1

Computational Problems in Formal Argumentation and their Complexity

Wolfgang Dvořák, Paul E. Dunne

ABSTRACT. In this chapter we give an overview of the core computational problems arising in formal argumentation together with a complexity analysis highlighting different sources of computational complexity. To this end we consider three of the previously discussed formalisms, that are Dung's abstract argumentation frameworks, assumption-based argumentation, and abstract dialectical frameworks, each of which allows to highlight different sources of computational complexity in formal argumentation. As most of these problems turn out to be of high complexity we also consider properties of instances, like being in a specific graph class, that reduce the complexity and thus allow for more efficient algorithms. Finally, we also show how to apply techniques from parametrized complexity that allow for a more fine-grained complexity classification.

1 Introduction

In the previous chapters of this handbook several models for formal argumentation were discussed. They propose different ways to construct arguments, draw conclusions and each of these models comes with several proposals for semantics as to how coherent sets of arguments or statements should be selected. In this chapter we address the computational issues appearing in argumentation formalisms and that have to be tackled when implementing argumentation systems. That is, we will identify core problems of abstract argumentation and present basic procedures to solve them together with hardness results, based on computational complexity theory, that show some problems to have inherent complexity that cannot be circumvented by any algorithm.

The computational problems of formal argumentation occur at several places in the argumentation process. (A1) First, when instantiating argumentation frameworks from knowledge bases one has to deal with the task of constructing arguments and identifying conflicts (or even more complex) relations between arguments. (A2) Second, given the arguments and conflicts between them one has to find coherent sets of arguments that can be simultaneously accepted (w.r.t. a selected semantics). (A3) Finally, given the coherent sets of arguments one has to draw conclusions based on this selection. The computations in the first and third item often correspond to problems that are purely located in the underlying logic, e.g., to evaluating an inference operator. The second item is at the core of formal argumentation. That is, we are given arguments and relations (e.g. attacks) between them, and have to evaluate them w.r.t. to an argumentation semantics. This may require computing all extensions of a given semantics, the acceptance status of some argument w.r.t. some semantics, or finding some witness or counter example for a claim.

In this chapter we consider computational problems in three argumentation formalisms introduced in earlier chapters of this handbook, i.e. Dung's abstract argument frameworks, assumption-based argumentation and abstract dialectical frameworks. Dung's abstract argument frameworks are a model for (A2) which only consists of abstract entities called arguments and a binary attack relation between them. There is no instantiation process or computation of conclusions. Thus, it is perfectly suited for studying the computational issues involved in (A2). Assumption-based argumentation models the whole process (A1), (A2) and (A3), starting from a knowledge base, constructing arguments and conflicts, and finally returning conclusions. By comparing assumption-based argumentation with the results for Dung's abstract argumentation we are able to highlight the computational costs for (A1) and (A3). Finally, we consider abstract dialectical frameworks (ADFs) which are a richer model for (A2). As for Dung's abstract argument frameworks, here only abstract entities are considered but instead of just a binary attack relation ADFs allow for more complex relations between these entities. On the basis of ADFs we will highlight the impact of the allowed relations between arguments on the computational complexity of the reasoning tasks.

Notice by the term "*computational problem*" we mean the task of when presented with a *description* of some *input*, e.g., the vertices and edges in a graph, a collection of numeric values, producing an output related in a specified way to this input. For example: reporting the set of vertices forming the endpoints of at least two edges, returning the collection of numerical values *sorted* in increasing order. One special type of computational problem is of particular interest: the class of so-called *decision* problems. These concern determining whether the given input structure has a particular property of interest, e.g. given a graph as before does it contain a cycle?, given a list of numbers, does the largest exceed 100?

The formal study of computational problems has two principal foci:

A. The construction of ("efficient") *algorithms* to *solve* the problem. That is methods which when presented with an input instance *always* report the *correct* output.

B. To categorise collections of computational problems that are "similar" in terms of their "best" algorithms, and thence provide a formal proof that *every* algorithmic approach must take some number of steps.

Thus (A) is concerned with positive constructive demonstration of an *upper* bound on a problem's *computational complexity* while (B) is a (more negative) statement prescribing *lower* bounds on computational complexity.

Why are these focal points of importance? To gain some insight to this consider the well-known computational problem of *sorting*: given a collection of N numbers $< a_1, a_2, \ldots, a_n >$ return this collection in increasing order of its members. Here are three informally presented "sorting algorithms":

S1 Generate each possible ordering, π, of $< a_1, a_2, \ldots, a_N >$ in turn: return the first ordering found that is correct.

S2 Form a new ordering by comparing for each $i > 1$ the (current) a_{i-1} and a_i: if $a_i > a_{i-1}$ exchange the pair. Repeat with the new ordering produced until the collection is sorted.

S3 If $N = 1$ the list is already sorted. Otherwise (recursively) sort the two list $< a_1, \ldots, a_{N/2} >$ and $< a_{N/2+1}, \ldots, a_N >$ and "merge" the two sorted lists to give the final output.

On the surface, in the sense that all three methods are correct there appears to be little to choose between these three methods. If, however, we examine their performance a very different picture emerges.

Method (S1) in the worst case (no matter how the successive ordering are produced) requires $N!$ steps: if $N = 100$ this is roughly 10^{200}

Method (S2) in the worst case needs N^2 steps: for $N = 100$ this is 10^4.

Method (S3) takes of the order of $N \log_2 N$ steps: with $N = 100$ this is about $10^{2.5}$.

Now (S1) is unusable as a *realistic* algorithm: even with a high-performance computer implementation capable of executing 10^{12} operations per second, in the worst case (S1) will require 10^{150} *years.* On a much slower machine (say 100 operations per second) even a "naive" implementation of (S2) will have finished in about 2 *minutes* and (S3) in just over 1 second.

Although a quite extreme case is being considered, this overview of one particular range of algorithmic methods for a computational problem does highlight two significant issues:

H1. The efficiency of an algorithm is a crucial factor in determining its practical usability: if (S1) were the *only known* sorting method, tasks such as organising records in a database would not be possible.

H2. Developments in technology – the platforms on which algorithms are realised – have minimal impact: a reasonable algorithm (S2 or S3) even running on an antiquated very slow machine (100 ops/sec) will easily outperform a very inefficient approach (such as S1) even if this is run on a machine with significant computational power (10^{12} ops/sec)

The study of algorithms for computational problems in argumentation has made notable advances over the last twenty years. There is, however, a significant issue that besets many of its computational concerns: that within the technical classifications of problem difficulty presented in the field of *computational complexity theory* there is powerful evidence that the prospects for identifying efficient solution methods are extremely limited: that is to say, in terms of the sorting method example given, the status of best known worst-case methods for important computational problems in argumentation is more likely to be characterised by (S1) than (S2) or (S3).

Our intention in this chapter is to present a survey of computational complexity results that have been obtained within formal argumentation.

Prior to embarking on this overview, in order to provide some necessarily technical background, we give an very informal basic introduction to the ideas and techniques used in this study.

From a practical point, complexity classification is in particular crucial when one considers implementing argumentation reasoning tasks by a *reduction approach*. That is, instead of designing and implementing complex algorithms and systems from scratch, one might reduce the new reasoning tasks to related formalisms where sophisticated solvers already exist. For instance, for a broad range of argumentation semantics one can reduce the task of computing a set of coherent arguments of an argumentation framework to computing a model of a propositional formula that can be efficiently constructed from the argumentation framework [Besnard and Doutre, 2004]. Now one can exploit the sophisticated systems to deal with propositional formulae to get an efficient system for the encoded argumentation semantics with relatively small effort. In the reduction approach the complexity of the actual problem and the corresponding problem in the target formalism are crucial for the following reasons: given that an actual problem has higher complexity than the designated target problem we know that there is no efficient encoding of our problem and we might consider a different target formalism. On the other hand if the target problem is of higher complexity we may end up with unnecessarily high computational costs. In such a case it might be a good idea to encode the problem within a restriction of the target formalism, providing lower complexity.

The remainder of the chapter is organised as follows. In Section 2 we give a brief introduction to computational complexity. That is we introduce the techniques and complexity classes we will use in the later parts of the chapter. In Section 3 we consider Dung's abstract argumentation frameworks and the main computational problems thereof. In Section 4 we consider computational problems in assumption-based argumentation. In Section 5 we consider computational problems in abstract dialectical frameworks. Finally, in Section 6 we summarise and discuss the presented results as well as related results not covered by this chapter.

2 A brief Introduction to Computational Complexity Theory

In very informal terms, computational complexity theory is the field of computer science concerned with grouping computational problems (in the sense we introduced above) into so-called "complexity classes". Such classes are captured by different resource requirements, typically measured by quantities such as Time (number of steps taken by an algorithm) or Space (amount of "memory" needed). Thus a *complexity class* $\mathcal{C}$ is a *set* of computational problems, and when we say that "problem P is in the complexity class $\mathcal{C}$" (or P has complexity $\mathcal{C}$) this indicates that there *exists* an algorithm that solves P and meets the resource criteria prescribed by $\mathcal{C}$. For example, as illustrated by the methods discussed in the introduction, the computational problem of "sorting n numbers" is in the (function) complexity class of problems solvable in time $n \log n$ (evidenced by method S3).

Now already this basic description raises many issues, among which we have:

a. How do we avoid proliferating "complexity classes" because of different technological capabilities, i.e., having to formulate a "complexity theory for Apple Mac machines", another for IBM hardware, and yet another for Windows O/S, etc. etc.?

b. How do we formalise notions of "input size" and relate such to the computational complexity of a problem?

c. How do we, in a precise sense, group distinct computational problems into collections of similar behaviour?

Before developing these questions further, we observe that it is convenient to focus on *decision problems.* That is to say, problems that separate input *instances* into two disjoint sets:

- **Positive** instances x of problem P: those on which P reports the answer **true** (equivalently, 1 or **yes**).
- **Negative** instances x of problem P: those on which P reports the answer **false** (equivalently, 0 or **no**).

In order to abstract away from the trivialities of platform specifics, algorithms are considered as realised on some standard "*model of computation*". While a huge number of such models have appeared in the technical literature[1] those adopted in computational complexity, ultimately, derive from *Turing machine (*TM*) programs.* The exact specification of these is unimportant for the purposes of this overview. The interested reader is referred to any standard textbook for further details (e.g., [Papadimitriou, 1994; Arora and Barak, 2009]).

By fixing a standard basis for specifying algorithms (that is, TM programs) we obtain methods for addressing questions (b) and (c). At the most rarefied abstract level of TM operation, "input size" is simply the total number of *characters* (symbols) appearing in the input data. Usually (although not invariably) this will take the form of a sequence of *binary* "digits". The important feature is that the input sequence uses only characters from a *fixed finite*

[1]In one form or another the abstraction "model of computation" can be traced back almost a hundred years: its first appearance being with respect to capturing the notion of "computational problems that *can* be solved".

set or *alphabet* no matter whether these characters are digits, letters, or any other type of characters. [2]

While "length of the input string" offers a common basis for comparison, it can be somewhat cumbersome for practical analysis. Fortunately (and certainly in the case of abstract argumentation problems which are our principal interest) there is, usually, some supporting structure to a problem instance which can serve as a size parameter. For example, returning to the example of "sorting", instead of considering the total number of *bits* to represent instances (which could be $n \log_2 k$ when non-negative integers of value $< 2^k$ are involved), since most sorting methods work at the level of numeric comparisons (as opposed to individual bit-level manipulation), the size of an instance can reasonably be viewed as the number of values (N) to be sorted. In the consideration of decision problems arising in Dung's formalism a typical instance will specify an argumentation framework, that is to say a *directed graph*, (A, R) and a subset S of arguments: hence the "obvious" input size parameter is simply "the number of arguments in A". Notice that, total input size is bounded polynomially in n, as the size of each part of input, i.e., of A, R and S, is bounded polynomially in n.

We can now deal with the second part of (b): relating such notions of "size" to problem complexity, in particular precise interpretations of "problem P has lower (time) complexity than problem Q". Notice that such statements combine *two separate* claims:

C1. That there *exists* an algorithm A_P solving P that runs in time $T_P(n)$ on instances of size n.

C2. That *every* algorithm A_Q solving Q takes times at least $T_Q(n)$ on instances of size n and $T_Q(n)$ is "larger" than $T_P(n)$.[3]

Let us focus now on problems concerning AFs in which the dominant input component is a directed graph, (A, R). Considering the character of the algorithm, A_P, associated with this there is an infinite sequence,

$$\{< A_P, R_P >_{(1)}, < A_P, R_P >_{(2)}, \ldots, < A_P, R_P >_{(k)}, \ldots\}$$

for which $< A_P, R_P >_k$ is an AF having exactly k arguments. In addition, the number of steps (run-time) of A_P on the instance $< A_P, R_P >_k$ is not exceeded by any other instance (A, R) in which A has exactly k arguments. Such an instance, $< A_P, R_P >_k$ is called a "*worst-case* input for A_P". In this way the run-time function, T_P, is just the

$T_P(n) =_{\text{def}}$ The number of steps A_P takes when given the input $< A_P, R_P >_{(n)}$

[2]In complexity matters, if such a set contains at least 2 distinct symbols, it makes little difference whether the alphabet has 2 or 1000 or more symbols. In contrast, however, *unary* (single symbol) encodings may lead to notably different algorithmic behaviour.

[3]Typically, one is interested in the asymptotic behaviour with growing input size n, i.e., whether there is an there exists n_0 such that $T_Q(n) >= T_P(n)$ for every $n >= n_0$.

Now, unless we are dealing with a highly artificial and contrived problem, P, one will typically have $T_P(n+1) > T_P(n)$.[4] This clarifies the precise meaning of (C1), and by a similar analysis we can associate run-time functions, T_Q with every algorithm solving Q. The statement "P has smaller complexity than Q" is thus a *positive* (upper bound) claim about algorithms for problem P and a *negative* lower bound claim about all algorithms for Q: as the number of arguments in A increases we will see a growing disparity between the worst-case time that P requires to deliver an answer (using A_P) compared to the worst-case time that *any algorithm*, A_Q takes to deliver its answer.

Much of the focus of computational complexity theory is in grouping problems into classes where this disparity is at its most extreme: these extremes and the techniques for placing problems at either end of the spectrum of difficulty are the subject of the next subsection.

2.1 Basic Complexity Classes

Here we briefly review the *complexity classes* used in this work and their relations. As discussed above the high-level idea of complexity theory is to group problems with similar resource requirements in complexity classes and also put these classes into an order so that we can distinguish between "easier" and "harder" problems.

2.1.1 Polynomial-Time

By convention, a problem is viewed as having an efficient algorithmic solution if it can be placed into the class P (*polynomial-time*) of all problems that have a polynomial-time algorithm, i.e., an algorithm that for each instance x (of size $|x|$) produces its answer after at most $|x|^k$ steps, for a fixed constant k. It is noted that this a rather coarse-grained classification: problems whose fastest algorithm runs in time n^{100} are considered to be "efficiently solvable". This may seem rather arbitrary, however, there is a very noticeable performance difference between methods whose run-time is bounded by n^k and those that cannot be so bounded.

An important subclass we will consider is L (*logarithmic space*), which consists of the problems that can be solved in logarithmic space (not counting input and output) and polynomial-time. Just as P is seen as the class of computational problems with efficient "sequential" algorithms, so L is the class having efficient "parallel" algorithms (see, e.g., [Greenlaw *et al.*, 1995]).

We consider problems in the classes L, P to be computationally tractable, while we will consider problems in all the other classes in this chapter to be intractable or computationally hard.

[4]We are, of course, ignoring minor issues whereby P requires (A, R) to have a particular structure rendering frameworks with some numbers of arguments unsuitable, e.g., problems in which A must have an even number of arguments and are ill-defined when the size of A is an odd number.

2.1.2 The classes NP, coNP and DP

Often a decision question can be solved by finding a witness for the instance satisfying the questioned property. For instance if we ask whether an AF has a stable extension, a way to answer that positively would be to actually compute a stable extension as witness.

Taking this view, we may associate with any instance x of a decision problem Q, a set $W(x)$ of potential *witnesses* that x has the property of interest. For example, for admissibility semantics if we are interested in whether a specified argument p is credulously accepted with respect to admissibility the instances have the form $((A,R),p)$ (with $p \in A$) and potential witnesses are all subsets of $A \setminus \{p\}$. A witness S in this set is valid for the instance if and only if the set $S \cup \{p\}$ is admissible.

The class NP. The complexity class NP (*non-deterministic polynomial-time*) can be characterised by such witnesses. A decision problem is in the class NP if (i) for each instance x there is a set $W(x)$ of potential witnesses, which are of polynomial size in $|x|$, such that (ii) one can verify that a $y \in W(x)$ is actually a witness for x in polynomial time and (iii) x is a "yes" instance if and only if at least one $y \in W(x)$ is a witness for x.

In the above example for an AF F the potential witnesses $W(F)$ would be all the subsets of arguments. Verifying whether a set is admissible is in polynomial time and F is a positive instance iff [5] at least one of these sets is a stable extension.

Formally the specification of a decision problem in terms of witness sets can be seen in the following way. Let x be an instance of a (decision) problem Q we write, $Q(x) = 1$ if x is a "yes" instance of Q, and $Q(x) = 2$ if x is a "no" instance of Q. We have a *binary* relation $W_Q(x,y)$ for which $< x,y > \in W_Q$ iff y is a valid witness that x is a positive instance of the (decision) problem Q. This yields,

$$Q(x) = 1 \quad \Leftrightarrow \quad \exists\ y \in W(x) : \ < x,y > \in W_Q$$

Thus the class NP can be interpreted as those decision problems, Q, for which the membership problem $< x,y > \in W_Q$ can be decided in time polynomial in the size of x. Notice that this constraint immediately forces y (a valid witness) also to have size polynomial in $|x|$.

The class coNP. The quantifier in our formalisation of NP is an existential one. If we modify this to

$$\forall\ y \in W(x) \ \ < x,y > \notin W_Q$$

then we obtain the important class coNP capturing instances that *do not* have the property of interest. For example if we wish to demonstrate that an argument, x is inadmissible then it suffices to show "for every subset S of A the set $S \cup \{x\}$ either is not conflict-free or has an undefended argument".

[5] We will frequently use "iff" as short form for "if and only if".

We, now, briefly summarise some developments of this view of "decision problems as witness testing".

The first of these is the concept of "oracles": in an *oracle* computation we are provided with a "black-box" for witness testing which given a problem instance x provides the answer for "$Q(x) = 1$?" in a single computational step. Now such oracle machines may be considered with respect to arbitrary complexity classes, so P^A describes the class of "decision problems that have a polynomial-time algorithm *that makes use of an oracle for a decision problem in the complexity class* A".

For example, for an NP oracle we might use "existence of a stable extension". In exploiting such an oracle to solve another decision problem B "in polynomial-time" we might use an algorithm which, given an instances p of B, constructs one or more (but at most polynomial in $|p|$) frameworks F_1^p, F_2^p, etc., using the answer to "does F_k^p have a stable extension?" to determine if p should be accepted as an instance of B.

The class DP. A number of important classes have been found to occur in complexity analysis of argumentation via such oracles. Among them we have DP, the so-called "difference class" of decision problems whose members are captured by the intersection of instances x accepted by a problem L_1 (with $L_1 \in \mathsf{NP}$) and x accepted by a problem L_2 (with $L_2 \in$ coNP). For example the set of pairs of propositional formulae $<\varphi_1, \varphi_2>$ in which φ_1 is satisfiable and φ_2 is not so (the SAT-UNSAT problem) is in DP since its positive instances are the intersection of

$$\begin{array}{rcl} L_1 & = & \{ <\varphi, \psi> \ : \ \varphi \text{ is satisfiable } \} \\ L_2 & = & \{ <\varphi, \psi> \ : \ \psi \text{ is unsatisfiable } \} \end{array}$$

2.1.3 The Polynomial-time Hierarchy

The notion of "oracle" can also be used in defining the important "Polynomial-time Hierarchy" (PH). Consider the quantifier formulation of NP and coNP

$$\begin{array}{c} \exists\ y \ \ <x, y> \in W_Q \\ \forall\ y \ \ <x, y> \notin W_Q \end{array}$$

This uses a (polynomial) time decidable *binary* relation and a *single* quantifier. We could, however, extend this further, e.g.

$$\begin{array}{c} \exists\ y_1\ \forall\ y_2\ W_Q^2(x, y_1, y_2) \\ \forall\ y_1\ \exists\ y_2\ \neg W_Q^2(x, y_1, y_2) \end{array}$$

or even

$$\begin{array}{c} \exists\ y_1\ \forall\ y_2\ \exists\ y_3\ W_Q^3(x, y_1, y_2, y_3) \\ \forall\ y_1\ \exists\ y_2\ \forall\ y_3\ \neg W_Q^3(x, y_1, y_2,\ y_3) \end{array}$$

and, generally

$$Q_1\ y_1\ Q_2\ y_2\ \ldots\ Q_k\ y_k\ W_Q^k(x, y_1, y_2, \ldots, y_k)$$

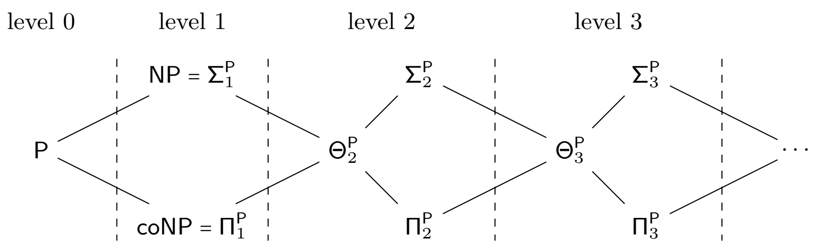

Figure 1. Levels of the polynomial-hierarchy. An edge denotes that all problems in the class on the left side are also contained in the class on the right side. Notice that only classes relevant for this chapter are shown.

In the last case we have k *alternating* quantifiers (that is $\exists$ is followed by $\forall$ and vice-versa) and the predicate $W_Q^k(x, y_1, y_2, \ldots, y_k)$ is decidable in time polynomial in $|x|$. When the opening (Q_1) quantifier is $\exists$ this defines the class of languages Σ_k^P; when this quantifier is $\forall$ we have Π_k^P. The *polynomial-hierarchy* (PH) is

$$\mathsf{PH} = \bigcup_{k=0}^{\infty} \Sigma_k^P = \bigcup_{k=0}^{\infty} \Pi_k^P$$

We sometimes refer to *levels of the polynomial-hierarchy*, where the k-th level is formed by the classes Σ_k^P and Π_k^P. For instance on the first level there are the classes NP and coNP while on the second level there are the classes Σ_2^P and Π_2^P. Moreover, we will later introduce a further family of complexity classes Θ_k^P, and will consider the class Θ_k^P to be in the k-th level of the polynomial hierarchy (cf. Figure 1).

How does this relate to the concept of "oracle machines"? The answer to this is given by examining the quantifier structure in more depth. We have required the inner most $(k+1)$-ary predicate W_Q^k to be (deterministic) polynomial-time computable. If we have an oracle for the decision problem implied by removing the *first* quantifier then this class of languages (when $Q_1 = \exists$) is formed by languages which belong to NP given access to a Σ_{k-1}^P oracle, conventionally denoted $\mathsf{NP}^{\Sigma_{k-1}^P}$, while Π_k^P (first quantifier is $\forall$) are those problems computable in coNP with a Σ_{k-1}^P oracle.

For example, consider the "quantified SAT problem" one version of which involves two disjoint sets of propositional variables, X and Y, and asks of a given formula $\varphi(X, Y)$ whether $\exists\, \alpha_X \forall\, \beta_Y\ \varphi(\alpha_X, \beta_Y)$, that is, "can we find an assignment of values to the X variables (α_X) which renders the formula $\varphi(\alpha_X, Y)$ a tautology?". Given an oracle for satisfiability we can test $\varphi(\alpha_X, Y) \equiv \top$ to be a "single step", by testing the negated formula for satisfiability. The implied NP question ("can we find ...") is handled by a "polynomial" algorithm with access to this oracle so that $\Sigma_2^P = \mathsf{NP}^{\mathsf{NP}}$. Notice that, in our example we can also directly use an oracle for the coNP problem of tautology which gives us $\mathsf{NP}^{\mathsf{NP}} = \mathsf{NP}^{\mathsf{coNP}}$. That is, for an oracle machine it does not matter whether it has access to a NP or coNP oracle (or more generally to a $\Sigma_{k-1}^{\mathsf{P}}$ or Π_{k-1}^{P} oracle) as it can easily switch "yes" and "no" answers after an oracle call.

In total we can treat PH as groups of problems described via alternation of a fixed number (k) of quantifiers or in terms of polynomial-time oracle machines exploiting oracles to the immediately lower level, i.e. both Σ_k^{P} and Π_k^{P} use access to a $\Sigma_{k-1}^{\mathsf{P}}$ oracle.

Moreover, we consider related oracle complexity classes that have only restricted access to their oracle. Concretely, the class $\Theta_k^{\mathsf{P}} = \mathsf{P}^{\Sigma_{k-1}^{\mathsf{P}}[\log(|x|)]}$ contains problems decidable by a deterministic polynomial-time algorithm that is allowed to make a logarithmic number (w.r.t. input size) of $\Sigma_{k-1}^{\mathsf{P}}$-oracle calls. An alternative characterisation for Θ_k^{P} is that the deterministic algorithm is allowed to make linearly (in the input size) so called non-adaptive calls to the $\Sigma_{k-1}^{\mathsf{P}}$-oracle, that is all oracle calls are evaluated in parallel. When using this alternative characterisation the class Θ_k^{P} is sometimes also denoted as $\mathsf{P}_{\parallel}^{\Sigma_{k-1}^{\mathsf{P}}}$.

Notice that all complexity classes we consider can be solved by in worst-case exponential time algorithms that only require polynomial space. However, problems on different levels of the polynomial hierarchy behave quite differently, and methods that work reasonable for problems at the NP, coNP level might not work as well for Σ_2^{P} or Π_2^{P}-hard problems.[6]

2.2 Reductions, Hardness and Completeness

At the conclusion of the preceding sub-section we referred to particular problems as "among the hardest Π_2^{P} problems". This (at the time of writing) does *not* mean we can formally demonstrate that *every* problem that can be classified as belonging to Π_2^{P} may be solved by a (deterministic) algorithm whose run-time is no worse than that of the best algorithm for, e.g., semi-stable skeptical reasoning. It does, however, mean the following: *if* we can find an NP (or even P) algorithm for skeptical semi-stable reasoning *then* we can construct NP (resp. P) algorithms *for every problem in the class* Π_2^{P}, i.e. it would follow that the classes Π_2^{P} and NP (resp. P) contained *exactly* the same decision problems. Despite this, throughout this work we will follow the standard assumptions in computational complexity theory and consider problems in higher levels of the polynomial hierarchy to be harder than problems in the lower levels of the polynomial-hierarchy.[7]

2.2.1 Polynomial Reducibility

The key idea used to support this claim is that of *polynomial reducibility.* Suppose we have two decision problems – F and G say. These have sets of instances I_F and and I_G. Now, while we may not be able to formally prove

[6]In the context of formal argumentation such a behaviour can be observed at the results of the First International Competition on Computational Models of Argumentation [Thimm and Villata, 2015; Thimm *et al.*, 2016].

[7]This relates to two famous open problems in complexity theory, namely to show that $\mathsf{P} \neq \mathsf{NP}$ and to show that the polynomial hierarchy is an infinite hierarchy and does not collapse at a certain level, i.e, $\Sigma_k^{\mathsf{P}} \neq \Sigma_{k+1}^{\mathsf{P}}$ for all $k > 0$. Both statement are widely believed but (at the time of writing) there are no formal proofs.

that either problem is intractable we can argue, using the following approach, that if G is decidable in polynomial time then F is also.

Build an *efficient* procedure, τ, transforming any instance of F into an instance of G, i.e., $\tau : I_F \rightarrow I_G$ and with the property that $x \in I_F$ is a positive instance of F iff $\tau(x) \in I_G$ is a positive instance G.

With such a transformation procedure any algorithm for G can be used as a sub-routine to give an algorithm for F. So were it the case that $G \in \mathsf{P}$, as τ is efficient, it follows that $F \in \mathsf{P}$ also. By contraposition, it can be shown that if $F \notin \mathsf{P}$ it must be the case that $G \notin \mathsf{P}$. When such a transformation can be found between decision problems F and G as above, we say that "F is *polynomially-reducible* to G" using the notation $F \leq_p G$ to describe this relationship.

Notice that the form of instances for F and G do not have to be identical: G could, for example, be a decision problem concerning propositional formulae and F one whose instances are AFs: a transformation between the two would define how a formula is constructed from a given AF.

2.2.2 Hardness and Completeness

The concept of reducibility offers a means to argue that the class NP differs from the class P and formalise the notion of "hardest" problem of a complexity class. Intuitively we consider a problem to be among the "hardest" problems of a complexity class if an efficient method for the problem would yield efficient methods for *all* problems in the class. That is, an efficient method for just *one* of the "hardest" problems would yield efficient methods for *all* problems in the class. Formally, for any complexity class, $\mathcal{C}$, a decision problem G is said to be *$\mathcal{C}$-hard* if

$$\forall\ F \in \mathcal{C} \quad F \leq_p G$$

If, in addition $G \in \mathcal{C}$ then G is said to be *$\mathcal{C}$-complete.*

So the class of NP-complete problems are those problems in NP to which any other problem in NP can be polynomially reduced. The class of known NP-complete problems includes many well-studied combinatorial, logic, and graph problems for which no efficient algorithm has been discovered, in some cases after several centuries of study. Among these are: deciding if a propositional formula has a model (SAT); deciding if a graph has a path that contains every vertex exactly once (a variant of the so-called Travelling Salesperson Problem), deciding if a given argument is acceptable w.r.t. Dung's stable semantics.

It is considered highly unlikely that every single one of these problems can be solved efficiently. In order to prove that no NP-complete problem can be solved in polynomial time it would suffice to show that just *one* could not be.

Thus, a proof that a problem G is NP-complete is seen as very strong evidence that F is intractable. Given the transitivity of $\leq_p$ all that is required to proof NP-hardness is a known NP–hard problem (F say) and a transformation, τ, to witness $F \leq_p G$. In order to obtain NP-completeness one has to additionally give a procedure that decides G and fits the definition of NP, we sometimes call such a procedure a NP-algorithm (more generally $\mathcal{C}$-algorithm for complexity class $\mathcal{C}$).

Next let us briefly reconsider our restrictions on reductions. All the complexity classes $\mathcal{C}$ considered in this handbook chapter, except L, are *closed under polynomial reductions*, that is whenever a problem A can be polynomial-time reduced to a problem $B \in \mathcal{C}$ then also A belongs to $\mathcal{C}$. Notice that any problem in the class P and in particular those in the class L would be complete for P with respect to polynomially-reducibility. Thus when differentiating between problems in L and P one uses the concept of *logspace-reducibility* where the transforming procedure is required to work in logarithmic space. In particular, P-completeness results are stated w.r.t. logspace-reducibility.

2.2.3 Complete Problems for the Polynomial Hierarchy.

To show that a problem A is hard for a specific complexity class $\mathcal{C}$ one typically starts from a problem B that is complete for the class $\mathcal{C}$ and provides a reduction from B to A. In the following we briefly introduce some canonical complete problems for the complexity classes in the polynomial-hierarchy.

As already mentioned a famous NP-complete problem is deciding if a propositional formula has a model (SAT). On the other side standard coNP-complete problems are verifying that a propositional formula is a tautology (TAUT) or that a propositional formula has no model (UNSAT). The canonical DP-complete problem is the earlier mentioned $\mathsf{SAT}-\mathsf{UNSAT}$ problem.

The complete problems for classes Σ_k^{P} and Π_k^{P} are given by quantified SAT problems (cf. Section 2.1.3). That is, one is given a propositional formula φ whose variables are split up in k disjoint sets $X_1, \dots X_k$ and the possible assignments for these sets X_1 are quantified with alternating existential and universal quantifiers. A quantified boolean formula (QBF) is then of the form

$$Q_1 X_1\, Q_2 X_2 \dots Q_k X_k\, \varphi(X_1, \dots X_k)$$

with Q_i being alternating $\exists, \forall$ quantifiers (i.e., $\exists$ is followed by $\forall$ and vice versa). Deciding whether a QBF with k quantifiers and $Q_1 = \exists$ is valid is the canonical Σ_k^{P}-complete problem while deciding whether a QBF with k quantifiers and $Q_1 = \forall$ is valid is the canonical Π_k^{P}-complete problem.

As the second level of the polynomial-hierarchy is of special interest in the setting of formal argumentation we next introduce minimal model satisfiability (MINSAT) as another problem that is Σ_2^{P}-complete [Eiter and Gottlob, 1993]. In the MINSAT problem one is given a propositional formula φ over variables X and a variable x thereof and has to decide whether the variable is true in some minimal model of φ.

2.3 Parametrized Complexity

Classical complexity theory deals with the complexity of problems w.r.t. the size of the instance. However, often the complexity of a problem does not mainly depend on the size of an instance but on some (structural) properties of the instance. That is, we can solve huge instances efficiently as long as some property is satisfied or the obstacles in the structure are bounded independent of the size. The field of parametrized complexity theory[8] deals with this observation. The idea is to consider parametrized problems, i.e., the problem description contains a designated parameter (typically an integer) which is instantiated by each problem instance. An example for a parametrized problem is given a graph G and an integer parameter k deciding whether G has a clique of size k.

Definition 2.1 *A parametrized (decision) problem is called* fixed-parameter tractable *(or in* FPT*) if it can be determined in time* $f(k) \cdot |x|^{O(1)}$ *for a computable function* f*.*

Now given that a problem is in FPT and just consider those instances where the parameter is bounded by some constant then we can decide an instance with a polynomial-time algorithm. Only the constants in the polynomial-time bound are affected by the parameter, but not the order of the polynomial.

Beside FPT there is also a weaker form of tractability w.r.t. a parameter allowing the order of the polynomial to depend on the parameter.

Definition 2.2 *A parametrized (decision) problem is* slice-wise polynomial *(or in* XP*) if it can be determined in time* $f(k) \cdot |x|^{g(k)}$ *for computable functions* f, g*.*

A problem in XP can be solved in polynomial time if we bound the parameter, but distinguishing it from FPT the order of the polynomial may highly depend on the bound of the parameter.

Let us briefly present the relations between the classes FPT, XP and P:

$$\mathsf{P} \subseteq \mathsf{FPT} \subseteq \mathsf{XP}$$

When considering unparametrized problems and talking about FPT we have to mention the used parameter explicitly. Thus we say a problem P is fixed-parameter tractable w.r.t. the parameter k iff the corresponding parametrized problem (P, k) is fixed-parameter tractable.

3 Complexity of Dung's Abstract Argumentation

We start our analysis with Dung's Abstract Argumentation Frameworks. These frameworks consist of a set of abstract arguments and a relation representing directed conflicts or attacks between these arguments. Then rules, so called semantics, are defined to select coherent sets of arguments that can be accepted simultaneously. That is, abstract argument frameworks focus on the core issue of argumentation, i.e., resolving conflicts between arguments.

This part of the chapter is organised as follows: In Section 3.1 we recall the basic definitions of Dung's Abstract Argumentation Frameworks and the

[8]We just briefly introduce the concepts relevant for this chapter; for comprehensive introductions to parametrized complexity the reader is referred to [Flum and Grohe, 2006; Niedermeier, 2006; Cygan *et al.*, 2015].

most popular semantics for it. That is, beside the semantics introduced by Dung [Dung, 1995], we consider ideal [Dung *et al.*, 2007], semi-stable [Verheij, 1996; Caminada *et al.*, 2012], stage [Verheij, 1996] and cf2 [Baroni *et al.*, 2005] semantics. Then in Section 3.2 we discuss the core computational Problems of Abstract Argumentation and define formal variants that serve as basis for the complexity analysis in Section 3.3. In Section 3.4 we consider potential computational advantages when the argumentation frameworks fall into some specific graph class. The potential of techniques from parametrized complexity theory is discussed in Section 3.5. In Section 3.6 we discuss some computational issues specific to labelling-based argumentation semantics. Finally, in Section 3.7 we summarise and discuss the presented results and give additional pointers to literature.

3.1 Dung's Abstract Argumentation Frameworks

In this section we introduce (abstract) argumentation frameworks [Dung, 1995] and recall the semantics we study (for a comprehensive introduction the reader is referred to [Baroni *et al.*, 2011a] or the earlier chapter of this handbook dedicated to Dung's Abstract Argumentation Frameworks).

Definition 3.1 *An* argumentation framework (AF) *is a pair $F = (A, R)$ where A is a (finite) set of arguments and $R \subseteq A \times A$ is the attack relation. The pair $(a, b) \in R$ means that a attacks b. We say that 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 exists $c \in S$ such that $(c, b) \in R$.*

Indeed when studying computational complexity we are only interested in AFs where the set A is finite.

Semantics for argumentation frameworks are defined as 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 *na*, *gr*, *st*, *ad*, *co*, *cf2*, *id*, *pr*, *sst* and *stg* which stand for naive, grounded, stable, admissible, complete, cf2, ideal, preferred, semi-stable and stage semantics, respectively. Towards the definition of these semantics we have to introduce a few more formal concepts.

Definition 3.2 *Given an AF $F = (A, R)$, the* characteristic function $\mathrm{F}_F : 2^A \to 2^A$ *of F is defined as $\mathrm{F}_F(S) = \{x \in A \mid x \text{ is defended by } S\}$.*

Definition 3.3 *For a set $S \subseteq A$ and an argument $a \in A$, we say S* attacks *a (resp. a attacks S) in case there is an argument $b \in S$, such that $(b, a) \in R$ (resp. $(a, b) \in R$). Moreover, for a set $S \subseteq A$, we denote the set of arguments attacked by (resp. attacking) S as $S_R^+ = \{x \mid S \text{ attacks } x\}$ (resp. $S_R^- = \{x \mid x \text{ attacks } S\}$), and define the* range *of S as $S_R^\oplus = S \cup S_R^+$.*

We are now prepared to give the formal definitions of the abstract argumentation semantics we will consider. Notice that we restrict ourselves to extension-based semantics, but some aspects of labelling-based semantics are discussed in Section 3.6).

Definition 3.4 *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 $(a, b) \in R$. $cf(F)$ denotes the collection of conflict-free sets of F. For a conflict-free set $S \in cf(F)$, it holds that*

- *$S \in na(F)$, if there is no $T \in cf(F)$ with $T \supset S$;*
- *$S \in st(F)$, if $S^+_R = A \setminus S$;*
- *$S \in ad(F)$, if $S \subseteq \mathrm{F}_F(S)$;*
- *$S \in co(F)$, if $S = \mathrm{F}_F(S)$;*
- *$S \in gr(F)$, if $S \in co(F)$ and there is no $T \in co(F)$ with $T \subset S$;*
- *$S \in pr(F)$, if $S \in ad(F)$ and there is no $T \in ad(F)$ with $S \subset T$;*
- *$S \in id(F)$ if S is $\subseteq$-maximal among $\{S' \mid S' \in ad(F)$ and $S' \subseteq E$ for each $E \in pr(F)\}$.*
- *$S \in sst(F)$, if $S \in ad(F)$ and there is no $T \in ad(F)$ with $S^\oplus_R \subset T^\oplus_R$;*
- *$S \in stg(F)$, if there is no $T \in cf(F)$, with $S^\oplus_R \subset T^\oplus_R$.*

We recall that for each AF F, the grounded semantics yields a unique extension, the grounded extension, which is the least fixed-point of the characteristic function F_F.

Finally, we give the recursive definition of cf2 semantics (see [Baroni *et al.*, 2005; Gaggl and Woltran, 2013] for further reference).

Definition 3.5 *Given an argumentation framework $F = (A, R)$, then $E \in cf2(F)$, if*

- *$E \in na(F)$ if $|SCCs_F| = 1$, and*
- *$\forall S \in SCCs_F\ (E \cap S) \in cf2(F{\downarrow}_{UP_F(S,E)})$ otherwise.*

Here $SCCs_F$ denotes the set of strongly connected components of F, and for any $E, S \subseteq A$, $UP_F(S, E) = \{a \in S \mid \nexists b \in E \setminus S : (b, a) \in R\}$. Moreover, for $S \subseteq A$ we use $F{\downarrow}_S$ to denote the AF $(A \cap S, R \cap S \times S)$, i.e., the AF that one obtains when restricting F to the arguments in S.

We recall some basic properties of these semantics. For each AF F we have the following subset relations:

$$st(F) \subseteq stg(F) \subseteq na(F) \subseteq cf(F),$$

$$st(F) \subseteq sst(F) \subseteq pr(F) \subseteq co(F) \subseteq ad(F) \subseteq cf(F),$$

and $st(F) \subseteq cf2(F) \subseteq na(F)$. Furthermore, for any of the considered semantics σ except stable semantics we have that $\sigma(F) \neq \emptyset$ holds, i.e., these semantics always propose at least one extension. Grounded and ideal semantics always yield exactly one extension, thus we also say that they are unique status semantics, and the ideal extension is always a complete extension. With slight abuse of notation we sometimes use $gr(F)$, resp. $id(F)$, to refer to the unique grounded, resp. ideal, extension of F. Moreover, stable, semi-stable, and stage semantics coincide for AFs with at least one stable extension.

3.2 Computational Problems

In general an argumentation semantics assigns several extensions to a single framework, but at the end of the day we want to make a conclusion about arguments. There are different ways to aggregate the acceptance status of an argument from the set of extensions, which mirrors different levels of scepticism. First it is quite clear that an argument which is in no extension at all should not be accepted, but in certain situations it might be fine to accept an argument that appears in just one extension, this is what we will call *credulous reasoning.* On the other hand in situations where one has to be cautious one might demand that an argument is in all extensions, we refer to this as *skeptical reasoning.*

These reasoning modes give rise to the following computational problems for argumentation semantics σ.

- *Credulous Acceptance $Cred_\sigma$*: Given AF $F = (A, R)$ and an argument $a \in A$. Is a contained in some $S \in \sigma(F)$?

- *Skeptical Acceptance $Skept_\sigma$*: Given AF $F = (A, R)$ and an argument $a \in A$. Is a contained in each $S \in \sigma(F)$?

If an AF has no stable extensions, according to our definition of skeptical acceptance, all arguments are skeptically accepted. This may be unwanted and hence one might consider a variation of the skeptical acceptance problem asking whether an argument is contained in all extensions and there exists at least one extension [Dunne and Wooldridge, 2009].

In practice, one often is interested in computing all extensions or a certain number of extensions. However, complexity theory provides much better tools for decision problems than for function problems and thus one usually sticks to decision problems when analysing the computational problems, in our case credulous and skeptical acceptance. Nevertheless, the complexities of credulous and skeptical acceptance together give a good impression of the complexity to actually compute the extensions.

Beside these reasoning problems there are also several other computational problems in the field of abstract argumentation. In this work we consider the most prominent ones of them. First of all one might be interested in verifying given extensions, which may come from another agent or potentially corrupted file, or simply as part of a reasoning algorithm.

- *Verification of an extension Ver_σ*: Given AF $F = (A, R)$ and a set of arguments $S \subseteq A$. Is $S \in \sigma(F)$?

Another task is deciding whether an AF provides any coherent conclusion. That can be deciding whether it has at least one extension, in the case of stable semantics, or whether it has an extension different from the empty set, for all the other semantics under our consideration.

- *Existence of an extension $Exists_\sigma$*: Given AF $F = (A, R)$. Is $\sigma(F) \neq \emptyset$?
- *Existence of a non-empty extension $Exists_\sigma^{\neg\emptyset}$*: Given AF $F = (A, R)$. Does there exist a set $S \neq \emptyset$ such that $S \in \sigma(F)$?

Finally, we will also consider the problem of deciding whether a semantics yields a unique extension for a given an AF (cf. Chapter 18 of this handbook).

- *Uniqueness of the solution $Unique_\sigma$*: Given AF $F = (A, R)$. Is there a unique set $S \in \sigma(F)$, i.e., is $\sigma(F) = \{S\}$?

3.3 Computational Complexity

A typical complexity analysis of a problem consists of two parts. First, we have to give an upper bound for the complexity of the problem. That is, we have to either give an algorithm showing the problem can be solved within a class $\mathcal{C}$ or we reduce the problem to another problem already shown to be in the class $\mathcal{C}$. Second, we want to prove lower bounds for the complexity of the problem. That is, we consider a problem that was shown to be hard for some complexity class $\mathcal{C}'$ and reduce it to the current problem. That is, we show the problem to be $\mathcal{C}'$-hard. In case that the classes $\mathcal{C}$ and $\mathcal{C}'$ coincide we obtain that the studied problem is $\mathcal{C}$-complete, and have an exact classification of the complexity of the problem.

The complexity landscape of abstract argumentation semantics is given in Table 1 and discussed below. For Dung's semantics the "in P" and "trivial" are immediately by properties of the corresponding semantics [Dung, 1995]; results for naive semantics are due to Coste-Marquis *et al.* [2005]; results for stable, admissible and preferred semantics follow from results on logic programs by Dimopoulos and Torres [1996], except for the Π_2^{P}-completeness of $Skept_{pr}$ which is due to Dunne and Bench-Capon [2002]; the complexity of ideal semantics is due to Dunne [2009]; results for complete semantics are due to Coste-Marquis *et al.* [2005]; results for semi-stable and stage semantics are due to Caminada *et al.* [2012] and Dvořák and Woltran [2010]; the results for *cf2* semantics are due to Gaggl and Woltran [2013] and the analysis of polynomial-time problems that distinguishes problems that can be solved in L from problems that are P-complete is due to Dvořák and Woltran [2011] [Dvořák, 2012a].

In accordance with the above we will first consider upper bounds for the introduced reasoning problems and then discuss hardness results for them.

3.3.1 Upper Bounds for the Computational Complexity

Most of the problems we introduced in the previous section will fall into one of the complexity classes based on non-deterministic algorithms, e.g. NP and

Table 1. Complexity of Dung's abstract argumentation ($\mathcal{C}$-c denotes completeness for class $\mathcal{C}$).

σ	$Cred_\sigma$	$Skept_\sigma$	Ver_σ	$Exists_\sigma$	$Exists_\sigma^{\neg\emptyset}$	$Unique_\sigma$
cf	in L	trivial	in L	trivial	in L	in L
na	in L	in L	in L	trivial	in L	in L
gr	P-c	P-c	P-c	trivial	in L	trivial
st	NP-c	coNP-c	in L	NP-c	NP-c	DP-c
ad	NP-c	trivial	in L	trivial	NP-c	coNP-c
co	NP-c	P-c	in L	trivial	NP-c	coNP-c
cf2	NP-c	coNP-c	in P	trivial	in L	in P
id	Θ_2^{P}-c	Θ_2^{P}-c	Θ_2^{P}-c	trivial	Θ_2^{P}-c	trivial
pr	NP-c	Π_2^{P}-c	coNP-c	trivial	NP-c	coNP-c
sst	Σ_2^{P}-c	Π_2^{P}-c	coNP-c	trivial	NP-c	in Θ_2^{P}
stg	Σ_2^{P}-c	Π_2^{P}-c	coNP-c	trivial	in L	in Θ_2^{P}

coNP, and thus most of the upper bound are by guess and check algorithms that first non-deterministically guess a potential extension and then verify that it is indeed an extension and satisfies the desired properties.

Standard Reasoning Procedures. The standard algorithm for credulous acceptance first non-deterministically guesses a set of arguments, and then verifies that the set is an extension for the considered semantics and contains the argument under question. The answer to the credulous acceptance query is yes if at least one of the possible guesses evaluates to true. Now let V be the complexity for verifying an extension then the above gives use a NP^{V} algorithm for credulous acceptance. We next consider skeptical acceptance and show that it has a $\mathsf{coNP}^{\mathsf{V}}$ algorithm. To show that a problem falls into a $\mathsf{coNP}^{\mathcal{C}}$ class one could follow the definition of coNP and give an algorithm that first guesses a potential witness and then tests whether the potential witness satisfies certain conditions that can be tested in P^{V}. This conditions have to be such that an instance is positive iff all possible guesses evaluate to true. However, often it is more convenient to consider the complementary problem and provide a $\mathsf{NP}^{\mathcal{C}}$ algorithm for that problem. That is, instead of skeptical acceptance we consider the problem of showing that an argument is not skeptically accepted. The standard algorithm non-deterministically guesses a set of arguments, and then verifies that it is an extension and does not contain the argument under question. The answer to the skeptical acceptance query is yes only if each possible guess evaluates to false. Let V be again the complexity for verifying an extension then the above gives use a $\mathsf{coNP}^{\mathsf{V}}$ algorithm for skeptical acceptance. Towards upper bounds for *Cred* and *Skept* we next consider upper bounds for the verification problems.

Verifying Extensions. For conflict-free, naive, stable, admissible and complete semantics we only have to check whether for the given set certain attacks exist, respectively do not exist. For instance to verify a stable extension we have to verify that (a) between arguments in the extension there is no attack and (b) that all arguments not in the extension are attacked by at least one argument in the extension. This can clearly be done in polynomial time and as it only needs two pointers to arguments also in logarithmic space.[9] Since polynomial time oracles do not add any computational power they can be neglected, i.e., $\mathsf{NP}^{\mathsf{P}} = \mathsf{NP}$ and $\mathsf{coNP}^{\mathsf{P}} = \mathsf{coNP}$. This gives NP, resp. coNP, upper bounds for credulous and skeptical acceptance under these semantics.

Next consider semantics that require maximisation, that are pr, sst, stg (we will see later that for ideal semantics no maximisation is required). Again the basic criterion of being admissible or conflict-free can be easily checked in polynomial time but the maximality criterion adds some complexity. To show that checking whether a set S is an extension is in coNP we again give a non-deterministic algorithm for the complimentary problem, of falsifying the set S to be an extension. This is done by first testing whether S is not admissible (for pr, sst) or not conflict-free (for stg) and then guessing a set $T \supset S$ and testing whether it is admissible (for pr, sst) or conflict-free (for stg). The algorithm successfully falsifies the set S to be an extension iff the first test succeeds or the second test succeeds for at least one guess. In other words, the set S is an extension only if the first and the second tests fails for all possible guessed sets T. Combined with the NP^{V}, $\mathsf{coNP}^{\mathsf{V}}$ resp., algorithm for credulous, skeptical resp., acceptance the above coNP-algorithms for verification give Σ_2^{P}, resp. Π_2^{P} algorithms, for the credulous, resp. skeptical, acceptance problems.

Improved Procedures. For many semantics the above upper bounds are already optimal, but there are some cases where we can improve over them.

First, consider *conflict-free* and naive sets. If an argument is not self-attacking then it certainly will appear in a conflict-free set and thus also in a naive extension. Thus credulous acceptance can be decided by just testing whether the argument under question is self-attacking. Considering skeptical acceptance we have that the empty set is always conflict-free and admissible. Thus for conflict-free and admissible semantics we can reply “no” to each skeptical acceptance query without looking at the actual framework.

For *naive sets* we know that an argument is in a naive set iff it has no self-attack and none of its neighbours is in the set. Thus for skeptical acceptance we just have to test whether the argument is not self-attacking and all of its neighbours are not credulously accepted, i.e., they are self-attacking.

The *grounded* semantics can be computed by iterating the characteristic function until the least fixed-point is reached [Dung, 1995]. The characteristic function can be computed in polynomial time and, as the least fixed-point is reached after at most linearly many iterations. That is, the grounded extension

[9]For the corresponding result for cf2 semantics see [Nieves *et al.*, 2009; Gaggl and Woltran, 2013].

can be computed in polynomial time and the decision problems can then be easily answered. Moreover, as the grounded extension is the unique minimal complete extension skeptical acceptance for complete semantics is exactly the problem of testing whether an argument is contained in the grounded extension and thus in polynomial time.

As each admissible set can be extended to a preferred extension and each preferred extension is admissible we have $Cred_{ad} = Cred_{pr}$. That is, for *credulous acceptance* under *preferred* semantics it suffices to consider admissible sets and thus an NP algorithm suffices.

Finally, for *ideal* semantics there is an alternative characterisation that allows for a Θ_2^P algorithm [Dunne, 2009]. That is, the ideal extension is the maximal admissible set that is not attacked by any other admissible set. The algorithm first computes the credulously accepted arguments (w.r.t. preferred semantics) via an NP-oracle and then considers the set of arguments that are credulously accepted but not attacked by any credulous accepted argument. Within this set one then computes the ideal extension by a polynomial-time algorithm that iteratively removes arguments which are not defended.

3.3.2 Hardness results

Given the complexity upper bounds from above we are now going for hardness results that show that these upper bounds are optimal. We start with what we call the *standard translation* from propositional formulae to argumentation frameworks and then discuss some prototypical hardness results that extend the standard translation.

Standard Translation. On the one hand the standard translation will give us our first hardness results and on the other hand it is part of almost all reductions in abstract argumentation. To show hardness one typically starts from the standard translation and adds modifications to match the actual problem and semantics.

Reduction 3.6 *Given a propositional formula φ in CNF given by a set of clauses C over the atoms Y, we define the* standard translation *from φ as $F_\varphi = (A, R)$, where*

$$\begin{aligned} A &= \{\varphi\} \cup C \cup Y \cup \bar{Y} \\ R &= \{(c, \varphi) \mid c \in C\} \cup \\ &\quad \{(x, c) \mid x \in c, c \in C\} \cup \{(\bar{x}, c) \mid \bar{x} \in c, c \in C\} \cup \\ &\quad \{(x, \bar{x}), (\bar{x}, x) \mid x \in Y\} \end{aligned}$$

The AF F_φ from Reduction 3.6 is illustrated in Figure 2. The intuition behind the construction is as follows. Assume we are arguing whether the formula φ is true. For an atom y_i we have two arguments, y_i claiming the atom is true, $\bar{y}_i$ claiming the atom is false and thus $\neg y_i$ is true. As exactly one of y_i and $\bar{y}_i$ is true they are mutually attacking. Now consider the argument φ,

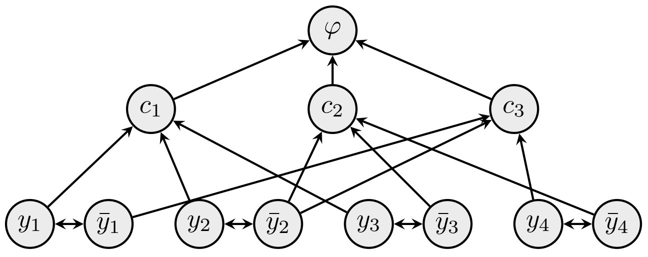

Figure 2. Illustration of the standard translation F_φ, for the propositional formula φ with clauses $\{\{y_1, y_2, y_3\}, \{\bar{y}_2, \bar{y}_3, \bar{y}_4)\}, \{\bar{y}_1, \bar{y}_2, y_4\}\}$.

which can be interpreted as "the formula φ is true". This argument is attacked by the arguments c_i which can be read as "clause c_i is not satisfied". Clearly if one clause is false the whole formula is not satisfied. Finally, if one of the literals in a clause is true the whole clause is true and thus an argument c_i is attacked by all arguments corresponding to literals in c_i.

Credulous Acceptance. For the NP-hardness of credulous acceptance consider the AF F_φ constructed by the above reduction. It is not to hard to show that each model I_Y[10] of φ corresponds to a stable extension of F_φ that consists of the argument φ, the arguments y_i for $y_i \in I_Y$, and the arguments $\bar{y}_i$ for $y_i \in Y \setminus I_Y$. Moreover also the converse holds, i.e., each stable extension of F_φ containing the argument φ corresponds to a model of φ. Thus, F_φ has a stable extension containing φ iff φ has a model. The same holds for admissible sets, complete, preferred and cf2 extensions. Thus Reduction 3.6 is a reduction from SAT to credulous reasoning under these semantics and as it can be clearly performed in polynomial time we obtain that credulous reasoning under these semantics is NP-hard.

Skeptical Acceptance. To show coNP-hardness of skeptical acceptance for stable, preferred and cf2 semantics we extend the standard translation F_φ by an additional argument $\bar{\varphi}$ that is attacked by φ. In the resulting AF G_φ the argument $\bar{\varphi}$ is skeptically accepted w.r.t. the mentioned semantics iff φ is not credulously accepted iff φ is unsatisfiable [Dimopoulos and Torres, 1996; Gaggl and Woltran, 2013]. Thus we have a reduction from UNSAT to skeptical acceptance showing coNP-hardness. Notice that this reduction does not work for admissible and complete semantics as for both the empty set is an extension neither containing φ nor $\bar{\varphi}$.

Skeptical Acceptance with Preferred Semantics. Skeptical acceptance with preferred semantics is a prototypical problem for the second level of the polynomial-hierarchy. The hardness proof is reported in [Dunne and Bench-Capon, 2002] and we next discuss a slight variation of the reduction presented

[10]A model I_Y is a subset of the variables Ysuch that if we set all variables in I_Y to true and all arguments in $Y \setminus I_Y$ to false the formula φ evaluates to true.

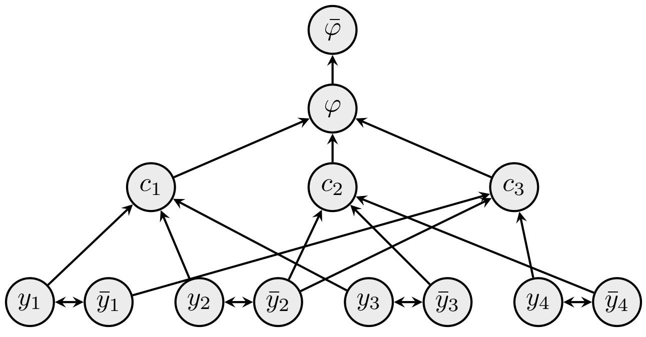

Figure 3. Illustration of the reduction G_φ, for the propositional formula φ with clauses $\{\{y_1, y_2, y_3\}, \{\bar{y}_2, \bar{y}_3, \bar{y}_4)\}, \{\bar{y}_1, \bar{y}_2, y_4\}\}$.

there. That is, we give a reduction from the Π_2^{P}-complete problem $QSAT_\forall^2$ of deciding whether a $QBF_\forall^2$ formula is valid to skeptical acceptance with preferred semantics. That is, given a $QBF_\forall^2$ formula $\forall Y \exists Z \, \varphi(Y, Z)$ with φ being a CNF formula we construct an AF as follows. We first apply the standard reduction from propositional CNF formulae to AFs and then add an additional argument $\bar{\varphi}$ which is attacked by φ, attacks itself and attacks all arguments z, $\bar{z}$ for $z \in Z$ (but not the arguments y, $\bar{y}$ for $y \in Y$). The full reduction is given below and illustrated in Figure 4.

Reduction 3.7 *Given a $QBF_\forall^2$ formula $\Phi = \forall Y \exists Z \, \varphi(Y, Z)$ with φ being a CNF formula given by a set of clauses C over atoms $X = Y \cup Z$, we define the following translation from Φ to $H_\Phi = (A, R)$, where*

$$\begin{aligned} A &= \{\varphi, \bar{\varphi}\} \cup C \cup X \cup \bar{X} \\ R &= \{(c, \varphi) \mid c \in C\} \cup \{(x, \bar{x}), (\bar{x}, x) \mid x \in X\} \cup \\ &\quad \{(x, c) \mid x \in c, c \in C\} \cup \{(\bar{x}, c) \mid \bar{x} \in c, c \in C\} \cup \\ &\quad \{(\varphi, \bar{\varphi}), (\bar{\varphi}, \bar{\varphi})\} \cup \{(\bar{\varphi}, z), (\bar{\varphi}, \bar{z}) \mid z \in Z\} \end{aligned}$$

In the Reduction 3.7 we have that each interpretation $I_Y \subseteq Y$ corresponds to an admissible set $\{y \mid y \in I_Y\} \cup \{\bar{y} \mid y \in Y \setminus I_Y\}$ in H_Φ while arguments z and $\bar{z}$ are attacked by φ and thus can only be in an admissible set if also φ is in that set. To make φ admissible we have to find $I_Y \subseteq Y$ and $I_Z \subseteq Y$ that together satisfy φ. Moreover, as each $c \in C$ is in conflict with φ and attacked by either z and $\bar{z}$ for some $z \in Z$ none of them can be in an admissible set. We then have that a set $\{y \mid y \in I_Y\} \cup \{\bar{y} \mid y \in Y \setminus I_Y\}$ is a preferred extension, i.e., a subset maximal admissible set, iff there is no I_Z such $I_Y \cup I_Z$ satisfies φ. That is, there is a preferred extension in H_Φ not containing the argument φ iff the $QBF_\forall^2$ formula $\forall Y \exists Z \, \varphi(Y, Z)$ is false. Thus, we have a polynomial reduction from the Π_2^{P}-complete problem of $QSAT_\forall^2$ to skeptical acceptance with preferred semantics which proves the Π_2^{P}-hardness of the latter.

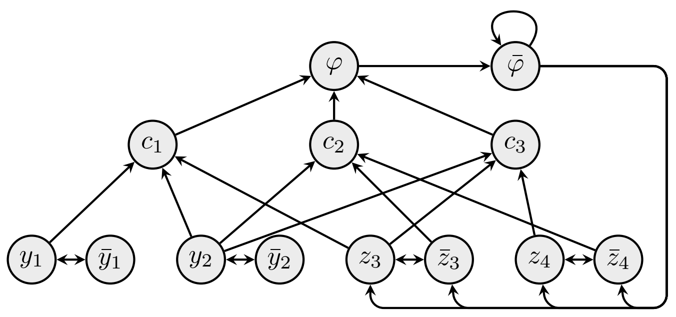

Figure 4. Illustration of the reduction for Π_2^{P}-hardness of *Skept*$_{pr}$. The AF H_Φ, for $\Phi = \forall y_1 y_2 \exists z_3 z_4\ ((y_1 \vee y_2 \vee z_3) \wedge (y_2 \vee \neg z_3 \vee \neg z_4) \wedge (y_2 \vee z_3 \vee z_4))$.

Acceptance with Grounded Semantics. Here we consider the problem of deciding whether an argument is in the grounded extension and show that it is P-hard. To this end we first have to introduce the P-complete problem HORNSAT. A *definite Horn-clause* c is a disjunction over literals from a countable domain U such that c contains exactly one positive literal. A definite Horn-formula is the conjunction over definite Horn-clauses. For example consider the definite Horn-formula $\varphi = x \wedge (\neg x \vee \neg y \vee z) \wedge (\neg y \vee \neg z \vee x)$. A more convincing way to denote definite Horn-formulae is as set of clauses and moreover denoting clauses as (logically equivalent) rules. Thus, our example formula φ can be denoted as $\varphi = \{\rightarrow x, x \wedge y \rightarrow z, y \wedge z \rightarrow x\}$. It is well-known that a definite Horn-formula has a unique minimal model which can be computed in polynomial time. Moreover, the problem HORNSAT of deciding whether an atom is in the minimal model of a definite Horn formula is P-complete [Kasif, 1986].

Next, in order to show P-hardness of *Cred*$_{gr}$, we give a logspace-reduction from the P-complete problem HORNSAT, to *Cred*$_{gr}$ (see Figure 5). Starting from a definite Horn formula one constructs an AF with one argument for each Horn clause; one argument $\bar{x}$ for each variable x; and an additional argument z for the variable that we are asking for being in the minimal model. All the arguments $\bar{x}$ are self attacking. Each argument corresponding to a rule is attacked by all arguments $\bar{x}$ for which the variable x is in the body of the rule and the argument attacks the argument $\bar{h}$ where h is the head of the rule. Finally, the argument z is only attacked by $\bar{z}$ but attacks all $\bar{x}$ arguments. The reduction is formally stated below.

Reduction 3.8 *Let* $\varphi = \{r_l : b_{l,1} \wedge \cdots \wedge b_{l,i_l} \rightarrow h_l \mid 1 \leq l \leq n\}$ *be a definite Horn*

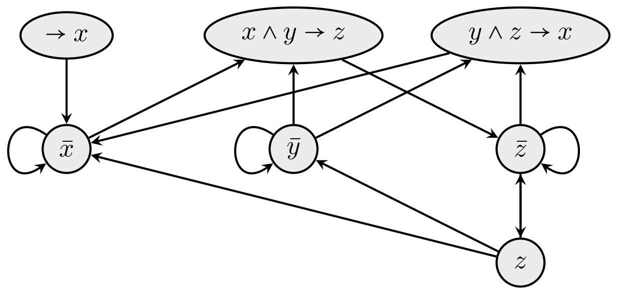

Figure 5. Illustration of the reduction for P-hardness of $Cred_{gr}$, that is $F_{\varphi,z}$ for $\varphi = \{\to x, x \land y \to z, y \land z \to x\}$.

theory over atoms X. We construct the AF $F_{\varphi,z} = (A, R)$ as follows:

$$\begin{aligned} A =& \varphi \cup \bar{X} \cup \{z\} \\ R =& \{(\bar{x},\bar{x}),(z,\bar{x}) \mid x \in X\} \cup \{(\bar{z},z)\} \cup \\ & \{(r_l,h_l),(b_{l,j},r_l) \mid r_l \in \varphi, 1 \le j \le i_l)\} \end{aligned}$$

The intuition behind the above reduction is that an argument corresponding to a rule is in the grounded extension only if all atoms in the rule body are in the minimal model of φ and an argument $\bar{x}$ is attacked by the grounded extension only if x is in the minimal model. That is, when computing the grounded extension via iteratively applying the characteristic function we simulate the following algorithm for deciding whether z is in the minimal model of φ. The algorithm starts with the rules with empty body and adds their rule heads to the minimal model. Then it iteratively considers all rules with the body already being part of the minimal model and adds their heads to the minimal model until either z is added or a fixed-point is reached. Notice that as soon as z is added to the grounded extension all arguments corresponding to rules are defended and thus also added to the grounded extension. We then have that z is in the minimal model of the Horn-formula φ iff z is in the grounded extension of $F_{\varphi,z}$ iff $\{r_l \mid 1 \le l \le n\} \cup \{z\}$ is the grounded extension of $F_{\varphi,z}$. This shows the P-hardness of credulous acceptance as well as of verifying the grounded extension.

3.3.3 Existence and Uniqueness of Extensions

Next, let us consider the results for the existence of (non-trivial) extensions and the uniqueness of solutions in Table 1.

Existence problems. First notice that the $Exists_\sigma$ problem is only relevant for stable semantics as all the other semantics always lead at least one extensions. Moreover, for AFs with at least one argument the problems $Exists_{st}$ and $Exists_{st}^{\neg\emptyset}$ coincide. The standard (non-deterministic) algorithm for $Exists_\sigma^{\neg\emptyset}$ first guesses a non-empty set and then verifies that it is an extension. Now let be V be the complexity for verifying an extension then the above gives use a NP^{V} algorithm for $\mathit{Exists}_{\sigma}^{\neg\emptyset}$. However, the algorithm is only adequate for *st*, *ad*, *co*, and *id* semantics, for the other semantics the problem can be solved more efficiently: for *cf* and *na* semantics it suffices to find one argument that is no self-attacking; for *gr* semantics one tests whether there is an argument that is not attacked by other arguments; for *cf2* and *stg* semantics the problem reduce to test whether there is a non-empty conflict-free set; and finally for *pr* and *sst* semantics the problem reduces to check whether there is a non-empty admissible set.

Uniqueness. When testing for the uniqueness of extensions again stable semantics has a special behaviour. While for all the other semantics we are guaranteed that there is at least one extension for stable semantics we have to perform an additional check that there exists an extension. To check that there are not two (or more) extensions we use the following NP^{V} procedure that shows that an AF has at least two extensions. It first non-deterministically guesses two sets and then verifies that they are different from each other and both are extensions (for the latter the V oracle is used). That is we have an $\mathsf{coNP}^{\mathsf{V}}$ for testing that an AF has at most one extension, which for all semantics, except stable, is equivalent to Unique_{σ}.

Let us now briefly discuss the results for the specific semantics listed in Table 1 (cf. [Dvořák, 2017]). First, *cf* semantics yields a unique extension iff all arguments in the AF are self-attacking, and naive semantics yields a unique extensions if there is no conflict between non-self-attacking arguments. Both criteria can be easily tested in L. Second, *gr* and *id* always yield a unique extension and thus an algorithm can answer “yes” without any computation. For *st*, *ad*, and *co* we can use the $\mathsf{coNP}^{\mathsf{V}}$ algorithm. However, for stable we have to use an additional NP-algorithm to test whether there exists an extension, resulting in a DP algorithm for Unique_{st}. The situation of *cf2* is different. As shown in [Kröll *et al.*, 2017] one can enumerate *cf2* extensions with polynomial delay and thus can also test uniqueness in polynomial time (by computing the first two *cf2* extensions). For *pr* semantics the $\mathsf{coNP}^{\mathsf{V}}$ algorithm can be improved by the observations that an AF has two (or more) preferred extensions iff it has two admissible sets that are in conflict with each other. Thus it suffices to guess two sets, and verify that both sets are admissible and there is a conflict between the two sets. For *sst*, and *stg* the exact complexity is still open but one can also do better than the standard algorithm. That is, the standard algorithm would give a Π_2^{P}-algorithm but one can actually decide uniqueness with a Θ_2^{P}-algorithm [Dvořák, 2017].

3.4 Computational Advantages of Specific Graph-Classes

As most of the reasoning tasks are hard for most of the semantics, one is interested in criteria that make concrete instances tractable. Here we consider special graph classes such that abstract argumentation frameworks within this graph class can be evaluated efficiently. However, these tractability results often only hold for specific semantics and not for the others. This section is based on [Dunne, 2007] and follow up work. In the following we omit semantics where the reasoning tasks are already in L in the general case.

3.4.1 Acyclic AFs

For acyclic AFs we have that each argument is either contained in the grounded extension or attacked by an argument in the grounded extension. Thus the grounded extension is the only stable extension and all the semantics under our consideration coincide. Thus, for all semantics reasoning reduces to computing the grounded extension, but which itself remains P-hard even for acyclic bipartite AFs [Dvořák, 2012a]. The results are summarised in Table 2. Notice that Skept_{ad} is trivially false even in the general case.

Table 2. Complexity for acyclic AFs.

σ	*gr*	*st*	*ad*	*co*	*pr*	*sst*	*stg*	*cf2*	*id*
Cred_σ	P-c	P-c	P-c	P-c	P-c	P-c	P-c	P-c	P-c
Skept_σ	P-c	P-c	trivial	P-c	P-c	P-c	P-c	P-c	P-c

For admissibility based semantics there is a conceptual difference how they deal with even (length) and odd (length) cycles. In an even-cycle there are three admissible sets, the empty set, the set of odd numbered arguments and the set of even numbered arguments, while for an odd-cycle the only admissible set is the empty set. Due to different treatments even and odd-cycles have a quite different impact on the computational complexity.

Even-cycle free AFs. Let us first consider the impact of even-cycles for admissibility based semantics. By an observation in [Dunne and Bench-Capon, 2001] each AF with at least two preferred extensions has an even-cycle. This even holds for complete extensions, i.e., each AF with two complete extensions has an even-cycle [Dvořák, 2012a]. The number of even-cycles in an AF bounds the number of complete and thus also preferred extensions. Thus if an AF has no even-cycles the grounded extension is as well the unique preferred extension and therefore the only candidate for being a stable extension. Again the reasoning tasks for the admissibility based semantics reduce to computing the grounded extension.

Table 3. Complexity results for even-cycle free AFs.

σ	*gr*	*st*	*ad*	*co*	*pr*	*sst*	*stg*	*cf2*	*id*
Cred_σ	P-c	P-c	P-c	P-c	P-c	P-c	Σ_2^{P}-c	NP-c	P-c
Skept_σ	P-c	P-c	trivial	P-c	P-c	P-c	Π_2^{P}-c	coNP-c	P-c

The picture is different for stage and cf2 semantics which are not based on admissibility and handle odd and even-cycles in a similar way. Both maintain their full complexity for even-cycle free AFs [Dvořák and Gaggl, 2016]. The results for even-cycle free AFs are summarised in Table 3.

Odd-cycle free AFs. Odd-cycles are of interest as they distinguish stable from preferred semantics. By a result from [Dung, 1995] in the absence of odd-cycles stable and preferred semantics coincide, i.e., the AF is coherent. As this implies that there is at least one stable extension also semi-stable and stage semantics coincide with stable and preferred semantics in odd-cycle free AFs. But then the complexity of preferred, semi-stable and stage drops down to the complexity of stable, which however stays the same as in the general case. Also admissible, complete and cf2 are not profiting from the absence of odd-cycles, which is proven by the fact that both the standard translation and the modification for skeptical acceptance do not make use of odd-cycles [Dimopoulos and Torres, 1996; Gaggl and Woltran, 2013]. An overview is given in Table 4.[11]

Table 4. Complexity results for odd-cycle free AFs.

σ	*gr*	*st*	*ad*	*co*	*pr*	*sst*	*stg*	*cf2*	*id*
$Cred_\sigma$	P-c	NP-c	NP-c	NP-c	NP-c	NP-c	NP-c	NP-c	coNP-c
$Skept_\sigma$	P-c	coNP-c	trivial	P-c	coNP-c	coNP-c	coNP-c	coNP-c	coNP-c

3.4.2 Bipartite AFs

Bipartite AFs, AFs where the arguments can be partitioned on two conflict-free sets, are a special case of odd-cycle free AFs and thus again stable, preferred, semi-stable, and stage semantics coincide. There is a polynomial time algorithm for computing the credulously accepted arguments [Dunne, 2007], which is based on the following observation. Let the arguments be partitioned in two conflict-free sets A, B. Arguments in A are only attacked by arguments in B and can only be defended by arguments in A. Now the algorithms starts with the set A and tests if it is admissible. If yes then all arguments in the set A are credulously accepted, otherwise all arguments in A which are not defended can not be in any admissible set, i.e., they are not credulously accepted. In the latter case the algorithm removes the undefended arguments and tests the new set for being admissible. It proceeds until it reaches an admissible set (which might be empty). At the end we have that all arguments which are in the computed admissible set are credulously accepted and the remaining arguments in A are not. We then apply the same algorithm to the set B to compute the remaining credulously accepted arguments.

To decide skeptical acceptance we can use that for stable semantics an argument is skeptically accepted iff none of its attackers is credulously accepted.[12] Hence given that we can compute all credulously accepted arguments in polynomial time we can also decide skeptical acceptance in P.

[11]The result for ideal has not been stated before, but is immediate by a generic result in [Dunne *et al.*, 2013] stating that $Cred_{id}$ belongs to $\mathsf{coNP}^{\mathsf{V}}$ where V is the complexity of Ver_{pr} and the fact that the reduction for coNP hardness in [Dunne, 2008] constructs an odd-cycle free AF.

[12]Each stable extension has to either contain the argument or one of its attackers.

In [Dvořák and Gaggl, 2016] it was shown that the above algorithm also works for cf2 semantics. The result for ideal semantics follows from the result for preferred semantics and the polynomial-time algorithm in [Dunne *et al.*, 2013] that computes the ideal extension given the skeptically accepted arguments w.r.t. preferred semantics.

The results are summarised in Table 5. Notice that while in bipartite AFs we can efficiently compute credulous and skeptical acceptance, in contrast to previous tractable fragments, we cannot compute all extensions nor have a good handle on them. This is mirrored by the fact that deciding whether two arguments appear together in one extension is NP-hard [Dunne, 2007]. One can also imagine to consider generalisations of bipartite graphs, so called k-partite graphs, where the arguments can be divided into k conflict-free sets. However, for $k \geq 3$ there are no computational advances from k-partite graphs [Dunne, 2007].

Table 5. Complexity results for bipartite AFs.

σ	*gr*	*st*	*ad*	*co*	*pr*	*sst*	*stg*	*cf2*	*id*
$Cred_\sigma$	P-c	P-c	P-c	P-c	P-c	P-c	P-c	P-c	P-c
$Skept_\sigma$	P-c	P-c	trivial	P-c	P-c	P-c	P-c	P-c	P-c

3.4.3 Symmetric AFs

Here we consider AFs where each attack is symmetric. As each attacker is immediately defended by the symmetric attack the notion of admissibility reduces to conflict-freeness. Considering grounded semantics, in each non-trivial connected component of arguments all arguments are attacked by at least one other argument and thus none of them can be in the grounded extension. Thus computing the grounded extension reduces to find all isolated arguments which can be done in L. Hence all reasoning tasks for grounded, admissible, complete, and preferred semantics are in L. Moreover, also cf2 coincides with naive semantics [Dvořák and Gaggl, 2016]. Finally for symmetric AFs the set of skeptically accepted arguments is always admissible and thus the skeptically accepted arguments coincide with the ideal extension.

Table 6. Complexity results for symmetric AFs.

σ	*gr*	*st*	*ad*	*co*	*pr*	*sst*	*stg*	*cf2*	*id*
$Cred_\sigma$	L	L/NP-c	L	L	L	L/Σ_2^P-c	L/Σ_2^P-c	L	L
$Skept_\sigma$	L	L/coNP-c	trivial	L	L	L/Π_2^P-c	L/Π_2^P-c	L	L

For symmetric AFs one often also requires [Coste-Marquis *et al.*, 2005] that the AF is irreflexive, i.e., it has no self attacks. In that case each naive extension is also a stable extension and thus also semi-stable, and stage coincides with

Table 7. Complexity of acceptance problems, parametrized by the distance from graph classes that allows for efficient algorithms.

graph class	*ad*	*co*	*pr*	*sst*	*st*	*stg*	*cf2*	*id*
acyclic	FPT	FPT	FPT	FPT	FPT	hard	hard	FPT
noeven	XP	XP	XP	XP	XP	hard	hard	XP
bipartite	hard	hard	hard	hard	hard	hard	hard	hard
symmetric	hard	hard	hard	hard	hard	hard	hard	hard

naive. That is, the corresponding reasoning tasks become tractable. However, if we allow self-attacks in the framework then these three semantics maintain their full complexity [Dvořák, 2012a].

3.5 Fixed-Parameter Tractable Fragments

In the previous section we considered properties of the graph structure that make argumentation problems tractable. In this section we study parameters that are quantitative measures for some kind of structure in the graph, with the goal to find parameters such that the complexity of the problem rather scales with the parameter than with input size. That is, we are looking for parameters that allow for fixed-parameter tractable algorithms.

Backdoors for abstract argumentation. One approach to fixed-parameter tractable algorithms is the so called backdoor approach [Dvořák *et al.*, 2012a]. The idea is to start from a tractable fragment, define some kind of distance to the tractable fragment, and then use the distance as a parameter for the reasoning problem. The hope behind this approach is that the running time will scale with the distance to the tractable fragment instead of jumping instantly to the full problem complexity when leaving the fragment. In argumentation one can use the graph classes discussed above as tractable fragments and as distance one considers the number of arguments that have to be deleted from an AF to fall into the graph class.

Definition 3.9 *Let $\mathcal{G}$ be a graph class and $F = (A, R)$ an AF. We define* $\mathrm{dist}_{\mathcal{G}}(F)$ *as the minimal number k such that there exists a set $S \subseteq A$ (the* backdoor set*) with $|S| = k$ and $(A \setminus S, R \cap ((A \setminus S) \times (A \setminus S)) \in \mathcal{G}$. If there is no such set S we define* $\mathrm{dist}_{\mathcal{G}}(F) = \infty$.

We will see that this parametrization only works for certain fragments and semantics, while for other fragments and semantics we have the full complexity even for AFs with constant distance to the fragment. Table 7 summarises the results for the semantics under our considerations (again we omit semantics already tractable in the general case). All results are due to [Ordyniak and Szeider, 2011; Dvořák *et al.*, 2012a] [13], except the results for cf2 semantics

[13]Notice, that ideal semantics is not explicitly mentioned in [Dvořák *et al.*, 2012a], but the

which are due to [Dvořák and Gaggl, 2016]. The entries in Table 7 are to be read as follows. FPT: all reasoning tasks are in FPT; XP: all reasoning tasks are in XP; hard: all problems are as hard as for general graphs even for instances with a fixed distance to the fragment (and are at least NP/coNP-hard for distance 1).

In the remainder of the section we will first present the algorithm that underlies the FPT and XP results and then exemplify some hardness results.

FPT backdoor-algorithms. The positive results are all based on the fact that the number of complete extensions is small and we can compute them efficiently. As soon as we have the complete extensions all the reasoning tasks for admissibility based semantics can be answered efficiently. The algorithms for complete semantics consist of two parts: first one has to compute the backdoor set; second given the backdoor set one has to compute the complete extensions.

Let us first consider computing a backdoor. The detection of *acyc*-backdoors for AFs is equivalent to the so-called *directed feedback vertex set* problem in graph theory, which is known to be fixed-parameter tractable [Chen *et al.*, 2008]. For detecting *noeven*-backdoors in AFs the following algorithm is known, which only shows the problem to be in XP. By a result of Robertson *et al.* [1999] one can test in polynomial time whether a graph is in *noeven* or not. Now, to find a backdoor of size k one can simply iterate over all sets of size k and test whether removing these arguments break all even-cycles. As there are $\Theta(n^k)$ many such sets the algorithm is not fixed-parameter tractable and thus only shows the problem to be in XP.

Now let us assume we already have a backdoor set. We consider labels for arguments that correspond to their status in the extension. An argument is labelled `in` if it is in the extension, `out` if it is not in the extension but attacked by an argument in the extension, and `undec` otherwise. The algorithm tests all possible assignments of labels to the arguments in the backdoor set (these are 3^k many) and for each of them propagates the labels to the remaining AF (which is acyclic or noeven) according to the characteristic function. That is, a node gets label `out` as soon as one attacker is labelled `in`, label `in` if all attackers are labelled `out` and label `undec` if all attackers are labelled and none of the above applies. Finally, one considers the set of arguments labelled `in` and keeps the set if it is a complete extension or withdraws them otherwise. As we do this for each possible labelling of the backdoor set we finally get the set of all complete extensions, which is of size at most 3^k. As one can propagate the labels in polynomial time the, total running time of the algorithm is 3^k multiplied by some polynomial and thus in FPT.

Finally combining the results for computing a backdoor set and for evaluating an AF given a backdoor we have an FPT algorithm for acyclic AFs and an XP algorithm for noeven AFs. Notice that the XP complexity for noeven AFs comes solely from the algorithm for computing the backdoor, the evaluation itself is in FPT.

results follow immediately from the results presented for preferred semantics.

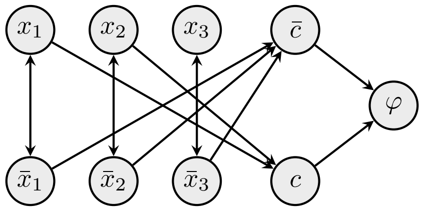

Figure 6. Hardness reduction for $Cred_{ad}$ and backdoors to bipartite graphs, illustrated for the propositional formula φ, with clauses $c = \{x_1, x_2\}$, and $\bar{c} = \{\bar{x}_1, \bar{x}_2, \bar{x}_3\}$.

Hardness Results. The hardness proofs work very much like for the general case, one has to give a reduction from a hard problem but additionally take into account the graph structure [Dvořák *et al.*, 2012a; Dvořák, 2012c; Dvořák *et al.*, 2014a; Dvořák and Gaggl, 2016]. We exemplify such a reduction for credulous reasoning under admissible, complete, preferred and cf2 semantics and backdoors for bipartite graphs. To this end consider the standard translation from proposition logic and the NP-hard problem monotone SAT, of deciding whether a formula in CNF where each clause either contains solely positive or solely negative literals is satisfiable. As each clause either contains solely positive literals or solely negative literals the graph constructed by the standard translation is almost bipartite (cf. Figure 6). That is, there are no edges between the arguments corresponding to positive literals and negative clauses and no edges between the arguments corresponding to negative literals and positive clauses. Thus, when deleting φ from the graph the graph becomes bipartite with two independent sets, one containing the positive literals and the negative clauses, and one containing the negative literals and the positive clauses. We obtain that credulous reasoning is NP-hard even for graphs with distance 1 to bipartite graphs.

Further FPT Results. Besides backdoors to tractable fragments several other approaches for parametrizations can be found in the literature. One approach is to consider graph parameters that measure structural properties, most prominently tree-width, a parameter that, roughly speaking, measures how tree-like a graph is. Results for tree-width (and the related parameter clique-width) can be either obtained by dynamic programming algorithms that exploit the structural properties or by powerful meta-theorems. These meta-theorems basically say that every property which can be characterised by a formula from monadic-second order logic (MSO) over a graph structure can be tested in FPT w.r.t. tree-width and clique-width. Results via the MSO meta-theorems are given in [Dunne, 2007; Dvořák *et al.*, 2012c] concrete dynamic programming algorithms are given in [Dvořák *et al.*, 2012b; Charwat, 2012] for tree-width and in [Dvořák *et al.*, 2010] for clique-width. Moreover, in [Dvořák *et al.*, 2012b] a lot of parameters specific to directed graphs, e.g. directed tree-width, are shown to be not applicable for FPT algorithms in abstract argumentation.

Finally for semantics harder than NP one can also think about backdoors to graph classes that allow to solve problems in NP or coNP [Dvořák *et al.*, 2014a]. While this does not give FPT results it still reduces complexity, with notable effects on the practical resolvability.

3.6 Computational Problems related to Labelling-Based Semantics

So far our complexity analysis was in terms of extension-based semantics (which is in accordance with the literature), in this section we discuss some computational aspects related to labelling-based semantics.

Labelling-based semantics. Beside the so-called extension-based semantics we have considered so far, there are several approaches defining argumentation semantics via certain kinds of argument labellings. As an example we consider the popular approach of 3-valued labellings by Caminada and Gabbay [2009] and in particular their complete labellings. Basically, such a labelling is a three-valued function $\mathcal{L}ab$ that assigns one of the labels `in`, `out` and `undec` to each argument, with the intuition behind these labels being the following. An argument is labelled with: `in` if it is accepted, i.e., it is defended by the `in` labelled arguments; `out` if there are strong reasons to reject it, i.e., it is attacked by an accepted argument; `undec` if the argument is undecided, i.e., neither accepted nor attacked by accepted arguments. Complete labellings can be one-to-one mapped to complete extensions by considering the set of `in` labelled arguments and vice versa, by labelling all arguments in the extension with `in` all arguments attacked by the extension with `out` and the remaining arguments with `undec` [Caminada and Gabbay, 2009]. Notice that this is not only a property of complete semantics but this one-to-one correspondence holds for most argumentation semantics.

Computational problems. Given the above correspondence between labellings and extensions, the tasks of computing all labellings and all extensions are, from a computational point of view, equivalent and the same holds for credulous and skeptical reasoning. However, three-valued labellings allow for more fine-grained acceptance statuses of arguments. Wu and Caminada 2010 introduced the notion of justification status of an argument w.r.t. a semantics which is given by the set of labels that are assigned by at least one labelling of the semantics. That is, given an AF F and a labelling-based semantics $\sigma_{\mathcal{L}ab}$ the *justification status* $\mathcal{JS}_{\sigma_{\mathcal{L}ab}}(F,a)$ of an argument a in F is given by $\mathcal{JS}_{\sigma_{\mathcal{L}ab}}(F,a) = \{\mathcal{L}ab(a) \mid \mathcal{L}ab \in \sigma_{\mathcal{L}ab}(F)\}$. The above definition gives rise to eight different justification statuses, most prominently the set $\{\texttt{in}\}$ called strong accept, which corresponds to skeptical acceptance, and the set $\{\texttt{in}, \texttt{undec}\}$ called weak accept.[14] We are now faced with the computational problem of *verifying the justification status* of an argument, which was studied in [Dvořák, 2012b].

[14]Notice that credulous acceptance of argument a corresponds to the query $\texttt{in} \in \mathcal{JS}_{\sigma}(F,a)$.

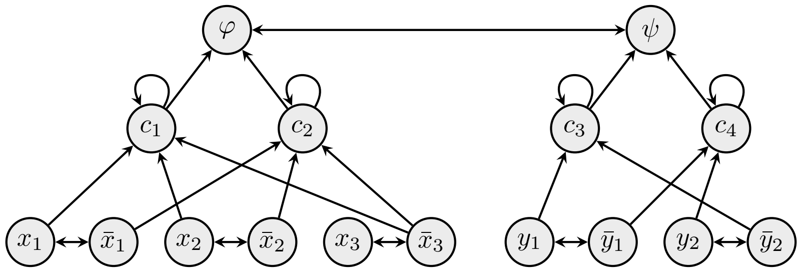

Figure 7. Reduction for showing the DP-hardness of weak acceptance. AF $F_{\varphi,\psi}$ for the propositional formulae φ, with clauses $\{x_1, x_2, \bar{x}_3\}$,$\{\bar{x}_1, \bar{x}_2, \bar{x}_3\}$, and ψ, with clauses $\{y_1, \bar{y}_2\}$, $\{\bar{y}_1, y_2\}$.

Algorithms. Compared with credulous and skeptical acceptance, where we either search for an extension containing a specific argument or for an extension not containing a specific argument, the problem of testing whether an argument has a specific justification status, e.g., whether it is a weakly accepted, has two sources of complexity. First, we have to search for labellings that assign the labels appearing in the justification status, e.g., `in` and `undec` for weak acceptance, and second we have to make sure that no labelling assigns one of the labels not in the in the justification status, e.g., `out` for weak acceptance, to a. For complete semantics this means we have to perform both an NP search for the good labels and a coNP search for the bad labels, which together gives a DP algorithm.

Hardness. To prove DP-hardness of weak acceptance w.r.t. complete semantics one starts from an instance (φ, ψ) of the DP-complete SAT−UNSAT problem and constructs the AF $F_{\varphi,\psi}$ (see Figure 7) as follows. First one applies the standard translation to each of the two formulae and then makes the arguments c corresponding to clauses unacceptable, by adding self-attacks. Finally the arguments φ and ψ are connected by a mutual attack. As in the standard translation we have that φ, respectively ψ, is credulously accepted iff φ, respectively ψ, is satisfiable. Moreover, (a) φ is labelled `out` iff ψ is labelled `in` by some labelling, i.e., if ψ is credulously accepted, and (b) the grounded labelling maps all arguments to `undec` and thus `undec` $\in \mathcal{JS}_{\sigma_{\mathcal{L}ab}}(F_{\varphi,\psi}, \varphi)$. We then have that the argument φ is weakly accepted iff φ is satisfiable and ψ is unsatisfiable, that is iff (φ, ψ) is a "yes" instance of SAT−UNSAT. Thus we have a reduction from SAT−UNSAT to weak acceptance and can conclude that also the latter is DP-hard.

3.7 Discussion

As illustrated in Table 1 there is a significant difference in the computational complexity between the different semantics. Let us first consider the polynomial-time computable semantics. Grounded semantics distinguishes itself from the remaining semantics by the fact that it has a unique extension which can be efficiently computed in an iterative fashion by applying the characteristic function. For conflict-free and naive sets the good complexity comes from the fact that we can decide the reasoning problems without computing the actual conflict-free, respectively naive, sets. However, there are AFs with exponentially many conflict-free, respectively naive, sets and there are non-standard problems the are computationally hard, for instance counting the number of conflict-free, respectively naive, sets [Baroni *et al.*, 2010].

On the NP, coNP layer of the polynomial-hierarchy we have semantics with potentially exponentially many extensions but where each set itself can be easily tested to be an extension. That is, the source of the computational hardness is the fact that one, in the worst case, has to check many sets to find a witness for credulous acceptance, respectively to find a counter-example for skeptical acceptance. However, these problems can be efficiently encoded in formalisms where the corresponding problems are NP- and coNP-hard, like propositional logic, and then can be evaluated with corresponding systems for these formalisms [Besnard and Doutre, 2004].

Finally, we have semantics that require some sort of subset maximisation which adds an additional source of complexity. Thus, these semantics are harder than NP and located at the second level of the polynomial-hierarchy. For reduction-based approaches this implies that one cannot efficiently translate them to a single instance of propositional logic but has either to consider richer formalisms like QBFs [Egly and Woltran, 2006; Arieli and Caminada, 2012] or ASP [Egly *et al.*, 2010] or consider iterative approaches [Cerutti *et al.*, 2014; Dvořák *et al.*, 2014a] that make several calls to a SAT-Solver. The different levels of hardness of different semantics are also mirrored by the results of the First International Competition on Computational Models of Argumentation [Thimm and Villata, 2015; Thimm *et al.*, 2016], where the computational tasks for preferred semantics appear significantly harder than the corresponding tasks for stable or complete semantics.

Notice that there are several established semantics which are beyond the scope of this chapter. First there is the scheme of resolution-based semantics [Baroni *et al.*, 2011c], with resolution-based grounded semantics being the most prominent instantiation. A comprehensive complexity analysis for resolution-based grounded semantics can be found in [Baroni *et al.*, 2011c], which is complemented by results in [Dvořák *et al.*, 2012c; Dvořák *et al.*, 2014b]. Another semantics we neglected is eager semantics [Caminada, 2007], whose complexity was studied in the generalised setting of parametrized ideal semantics [Dunne *et al.*, 2013].

4 Complexity of Assumption-based Argumentation

With Dung's abstract argumentation frameworks we focused on the issue of finding coherent sets of simultaneously acceptable arguments, but neglected the effort for constructing these frameworks and for drawing conclusions from the accepted arguments. With Assumption-based Argumentation [Bondarenko *et al.*, 1997] we now switch to a formalism that covers the whole argumentation process. That is, arguments and conflicts are constructed from a knowledge base, then acceptable sets, i.e., extensions, are identified, and finally one draws conclusions from the extensions. We are in particular interested in how these additional steps affect the overall computational complexity.

In this section will discuss complexity results for Assumption-based Argumentation which are due to the work of Dimopoulos *et al.* [1999; 2000; 2002] and the later work on ideal semantics [Dunne, 2009].[15] We first briefly introduce assumption-based frameworks and the different semantics thereof and define the core reasoning problems in assumption-based argumentation. We then discuss procedures to solve the reasoning problems, which give us upper bounds for the computational complexity. As most of these procedures are of high complexity we also discuss the special case of flat ABFs which allows for a milder complexity. Finally, we discuss some hardness results showing that the presented procedures are essentially optimal.

4.1 Assumption-based Argumentation

We first briefly recall the definitions of assumption-based argumentation, for a comprehensive introduction the reader is referred to [Toni, 2014] or the chapter on Assumption-based Argumentation in this handbook.

For an assumption-based framework we assume a *deductive system* $(\mathcal{L}, \mathcal{R})$, where $\mathcal{L}$ is a formal language and $\mathcal{R}$ a set of inference rules that induces a derivability relation $\vdash$. Given a theory $T \subseteq \mathcal{L}$ the *deductive closure* $Th(T)$ of T is defined as $Th(T) = \{\alpha \in \mathcal{L} \mid T \vdash \alpha\}$.

Definition 4.1 *An abstract* assumption-based framework *(ABF) is a tuple* $\langle \mathcal{L}, \mathcal{R}, A, \overline{} \rangle$ *with* $(\mathcal{L}, \mathcal{R})$ *a deductive system,* $A \subseteq \mathcal{L}$ *is a (non-empty finite) set, with elements referred to as assumptions; and the* contrary *function* $\overline{}$, *a total mapping from* A *into* $\mathcal{L}$.

An extension of an ABF is a set of assumptions $\Delta \subseteq A$ meeting some requirements.

Definition 4.2 *Given an ABF and an assumption set* $\Delta \subseteq A$ *we say that* Δ attacks *an assumption* $\alpha \in A$ *if* $\bar{\alpha} \in Th(\Delta)$. *Further we say that an assumption set* Δ *attacks an assumption set* Δ' *if* Δ *attacks at least one* $\alpha \in \Delta'$

We will further require that assumptions sets are closed, i.e., we can not derive additional assumptions.

[15]The complexity of Assumption-based Argumentation was also briefly discussed in the earlier survey on the complexity of argumentation [Dunne and Wooldridge, 2009].

Definition 4.3 *We call an assumption set* Δ closed *if* $Th(\Delta) \cap A = \Delta$.

It is often the case that the derivability relation is such that all assumption sets are closed, in that case we call the ABF *flat*.

We are now prepared to define the standard semantics for ABFs.

Definition 4.4 *Given an ABF F and an assumption set $\Delta \subseteq A$. Δ is called*

- *stable extension ($\Delta \in st(F)$), if Δ is closed, Δ does not attack itself, and Δ attacks each assumption $\alpha \in A \setminus \Delta$.*
- *admissible set ($\Delta \in ad(F)$), if Δ is closed, Δ does not attack itself, and for all closed assumption sets $\Delta' \subseteq A$, if Δ' attacks Δ then also Δ attacks Δ'.*
- *preferred extension ($\Delta \in pr(F)$), if Δ is a subset-maximal admissible assumption set.*

Moreover, for flat frameworks also *ideal semantics* can be defined [Dung *et al.*, 2006; Dung *et al.*, 2007]. The unique ideal extensions $id(F)$ is the maximal admissible set Δ that is contained in all preferred extensions.[16]

We have that every stable assumption set is also a preferred assumption set, and every preferred assumption set is an admissible assumptions set, but not vice versa. However, each admissible assumption set is a subset of some preferred assumption set. Moreover, if the ABF is flat the empty assumption set is always admissible.

4.2 Reasoning Problems

As for abstract argumentation we are mainly interested in computing acceptance statuses of statements instead of extensions. However, the reasoning tasks we consider will give us a good impression of the complexity of computing extensions. That is, we again consider credulous and skeptical acceptance but now of a sentence $\varphi \in \mathcal{L}$ instead of an argument. More concretely we either want to decide whether there is at least one extension that entails φ (credulous reasoning) or whether φ is entailed by each extension. This gives rise to the following computational problems for an assumption-based argumentation semantics σ.

- *Credulous Acceptance $Cred_\sigma$*: Given ABF F and a sentence $\varphi \in \mathcal{L}$. Is $\varphi \in Th(\Delta)$ for some assumption set $\Delta \in \sigma(F)$?
- *Skeptical Acceptance $Skept_\sigma$*: Given ABF F and a sentence $\varphi \in \mathcal{L}$. Is $\varphi \in Th(\Delta)$ for all assumption sets $\Delta \in \sigma(F)$?

Beside the above reasoning problems we again consider the task of verifying extensions, i.e., one is given an assumption set and has to verify that it is an extension of a given semantics σ.

[16]Notice that uniqueness and other properties of the ideal extension are only guaranteed for flat ABFs.

Table 8. Complexity upper bounds for different types of ABFs. $\mathcal{C}$ denotes the complexity of deciding the $\vdash$ relation.

	General ABFs			*Flat ABFs*		
σ	$Cred_\sigma$	$Skept_\sigma$	Ver_σ	$Cred_\sigma$	$Skept_\sigma$	Ver_σ
st	$\mathsf{NP}^{\mathcal{C}}$	$\mathsf{coNP}^{\mathcal{C}}$	$\mathsf{P}^{\mathcal{C}}$	$\mathsf{NP}^{\mathcal{C}}$	$\mathsf{coNP}^{\mathcal{C}}$	$\mathsf{P}^{\mathcal{C}}$
ad	$\mathsf{NP}^{\mathsf{NP}^{\mathcal{C}}}$	$\mathsf{coNP}^{\mathsf{NP}^{\mathcal{C}}}$	$\mathsf{coNP}^{\mathcal{C}}$	$\mathsf{NP}^{\mathcal{C}}$	$\mathcal{C}$	$\mathsf{P}^{\mathcal{C}}$
id	–	–	–	$\mathsf{P}_{\parallel}^{\mathsf{NP}^{\mathcal{C}}}$	$\mathsf{P}_{\parallel}^{\mathsf{NP}^{\mathcal{C}}}$	$\mathsf{P}_{\parallel}^{\mathsf{NP}^{\mathcal{C}}}$
pr	$\mathsf{NP}^{\mathsf{NP}^{\mathcal{C}}}$	$\mathsf{coNP}^{\mathsf{NP}^{\mathsf{NP}^{\mathcal{C}}}}$	$\mathsf{NP}^{\mathsf{NP}^{\mathcal{C}}}$	$\mathsf{NP}^{\mathcal{C}}$	$\mathsf{coNP}^{\mathsf{NP}^{\mathcal{C}}}$	$\mathsf{NP}^{\mathcal{C}}$

- *Verification of an Extension Ver_σ*: Given ABF $F = \langle T, A, \bar{\ } \rangle$ and an assumption set $\Delta \subseteq A$. Is $\Delta \in \sigma(F)$?

4.3 Procedures to solve ABA Reasoning Problems

In ABA, new computational challenges come up when compared with Dung's abstract argumentation. While in Dung's abstract argumentation arguments and attacks are given explicitly, they are only given implicitly in ABFs and depend on the set of assumptions and the derivability relation $\vdash$. That is, we get two additional sources of complexity: (1) the construction of arguments, and (2) the identification of conflicts between them. Both highly depend on the complexity of deciding the derivability relation $\vdash$. Thus, upper bounds for the complexity in assumption-based argumentation usually assume that the derivability relation $\vdash$ can be decided in some complexity class $\mathcal{C}$ and the actual complexity results are then given in terms of some $\mathcal{C}$-oracle complexity classes.

Verifying an Assumption Set. First, we consider the problem of verifying an assumption set Δ as an extension and start with stable semantics. We have to check that (i) Δ is closed, (ii) Δ is conflict-free, and (iii) Δ attacks every assumption $\alpha \in A \setminus \Delta$. Each of these checks can be done in $\mathsf{P}^{\mathcal{C}}$ as follows: For (i) one has to check whether $\Delta \vdash \alpha$ for $\alpha \in A \setminus \Delta$ which just requires a linear number of $\vdash$ computations. For (ii) one has to check whether $\Delta \nvdash \bar{\alpha}$ for $\alpha \in \Delta$ which again just requires a linear number of $\vdash$ computations. Finally, (iii) can also be checked by a linear number of $\vdash$ computations and thus a stable set can be verified in $\mathsf{P}^{\mathcal{C}}$. For admissible semantics verification is a bit harder. Here instead of condition (iii) we have to verify that for all closed assumption sets $\Delta' \subseteq A$, if Δ' attacks Δ then also Δ attacks Δ'. This can be done with a $\mathsf{coNP}^{\mathcal{C}}$-algorithm that guesses a counter-example Δ' and then verifies via the $\mathcal{C}$ oracle that Δ' is closed, Δ' attacks Δ, and Δ' is not attacked by Δ. In total we have that verifying an admissible extension is in $\mathsf{coNP}^{\mathcal{C}}$. For preferred semantics we additionally have to take into account the maximality check which leads to a $\mathsf{coNP}^{\mathsf{NP}^{\mathcal{C}}}$-algorithm.

Reasoning. The complexity upper bounds for skeptical and credulous reasoning are immediate by the algorithms for verifying extensions. We can decide the acceptance of a sentence by first guessing an assumption set, second verifying that the guessed set is an extension and finally deciding via a $\mathcal{C}$ oracle whether the extension entails the queried sentence. The corresponding complexity results are given in the left part of Table 8 (recall that ideal semantics was only introduced for flat ABFs). As for Dung's AFs, credulous reasoning with preferred semantics reduces to credulous reasoning with admissible semantics and thus has a lower complexity than skeptical reasoning. Moreover, in contrast to Dung's AFs, we have a complexity gap between stable and admissible semantics, which is due to the fact that for admissible extensions for each attacking assumption set we have to test whether it is closed or not.

Flat ABFs. Flat ABFs as a special class of ABFs that provide milder complexity. Recall that in flat ABFs each assumption set is already closed and we thus do not have to check this in the algorithms. Let us now reconsider the problem of verifying an admissible extension Δ. As Δ is closed we only have to check whether (i) Δ is conflict-free, and (ii) for all assumption sets $\Delta' \subseteq A$, if Δ' attacks Δ then also Δ attacks Δ'. The latter simplifies to checking whether $\{\alpha \in A \mid \Delta \nvdash \bar{\alpha}\}$ does not attack Δ, which can be decided in $\mathsf{P}^{\mathcal{C}}$. Thus, verifying admissible extensions in flat ABFs is in $\mathsf{P}^{\mathcal{C}}$ and hence also verifying preferred extensions is in $\mathsf{coNP}^{\mathcal{C}}$. This gives improved complexity bounds for credulous and skeptical acceptance listed in in Table 8 in the column *flat*. Finally, notice that in flat ABFs the empty set is always admissible and thus only the assumptions contained in $Th(\emptyset)$ are skeptically accepted. That is, skeptical reasoning reduces to testing whether $\varphi \in Th(\emptyset)$, which is in $\mathcal{C}$.

The *ideal extension* can be computed by the same algorithm as for Dung AFs [Dunne, 2009]. That is, one first determines the credulously accepted assumptions w.r.t. admissible semantics that are not attacked by other credulously accepted assumptions. Given those arguments one iteratively removes assumptions that can not be defended until an admissible set, the ideal extension, is reached. Overall, this gives an $\mathsf{P}_{\parallel}^{\mathsf{NP}^{\mathcal{C}}}$ algorithm for credulous and skeptical reasoning as well as for the verification problem.

4.4 Complexity lower bounds

While the upper bounds can be given in a generic fashion, which immediately gives upper bounds/algorithms for each instantiation, hardness results only exist for concrete formalisms. However, the complexity results for the concrete instantiations [Dimopoulos *et al.*, 2002] show that the generic upper bounds are tight in the sense that there are formalisms where the lower bounds match the generic upper bounds. In Table 9 we list the complexity results for Autoepistemic Logic (AEL) [Moore, 1985], Logic Programming (LP) [Gelfond and Lifschitz, 1988], and Default Logic (DL) [Reiter, 1980] all the results are due to Dimopoulos *et al.* [2002] and Dunne [2009]. For Autoepistemic Logic we have that the ABF is not flat and deciding the $\vdash$ relation is coNP-complete. Thus the complexity results in Table 9 exactly match the generic upper bounds for general ABFs. In contrast, Logic Programming and Default Logic result flat ABFs and for the former the $\vdash$ relation is in P and for the latter the $\vdash$ relation is coNP-complete. In both cases the complexity results in Table 9 exactly match the generic upper bounds for flat ABFs.

Table 9. Completeness results for instantiations of ABA.

	type	stability		Admissibility		Preferability		Ideal
		cred.	skept.	cred.	skept.	cred.	skept.	cred.
AEL	general	Σ_2^{P}-c	Π_2^{P}-c	Σ_3^{P}-c	Π_3^{P}-c	Σ_3^{P}-c	Π_4^{P}-c	–
LP	flat	NP-c	coNP-c	NP-c	P-c	NP-c	Π_2^{P}-c	Θ_2^{P}-c
DL	flat	Σ_2^{P}-c	Π_2^{P}-c	Σ_2^{P}-c	coNP-c	Σ_2^{P}-c	Π_3^{P}-c	Θ_3^{P}-c

For a hardness proof in ABA one has to construct a certain knowledge base in the considered formalism instead of arguments interlinked with conflicts. Thus hardness proofs in the context of ABA are of a different nature than in Dung's abstract argumentation. To exemplify such an hardness proof we next present the hardness result for credulous admissible reasoning in Default Logic which is Σ_2^{P}-complete [Dimopoulos *et al.*, 2002].

Default logic as ABA. A Default theory (W, D) that consists of a set W of propositional formulae[17], called background theory, and a set D of default rules of the form $\frac{\alpha : M\beta_1 \dots M\beta_n}{\gamma}$, where α, β_i, γ are sentences in propositional logic, can be interpreted as assumption-based framework $\langle \mathcal{L}, \mathcal{R}, A, \overline{} \rangle$ [Bondarenko *et al.*, 1997]. As deductive system one uses the deductive system of propositional logic extended by the set D of default rules, where the intuitive meaning of a default rule is that if we know α is the case and have no basis on which to suppose any $\neg\beta_i$ holds it is reasonable to assume γ. The ABF is now built as follows: the set of assumptions A consists of the expressions of the form $M\beta$, the contrary $\overline{M\beta}$ of an assumption $M\beta$ is $\neg\beta$ and the derivability relation $\vdash$ is given by $\Delta \vdash \phi$ iff $\phi \in Th_{DL}(W \cup \Delta)$ where Th_{DL} is the deductive closure of the deductive system described above.

Hardness of credulous admissible reasoning in DL. To show hardness for credulous admissible reasoning in Default Logic, we give a reduction from the Σ_2^{P}-hard problem $QSAT_\exists^2$ of deciding whether a $QBF_\exists^2$ is valid. That is, we start with a QBF $\exists Y \forall Z\, \varphi(Y, Z)$ and construct a DL theory $(\emptyset, D)$ and thus the corresponding ABF F as follows. To construct the set of default rules D, we add the two default rules (i) $\frac{My}{y}$ and (ii) $\frac{M\neg y}{\neg y}$ for each variable $y \in Y$. This corresponds to the ABF F with $A = \{My, M\neg y \mid y \in Y\}$ and $\overline{My} = \neg y$, $\overline{M\neg y} = y$ for all $y \in Y$. By that we have that an admissible set can only contain either My or $M\neg y$ but not both, and thus that the admissible sets correspond to the

[17]Notice that instead of using propositional logic one could also define Default logic on top of first-order logic or any other formal logic.

partial truth assignments of Y. Moreover for an admissible set E we have that $E \vdash \varphi$ iff $\varphi(Y, Z)$ is true for all assignments Z under the partial assignment given for Y. That is, φ is credulously accepted iff there is a partial assignment of Y such that for each assignment to Z the formula $\varphi(Y, Z)$ evaluates to true, that is iff $\exists Y \forall Z \varphi(Y, Z)$ is valid.

Example 4.5 *Consider the QBF* $\Phi = \exists y_1, y_2 \forall z_1, z_2\, (y_1 \vee z_2 \vee \neg z_3) \wedge (\neg y_2 \vee z_3)$. *The above reduction would construct*

- *the default rules* $\frac{My_1}{y_1}$, $\frac{My_2}{y_2}$, $\frac{M\neg y_1}{\neg y_1}$ *and* $\frac{M\neg y_1}{\neg y_1}$;
- *the assumption set* $A = \{My_1, My_2, M\neg y_1, M\neg y_2\}$; *and*
- *the contrary function* $^-$ *with* $\overline{My_1} = \neg y_1$, $\overline{My_2} = \neg y_2$, $\overline{M\neg y_1} = y_1$, *and* $\overline{M\neg y_2} = \neg y_2$.

Now, by the above, it must be that Φ *is valid if and only if there is an admissible set* $E \subseteq A$ *such that* $E \vdash (y_1 \vee z_2 \vee \neg z_3) \wedge (\neg y_2 \vee z_3)$. *The formula* Φ *is valid as setting* y_1 *to true and* y_2 *to false makes both clauses true no matter which truth value is assigned to the variables* z_1 *and* z_2. *On the other hand also the set* $E = \{My_1, M\neg y_2\}$ *is admissible and, by our default rules, we have* $E \vdash (y_1 \vee z_2 \vee \neg z_3) \wedge (\neg y_2 \vee z_3)$. $\Diamond$

4.5 Discussion

The upper bounds for the complexity of the reasoning problems in Table 8 indicate that assumption-based argumentation indeed has a higher complexity than just Dung style argumentation. However, by the discussed results for flat argumentation in Table 8 and the concrete instantiations of ABA in Table 9 one can see that the actual complexity heavily depends on the complexity of derivability relation $\vdash$ and the type of the assumption-based framework. For instance for LP we have a flat assumption-based framework and a tractable derivability relation and end up with the same complexity bounds as for Dung's abstract argumentation frameworks. The complexity of deciding the derivability relation directly corresponds to the costs of constructing an argument, or drawing some conclusion when already given an extension. Thus the parameter $\mathcal{C}$ in Table 8 can be interpreted as the costs of these two steps, i.e., constructing arguments and drawing conclusions, in the argumentation process.

For general ABFs the complexity of assumption-based argumentation is quite high and thus it is promising to consider some restrictions of the formalism to get better algorithms. In this chapter we considered flat ABFs which reduced the complexity significantly. In [Dimopoulos *et al.*, 2002] also two other classes, namely so called simple and normal ABFs, are studied and shown to have computational advantages for certain problems.

5 Computational Problems in Abstract Dialectical Frameworks

In this section we consider abstract dialectical frameworks, a generalisation of Dung style abstract argumentation frameworks. While arguments are still abstract entities abstract dialectical frameworks allow for more complex relations between the arguments. That is, each abstract dialectical framework has a link relation between the arguments, which is not necessarily an attack relation. The semantics of the links is given by acceptance conditions for each argument that define the acceptance status of an argument in dependence on the acceptance status of the predecessor arguments. This allows for classical binary attacks between arguments but also for joint attacks, support and more complex dependencies.

This section is based on the works of Brewka *et al.* [2013], Wallner [2014, Chapter 4], and Strass and Wallner [2015] and organised as follows. We first define abstract dialectical frameworks and semantics thereof. We then discuss and formally define the core computational problems and consider the general computational complexity of abstract dialectical frameworks. Moreover, we discuss a restricted class of ADFs, so called bipolar ADFs, that only allow for links that are attacking or supporting (but might be both), and their computational advantages.

5.1 Abstract Dialectical Frameworks (ADFs)

Here we give a very brief discussion of Abstract Dialectical Frameworks.[18] Notice that in the literature there are several proposals how to define semantics for ADFs, here we will follow the lines of Brewka et al. [Brewka *et al.*, 2013].

Definition 5.1 ([Brewka *et al.*, 2013]) *An abstract dialectical framework is a tuple $D = (S, L, C)$ where*

- *S is a (finite) set of abstract arguments / statements,*
- *$L \subseteq S \times S$ is a set of links,*
- *$C = \{C_s\}_{s \in S}$ is a set of total functions $C_s : 2^{par(s)} \to \{\mathbf{t}, \mathbf{f}\}$, one for each statement $s \in S$. C_s is called acceptance condition of s.*

Here, we will assume that each acceptance condition C_s is given by a propositional formula φ_s over the predecessors of s. An example is provided in Figure 8.

As first semantics we define (two-valued) *models* of ADFs. To this end we consider two-valued interpretations I that to each $s \in S$ assign either $\mathbf{t}$ or $\mathbf{f}$. Given an interpretation I we will use $I^{\mathbf{t}}$ to denote the set $\{s \in S \mid I(s) = \mathbf{t}\}$ and $I^{\mathbf{f}}$ to denote the set $\{s \in S \mid I(s) = \mathbf{f}\}$.

[18]For a more detailed discussion the reader is referred to the the chapter of this handbook dedicated to ADFs or [Brewka *et al.*, 2013].

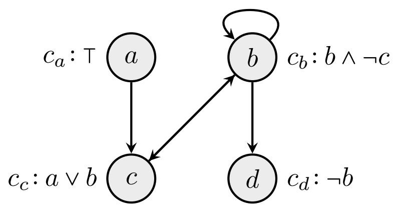

Figure 8. Illustration of an ADF $D = (S, L, C)$ with $S = \{a, b, c, d\}$, $L = \{(a, c), (b, b), (b, c), (c, b), (b, d)\}$, and $C = \{c_a : \top, c_b : b \wedge \neg c,\ c_c : a \vee b, c_d : \neg b\})$.

Definition 5.2 *Let $D = (S, L, C)$ be an ADF, a two-valued interpretation I defined over S is a two-valued model of D if $I \models \varphi_s$ for each $s \in I^{\mathbf{t}}$ and $I \not\models \varphi_s$ for each $s \in I^{\mathbf{f}}$.*

Most of the ADF semantics are based on *3-valued interpretations* [Kleene, 1952] that map each argument in S to one of the values $\mathbf{t}$, $\mathbf{f}$ and $\mathbf{u}$. The three values $\mathbf{t}$, $\mathbf{f}$, $\mathbf{u}$ are ordered, by $<_i$, such that $\mathbf{u} <_i \mathbf{t}$, $\mathbf{u} <_i \mathbf{f}$, and $\mathbf{t}$,$\mathbf{f}$ are incomparable. This ordering is then extended to interpretations such that for 3-valued interpretations I, J we have $I \leq_i J$ iff $I(s) \leq_i J(s)$ for all $s \in S$. We say that a two-valued interpretation I extends a 3-valued interpretation J iff $I \leq_i J$. That is, all arguments mapped to $\mathbf{f}$ or $\mathbf{t}$ by J are mapped to the same by I and all arguments that are mapped to $\mathbf{u}$ by J are mapped to either $\mathbf{t}$ or $\mathbf{f}$ by I. Given a 3-valued interpretation J, by $[J]_2$ we denote the set of all two-valued interpretations that extend J.

In Dung's abstract argumentation frameworks the characteristic function and its fixed-points are central in the definition of the semantics. We next define the operator Γ_D that will be central in our definitions of ADF semantics. Γ_D generalises the characteristic function in two directions: (i) it gives a three valued assignment on arguments, i.e., beside marking arguments as accepted it also explicitly marks arguments as rejected; and (ii) it allows for the more general acceptance conditions of ADFs.[19] Given an interpretation I, the operator Γ_D computes the arguments that should be set to $\mathbf{t}$ or $\mathbf{f}$ under the current interpretation I.

Definition 5.3 *For an ADF D and a three-valued interpretation I, the interpretation $\Gamma_D(I)$ is given by*

$$\Gamma_D(I)(s) = \bigsqcap\{w(\varphi_s) \mid w \in [I]_2\}$$

where $\sqcap$ is the consensus operation that assigns $\mathbf{t} \sqcap \mathbf{t} = \mathbf{t}$, $\mathbf{f} \sqcap \mathbf{f} = \mathbf{f}$, and assigns $\mathbf{u}$ otherwise.

We are now prepared to define admissibility based semantics.

[19]For a in depth discussion of the relation between the characteristic function in AFs and the Γ_D operator in ADFs in the context of approximation fixed-point theory the reader is referred to the chapter about Abstract Dialectical Frameworks in this handbook.

Definition 5.4 ([Brewka *et al.*, 2013]) *A three-valued interpretation I for an ADF D is*

- *the grounded interpretation iff it is the least fixed point of Γ_D.*
- *admissible iff $I \leq_i \Gamma_D(I)$;*
- *complete iff $I = \Gamma_D(I)$.*
- *preferred iff it is $\leq_i$-maximal admissible.*

Finally, one can define stable semantics which, in order to avoid cyclic support, makes use of a reduced ADF in the definition.

Definition 5.5 ([Brewka *et al.*, 2013]) *Let $D = (S, L, C)$ be an ADF with $C = \{\varphi_s\}_{s \in S}$. A two-valued model I of D is a stable model of D iff $E_I = \{s \in S : I(s) = \mathbf{t}\}$ equals the set of statements that are $\mathbf{t}$ in the grounded interpretation of the reduced ADF $D^I = (E_I, L^I, C^I)$, where $L^I = L \cap (E_I \times E_I)$ and for $s \in E_I$ we set $\varphi_s^I = \varphi_s[b/\mathbf{f} : I(b) = \mathbf{f}]$.*

5.2 Computational Problems

As the nature of abstract dialectical frameworks is quite similar to the nature of Dung's abstract argumentation frameworks also the core computational problems coincide. That is, we first have *credulous reasoning*, i.e., an argument is accepted if it is mapped to $\mathbf{t}$ by at least one interpretation, and *skeptical reasoning*, i.e., an argument is accepted only if it is mapped to $\mathbf{t}$ by all interpretations. These two reasoning modes again give rise to the following computational problems for argumentation semantics σ.

- *Credulous Acceptance Cred$_\sigma$*: Given ADF $D = (S, L, C)$ and an argument $a \in S$. Is there an interpretation $I \in \sigma(D)$ with $I(a) = \mathbf{t}$?
- *Skeptical Acceptance Skept$_\sigma$*: Given ADF $D = (S, L, C)$ and an argument $a \in S$. Is $I(a) = \mathbf{t}$ for each interpretation $I \in \sigma(D)$?

Beside these reasoning problems we also consider the problem of *verifying* a given interpretation, and deciding whether an ADF provides any *coherent conclusion.* Depending on the actual semantics the latter can corresponds to deciding whether the ADF has at least one interpretation, or whether the ADF has an interpretation that maps at least one statement to either $\mathbf{t}$ or $\mathbf{f}$.

- *Verification of an interpretation Ver$_\sigma$*: Given an ADF $D = (S, L, C)$ and an interpretation I. Is $I \in \sigma(F)$?
- *Existence of an interpretation Exists$_\sigma$*: Given an ADF $D = (S, L, C)$. Is $\sigma(F) \neq \emptyset$?
- *Existence of a non-trivial interpretation Exists$_\sigma^{\neg\emptyset}$*: Given an ADF $D = (S, L, C)$. Does there exist an interpretation I with $I(a) \in \{\mathbf{t}, \mathbf{f}\}$ for some argument $a \in S$.

Table 10. Complexity of ADFs ($\mathcal{C}$-c denotes completeness for class $\mathcal{C}$).

σ	$Cred_\sigma$	$Skept_\sigma$	Ver_σ	$Exists_\sigma$	$Exists_\sigma^{\neg\emptyset}$
gr	coNP-c	coNP-c	DP-c	trivial	coNP-c
model	NP-c	coNP-c	in P	NP-c	NP-c
st	Σ_2^{P}-c	Π_2^{P}-c	coNP-c	Σ_2^{P}-c	Σ_2^{P}-c
ad	Σ_2^{P}-c	trivial	coNP-c	trivial	Σ_2^{P}-c
co	Σ_2^{P}-c	coNP-c	DP-c	trivial	Σ_2^{P}-c
pr	Σ_2^{P}-c	Π_3^{P}-c	Π_2^{P}-c	trivial	Σ_2^{P}-c

5.3 Complexity Results for ADFs

The ability of ADFs to express more complex relations between arguments resulted in more evolved definitions of the semantics and as we will discuss next also increases the complexity of the core reasoning tasks. As the complexity results for ADFs in Table 10 show, all the non-trivial reasoning tasks are one level higher in the polynomial-hierarchy than for Dung's AFs. The main reason for that is the complexity of the Γ_D operator which replaces the characteristic function. While the characteristic function can be evaluated in polynomial time (and even logarithmic space) deciding problems associated with the Γ_D operator are in general NP/coNP-hard.[20]

In this section we discuss the complexity of admissible and grounded semantics in more detail.

Complexity of admissible semantics. Again the most fundamental problem is to verify that an interpretation is admissible. To show that the problem is in coNP we give an NP algorithm [Wallner, 2014] to falsify the admissibility of an interpretation I. Such an algorithm would guess an argument s, such that (i) $I(s) = \mathbf{t}$ or (ii) $I(s) = \mathbf{f}$, and a 2-valued interpretation $J \in [I]_2$ extending I such that either, in case (i), $J(\varphi_s) = \mathbf{f}$ or, in case (ii), $J(\varphi_s) = \mathbf{t}$. As I is admissible iff no such pair s, J exists this is a NP algorithm that falsifies I being admissible and thus the complementary problem of verifying an admissible interpretation is in coNP.

The coNP-hardness is by reduction from the coNP-complete UNSAT problem [Wallner, 2014] of testing whether a propositional formula is unsatisfiable. To this end consider a propositional formula φ over variables X and construct an ADF $D = (S, L, C)$ as follows: The set of arguments S consists of X and an additional argument a, each $x \in X$ is linked towards a, and the acceptance conditions are given by $c_x = x$ for $x \in X$ and $c_a = \varphi$. Now we consider the interpretation I mapping all $x \in X$ to $\mathbf{u}$ and a to $\mathbf{f}$. We have that the two-valued interpretations $J \in [I]_2$ correspond to the two-valued interpretations of φ and thus $\Gamma_D(I)(a) = \mathbf{f}$ iff φ has no model. That is, I is admissible iff

[20]For instance testing whether an argument is mapped to $\mathbf{t}$ is basically the validity problem of propositional logic.

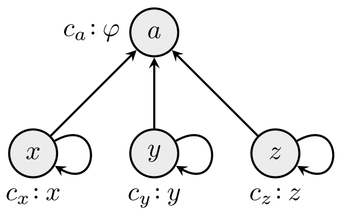

Figure 9. Illustration of the ADF constructed in the reduction from UNSAT to the problem of verifying an admissible interpretation in an ADF, for a propositional formula φ over atoms x, y, z.

φ is unsatisfiable and, as the ADF D can be constructed in polynomial time, coNP-hardness follows.

Combining the coNP verification algorithm for admissible semantics with the standard guess and check algorithms gives a Σ_2^{P} upper bound for credulous reasoning with admissible, complete and preferred semantics, and a Π_2^{P} upper bound for verifying a preferred interpretation. The latter then gives a Π_3^{P} algorithm for skeptical reasoning with preferred semantics.

Complexity of grounded semantics. The computational properties of grounded semantics in ADFs are quite in contrast to the computational properties of grounded semantics in AFs. When considering grounded semantics in ADFs, a straight forward algorithm is, starting from the three-valued model mapping all arguments to $\mathbf{u}$, and then iteratively apply the operator Γ_D until a fixed-point is reached. The straight forward algorithm is only a P^{NP}-algorithm, because of the costly evaluation of Γ_D. However, due to a sophisticated characterisation of grounded semantics [Wallner, 2014] there is a more efficient way to test whether an argument is mapped to $\mathbf{t}$ in the grounded interpretation of an ADF. Also notice that verifying the grounded interpretation is in DP as we have to do verify both the $\mathbf{t}$, $\mathbf{f}$ assignments and the $\mathbf{u}$ assignments.

The DP-hardness of verifying the grounded interpretation is by a reduction from the DP-complete SAT–UNSAT problem [Brewka and Woltran, 2010; Wallner, 2014]. To this end consider an instance (φ, ψ) of SAT–UNSAT where φ is a propositional formula over atoms X and ψ is a propositional formula over different atoms Y. In polynomial time we construct the ADF $D = (S, L, C)$ with $S = X \cup Y \cup \{d, s, v\}$, $L = \{(x, s) \mid x \in X\} \cup \{(y, v) \mid y \in Y\} \cup \{(d, s)\}$ and the acceptance conditions $c_x = x$ for $x \in X \cup Y$, $c_d = d$, $c_s = \varphi \wedge d$ and $c_v = \psi$. Now we consider the interpretation I with $I(v) = \mathbf{f}$ and $I(a) = \mathbf{u}$ for all the other arguments $a \in S \setminus \{v\}$. We next argue that I is the grounded model iff (φ, ψ) is a “yes” instance of SAT–UNSAT. Let G be the grounded model. First notice that the arguments $a \in X \cup Y \cup \{d\}$ do not have incoming edges from other arguments. Whenever $J(a) = \mathbf{u}$ then there are both an $I_1 \in [J]_2$ with $I_1(a) = \mathbf{t}$ and an $I_2 \in [J]_2$ with $I_2(a) = \mathbf{f}$, and thus also $\Gamma_D(J)(a) = \mathbf{u}$. That is, the grounded model G maps all arguments in $X \cup Y \cup \{d\}$ to $\mathbf{u}$. Now consider the

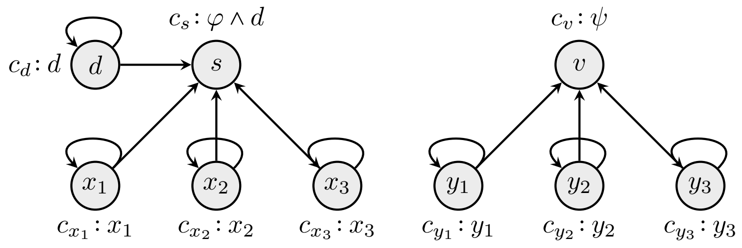

Figure 10. Illustration of the ADF constructed in the reduction from SAT–UNSAT to the problem of verifying the grounded model of an ADF, for an propositional formulae φ, ψ over atoms $X = \{x_1, x_2, x_3\}$ and $Y = \{y_1, y_2 y_3\}$ respectively.

argument s and $c_s = \varphi \wedge d$. The two-valued interpretations $J \in [G]_2$ correspond to the two-valued interpretations over $X \cup Y \cup \{d\}$. That is, either (a) φ has a model and we can satisfy $\varphi \wedge d$ by setting d to $\mathbf{t}$ as well as falsify $\varphi \wedge d$ by setting d to $\mathbf{f}$ and thus $\Gamma_D(G)(s) = G(s) = \mathbf{u}$, or (b) φ is unsatisfiable and thus $\Gamma_D(J)(s) = G(s) = \mathbf{f}$. One the other hand, for v and $c_v = \psi$, we have that either ψ is unsatisfiable and $\Gamma_D(G)(v) = G(v) = \mathbf{f}$, ψ is satisfiable but not valid and $\Gamma_D(G)(v) = G(v) = \mathbf{u}$, or ψ is valid and $\Gamma_D(G)(v) = G(v) = \mathbf{t}$. Hence, we have that $G = I$ iff φ is satisfiable and ψ is unsatisfiable.

5.4 Complexity of Bipolar ADFs with Known Link Types

Again there are certain instances of ADFs that do not have the worst-case complexity, but can be processed with milder complexity. Here we discuss so called *Bipolar ADFs* which put some restriction on the link structure, i.e., each link has to be supporting or attacking (but might be both). For a given set $X \subseteq S$ let I_X be the two-valued interpretation with $I^{\mathbf{t}} = X$ and $I^{\mathbf{f}} = S \setminus X$. A link (a, b) is called *supporting* if there is no $X \subseteq S$ such that $I_X \models \varphi_b$ and $I_{X \cup \{a\}} \not\models \varphi_b$; whereas it is called *attacking* if there is no $X \subseteq S$ such that $I_X \not\models \varphi_b$ and $I_{X \cup \{a\}} \models \varphi_b$. While in general testing the link type is itself coNP-complete [Brewka and Woltran, 2010; Ellmauthaler, 2012] there are certain applications of ADFs where the link type is known beforehand [Brewka and Gordon, 2010; Strass, 2013]. This motivates the research on *bipolar AFs with know link types* which we discuss in the remainder of this section.

The main observation that leads to the better complexity results for bipolar AFs (see Table 11) is that the operator Γ_D can be efficiently computed when all the links are attacking or supporting. The matching hardness results are then by the lower bounds for Dung's abstract argumentation (cf. Table 1) and the observation that AFs can be interpreted as bipolar ADFs with known link types [Brewka *et al.*, 2013] as follows.[21] Given an AF (A, R) the equivalent

[21]Notice that both ADF semantics models and stable models are generalisations of Dung's stable semantics.

Table 11. Complexity of Bipolar ADFs with know link types ($\mathcal{C}$-c denotes completeness for class $\mathcal{C}$).

σ	$Cred_\sigma$	$Skept_\sigma$	Ver_σ	$Exists_\sigma$	$Exists_\sigma^{\neg\emptyset}$
gr	P-c	P-c	P-c	trivial	in P
model	NP-c	coNP-c	in P	NP-c	NP-c
st	NP-c	coNP-c	in P	NP-c	NP-c
ad	NP-c	trivial	in P	trivial	NP-c
co	NP-c	P-c	P-c	trivial	NP-c
pr	NP-c	Π_2^{P}-c	coNP-c	trivial	NP-c

ADF is given by (A, R, C) with $C = \{c_a : \bigwedge_{(b,a)\in R} \neg b \mid a \in A\}$. Notice that all the links are indeed attacking.

To compute $\Gamma_D(I)$ in general ADFs, we have to consider all 2-valued interpretations that extend I, which is coNP-hard, but given the link type of each link we only have to check two 2-valued interpretations for each argument as follows [Wallner, 2014; Strass and Wallner, 2014]. Let *Supp* be the set of supporting links and *Att* the set of attacking links, but there might be links that are both supporting and attacking (such links are called redundant links). For $s \in S$ consider the the formula φ_s and the interpretations J_1, J_2 as follows.

1. $J_1(s) = \begin{cases} \mathbf{t} & \text{if } I(s) = \mathbf{t}, \text{ or } I(s) = \mathbf{u} \text{ and } (s,a) \in \mathit{Supp} \\ \mathbf{f} & \text{if } I(s) = \mathbf{f}, \text{ or } I(s) = \mathbf{u} \text{ and } (s,a) \in \mathit{Att} \setminus \mathit{Supp} \end{cases}$

2. $J_2(s) = \begin{cases} \mathbf{t} & \text{if } I(s) = \mathbf{t}, \text{ or } I(s) = \mathbf{u} \text{ and } (s,a) \in \mathit{Att} \\ \mathbf{f} & \text{if } I(s) = \mathbf{f}, \text{ or } I(s) = \mathbf{u} \text{ and } (s,a) \in \mathit{Supp} \setminus \mathit{Att} \end{cases}$

The interpretation J_1 sets all yet undecided supporters to true and all yet undecided (non-redundant) attackers to false. If $J_1(\varphi_s) = \mathbf{f}$ then no 2-valued interpretation extending I satisfies φ_s, and thus $\Gamma_D(I)(s) = \mathbf{f}$. Otherwise if $J_1(\varphi_s) = \mathbf{t}$ then clearly $\Gamma_D(I)(s) \neq \mathbf{f}$. The interpretation J_2 sets all yet undecided (non-redundant) supporters to false and all yet undecided attackers to true. Now, whenever $J_2(\varphi_s) = \mathbf{t}$ then all 2-valued interpretations extending I satisfy φ_s, and thus $\Gamma_D(I)(s) = \mathbf{t}$. Otherwise if $J_2(\varphi_s) = \mathbf{f}$ then clearly $\Gamma_D(I)(s) \neq \mathbf{t}$. Hence, we can compute $\Gamma_D(I)$ by just considering J_1 and J_2 and set $\Gamma_D(I)(s) = \mathbf{t}$ if $J_2(\varphi_s) = \mathbf{t}$; $\Gamma_D(I)(s) = \mathbf{f}$ if $J_1(\varphi_s) = \mathbf{f}$; and $\Gamma_D(I)(s) = \mathbf{u}$ otherwise.

Now, as $\Gamma_D(I)$ can be computed in polynomial time, we can also (i) efficiently compute the grounded model by iteratively applying $\Gamma_D(I)$. Moreover, (ii) verifying an admissible or complete interpretation just requires to apply the Γ_D operator once and thus can be done in polynomial time. The remaining results in Table 11 are by the combination of the polynomial-time verification algorithms with the standard guess and check algorithms as used for Dung's AFs.

6 Discussion

In this chapter we presented complexity results for the three argumentation formalisms of Dung's Abstract Argumentation Frameworks, Assumption-based Argumentation and Abstract Dialectical Frameworks. We have identified several sources of computational complexity: (i) the construction of arguments and the interlinking structure, e.g., the attack relation, (ii) the search for coherent sets of arguments, and (iii) the decision about certain conclusions. Points (i) and (iii) are present in the complexity results for Assumption-based Argumentation where the complexity of algorithms heavily depends on the complexity of the derivability relation, which is the essential ingredient to build arguments, identify conflicts, and draw conclusions. Point (ii) is present in all three formalisms. The discussed results show that the actual computational complexity in this step may highly depend on the chosen semantics and reasoning task. Moreover, faced with the typically high complexity we discussed approaches to identify instances with lower complexity and solve them more efficiently.

Implications for the Design of Systems and Algorithms. First given the upper bounds of the complexity analysis we have first guidelines how to implement argumentation semantics, and on the computational resources required for that. An efficient system should process any instance within the resources given by the upper bound but moreover also should perform better on easier instances. In particular an efficient system should also be able to process instances that fall into one of the tractable fragments with milder complexity more efficiently. A system for abstract argumentation that is explicitly built around this idea is CEGARTIX [Dvořák *et al.*, 2014a], that is based on easier fragments for semantics on the second level of the polynomial-hierarchy.

The complexity classification of a semantics is also crucial for reduction-based implementations. To get an appropriate reduction the target formalism should have a similar complexity as the argumentation semantics, or one should only use a fragment of the target formalism with similar complexity. One example is the ASPARTIX [Egly *et al.*, 2010] system that encodes abstract argumentation problems in logic-programming in a query-based fashion. That is, the system provides fixed encodings for the supported argumentation semantics (the queries) that are then evaluated on the encoding of considered AF (the input data).[22] The polynomial-time computable grounded semantics is encoded as stratified logic program, a fragment whose data-complexity is in polynomial time (even P-complete), the semantics at the NP, coNP level are encoded as programs without disjunction in the rule heads, the data-complexity of this kind of logic programs is on the NP / coNP level, and the full expressiveness of disjunctive logic programs is only used for the argumentation semantics whose

[22]The specific encoding of an AF as logic program has became popular beyond the logic programming setting as the so-called ASPARTIX-format for encoding AFs.

complexity is at the second level of the polynomial-hierarchy.[23]

Another example is the work on intertranslatability of abstract argumentation semantics where one aims to efficiently translate one argumentation semantics to another, by modifying the argumentation framework [Dvořák and Woltran, 2011; Dvořák and Spanring, 2016]. Here a gap in the complexities of the semantics immediately gives a negative result.

Function Complexity. In this chapter we restricted ourselves to what we consider to be the core computational problems and in particular to decision problems. In terms of computational complexity function problems, problems where one wants to compute a number, extensions or the set of extensions, are only rarely studied, notable exceptions are the research line on ideal semantics [Dunne, 2009; Dunne *et al.*, 2013], the work on counting the number of extensions [Baroni *et al.*, 2010], and the work on computing an admissible set that results in a minimal socratic discussion [Caminada *et al.*, 2016]. Recently, Kröll *et al.* started the research on enumeration complexity in abstract argumentation [Kröll *et al.*, 2017], where one is interested in the computational cost per extension.

Fine-Grained Lower Bounds. Lower bounds from classical computational complexity theory like NP-hardness indicate that there are no polynomial-time algorithms. However, they neither indicate lower bounds for the constants in the exponent of exponential running times nor rule out subexponential algorithms at all. That is, there is still some gap between the best known algorithms for the hard problems, they are exponential-time (see, e.g., [Nofal *et al.*, 2014]), and the existing lower bounds. To overcome this gap, in the field of combinatorial algorithms, so called conditional lower bounds are studied (see, e.g., [Abboud and Williams, 2014]). That is, one uses conjectures about lower bounds for well studied algorithmic problems. To obtain a lower bound for a new problem one then reduces the problem from the conjecture to the problem under question such that a faster algorithm for the new problem would imply a faster algorithm for the original problem and thus would contradict the conjecture. By that one gets an algorithmic lower bound for the new problem conditioned on the original conjecture. Probably the most prominent such conjecture is the (strong) exponential-time hypothesis (S)ETH [Impagliazzo and Paturi, 1999], with ETH conjecturing that there is no subexponential algorithm for 3-SAT, and SETH conjecturing that there is no algorithm for CNF-SAT that runs in time $2^{(1-\epsilon)n} \cdot poly(n,m)$, for every constant $\epsilon > 0$ and polynomial $poly(n,m)$. As many of the existing reductions in formal argumentation are based on propositional logic (S)ETH is also a promising starting point for closing these complexity gaps in formal argumentation.

Complexity Analysis of further Argumentation Formalisms. In this chapter we only cover three argumentation formalisms, while there are many more around and many of them come with a complexity analysis. Below we give a brief overview and pointers to the relevant literature. First, there are

[23]For a survey on the complexity of logic-programs see [Dantsin *et al.*, 2001].

formalisms, e.g. AFRAs [Baroni *et al.*, 2011b], that extend Dung's Abstract argumentation frameworks and can be efficiently reduced to them. For such formalisms the complexity results for AFs directly extend to the new formalism. Second, there are extensions of AFs that can not be reduced in such a direct way and thus need their own complexity analysis. Most prominently: The complexity of Extended argumentation frameworks was studied in [Dunne *et al.*, 2010] and later complemented by a result in [Dvořák *et al.*, 2015]; Valued-based argumentation has been discussed in an earlier survey [Dunne and Wooldridge, 2009] on the complexity of abstract argumentation and more recent results can be found in, e.g., [Dunne, 2010; Kim *et al.*, 2011]; weighted argumentation systems and their complexity have been studied in [Dunne *et al.*, 2011]; and Constrained Argumentation Frameworks [Coste-Marquis *et al.*, 2006]. Finally, there are complexity results for logic-based argumentation formalisms. Complexity aspects of Deductive Argumentation were for instance considered in [Besnard *et al.*, 2009; Creignou *et al.*, 2011], while the complexity of Defeasible Logic Programming (DeLP) was studied in [Cecchi *et al.*, 2006].

Complexity Analysis in Formal Argumentation. At the current state of the field of formal argumentation in most argumentation formalisms we already have a good understanding of the computational complexity of the fundamental problems and the important semantics. However, this by no means says that all research questions in that direction are solved. Indeed the field of formal argumentation is very active and with almost every new research topic there come associated computational problems that should be analysed w.r.t. their computational complexity. Let us exemplify three such occasion where a complexity analysis can deepen our understanding: (a) For a newly proposed semantics the complexity of the fundamental reasoning problems should be analysed in order to compare it with existing semantics and identified computational benefits/drawbacks. (b) When expanding existing argumentation formalisms with additional (syntactic) concepts one is interested in the (additional) computational costs of these concepts. That is by how much the complexity increases or whether one can add these concepts without any computational drawbacks. (c) When considering novel tasks for argumentation systems a complexity classification gives a first impression on the feasibility of the new task and guides the way to efficient implementations. For instance, recently the field of dynamics of argumentation received some attention [Diller *et al.*, 2015; Snaith and Reed, 2016; Wallner *et al.*, 2016; Kim *et al.*, 2013] and raised a couple of computational problems, e.g., the so-called extension enforcement problem [Baumann and Brewka, 2010] where one aims to modify an AF such that a certain set of arguments becomes acceptable. The work of [Wallner *et al.*, 2016] first gives a comprehensive complexity analysis of the enforcement problem and then turns these results into algorithms and the prototype system Pakota [24].

[24] `https://www.cs.helsinki.fi/group/coreo/pakota/`

Acknowledgments

The authors are grateful to Hannes Strass, Johannes P. Wallner and an anonymous reviewer for their thoughtful comments on earlier versions of this chapter which helped to improve the quality of presentation.

BIBLIOGRAPHY

[Abboud and Williams, 2014] Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 434–443. IEEE Computer Society, 2014.

[Arieli and Caminada, 2012] Ofer Arieli and Martin W. A. Caminada. A general QBF-based formalization of abstract argumentation theory. In Bart Verheij, Stefan Szeider, and Stefan Woltran, editors, *Computational Models of Argument: Proceedings of COMMA 2012, Vienna, Austria, September 10-12, 2012*, volume 245 of *Frontiers in Artificial Intelligence and Applications*, pages 105–116. IOS Press, 2012.

[Arora and Barak, 2009] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[Baroni *et al.*, 2005] Pietro Baroni, Massimiliano Giacomin, and Giovanni Guida. SCC-recursiveness: A general schema for argumentation semantics. *Artif. Intell.*, 168(1-2):162–210, 2005.

[Baroni *et al.*, 2010] Pietro Baroni, Paul E. Dunne, and Massimiliano Giacomin. On extension counting problems in argumentation frameworks. In Pietro Baroni, Federico Cerutti, Massimiliano Giacomin, and Guillermo Ricardo Simari, editors, *Computational Models of Argument: Proceedings of COMMA 2010, Desenzano del Garda, Italy, September 8-10, 2010*, volume 216 of *Frontiers in Artificial Intelligence and Applications*, pages 63–74. IOS Press, 2010.

[Baroni *et al.*, 2011a] Pietro Baroni, Martin Caminada, and Massimiliano Giacomin. An introduction to argumentation semantics. *Knowledge Eng. Review*, 26(4):365–410, 2011.

[Baroni *et al.*, 2011b] Pietro Baroni, Federico Cerutti, Massimiliano Giacomin, and Giovanni Guida. AFRA: Argumentation framework with recursive attacks. *Int. J. Approx. Reasoning*, 52(1):19–37, 2011.

[Baroni *et al.*, 2011c] Pietro Baroni, Paul E. Dunne, and Massimiliano Giacomin. On the resolution-based family of abstract argumentation semantics and its grounded instance. *Artif. Intell.*, 175(3-4):791–813, 2011.

[Baumann and Brewka, 2010] Ringo Baumann and Gerhard Brewka. Expanding argumentation frameworks: Enforcing and monotonicity results. In Pietro Baroni, Federico Cerutti, Massimiliano Giacomin, and Guillermo Ricardo Simari, editors, *Computational Models of Argument: Proceedings of COMMA 2010, Desenzano del Garda, Italy, September 8-10, 2010*, volume 216 of *Frontiers in Artificial Intelligence and Applications*, pages 75–86. IOS Press, 2010.

[Besnard and Doutre, 2004] Philippe Besnard and Sylvie Doutre. Checking the acceptability of a set of arguments. In James P. Delgrande and Torsten Schaub, editors, *Proceedings of the 10th International Workshop on Non-Monotonic Reasoning (NMR 2004)*, pages 59–64, 2004.

[Besnard *et al.*, 2009] Philippe Besnard, Anthony Hunter, and Stefan Woltran. Encoding deductive argumentation in quantified boolean formulae. *Artificial Intelligence*, 173(15):1406 – 1423, 2009.

[Bondarenko *et al.*, 1997] Andrei Bondarenko, Phan Minh Dung, Robert A. Kowalski, and Francesca Toni. An abstract, argumentation-theoretic approach to default reasoning. *Artif. Intell.*, 93:63–101, 1997.

[Brewka and Gordon, 2010] Gerhard Brewka and Thomas F. Gordon. Carneades and abstract dialectical frameworks: A reconstruction. In Pietro Baroni, Federico Cerutti, Massimiliano Giacomin, and Guillermo Ricardo Simari, editors, *Computational Models of Argument: Proceedings of COMMA 2010, Desenzano del Garda, Italy, September 8-10, 2010*, volume 216 of *Frontiers in Artificial Intelligence and Applications*, pages 3–12. IOS Press, 2010.

[Brewka and Woltran, 2010] Gerhard Brewka and Stefan Woltran. Abstract dialectical frameworks. In Fangzhen Lin, Ulrike Sattler, and Miroslaw Truszczynski, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference, KR 2010, Toronto, Ontario, Canada, May 9-13, 2010*, pages 780–785. AAAI Press, 2010.

[Brewka *et al.*, 2013] Gerhard Brewka, Hannes Strass, Stefan Ellmauthaler, Johannes Peter Wallner, and Stefan Woltran. Abstract dialectical frameworks revisited. In Francesca Rossi, editor, *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, pages 803–809. IJCAI/AAAI, 2013.

[Caminada and Gabbay, 2009] Martin Caminada and Dov M. Gabbay. A logical account of formal argumentation. *Studia Logica*, 93(2):109–145, 2009.

[Caminada *et al.*, 2012] Martin Caminada, Walter A. Carnielli, and Paul E. Dunne. Semi-stable semantics. *J. Log. Comput.*, 22:1207–1254, 2012.

[Caminada *et al.*, 2016] Martin Caminada, Wolfgang Dvořák, and Srdjan Vesic. Preferred semantics as socratic discussion. *J. Log. Comput.*, 26:1257–1292, 2016.

[Caminada, 2007] Martin Caminada. Comparing two unique extension semantics for formal argumentation: ideal and eager. In *Proceedings of the 19th Belgian-Dutch Conference on Artificial Intelligence (BNAIC 2007)*, pages 81–87, 2007.

[Cecchi *et al.*, 2006] Laura A. Cecchi, Pablo R. Fillottrani, and Guillermo R. Simari. On the complexity of DeLP through game semantics. In *11th. Intl. Workshop on Nonmonotonic Reasoning*, pages 386–394, 2006.

[Cerutti *et al.*, 2014] Federico Cerutti, Massimiliano Giacomin, Mauro Vallati, and Marina Zanella. An SCC recursive meta-algorithm for computing preferred labellings in abstract argumentation. In Chitta Baral, Giuseppe De Giacomo, and Thomas Eiter, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR 2014, Vienna, Austria, July 20-24, 2014*, pages 42–51. AAAI Press, 2014.

[Charwat, 2012] Günther Charwat. Tree-decomposition based algorithms for abstract argumentation frameworks. Master's thesis, Vienna University of Technology, 2012. Stefan Woltran and Wolfgang Dvořák advisors.

[Chen *et al.*, 2008] Jianer Chen, Yang Liu, Songjian Lu, Barry O'Sullivan, and Igor Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. *J. ACM*, 55(5):21:1–21:19, 2008.

[Coste-Marquis *et al.*, 2005] Sylvie Coste-Marquis, Caroline Devred, and Pierre Marquis. Symmetric argumentation frameworks. In Lluis Godo, editor, *Proceedings of the 8th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2005)*, volume 3571 of *Lecture Notes in Computer Science*, pages 317–328. Springer, 2005.

[Coste-Marquis *et al.*, 2006] Sylvie Coste-Marquis, Caroline Devred, and Pierre Marquis. Constrained argumentation frameworks. In Patrick Doherty, John Mylopoulos, and Christopher A. Welty, editors, *Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning, Lake District of the United Kingdom, June 2-5, 2006*, pages 112–122. AAAI Press, 2006.

[Creignou *et al.*, 2011] Nadia Creignou, Johannes Schmidt, Michael Thomas, and Stefan Woltran. Complexity of logic-based argumentation in Post's framework. *Argument & Computation*, 2(2-3):107–129, 2011.

[Cygan *et al.*, 2015] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms.* Springer, 2015.

[Dantsin *et al.*, 2001] Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov. Complexity and expressive power of logic programming. *ACM Comput. Surv.*, 33(3):374–425, 2001.

[Diller *et al.*, 2015] Martin Diller, Adrian Haret, Thomas Linsbichler, Stefan Rümmele, and Stefan Woltran. An extension-based approach to belief revision in abstract argumentation. In Qiang Yang and Michael Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 2926–2932. AAAI Press, 2015.

[Dimopoulos and Torres, 1996] Yannis Dimopoulos and Alberto Torres. Graph theoretical structures in logic programs and default theories. *Theor. Comput. Sci.*, 170(1-2):209–244, 1996.

[Dimopoulos *et al.*, 1999] Yannis Dimopoulos, Bernhard Nebel, and Francesca Toni. Preferred arguments are harder to compute than stable extension. In Thomas Dean, editor, *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99, Stockholm, Sweden, July 31 - August 6, 1999. 2 Volumes, 1450 pages*, pages 36–43. Morgan Kaufmann, 1999.

[Dimopoulos *et al.*, 2000] Yannis Dimopoulos, Bernhard Nebel, and Francesca Toni. Finding admissible and preferred arguments can be very hard. In Anthony G. Cohn, Fausto Giunchiglia, and Bart Selman, editors, *KR 2000, Principles of Knowledge Representation and Reasoning Proceedings of the Seventh International Conference, Breckenridge, Colorado, USA, April 11-15, 2000.*, pages 53–61. Morgan Kaufmann, 2000.

[Dimopoulos *et al.*, 2002] Yannis Dimopoulos, Bernhard Nebel, and Francesca Toni. On the computational complexity of assumption-based argumentation for default reasoning. *Artif. Intell.*, 141(1/2):57–78, 2002.

[Dung *et al.*, 2006] Phan Minh Dung, Paolo Mancarella, and Francesca Toni. A dialectic procedure for sceptical, assumption-based argumentation. In Paul E. Dunne and Trevor J. M. Bench-Capon, editors, *Computational Models of Argument: Proceedings of COMMA 2006, September 11-12, 2006, Liverpool, UK*, volume 144 of *Frontiers in Artificial Intelligence and Applications*, pages 145–156. IOS Press, 2006.

[Dung *et al.*, 2007] Phan Minh Dung, Paolo Mancarella, and Francesca Toni. Computing ideal sceptical argumentation. *Artif. Intell.*, 171(10-15):642–674, 2007.

[Dung, 1995] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.*, 77(2):321–358, 1995.

[Dunne and Bench-Capon, 2001] Paul E. Dunne and Trevor J. M. Bench-Capon. Complexity and combinatorial properties of argument systems. Technical report, Dept. of Computer Science, University of Liverpool, 2001.

[Dunne and Bench-Capon, 2002] Paul E. Dunne and Trevor J. M. Bench-Capon. Coherence in finite argument systems. *Artif. Intell.*, 141(1/2):187–203, 2002.

[Dunne and Wooldridge, 2009] Paul E. Dunne and Michael Wooldridge. Complexity of abstract argumentation. In Guillermo Simari and Iyad Rahwan, editors, *Argumentation in Artificial Intelligence*, pages 85–104. Springer US, 2009.

[Dunne *et al.*, 2010] Paul E. Dunne, Sanjay Modgil, and Trevor J. M. Bench-Capon. Computation in extended argumentation frameworks. In *ECAI 2010 - 19th European Conference on Artificial Intelligence, Lisbon, Portugal, August 16-20, 2010, Proceedings*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, pages 119–124. IOS Press, 2010.

[Dunne *et al.*, 2011] Paul E. Dunne, Anthony Hunter, Peter McBurney, Simon Parsons, and Michael Wooldridge. Weighted argument systems: Basic definitions, algorithms, and complexity results. *Artif. Intell.*, 175(2):457–486, 2011.

[Dunne *et al.*, 2013] Paul E. Dunne, Wolfgang Dvořák, and Stefan Woltran. Parametric properties of ideal semantics. *Artif. Intell.*, 202(0):1 – 28, 2013.

[Dunne, 2007] Paul E. Dunne. Computational properties of argument systems satisfying graph-theoretic constraints. *Artif. Intell.*, 171(10-15):701–729, 2007.

[Dunne, 2008] Paul E. Dunne. The computational complexity of ideal semantics I: Abstract argumentation frameworks. In Philippe Besnard, Sylvie Doutre, and Anthony Hunter, editors, *Computational Models of Argument: Proceedings of COMMA 2008, Toulouse, France, May 28-30, 2008*, volume 172 of *Frontiers in Artificial Intelligence and Applications*, pages 147–158. IOS Press, 2008.

[Dunne, 2009] Paul E. Dunne. The computational complexity of ideal semantics. *Artif. Intell.*, 173(18):1559–1591, 2009.

[Dunne, 2010] Paul E. Dunne. Tractability in value-based argumentation. In Pietro Baroni, Federico Cerutti, Massimiliano Giacomin, and Guillermo Ricardo Simari, editors, *Computational Models of Argument: Proceedings of COMMA 2010, Desenzano del Garda, Italy, September 8-10, 2010*, volume 216 of *Frontiers in Artificial Intelligence and Applications*, pages 195–206. IOS Press, 2010.

[Dvořák and Gaggl, 2016] Wolfgang Dvořák and Sarah Alice Gaggl. Stage semantics and the SCC-recursive schema for argumentation semantics. *J. Log. Comput.*, 26(4):1149–1202, 2016.

[Dvořák and Spanring, 2016] Wolfgang Dvořák and Christof Spanring. Comparing the expressiveness of argumentation semantics. *J. Log. Comput.*, in press (available online), 2016.

[Dvořák and Woltran, 2010] Wolfgang Dvořák and Stefan Woltran. Complexity of semi-stable and stage semantics in argumentation frameworks. *Inf. Process. Lett.*, 110(11):425–430, 2010.

[Dvořák and Woltran, 2011] Wolfgang Dvořák and Stefan Woltran. On the intertranslatability of argumentation semantics. *J. Artif. Intell. Res. (JAIR)*, 41:445–475, 2011.

[Dvořák *et al.*, 2010] Wolfgang Dvořák, Stefan Szeider, and Stefan Woltran. Reasoning in argumentation frameworks of bounded clique-width. In Pietro Baroni, Federico Cerutti, Massimiliano Giacomin, and Guillermo Ricardo Simari, editors, *Computational Models of Argument: Proceedings of COMMA 2010, Desenzano del Garda, Italy, September 8-10, 2010*, volume 216 of *Frontiers in Artificial Intelligence and Applications*, pages 219–230. IOS Press, 2010.

[Dvořák *et al.*, 2012a] Wolfgang Dvořák, Sebastian Ordyniak, and Stefan Szeider. Augmenting tractable fragments of abstract argumentation. *Artif. Intell.*, 186(0):157–173, 2012.

[Dvořák *et al.*, 2012b] Wolfgang Dvořák, Reinhard Pichler, and Stefan Woltran. Towards fixed-parameter tractable algorithms for abstract argumentation. *Artif. Intell.*, 186(0):1 – 37, 2012.

[Dvořák *et al.*, 2012c] Wolfgang Dvořák, Stefan Szeider, and Stefan Woltran. Abstract argumentation via monadic second order logic. In Eyke Hüllermeier, Sebastian Link, Thomas Fober, and Bernhard Seeger, editors, *Scalable Uncertainty Management - 6th International Conference, SUM 2012, Marburg, Germany, September 17-19, 2012. Proceedings*, volume 7520 of *Lecture Notes in Computer Science*, pages 85–98. Springer, 2012.

[Dvořák *et al.*, 2014a] Wolfgang Dvořák, Matti Järvisalo, Johannes Peter Wallner, and Stefan Woltran. Complexity-sensitive decision procedures for abstract argumentation. *Artif. Intell.*, 206(0):53 – 78, 2014.

[Dvořák *et al.*, 2014b] Wolfgang Dvořák, Thomas Linsbichler, Emilia Oikarinen, and Stefan Woltran. Resolution-based grounded semantics revisited. In Simon Parsons, Nir Oren, Chris Reed, and Federico Cerutti, editors, *Computational Models of Argument: Proceedings of COMMA 2014, Atholl Palace Hotel, Scottish Highlands, UK, September 9-12, 2014*, volume 266 of *Frontiers in Artificial Intelligence and Applications*, pages 269–280. IOS Press, 2014.

[Dvořák *et al.*, 2015] Wolfgang Dvořák, Sarah Alice Gaggl, Thomas Linsbichler, and Johannes Peter Wallner. Reduction-based approaches to implement Modgil's extended argumentation frameworks. In Thomas Eiter, Hannes Strass, Miroslaw Truszczynski, and Stefan Woltran, editors, *Advances in Knowledge Representation, Logic Programming, and Abstract Argumentation - Essays Dedicated to Gerhard Brewka on the Occasion of His 60th Birthday*, volume 9060 of *Lecture Notes in Computer Science*, pages 249–264. Springer, 2015.

[Dvořák, 2012a] Wolfgang Dvořák. *Computational Aspects of Abstract Argumentation.* PhD thesis, Vienna University of Technology, Institute of Information Systems, 2012.

[Dvořák, 2012b] Wolfgang Dvořák. On the complexity of computing the justification status of an argument. In Sanjay Modgil, Nir Oren, and Francesca Toni, editors, *Theory and Applications of Formal Argumentation - First International Workshop, TAFA 2011. Barcelona, Spain, July 16-17, 2011, Revised Selected Papers*, volume 7132 of *Lecture Notes in Computer Science*, pages 32–49. Springer, 2012.

[Dvořák, 2012c] Wolfgang Dvořák. Technical note: Exploring Σ_2^P / Π_2^P -hardness for argumentation problems with fixed distance to tractable classes. *CoRR*, abs/1201.0478, 2012.

[Dvořák, 2017] Wolfgang Dvořák. Technical note: On the complexity of the uniqueness problem in abstract argumentation. Technical Report DBAI-TR-2008-xy, Technische Universität Wien, 2017.

[Egly and Woltran, 2006] Uwe Egly and Stefan Woltran. Reasoning in argumentation frameworks using quantified boolean formulas. In Paul E. Dunne and Trevor J. M. Bench-Capon, editors, *Computational Models of Argument: Proceedings of COMMA 2006, September*

11-12, 2006, Liverpool, UK, volume 144 of *Frontiers in Artificial Intelligence and Applications*, pages 133–144. IOS Press, 2006.

[Egly *et al.*, 2010] Uwe Egly, Sarah Alice Gaggl, and Stefan Woltran. Answer-set programming encodings for argumentation frameworks. *Argument and Computation*, 1(2):147–177, 2010.

[Eiter and Gottlob, 1993] Thomas Eiter and Georg Gottlob. Propositional circumscription and extended closed-world reasoning are Π_2^P-complete. *Theor. Comput. Sci.*, 114(2):231–245, 1993.

[Ellmauthaler, 2012] Stefan Ellmauthaler. Abstract dialectical frameworks: Properties, complexity, and implementation. Master's thesis, Vienna University of Technology, 2012. Stefan Woltran and Johannes Peter Wallner advisors.

[Flum and Grohe, 2006] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006.

[Gaggl and Woltran, 2013] Sarah Alice Gaggl and Stefan Woltran. The cf2 argumentation semantics revisited. *J. Log. Comput.*, 23(5):925–949, 2013.

[Gelfond and Lifschitz, 1988] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert A. Kowalski and Kenneth A. Bowen, editors, *Logic Programming, Proceedings of the Fifth International Conference and Symposium, Seattle, Washington, August 15-19, 1988 (2 Volumes)*, pages 1070–1080. MIT Press, 1988.

[Greenlaw *et al.*, 1995] Raymond Greenlaw, H.James Hoover, and Walter L. Ruzzo. *Limits to parallel computation: P-completeness theory*. Oxford University Press, 1995.

[Impagliazzo and Paturi, 1999] Russell Impagliazzo and Ramamohan Paturi. Complexity of k-sat. In *Proceedings of the 14th Annual IEEE Conference on Computational Complexity, Atlanta, Georgia, USA, May 4-6, 1999*, pages 237–240. IEEE Computer Society, 1999.

[Kasif, 1986] Simon Kasif. On the parallel complexity of some constraint satisfaction problems. In Tom Kehler, editor, *Proceedings of the 5th National Conference on Artificial Intelligence. Philadelphia, PA, August 11-15, 1986. Volume 1: Science*, pages 349–353. Morgan Kaufmann, 1986.

[Kim *et al.*, 2011] Eun Jung Kim, Sebastian Ordyniak, and Stefan Szeider. Algorithms and complexity results for persuasive argumentation. *Artif. Intell.*, 175(9-10):1722–1736, 2011.

[Kim *et al.*, 2013] Eun Jung Kim, Sebastian Ordyniak, and Stefan Szeider. The complexity of repairing, adjusting, and aggregating of extensions in abstract argumentation. In Elizabeth Black, Sanjay Modgil, and Nir Oren, editors, *Theory and Applications of Formal Argumentation - Second International Workshop, TAFA 2013, Beijing, China, August 3-5, 2013, Revised Selected papers*, volume 8306 of *Lecture Notes in Computer Science*, pages 158–175. Springer, 2013.

[Kleene, 1952] S.C. Kleene. *Introduction to Metamathematics*. Bibliotheca Mathematica. North-Holland, 1952.

[Kröll *et al.*, 2017] Markus Kröll, Reinhard Pichler, and Stefan Woltran. On the complexity of enumerating the extensions of abstract argumentation frameworks. In *IJCAI 2017 (to appear)*, 2017.

[Moore, 1985] Robert C. Moore. Semantical considerations on nonmonotonic logic. *Artif. Intell.*, 25(1):75–94, 1985.

[Niedermeier, 2006] Rolf Niedermeier. *Invitation to fixed-parameter algorithms*, volume 31. Oxford University Press, USA, 2006.

[Nieves *et al.*, 2009] Juan Carlos Nieves, Mauricio Osorio, and Claudia Zepeda. Expressing extension-based semantics based on stratified minimal models. In Hiroakira Ono, Makoto Kanazawa, and Ruy J. G. B. de Queiroz, editors, *Logic, Language, Information and Computation, 16th International Workshop, WoLLIC 2009, Tokyo, Japan, June 21-24, 2009. Proceedings*, volume 5514 of *Lecture Notes in Computer Science*, pages 305–319. Springer, 2009.

[Nofal *et al.*, 2014] Samer Nofal, Katie Atkinson, and Paul E. Dunne. Algorithms for decision problems in argument systems under preferred semantics. *Artif. Intell.*, 207:23–51, 2014.

[Ordyniak and Szeider, 2011] Sebastian Ordyniak and Stefan Szeider. Augmenting tractable fragments of abstract argumentation. In Toby Walsh, editor, *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 1033–1038. IJCAI/AAAI, 2011.

[Papadimitriou, 1994] Christos H. Papadimitriou. *Computational Complexity.* Addison-Wesley, Reading, Massachusetts, 1994.

[Reiter, 1980] Raymond Reiter. A logic for default reasoning. *Artif. Intell.*, 13(1-2):81–132, 1980.

[Robertson *et al.*, 1999] Neil Robertson, P. D. Seymour, and Robin Thomas. Permanents, Pfaffian orientations, and even directed circuits. *Ann. of Math. (2)*, 150(3):929–975, 1999.

[Snaith and Reed, 2016] Mark Snaith and Chris Reed. Argument revision. *J. Log. Comput.*, in press (available online), 2016.

[Strass and Wallner, 2014] Hannes Strass and Johannes P. Wallner. Analyzing the Computational Complexity of Abstract Dialectical Frameworks via Approximation Fixpoint Theory. In Chitta Baral, Giuseppe De Giacomo, and Thomas Eiter, editors, *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning, KR 2014*, pages 101–110. AAAI Press, 2014.

[Strass and Wallner, 2015] Hannes Strass and Johannes Peter Wallner. Analyzing the computational complexity of abstract dialectical frameworks via approximation fixpoint theory. *Artif. Intell.*, 226:34–74, 2015.

[Strass, 2013] Hannes Strass. Instantiating knowledge bases in abstract dialectical frameworks. In João Leite, Tran Cao Son, Paolo Torroni, Leon van der Torre, and Stefan Woltran, editors, *Computational Logic in Multi-Agent Systems - 14th International Workshop, CLIMA XIV, Corunna, Spain, September 16-18, 2013. Proceedings*, volume 8143 of *Lecture Notes in Computer Science*, pages 86–101. Springer, 2013.

[Thimm and Villata, 2015] Matthias Thimm and Serena Villata. System descriptions of the first international competition on computational models of argumentation (iccma'15). *CoRR*, abs/1510.05373, 2015.

[Thimm *et al.*, 2016] Matthias Thimm, Serena Villata, Federico Cerutti, Nir Oren, Hannes Strass, and Mauro Vallati. Summary report of the first international competition on computational models of argumentation. *AI Magazine*, 37(1):102, 2016.

[Toni, 2014] Francesca Toni. A tutorial on assumption-based argumentation. *Argument & Computation*, 5(1):89–117, 2014.

[Verheij, 1996] Bart Verheij. Two approaches to dialectical argumentation: admissible sets and argumentation stages. In J. Meyer and L. van der Gaag, editors, *Proceedings of the 8th Dutch Conference on Artificial Intelligence (NAIC'96)*, pages 357–368, 1996.

[Wallner *et al.*, 2016] Johannes Peter Wallner, Andreas Niskanen, and Matti Järvisalo. Complexity results and algorithms for extension enforcement in abstract argumentation. In Dale Schuurmans and Michael P. Wellman, editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 1088–1094. AAAI Press, 2016.

[Wallner, 2014] Johannes P. Wallner. *Complexity Results and Algorithms for Argumentation - Dung's Frameworks and Beyond.* PhD thesis, Vienna University of Technology, Institute of Information Systems, 2014.

[Wu and Caminada, 2010] Yining Wu and Martin Caminada. A labelling-based justification status of arguments. *Studies in Logic*, 3(4):12–29, 2010.

Wolfgang Dvořák
Institute of Information Systems
TU Wien
Vienna, Austria
Email: dvorak@dbai.tuwien.ac.at

Paul E. Dunne
Department of Computer Science
University of Liverpool
Liverpool, United Kingdom
Email: ped@csc.liv.ac.uk