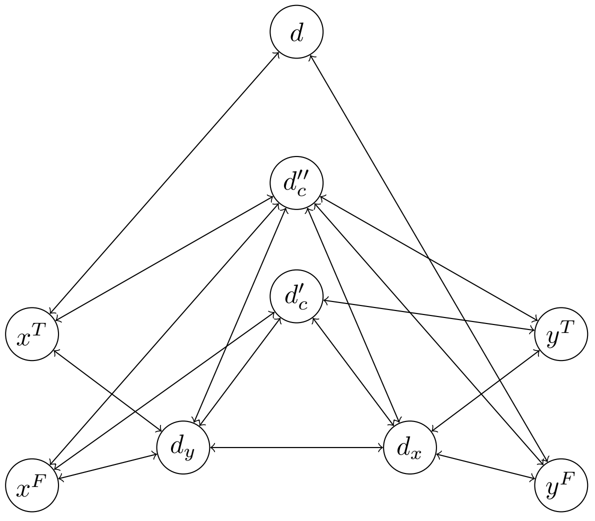

Figure 15: Illustration of reduction in proof of Proposition 10 for formula $c = (x \vee \neg y)$.

We now show that F has cost n for P_{stb} under the assumption that ϕ is satisfiable. The constructed AF F has bidirectional attacks between arguments to ensure that all the examples (11) and (12) are stable. By construction of F it is immediate that the sets of the examples are conflict-free. Consider set $\{d'_c, d''_c, d\}$ for some $c \in C$. Then $A \setminus \{d'_c, d''_c, d\}$ is attacked by this set, directly by construction of R. The same holds for $\{x^T, x^F, d_x\}$ for $x \in X$.

A mutual attack is added between x^T and d (x^F and d) based on the truth assignment τ, violating one of the unit-weighted examples. The other unit-weighted example is satisfied. Finally, examples of form (10) are satisfied, since one of the arguments in these sets (except d'_c) attacks d due to assumption that τ satisfies ϕ. Intuitively, the examples encoding the clauses (10) are satisfied since one of the arguments corresponding to the chosen truth assignment that satisfies one of the literals attacks d. $\square$

References

Airiau, S., Bonzon, E., Endriss, U., Maudet, N., & Rossit, J. (2017). Rationalisation of profiles of abstract argumentation frameworks: Characterisation and complexity. *Journal of Artificial Intelligence Research*, *60*, 149–177.

Alviano, M., Dodaro, C., & Ricca, F. (2015). A MaxSAT algorithm using cardinality constraints of bounded size. In Yang, Q., & Wooldridge, M. (Eds.), *Proc. IJCAI*, pp. 2677–2683. AAAI Press / IJCAI.

Ansótegui, C., & Gabàs, J. (2013). Solving (weighted) partial MaxSAT with ILP. In Gomes, C. P., & Sellmann, M. (Eds.), *Proc. CPAIOR*, Vol. 7874 of *Lecture Notes in Computer Science*, pp. 403–409. Springer.

Ansótegui, C., & Gabàs, J. (2017). WPM3: An (in)complete algorithm for weighted partial MaxSAT. *Artificial Intelligence*, *250*, 37–57.

Argelich, J., Lynce, I., & Marques-Silva, J. P. (2009). On solving Boolean multilevel optimization problems. In Boutilier, C. (Ed.), *Proc. IJCAI*, pp. 393–398. AAAI Press.

Baroni, P., Caminada, M., & Giacomin, M. (2011). An introduction to argumentation semantics. *Knowledge Engineering Review*, *26*(4), 365–410.

Baroni, P., Caminada, M., & Giacomin, M. (2018a). Abstract argumentation frameworks and their semantics. In Baroni, P., Gabbay, D., Giacomin, M., & van der Torre, L. (Eds.), *Handbook of Formal Argumentation*, chap. 4, pp. 159–236. College Publications.

Baroni, P., Gabbay, D., Giacomin, M., & van der Torre, L. (Eds.). (2018b). *Handbook of Formal Argumentation*. College Publications.

Baumann, R. (2012). What does it take to enforce an argument? Minimal change in abstract argumentation. In Raedt, L. D., Bessière, C., Dubois, D., Doherty, P., Frasconi, P., Heintz, F., & Lucas, P. J. F. (Eds.), *Proc. ECAI*, Vol. 242 of *Frontiers in Artificial Intelligence and Applications*, pp. 127–132.

Baumann, R., & Brewka, G. (2010). Expanding argumentation frameworks: Enforcing and monotonicity results. In Baroni, P., Cerutti, F., Giacomin, M., & Simari, G. R. (Eds.), *Proc. COMMA*, Vol. 216 of *Frontiers in Artificial Intelligence and Applications*, pp. 75–86. IOS Press.

Baumann, R., & Brewka, G. (2015). AGM meets abstract argumentation: Expansion and revision for Dung frameworks. In Yang, Q., & Wooldridge, M. (Eds.), *Proc. IJCAI*, pp. 2734–2740. AAAI Press.

Baumann, R. (2018). On the nature of argumentation semantics: Existence and uniqueness, expressibility, and replaceability. In Baroni, P., Gabbay, D., Giacomin, M., & van der Torre, L. (Eds.), *Handbook of Formal Argumentation*, chap. 17, pp. 839–936. College Publications.

Baumann, R., Dvořák, W., Linsbichler, T., Spanring, C., Strass, H., & Woltran, S. (2016). On rejected arguments and implicit conflicts: The hidden power of argumentation semantics. *Artificial Intelligence*, *241*, 244–284.

Baumann, R., Dvořák, W., Linsbichler, T., Strass, H., & Woltran, S. (2014). Compact argumentation frameworks. In Schaub, T., Friedrich, G., & O'Sullivan, B. (Eds.), *Proc. ECAI*, Vol. 263 of *Frontiers in Artificial Intelligence and Applications*, pp. 69–74. IOS Press.

Baumann, R., & Strass, H. (2013). On the maximal and average numbers of stable extensions. In Black, E., Modgil, S., & Oren, N. (Eds.), *Proc. TAFA*, Vol. 8306 of *Lecture Notes in Computer Science*, pp. 111–126. Springer.

Bench-Capon, T., & Dunne, P. E. (2007). Argumentation in artificial intelligence. *Artificial Intelligence*, *171*(10-15), 619–641.

Bergadano, F., & Gunetti, D. (1996). *Inductive logic programming - from machine learning to software engineering*. MIT Press.

Besnard, P., & Doutre, S. (2004). Checking the acceptability of a set of arguments. In Delgrande, J. P., & Schaub, T. (Eds.), *Proc. NMR*, pp. 59–64.

Bessiere, C., Koriche, F., Lazaar, N., & O'Sullivan, B. (2017). Constraint acquisition. *Artificial Intelligence*, *244*, 315–342.

Bistarelli, S., & Santini, F. (2011). ConArg: A constraint-based computational framework for argumentation systems. In Khoshgoftaar, T. M., & Zhu, X. (Eds.), *Proc. ICTAI*, pp. 605–612. IEEE Computer Society.

Black, E., Coles, A. J., & Bernardini, S. (2014). Automated planning of simple persuasion dialogues. In Bulling, N., van der Torre, L. W. N., Villata, S., Jamroga, W., & Vasconcelos, W. W. (Eds.), *Proc. CLIMA*, Vol. 8624 of *Lecture Notes in Computer Science*, pp. 87–104. Springer.

Black, E., & Hunter, A. (2012). A relevance-theoretic framework for constructing and deconstructing enthymemes. *Journal of Logic and Computation*, *22*(1), 55–78.

Black, E., & Hunter, A. (2015). Reasons and options for updating an opponent model in persuasion dialogues. In Black, E., Modgil, S., & Oren, N. (Eds.), *Proc. TAFA, Revised Selected Papers*, Vol. 9524 of *Lecture Notes in Computer Science*, pp. 21–39. Springer.

Bodanza, G. A., Tohmé, F., & Auday, M. (2017). Collective argumentation: A survey of aggregation issues around argumentation frameworks. *Argument & Computation*, *8*(1), 1–34.

Bonzon, E., Delobelle, J., Konieczny, S., & Maudet, N. (2016). A comparative study of ranking-based semantics for abstract argumentation. In Schuurmans, D., & Wellman, M. P. (Eds.), *Proc. AAAI*, pp. 914–920. AAAI Press.

Brewka, G., Eiter, T., & Truszczynski, M. (2011). Answer set programming at a glance. *Communications of the ACM*, *54*(12), 92–103.

Brewka, G., Ellmauthaler, S., Strass, H., Wallner, J. P., & Woltran, S. (2018). Abstract dialectical frameworks. In Baroni, P., Gabbay, D., Giacomin, M., & van der Torre, L. (Eds.), *Handbook of Formal Argumentation*, chap. 5, pp. 237–285. College Publications.

Calimeri, F., Faber, W., Gebser, M., Ianni, G., Kaminski, R., Krennwallner, T., Leone, N., Ricca, F., & Schaub, T. (2012). ASP-Core-2 input language format. Available at `https://www.mat.unical.it/aspcomp2013/files/ASP-CORE-2.0.pdf`.

Caminada, M. (2007). Comparing two unique extension semantics for formal argumentation: Ideal and eager. In Dastani, M., & de Jong, E. (Eds.), *Proc. BNAIC*, pp. 81–87.

Cayrol, C., de Saint-Cyr, F. D., & Lagasquie-Schiex, M. (2010). Change in abstract argumentation frameworks: Adding an argument. *Journal of Artificial Intelligence Research*, *38*, 49–84.

Cerutti, F., Dunne, P. E., Giacomin, M., & Vallati, M. (2014a). Computing preferred extensions in abstract argumentation: A SAT-based approach. In Black, E., Modgil, S., & Oren, N. (Eds.), *Proc. TAFA 2013 Revised Selected Papers*, Vol. 8306 of *Lecture Notes in Computer Science*, pp. 176–193. Springer.

Cerutti, F., Giacomin, M., & Vallati, M. (2014b). ArgSemSAT: Solving argumentation problems using SAT. In Parsons, S., Oren, N., Reed, C., & Cerutti, F. (Eds.), *Proc. COMMA*, Vol. 266 of *Frontiers in Artificial Intelligence and Applications*, pp. 455–456. IOS Press.

Cerutti, F., Gaggl, S. A., Thimm, M., & Wallner, J. P. (2018). Foundations of implementations for formal argumentation. In Baroni, P., Gabbay, D., Giacomin, M., & van der Torre, L. (Eds.), *Handbook of Formal Argumentation*, chap. 15, pp. 688–767. College Publications.

Clarke, E. M., Gupta, A., & Strichman, O. (2004). SAT-based counterexample-guided abstraction refinement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *23*(7), 1113–1123.

Clarke, E., Grumberg, O., Jha, S., Lu, Y., & Veith, H. (2003). Counterexample-guided abstraction refinement for symbolic model checking. *Journal of the ACM*, *50*(5), 752–794.

Coste-Marquis, S., Konieczny, S., Mailly, J.-G., & Marquis, P. (2014a). On the revision of argumentation systems: Minimal change of arguments statuses. In Baral, C., Giacomo, G. D., & Eiter, T. (Eds.), *Proc. KR*, pp. 52–61. AAAI Press.

Coste-Marquis, S., Konieczny, S., Mailly, J.-G., & Marquis, P. (2014b). A translation-based approach for revision of argumentation frameworks. In Fermé, E., & Leite, J. (Eds.), *Proc. JELIA*, Vol. 8761 of *Lecture Notes in Computer Science*, pp. 397–411. Springer.

Coste-Marquis, S., Devred, C., Konieczny, S., Lagasquie-Schiex, M., & Marquis, P. (2007). On the merging of Dung's argumentation systems. *Artificial Intelligence*, *171*(10-15), 730–753.

Coste-Marquis, S., Konieczny, S., Mailly, J., & Marquis, P. (2015). Extension enforcement in abstract argumentation as an optimization problem. In Yang, Q., & Wooldridge, M. (Eds.), *Proc. IJCAI*, pp. 2876–2882. AAAI Press.

Davies, J., & Bacchus, F. (2013). Exploiting the power of MIP solvers in MAXSAT. In Järvisalo, M., & Gelder, A. V. (Eds.), *Proc. SAT*, Vol. 7962 of *Lecture Notes in Computer Science*, pp. 166–181. Springer.

Davis, J., & Ramon, J. (Eds.). (2015). *Inductive Logic Programming - 24th International Conference, ILP 2014, Nancy, France, September 14-16, 2014, Revised Selected Papers*, Vol. 9046 of *Lecture Notes in Computer Science*. Springer.

de Saint-Cyr, F. D., Bisquert, P., Cayrol, C., & Lagasquie-Schiex, M. (2016). Argumentation update in YALLA (Yet Another Logic Language for Argumentation). *International Journal of Approximate Reasoning*, *75*, 57–92.

Delobelle, J., Haret, A., Konieczny, S., Mailly, J.-G., Rossit, J., & Woltran, S. (2016). Merging of abstract argumentation frameworks. In Baral, C., Delgrande, J. P., & Wolter, F. (Eds.), *Proc. KR*, pp. 33–42. AAAI Press.

Delobelle, J., Konieczny, S., & Vesic, S. (2015). On the aggregation of argumentation frameworks. In Yang, Q., & Wooldridge, M. (Eds.), *Proc. IJCAI*, pp. 2911–2917. AAAI Press.

Diller, M., Haret, A., Linsbichler, T., Rümmele, S., & Woltran, S. (2018). An extension-based approach to belief revision in abstract argumentation. *International Journal of Approximate Reasoning*, *93*, 395–423.

Dimopoulos, Y., & Torres, A. (1996). Graph theoretical structures in logic programs and default theories. *Theoretical Computer Science*, *170*(1-2), 209–244.

Dimopoulos, Y., Mailly, J., & Moraitis, P. (2018). Control argumentation frameworks. In McIlraith, S. A., & Weinberger, K. Q. (Eds.), *Proc. AAAI*, pp. 4678–4685. AAAI Press.

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.

Dung, P. M., & Thang, P. M. (2018). Fundamental properties of attack relations in structured argumentation with priorities. *Artificial Intelligence*, *255*, 1–42.

Dunne, P. E., Hunter, A., McBurney, P., Parsons, S., & Wooldridge, M. (2011). Weighted argument systems: Basic definitions, algorithms, and complexity results. *Artificial Intelligence*, *175*(2), 457–486.

Dunne, P. E., & Wooldridge, M. (2009). Complexity of abstract argumentation. In Simari, G., & Rahwan, I. (Eds.), *Argumentation in Artificial Intelligence*, pp. 85–104. Springer.

Dunne, P. E., Dvořák, W., Linsbichler, T., & Woltran, S. (2015). Characteristics of multiple viewpoints in abstract argumentation. *Artificial Intelligence*, *228*, 153–178.

Dunne, P. E., Marquis, P., & Wooldridge, M. (2012). Argument aggregation: Basic axioms and complexity results. In Verheij, B., Szeider, S., & Woltran, S. (Eds.), *Proc. COMMA*, Vol. 245 of *Frontiers in Artificial Intelligence and Applications*, pp. 129–140. IOS Press.

Dunne, P. E., Spanring, C., Linsbichler, T., & Woltran, S. (2016). Investigating the relationship between argumentation semantics via signatures. In Kambhampati, S. (Ed.), *Proc. IJCAI*, pp. 1051–1057. IJCAI/AAAI Press.

Dvorák, W., Gaggl, S. A., Wallner, J. P., & Woltran, S. (2011). Making use of advances in answer-set programming for abstract argumentation systems. In Tompits, H., Abreu, S., Oetsch, J., Pührer, J., Seipel, D., Umeda, M., & Wolf, A. (Eds.), *Proc. INAP, Revised Selected Papers*, Vol. 7773 of *Lecture Notes in Computer Science*, pp. 114–133. Springer.

Dvořák, W., Järvisalo, M., Wallner, J. P., & Woltran, S. (2014). Complexity-sensitive decision procedures for abstract argumentation. *Artificial Intelligence*, *206*, 53–78.

Dvořák, W., & Dunne, P. E. (2018). Computational problems in formal argumentation and their complexity. In Baroni, P., Gabbay, D., Giacomin, M., & van der Torre, L. (Eds.), *Handbook of Formal Argumentation*, chap. 14. College Publications.

Dyrkolbotn, S. K. (2014). How to argue for anything: Enforcing arbitrary sets of labellings using AFs. In Baral, C., Giacomo, G. D., & Eiter, T. (Eds.), *Proc. KR*, pp. 626–629. AAAI Press.

Eén, N., & Sörensson, N. (2004). An extensible SAT-solver. In Giunchiglia, E., & Tacchella, A. (Eds.), *SAT 2003 Selected Revised Papers*, Vol. 2919 of *Lecture Notes in Computer Science*, pp. 502–518. Springer.

Egly, U., Gaggl, S., & Woltran, S. (2010). Answer-set programming encodings for argumentation frameworks. *Argument & Computation*, *1*(2), 147–177.

Eiter, T., Ianni, G., & Krennwallner, T. (2009). Answer set programming: A primer. In Tessaris, S., Franconi, E., Eiter, T., Gutierrez, C., Handschuh, S., Rousset, M., & Schmidt, R. A. (Eds.), *Reasoning Web. Semantic Technologies for Information Systems, 5th International Summer School 2009, Brixen-Bressanone, Italy, August 30 - September 4, 2009, Tutorial Lectures*, Vol. 5689 of *Lecture Notes in Computer Science*, pp. 40–110. Springer.

Gaggl, S. A., Manthey, N., Ronca, A., Wallner, J. P., & Woltran, S. (2015). Improved answer-set programming encodings for abstract argumentation. *Theory and Practice of Logic Programming*, *15*(4-5), 434–448.

Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., & Wanko, P. (2016). Theory solving made easy with clingo 5. In *Technical Communications of the 32nd International Conference on Logic Programming*, pp. 2:1–2:15.

Gebser, M., Kaminski, R., Kaufmann, B., & Schaub, T. (2012). *Answer Set Solving in Practice*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Hosseini, S. A., Modgil, S., & Rodrigues, O. (2014). Enthymeme construction in dialogues using shared knowledge. In Parsons, S., Oren, N., Reed, C., & Cerutti, F. (Eds.), *Proc. COMMA*, Vol. 266 of *Frontiers in Artificial Intelligence and Applications*, pp. 325–332. IOS Press.

Hunter, A. (2007). Real arguments are approximate arguments. In *Proc. AAAI*, pp. 66–71. AAAI Press.

Hunter, A. (2014). Probabilistic qualification of attack in abstract argumentation. *International Journal of Approximate Reasoning*, *55*(2), 607–638.

Hunter, A., & Williams, M. (2012). Aggregating evidence about the positive and negative effects of treatments. *Artificial Intelligence in Medicine*, *56*(3), 173–190.

Li, H., Oren, N., & Norman, T. J. (2011). Probabilistic argumentation frameworks. In Modgil, S., Oren, N., & Toni, F. (Eds.), *Proc. TAFA*, Vol. 7132 of *Lecture Notes in Computer Science*, pp. 1–16. Springer.

Liao, B., Jin, L., & Koons, R. C. (2011). Dynamics of argumentation systems: a division-based method. *Artificial Intelligence*, *175*(11), 1790–1814.

Linsbichler, T. (2018). Characteristics of multiple viewpoints in abstract argumentation under complete semantics. Tech. rep. DBAI-TR-2018-113, Vienna University of Technology, Institut für Logic and Computation, Abteilung Datenbanken und Artificial Intelligence.

Linsbichler, T., Pührer, J., & Strass, H. (2016a). Characterizing realizability in abstract argumentation. In Kern-Isberner, G., & Wassermann, R. (Eds.), *Proc. NMR*, pp. 85–94.

Linsbichler, T., Pührer, J., & Strass, H. (2016b). A uniform account of realizability in abstract argumentation. In Kaminka, G. A., Fox, M., Bouquet, P., Hüllermeier, E., Dignum, V., Dignum, F., & van Harmelen, F. (Eds.), *Proc. ECAI*, Vol. 285 of *Frontiers in Artificial Intelligence and Applications*, pp. 252–260. IOS Press.

Linsbichler, T., Spanring, C., & Woltran, S. (2015). The hidden power of abstract argumentation semantics. In Black, E., Modgil, S., & Oren, N. (Eds.), *Proc. TAFA*, Vol. 9524 of *Lecture Notes in Computer Science*, pp. 146–162. Springer.

Mailly, J.-G. (2016). Using enthymemes to fill the gap between logical argumentation and revision of abstract argumentation frameworks. In *Proc. NMR*.

Martins, R., Manquinho, V. M., & Lynce, I. (2014). Open-WBO: A modular MaxSAT solver. In Sinz, C., & Egly, U. (Eds.), *Proc. SAT*, Vol. 8561 of *Lecture Notes in Computer Science*, pp. 438–445. Springer.

Morgado, A., Ignatiev, A., & Marques-Silva, J. (2015). MSCG: Robust core-guided MaxSAT solving. *Journal on Satisfiability, Boolean Modeling and Computation*, *9*, 129–134.

Murphy, J., Black, E., & Luck, M. (2016). A heuristic strategy for persuasion dialogues. In Baroni, P., Gordon, T. F., Scheffler, T., & Stede, M. (Eds.), *Proc. COMMA*, Vol. 287 of *Frontiers in Artificial Intelligence and Applications*, pp. 411–418. IOS Press.

Niskanen, A., Wallner, J. P., & Järvisalo, M. (2016a). Pakota: A system for enforcement in abstract argumentation. In Michael, L., & Kakas, A. C. (Eds.), *Proc. JELIA*, Vol. 10021 of *Lecture Notes in Computer Science*, pp. 385–400. Springer.

Niskanen, A., Wallner, J. P., & Järvisalo, M. (2016b). Synthesizing argumentation frameworks from examples. In Kaminka, G. A., Fox, M., Bouquet, P., Hüllermeier, E., Dignum, V., Dignum, F., & van Harmelen, F. (Eds.), *Proc. ECAI*, Vol. 285 of *Frontiers in Artificial Intelligence and Applications*, pp. 551–559. IOS Press.

Nofal, S., Atkinson, K., & Dunne, P. E. (2014). Algorithms for decision problems in argument systems under preferred semantics. *Artificial Intelligence*, *207*, 23–51.

Ontañón, S., Dellunde, P., Godo, L., & Plaza, E. (2012). A defeasible reasoning model of inductive concept learning from examples and communication. *Artificial Intelligence*, *193*, 129–148.

Oren, N., & Norman, T. J. (2009). Arguing using opponent models. In Peter, M., Iyad, R., & Parsons Simon, M. N. (Eds.), *Proc. ArgMAS*.

Pührer, J. (2015). Realizability of three-valued semantics for abstract dialectical frameworks. In Yang, Q., & Wooldridge, M. (Eds.), *Proc. IJCAI*, pp. 3171–3177. AAAI Press.

Rienstra, T., Thimm, M., & Oren, N. (2013). Opponent models with uncertainty for strategic argumentation. In Rossi, F. (Ed.), *Proc. IJCAI*, pp. 332–338. IJCAI/AAAI.

Riveret, R. (2016). On learning abstract argumentation graphs from bivalent statement labellings. In *Proc. ICTAI*, pp. 190–195. IEEE Computer Society.

Riveret, R., & Governatori, G. (2016). On learning attacks in probabilistic abstract argumentation. In Jonker, C. M., Marsella, S., Thangarajah, J., & Tuyls, K. (Eds.), *Proc. AAMAS*, pp. 653–661. ACM.

Riveret, R., Korkinof, D., Draief, M., & Pitt, J. V. (2015). Probabilistic abstract argumentation: an investigation with Boltzmann machines. *Argument & Computation*, *6*(2), 178–218.

Saikko, P., Berg, J., & Järvisalo, M. (2016). LMHS: A SAT-IP hybrid MaxSAT solver. In Creignou, N., & Berre, D. L. (Eds.), *Proc. SAT*, Vol. 9710 of *Lecture Notes in Computer Science*, pp. 539–546. Springer.

Thimm, M., & Villata, S. (2017). The first international competition on computational models of argumentation: Results and analysis. *Artificial Intelligence*, *252*, 267–294.

Thimm, M., Villata, S., Cerutti, F., Oren, N., Strass, H., & Vallati, M. (2016). Summary report of the first international competition on computational models of argumentation. *AI Magazine*, *37*(1), 102–104.

Tohmé, F. A., Bodanza, G. A., & Simari, G. R. (2008). Aggregation of attack relations: A social-choice theoretical analysis of defeasibility criteria. In Hartmann, S., & Kern-Isberner, G. (Eds.), *Proc. FoIKS*, Vol. 4932 of *Lecture Notes in Computer Science*, pp. 8–23. Springer.

Valiant, L. G. (1979a). The complexity of computing the permanent. *Theoretical Computer Science*, *8*, 189–201.

Valiant, L. G. (1979b). The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, *8*(3), 410–421.

Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM*, *27*(11), 1134–1142.

Wallner, J. P., Niskanen, A., & Järvisalo, M. (2017). Complexity results and algorithms for extension enforcement in abstract argumentation. *Journal of Artificial Intelligence Research*, *60*, 1–40.