TU WIEN Informatics

The Computational Cost and Benefit of Collective Attacks in Abstract Argumentation

DISSERTATION

zur Erlangung des akademischen Grades

Doktor der Technischen Wissenschaften

eingereicht von

Dipl.-Ing. Matthias König, BSc
Matrikelnummer 01425517

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Dipl.-Ing. Dr.techn. Stefan Woltran
Zweitbetreuung: Dipl.-Ing. Dipl.-Ing. Dr.techn. Wolfgang Dvořák

Diese Dissertation haben begutachtet:

Jörg Rothe	Simon Parsons

Wien, 11. November 2024

Matthias König

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

TU WIEN Informatics

The Computational Cost and Benefit of Collective Attacks in Abstract Argumentation

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktor der Technischen Wissenschaften

by

Dipl.-Ing. Matthias König, BSc
Registration Number 01425517

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dipl.-Ing. Dr.techn. Stefan Woltran
Second advisor: Dipl.-Ing. Dipl.-Ing. Dr.techn. Wolfgang Dvořák

The dissertation has been reviewed by:

Jörg Rothe

Simon Parsons

Vienna, November 11, 2024

Matthias König

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Declaration of Authorship

Dipl.-Ing. Matthias König, BSc

I hereby declare that I have written this Doctoral Thesis independently, that I have completely specified the utilized sources and resources and that I have definitely marked all parts of the work - including tables, maps and figures - which belong to other works or to the internet, literally or extracted, by referencing the source as borrowed.
I further declare that I have used generative AI tools only as an aid, and that my own intellectual and creative efforts predominate in this work. In the appendix "Overview of Generative AI Tools Used" I have listed all generative AI tools that were used in the creation of this work, and indicated where in the work they were used. If whole passages of text were used without substantial changes, I have indicated the input (prompts) I formulated and the IT application used with its product name and version number/date.

Vienna, November 11, 2024

Matthias König

Acknowledgements

I want to thank the many people who helped me to successfully complete my studies and ultimately craft this doctoral thesis. First I want to thank my advisors, Stefan Woltran and Wolfgang Dvořák, who helped me in all kinds of situations, ranging from long whiteboard sessions and strategic decisions for academic success, to personal growth in many aspects. I value your flexibility and understanding, and especially how you supported me in balancing my work and family life.

Special thanks go to Michael Bernreiter and Anna Rapberger for being the best possible office mates and personal confidants for all matters. Moreover, I am very thankful for my "core" team of co-workers and friends, Giovanni Buraglio, Jan Maly, Oliviero Nardi, and Markus Ulbricht, who not only proved themselves to be reliable and diligent colleagues time and again, but who also made our work very fun and enjoyable. In addition, I am grateful for the great collaboration with all my co-authors I have not mentioned so far, Viktor Besin, Lydia Blümel, Martin Caminada, Yannis Dimopoulos, Markus Hecher, Andre Schidler, Stefan Szeider, and Johannes P. Wallner.

I am very grateful to Juliane Auerböck for helping me in bureaucratic regards, organizing trips and events, and basically running the show. I want to thank Markus Bartel and Matthias Nitzschke for their support in all technical regards.

I also want to thank the reviewers of this thesis, Jörg Rothe and Simon Parsons, for taking the time to give me valuable feedback.

I would not have been able to complete this work without the support of my family and friends, who were always there to celebrate my victories and help me through setbacks. Loving thanks go to my ever-supportive parents Alfred, Gerald, and Susanne, as well as my siblings and their families, Becky, Tobi and Simon, as well as Philipp and Claudia. Likewise, I want to thank Christoph, David, Giulio, Hanna, Julia, Kathi, Marcel, Pamina, Sanja, Tamara, Thomas, and Tim. Your encouragement and the fruitful exchanges helped me to set things into perspective.

Above all, I want to thank my wife, Franzi, for her unlimited support and love, and my dear Caroline for her endless supply of smiles.

Kurzfassung

Sinnvolle Schlussfolgerungen aus möglicherweise widersprüchlichen Informationen zu ziehen ist von zentraler Bedeutung für verlässliche und nachvollziehbare Anwendungen von Künstliche-Intelligenz (KI). Formalismen für derartige Schlussfolgerungen werden in der formalen Argumentationstheorie erforscht; ein solcher Formalismus sind Argumentation Frameworks (AFs). AFs zeichnen sich—belegt durch laufend neue Forschungsergebnisse in den letzten Jahren—durch ihre Vielseitigkeit, einschließlich zahlreicher aufgezeigter Verbindungen zu anderen nicht-monotonen Formalismen, aus. Sie gelten daher ob ihrer einfachen Struktur als naheliegende Wahl, um Probleme in einem argumentativen Setting darzustellen und zu lösen. Die Eleganz dieses Formalismus liegt in der einfachen Struktur der AFs: Argumente werden als abstrakte Knoten eines Graphen modelliert, gerichtete Kanten stellen Attacken zwischen Argumenten dar. Eine weiterführende Idee ist, dass anstatt eines einzelnen Arguments eine Menge von Argumenten den Ursprung einer Attacke bilden—und keine kleinere Teilmenge davon stark genug ist, das Ziel-Argument anzugreifen. Diese Art der Attacke ist in AFs nicht ohne zusätzliche Argumente ohne klare Bedeutung möglich. In SETAFs können derartige "kollektive Attacken" zusätzlich zu den von AFs bekannten Attacken vorkommen; damit verallgemeinern SETAFs die einfachen AFs auf natürliche Weise. SETAFs wurden erstmals 2006 von Nielsen und Parsons formalisiert.

Obwohl gezeigt werden konnte, dass SETAFs eine größere Ausdrucksstärke bieten als AFs, wurde die naheliegende Idee von kollektiven Attacken bisher verhältnismäßig wenig erforscht. Oft muss man bei großer Ausdrucksstärke negative Auswirkungen in anderen Aspekten in Kauf nehmen: etwa eine erhöhte Rechenkomplexität (und daher schlussendlich auch längere Laufzeit) von Algorithmen, den Verlust wichtiger Eigenschaften, oder, dass komplexere algorithmische Ansätze nicht anwendbar sind. In dieser Arbeit werden wir allerdings sehen, dass derartige Befürchtungen im Falle der SETAFs nicht zutreffend sind: bemerkenswerterweise gelten die wichtigsten Eigenschaften von AFs auch für SETAFs, wie wir durch eine sogenannte "principle-based analysis" zeigen. Dieses Werkzeug bedient sich vordefinierter Eigenschaften (der "principles") um anhand ihrer Erfüllung oder Nichterfüllung zu zeigen, welche Merkmale die Argumentationssemantiken aufweisen.

Darüber hinaus zeigen wir, dass auch komplexere algorithmische Ansätze, welche für AFs bereits vielversprechende Ergebnisse zeigen, ebenfalls auf SETAFs anwendbar sind—unter der Voraussetzung dass diese auf clevere Weise an das neue Setting angepasst

werden. Wir untersuchen den backdoor-Ansatz und den treewidth-Ansatz. Eine backdoor ("Hintertür") kann man sich als den Teil eines Problems vorstellen, welcher für den größten Berechnungsaufwand verantwortlich ist; wird dieser Teil entfernt, so ist die Struktur des übrigen Teils simpel. Das nutzen wir um Teillösungen des komplizierten Teils auf den simplen Teil auszuweiten, um dadurch schnellere Laufzeiten zu erzielen. Der Parameter treewidth ("Baumweite") bestimmt wie ähnlich der Argumentationsgraph zu einem Baum ist. Da die von uns untersuchten Probleme auf baumförmigen SETAFs leicht lösbar sind, verspricht dieser Parameter ebenfalls verbesserte Laufzeiten. In der Tat zeigen wir durch auf dynamischer Programmierung basierenden Algorithmen, dass diese Ideen ebenfalls auf SETAFs anwendbar sind, und beweisen auch hier verbesserte Laufzeitschranken. In AFs sind durch ihre einfache Struktur viele interessante Phänomene nicht sichtbar, welche bei SETAFs einen deutlicheren Effekt zeigen. Wir können ebendiese Phänomene oft ausnutzen, um bessere Ergebnisse zu erzielen—und diese sind sogar ebenfalls auf AFs anwendbar. Das kommt daher, dass ein AF als Spezialfall eines SETAF gesehen werden kann, wo jede Attacke von einer genau ein-elementigen Menge ausgeht. Wir beweisen also nicht nur, dass SETAFs mit den genannten strukturellen Eigenschaften effizienter verarbeitet werden können, sondern zeigen auch Verbesserungen für AFs auf.

Abstract

Resolving conflicts and reasoning in an argumentative setting is a discipline that is essential for reliable and explainable artificial intelligence (AI). In the last decades, argumentation frameworks (AFs) by Dung have stood out in the research community as a well-investigated simple formalism for these tasks. The elegant structure of arguments as nodes and attacks as edges of a directed graph not only served well for modeling argumentation scenarios, but proved quite versatile with many connections to other non-monotonic reasoning formalisms. However, the seemingly basic notion of collective attacks, i.e., the scenario where a set of arguments rather than a single one is needed to defeat another, cannot natively be modeled in AFs without additional arguments. Hence, Nielsen and Parsons introduced a conservative generalization of AFs in 2006 incorporating this basic feature—the resulting frameworks are referred to as SETAFs.

While it was shown that SETAFs are strictly more expressive than AFs, comparatively little research was performed on them. Oftentimes increased expressive power goes hand in hand with undesirable side effects—like high computational cost, the loss of essential properties, or the impossibility of applying popular algorithmic ideas to reason even more efficiently on non-random data. We will see in this thesis that these worries are mostly groundless in the context of SETAFs: remarkably, even facing all apparent advantages of SETAFs over AFs, most desirable properties generalize to SETAFs—as captured by a *principle-based analysis*. In this prominent approach, the semantics of the frameworks are investigated and categorized by the satisfaction or violation of principles—i.e., properties that succinctly capture key features and behaviors in common argumentative scenarios.

Moreover, we show that when carefully generalized, advanced algorithmic ideas that are promising on AFs in general also work for SETAFs. In particular, we investigate the *backdoor*- and *treewidth*-approaches. In a nutshell, a backdoor is a part of a SETAF, such that its removal renders the remaining framework structurally simple. We apply this idea to the hypergraph structure of SETAFs by means of efficient propagation algorithms. The parameter treewidth describes how tree-like a graph is. Since many problems become efficiently solvable in tree-shaped SETAFs, we exploit a tree-like structure via dynamic programming algorithms. Interesting phenomena that can be exploited in several ways are hidden or trivialized by the simple structure of AFs—we utilize these phenomena for computational improvements. By means of our thorough analysis of the structure of SETAFs we not only deepen our understanding of collective attacks and their effects,

but also shine a new light on AFs. This is because AFs can be seen as a special case of SETAFs, where for each attack the attacking set is of size 1. Our close look pays off measurably: our algorithms for backdoor- and treewidth-techniques not only establish for the first time that SETAFs can be efficiently treated in the aforementioned scenarios, but also show improvements in the AF case.

Contents

Kurzfassung **ix**

Abstract **xi**

Contents **xiii**

1 Introduction **1**
1.1 Motivation . . . 1
1.2 State of the Art & Related Work . . . 2
1.3 Contributions and Overview . . . 3
1.4 Publications . . . 5

2 Background **9**
2.1 Abstract Argumentation and Collective Attacks . . . 9
2.2 Complexity of Reasoning in Abstract Argumentation . . . 15

3 Principle-Based Analysis **21**
3.1 Basic Principles . . . 22
3.2 Reduct and Modularization . . . 32
3.3 Directionality and Non-Interference . . . 38
3.4 SCC-Recursiveness . . . 42
3.5 Incremental Computation . . . 61
3.6 Discussion . . . 68

4 Backdoor-Based Evaluation **71**
4.1 Towards SETAF Backdoors . . . 73
4.2 Backdoor Evaluation . . . 75
4.3 Conditional Lower Bounds for Backdoor Evaluation . . . 89
4.4 Discussion . . . 90

5 Treewidth-Based Evaluation **93**
5.1 Towards SETAF Treewidth . . . 95
5.2 Dynamic Programming on SETAFs . . . 100
5.3 Characterizing Stable Extensions . . . 102

5.4 Characterizing Admissible Sets 119
5.5 Characterizing Complete Extensions 137
5.6 Discussion 152

6 Conclusion **153**
6.1 Summary & Insights 154
6.2 Future Work & Outlook 157

Overview of Generative AI Tools Used **159**

List of Figures **161**

List of Tables **163**

Bibliography **165**

CHAPTER 1

Introduction

Scientific research in artificial intelligence (AI) has shed a light on the pressing need for reliable, explainable, and furthermost correct decision making [RS09, LAD+24]. At the same time, we expect the systems we use everyday to be *fast*—we often expect real-time results. This is especially the case in sensitive areas like deciding on financial issues, making diagnoses in a medical setting, or when a seemingly overwhelming flood of information and fake news clouds the judgment that is the foundation of a democratic society. Hence, we want to reason both reliably correct and fast. While sub-symbolic AI systems like neuronal networks that form the basis e.g. for modern large language models at its current stage struggle with the former property, symbolic AI-approaches based on logic oftentimes provide disappointing runtimes. This problem of slow computation is largely due to the large solution space that has to be exhaustively explored to guarantee correct outputs. Also in the field of formal argumentation—which is a branch of symbolic AI that is concerned with reasoning with conflicting arguments—one has to deal with in general intractable problems. To mitigate this issue, in this thesis, we discuss argumentation frameworks with collective attacks (SETAFs), classify them via a principle-based approach, and ultimately use the gained insights to provide fast reasoning algorithms in this setting. In the following, we discuss in more detail the motivation for this work (Section 1.1), the current state-of-the-art which we build our results upon as well as related work (Section 1.2), the contributions of this thesis in broad strokes (Section 1.3), and finally list the publications that comprise this thesis, as well as other related publications of the author (Section 1.4).

1.1 Motivation

To make educated and deliberate decisions, while being aware of possibly conflicting viewpoints and possible counter-arguments to our own standpoint, in a systematic manner, the field of *formal argumentation* emerged and has since Dung's seminal paper in 1995 [Dun95]

become an active research area within the field of AI [RS09, BGGvdT18, GGST21]. Dung's *Abstract argumentation* is concerned with what can be seen as the *last* step of an argumentation pipeline: we face arguments as abstract entities, and draw conclusions based only on their relation to each other, while ignoring their internal structure. We focus on the conflicts between arguments; this is formally captured by *abstract argumentation frameworks* (AFs), as introduced by Dung. An AF can be seen as a directed graph, with the *arguments* as its nodes, and *attacks* as its directed edges. A coherent world-view, called an *extension*, is then a set of arguments that follows certain requirements (given by a *semantics*). Despite the elegance of this light-weigh formalism, AFs have been proven to be a versatile alternative in the world of non-monotonic reasoning with crosslinks to many other formalisms, such as logic programming (stable and well-founded semantics) and n-person-games [Dun95], defeasible logic [GMAB04], or logic programming (3-valued stable model semantics) [WCG09]. The simplicity of AFs however comes at a cost: many real-world scenarios require additional modeling effort in AFs including the addition of artificial, technical arguments without an intuitive real-world meaning. For this reason, many generalizations of AFs have been proposed, accounting for e.g. argument support [CGGS15], recursion (e.g., attacks *on* attacks) [CGGS15], weights [DHM$^+$11], preferences [Mod09], intricate acceptance conditions [BW10], incomplete information [BJN$^+$21], or time and probabilities [BDST23]. While these generalizations extend the syntax of AFs to account for their respective domains, this oftentimes comes at the cost of additional computational effort or additional syntactic entities. In this thesis, we study argumentation frameworks with *collective attacks* (referred to as SETAFs due to their "set"-attacks) [NP06b, BCD$^+$21]. SETAFs generalize Dung-style AFs in the sense that some arguments can only be effectively defeated by a collection of attackers, yielding a natural representation as a directed hypergraph. Hence, SETAFs only slightly extend the syntax of AFs with no entirely new syntactic entities while at the same time providing most advantages of AFs (as we will illustrate in this thesis), as well as extended modeling power. While many fundamental properties for SETAFs have already been shown by Nielsen and Parsons as they introduced SETAFs [NP06b], a thorough analysis of SETAFs with a focus on advanced algorithmic ideas has not yet been performed—in this thesis we close this gap.

1.2 State of the Art & Related Work

In his seminal paper from 1995, Dung introduced Abstract Argumentation Frameworks (AFs) as a simple approach to model various argumentative scenarios [Dun95]. In 2006, Nielsen and Parsons introduced SETAFs to account for collective attacks [NP06b]. They showed that desirable properties like the fundamental lemma of Dung's AFs carry over to SETAFs (cf. Lemma 2.7), as well as many other key semantic properties. Moreover, it has been shown that the addition of collective attacks offers strictly more expressive power [DFW19] while retaining the advantageous computational properties of AFs [DGW18]. Subsequently, labelings have been applied to SETAFs as an alternative way to extensions to characterize semantics, as well as a generalization of additional

popular argumentation semantics [FB19]. Furthermore, translations from SETAFs to AFs have been investigated [Pol17, FB19] (translating AFs to SETAFs is trivial, as AFs can be seen as a special case of SETAFs).

SETAFs have proven useful to model various sorts of situations. Recent investigations have shown that collective attacks are especially well suited as a target formalisms for instantiations from structured argumentation [KRU22, CKRU24, DDK+24, BDKU24, BKU24]. This is because a SETAF instantiation in general yields fewer arguments which can easily be mapped back to the original assumptions. Furthermore, collective attacks have been considered in the context of "higher level attacks" (i.e., attacks are possible onto arguments *or* attacks of any "order") [Gab09], evidence-based reasoning in connection with collective support [ON08], or in the context of forming coalitions (inspired by political models) [AS17]. As a target formalism for modeling various scenarios, SETAFs have e.g. been used in the context of preferences [BB20], inconsistent Datalog knowledge bases [YVC20], and choice logics [BK23].

The main goal of this thesis is to analyze the structure of SETAFs in order to obtain efficient methods for solving reasoning problems. It is known for AFs that computational advantages can be obtained in several ways, such as via the backdoor-approach [DOS12], the treewidth-approach [DPW12], or following the strongly-connected-components (SCCs) of the graph structure [Bau11, LJK11, BGL14, CGVZ14]. The backdoor-approach has also recently been applied in the context of structured argumentation [AU24]. SCC-recursiveness has also been considered for Abstract Dialectical Frameworks (ADFs) [GRS21]. Support-free ADFs can be modeled as SETAFs [DKW23], the relation between our approach towards SCC-recursiveness and the respective results for ADFs is discussed in Chapter 3.

Despite the apparent advantages of SETAFs over AFs, a thorough investigation of the computational cost and benefit of collective attacks is not yet available. In this thesis, we will close this gap by investigating properties that are shared by AFs and SETAFs, and discover intricate differences that shed a light also on implicit properties of AFs that are hidden by their simplistic structure. The focus of this work lies on the computational properties of SETAFs, where we establish that most of the "shortcuts" one can take for AFs generalize to SETAFs—given that one knows the important differentiating details, which we will discuss in the following.

1.3 Contributions and Overview

We are interested in *computational* "shortcuts" for SETAFs, i.e., ways to avoid enumerating the exponentially large search space for reasoning problems. We achieve this by exploiting properties of typical data, which usually is not entirely random but rather admits some sort of *structure* [DBC01, CMDM05, Dun07]. While in arbitrary data the exponential worst-case runtime of the best known approaches for solving NP-hard problems cannot be avoided—even in state-of-the-art systems, problems on structured data can oftentimes be solved much faster. To this end, we identify *parameters* that characterize

the structure and quantify the achievable speedup utilizing these parameter values. This can be seen as a response to Paul Dunne's plea at the COMMA 2022 conference, where he presented his "personal view of complexity in argumentation after 20 years" [Dun22]. In a nutshell, Dunne finds that proving intractability for argumentation problems should not be the *end* of the investigation of their computational complexity, but the *start* of finding and characterizing those cases where we can indeed reason efficiently.

We start with the analysis of *principles* for SETAF semantics. The principle-guided approach is the "traditional" way in the argumentation community of comparing the features and shortcomings of different semantics in order to guide the choice of the right semantics for any given purpose [vdTV17]. In Chapter 3 we generalize known AF-principles to be applicable in the context of collective attacks, and analyze which ones still hold. We will show that with the notable exception of *tightness*[1] all established principles hold—which we can count as a major *benefit* of SETAFs, as one would expect the richer syntax (compared to AFs) to come at the cost of losing many desirable properties. In the context of principles we transition into the main focus of this thesis, namely computational speedup via structural exploits. We first analyze the principles *directionality* and *SCC-recursiveness* for SETAFs, which ultimately yield fast reasoning algorithms based on incremental computation of extensions. We generalize the concepts to the rich structure of SETAFs and establish techniques to perform the computation along strongly connected components (SCCs) of the directed hypergraph. Finally, we enrich this approach by showing how traditional graph-properties of directed graphs can help to obtain an even faster theoretical runtime bounds.

Chapter 4 continues the investigation of advantageous structure for SETAF reasoning; we provide a thorough analysis of *backdoors*. The *backdoor*-concept is used in different contexts such as constraint satisfaction problems (CSP), satisfiability checking (SAT), answer set programming (ASP) (see e.g. [GMO$^+$14, FS15, GOS17, OSS21] for recent work), and has also been investigated in the context of AFs [DOS12]. Intuitively, a backdoor is one part of an instance that "makes it hard to solve". We focus on *deletion-backdoors*, i.e. if this backdoor is removed from the instance, the remaining (sub-) instance is computationally easy. In the context of formal argumentation this means that a framework belongs to a tractable class (such as acyclicity) after removing certain arguments. Hence, we can utilize the tractability results of these easy classes for general frameworks without restrictions, as arbitrary distances to the tractable fragments are allowed. If this distance is small (constant), we can reason in polynomial time [DOS12], as the exponential factor is only in the size of the backdoor. We show that the backdoor-approach for AFs generalizes to SETAFs, and provide algorithms for fast reasoning. In fact, we show that, even though our approach is tailored to SETAFs with their rich

[1] Intuitively, tightness (cf. Principle 3.27) means that whenever an argument a is not acceptable in addition to an extension E, there has to be an argument $b \in E$ which is "responsible" for this non-acceptability—in that a and b are never jointly accepted. While some AF semantics satisfy tightness, due to their increased expressive power the only SETAF semantics under our consideration satisfying tightness are the ones admitting only a single extension.

structure, our algorithms improve on the runtime of existing approaches in the special case of AFs.

In Chapter 5 we analyze the parameter *treewidth*. A low *treewidth* indicates a certain "tree-likeness" of a graph, and as problems often become easy on trees, adapted versions of these easy algorithms can often be applied to instances with low treewidth. In AFs, it has been shown that reasoning is indeed fixed-parameter tractable w.r.t. treewidth [DPW12, DSW12]. Also in the field of structured argumentation, the parameter treewidth has recently been investigated w.r.t. assumption-based argumentation by Popescu and Wallner [PW23], who show fixed-parameter tractability for several reasoning problems via monadic second-order logic and tailored algorithms, similarly to the work we present in this thesis. We investigate how this notion of treewidth is applicable to the directed hypergraph-structure of SETAFs and show that certain generalizations admit parameterized-tractable algorithms (FPT), while other natural attempts do not.

In a nutshell, we show that SETAFs admit almost all advantages of AFs, whereas the few exceptions prove interesting behavior (for example, the non-satisfaction of the *tightness* principle is due to the improved expressiveness of SETAFs over AFs). At the same time we introduce algorithms to efficiently reason on SETAFs. By providing algorithms with runtimes that improve even over known AF approaches with their simpler structure we show that it pays off to dive deep into the idea of collective attacks, since due to the deeper understanding of the underlying concepts by investigating the rich structure of SETAFs improvements over the state-of-the-art can be achieved. This can be interpreted as an invitation to other researchers to consider collective attacks as a modeling formalism when it becomes apparent that the AF syntax is too restrictive, but one does not want to add too much additional formal machinery.

1.4 Publications

This thesis is based on the following publications:

Chapter 3:

- Wolfgang Dvořák, Matthias König, Markus Ulbricht, and Stefan Woltran. A reduct-driven study of argumentation frameworks with collective attacks. In *Proceedings of the 19th International Workshop on Non-Monotonic Reasoning, NMR 2021*, pages 285–294, 2021.
- Wolfgang Dvořák, Matthias König, and Stefan Woltran. Graph-classes of argumentation frameworks with collective attacks. In *Proceedings of the 17th European Conference on Logics in Artificial Intelligence, JELIA 2021*, pages 3–17, 2021.
- Wolfgang Dvořák, Matthias König, Markus Ulbricht, and Stefan Woltran. Rediscovering argumentation principles utilizing collective attacks. In *Proceedings of*

the 19th International Conference on Principles of Knowledge Representation and Reasoning, KR 2022, pages 122–131, 2022.

- Wolfgang Dvořák, Matthias König, Markus Ulbricht, and Stefan Woltran. Principles and their computational consequences for argumentation frameworks with collective attacks. *J. Artif. Intell. Res.*, 79:69–136, 2024.

Chapter 4:

- Wolfgang Dvořák, Matthias König, and Stefan Woltran. Deletion-backdoors for argumentation frameworks with collective attacks. In *Proceedings of the 4th International Workshop on Systems and Algorithms for Formal Argumentation, SAFA 2022*, pages 98–110, 2022.
- Wolfgang Dvořák, Matthias König, and Stefan Woltran. Parameterized Complexity of Abstract Argumentation with Collective Attacks. *Under review at Argument & Computation*

Chapter 5:

- Wolfgang Dvořák, Matthias König, and Stefan Woltran. Treewidth for argumentation frameworks with collective attacks. In *Proceedings the 9th International Conference on Computational Models of Argument, COMMA 2022*, pages 140–151, 2022.
- Wolfgang Dvořák, Matthias König, and Stefan Woltran. Parameterized Complexity of Abstract Argumentation with Collective Attacks. *Under review at Argument & Computation*

Additional Publications of the Author:

- Wolfgang Dvořák, Matthias König, and Stefan Woltran. On the complexity of preferred semantics in argumentation frameworks with bounded cycle length. In *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021*, pages 671–675, 2021.
- Wolfgang Dvořák, Matthias König, Johannes P. Wallner, Stefan Woltran. ASPARTIX-V21. In *Fourth International Competition on Computational Models of Argumentation, ICCMA 2021*, 2021
- Wolfgang Dvořák, Markus Hecher, Matthias König, André Schidler, Stefan Szeider, and Stefan Woltran. Tractable abstract argumentation via backdoor-treewidth. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence, AAAI 2022*, pages 5608–5615, 2022.

- Matthias König, Anna Rapberger, and Markus Ulbricht. Just a matter of perspective – Intertranslating Expressive Argumentation Formalisms. In *Proceedings the 9th International Conference on Computational Models of Argument, COMMA 2022*, pages 212–223, 2022.
- Michael Bernreiter and Matthias König. From qualitative choice logic to abstract argumentation. In *Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning, KR 2023*, pages 737–741, 2023.
- Yannis Dimopoulos, Wolfgang Dvořák, Matthias König, Anna Rapberger, Markus Ulbricht, and Stefan Woltran. Sets attacking sets in abstract argumentation. In *Proceedings of the 21st International Workshop on Non-Monotonic Reasoning, NMR 2023*, pages 22–31, 2023.
- Yannis Dimopoulos, Wolfgang Dvořák, Matthias König, Anna Rapberger, Markus Ulbricht, and Stefan Woltran. Redefining ABA+ semantics via abstract set-to-set attacks. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence, AAAI 2024*, pages 10493–10500, 2024.
- Yannis Dimopoulos, Wolfgang Dvořák, Matthias König. Connecting Abstract Argumentation and Boolean Networks. In *Proceedings the 10th International Conference on Computational Models of Argument, COMMA 2024*, pages 85-96, 2024.
- Giovanni Buraglio, Wolfgang Dvořák, Matthias König, Markus Ulbricht. Justifying Argument Acceptance with Collective Attacks: Discussions and Disputes. In *Proceedings of the 33rd International Joint Conference on Artificial Intelligence, IJCAI 2024*, pages 3281–3288, 2024.
- Martin Caminada, Matthias König, Anna Rapberger, and Markus Ulbricht. Attack semantics and collective attacks revisited. *Argument and Computation*, 2024. Pre-press.
- Lydia Blümel, Matthias König, and Markus Ulbricht. Weak admissibility for ABA via abstract set attacks. In *Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning, KR 2024*, pages 178–188, 2024.
- Giovanni Buraglio, Wolfgang Dvořák, Matthias König, Stefan Woltran. Splitting Argumentation Frameworks with Collective Attacks. In *Proceedings of the 5th International Workshop on Systems and Algorithms for Formal Argumentation, SAFA 2024*, pages 41–55, 2024.

CHAPTER 2

Background

In this chapter, we recall the definitions of abstract argumentation frameworks with collective attacks (argumentation frameworks with set-attacks, a.k.a. SETAFs) which serve as the basis for the rest of this thesis. These frameworks due to Nielsen and Parsons [NP06b] are a generalization of Dung's popular abstract argumentation frameworks (AFs) [Dun95]. Both formalisms are based on a set of *arguments*, which in the corresponding graphical representation comprise the nodes of a (hyper-)graph. Directed conflicts between these arguments form the *attacks* of the framework: while in AFs, attacks are from a single argument to another argument, a SETAF attack stems from a non-empty *set* of arguments, and again targets a single argument[2]. The relevant formal definitions for the syntax and semantics of SETAFs are listed in Section 2.1.

Moreover, we give a brief overview of the computational complexity of reasoning and verification problems for SETAFs in Section 2.2. Finally, we also list the known *tractable fragments* for SETAFs, i.e., graph classes which allow for efficient reasoning. In the later chapters we will utilize these results for refined methods for efficient reasoning, be it with incremental computation where parts of the framework belong to a graph class (Chapter 3), backdoors where the removal of a set of arguments renders the remaining framework in a graph class (Chapter 4), or treewidth where we exploit the structural advantages of a tree's acyclicity in a decomposition-based setting (Chapter 5).

2.1 Abstract Argumentation and Collective Attacks

We recall the definitions of SETAFs and their semantics [NP06b], see, e.g. [BCD+21] for an overview. Throughout the thesis, we assume a countably infinite domain $\mathfrak{A}$ of possible arguments.

[2] Also attacks *towards* sets of arguments have been considered [Ver96, Boc03, DDK+23, DDK+24].

Definition 2.1. *A SETAF is a pair* $SF = (A, R)$ *where* $A \subseteq \mathfrak{A}$ *is a finite*[3] *set of* arguments*, and* $R \subseteq (2^A \setminus \{\emptyset\}) \times A$ *is the* attack relation*. For an attack* $(T, h) \in R$ *we call* T *the* tail *and* h *the* head *of the attack.*

If the tail T of an attack (T, h) is a single argument, we usually write (t, h) to denote the set-attack $(\{t\}, h)$. The class of SETAFs where all attacks are of this form amounts to (standard Dung) AFs. Hence, the structure of an AF is a directed graph, while the structure of a SETAF is a *directed hypergraph*.

If the tail T of an attack (T, h) also contains its head h (i.e., $h \in T$), we call (T, h) a "self-attack". This generalizes the special case of self-attacks in AFs, which due to the syntactic restrictions only allow for self-attacks of the form (t, t).

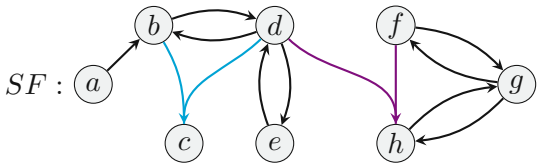
Note that attacks can only stem from a *non-empty* tail. This is in line with the original definition due to Nielsen and Parsons [NP06b]. However, it was shown in [CKRU24] that one can allow for attacks from the empty set while faithfully generalizing the semantics without changing the key properties of the well-known semantics. Intuitively, an attack $(\emptyset, a)$ can be dealt with in a "pre-processing" step, where the argument a as well as all outgoing attacks (T, h) with $a \in T$ are removed from the SETAF. This is due to the fact that a is automatically defeated, and each argument h that is the head of an outgoing attack of a is defended against this attack. One then obtains a SETAF in accordance with Definition 2.1 which admits the same extensions.

Given a SETAF $SF = (A, R)$ and $S, S' \subseteq A$, we write $S \mapsto_R a$ if there is a set $T \subseteq S$ with $(T, a) \in R$. Furthermore, we write $S' \mapsto_R S$ if $S' \mapsto_R a$ for some $a \in S$. For $S \subseteq A$, we use S_R^+ to denote the set $\{a \in A \mid S \mapsto_R a\}$ and define the *range* of S (w.r.t. R), denoted $S_R^\oplus$, as the set $S \cup S_R^+$. Moreover, we use $A(SF)$ and $R(SF)$ to identify its arguments A and its attack relation R, respectively.

Example 2.2. *Consider the following SETAF* $SF = (A, R)$ *with arguments* $A = \{a, b, c, d, e, f, g, h\}$ *and the attack relation*

$$R = \{(a, b), (\{b, d\}, c), (b, d), (d, b), (d, e), (e, d),$$
$$(\{d, f\}, h), (f, g), (g, f), (g, h), (h, g)\};$$

the collective attacks $(\{b, d\}, c), (\{d, f\}, h)$ *are highlighted.*



[3] While in theory it is not unreasonable to consider infinite frameworks (see e.g. [Dun24] for recent work on *infinite* AFs), since our focus lies on computational properties we only consider finite sets of arguments (as otherwise the complexity classes we consider trivialize or become inapplicable.)

For example, the arguments $\{d, f\}$ *are only effectively attacking* h *(through the violet attack) if* both d *and* f *are accepted.* $\{d, f\}$ *is the tail of the attack* $(\{d, f\}, h)$ *and* h *is its head. For the set* $S = \{d, f\}$ *we have* $S_R^+ = \{b, e, g, h\}$ *and* $S_R^{\oplus} = \{b, d, e, f, g, h\}$. *Even though* d *is part of an attack* $(\{b, d\}, c)$ *towards* c *we have* $c \notin S_R^+$, *since* $b \notin S$.

While Definition 2.1 covers the *syntax* of SETAFs as an extension of Dung AFs, in the following we discuss the *semantics* of SETAFs and the meaning of "set-attacks", i.e., attacks with sets of attacking arguments. Assume three arguments a, b, and c, where the acceptance of a as well as b counter the acceptance of c. Note that there is some ambiguity in this statement: (1) is the acceptance of *one of* a and b sufficient to counter the acceptance of c or (2) do we have to accept *both* a and b to counter c? Intuitively, in an AF with its 1-to-1 attack structure we can consider the attacks (a, c) and (b, c) as a starting point, which turns out to capture case (1). Furthermore, in order to accept c both a and b have to be countered. However, case (2) cannot be expressed with these three arguments and AF-attacks alone. SETAFs on the other hand offer the possibility to model both cases, as Example 2.3 illustrates.

Example 2.3. *Left we have an AF (and, hence, also a SETAF) with two attacks* $R_1 = \{(a, c), (b, c)\}$. *On the right we have a SETAF with one collective attack* $R_2 = \{(\{a, b\}, c)\}$.

In the left framework, accepting one of a *or* b *renders* c *defeated, as* $\{a\}_{R_1}^+ = \{b\}_{R_1}^+ = \{c\}$. *Conversely, to accept* c both a *and* b *have to be countered. In the framework on the right however, neither* a *nor* b *on their own attack* c, *as* $\{a\}_{R_2}^+ = \{b\}_{R_2}^+ = \emptyset$. *Accepting both* a *and* b *effectively attacks* c: $\{a, b\}_{R_2}^+ = \{c\}$. *In order to accept* c, *it suffices to counter* a *or* b *(or both). In the following, we formally recall these concepts of* conflicts *and* defense.

The well-known notions of conflict and defense from classical Dung-style AFs naturally generalize to SETAFs. Note that in a SETAF, a set of arguments S is conflicting only if for some attack (T, h) both the full tail T and the head h are in S, i.e., $T \cup \{h\} \subseteq S$. Consequently, this means in particular that it is possible to have $T' \cup \{h\}$ in a conflict-free set, where $T' \subset T$ (this concept of "partial conflicts" will be investigated in more detail in Chapter 3).

Definition 2.4. *Given a SETAF* $SF = (A, R)$, *a set* $S \subseteq A$ *is* conflicting *in* SF *if* $S \mapsto_R a$ *for some* $a \in S$. *A set* $S \subseteq A$ *is* conflict-free *in* SF, *if* S *is not conflicting in* SF, *i.e. if* $T \cup \{h\} \not\subseteq S$ *for each* $(T, h) \in R$. $cf(SF)$ *denotes the set of all conflict-free sets in* SF.

Defense is a central concept in abstract argumentation: the idea of countering every incoming attack. For SETAFs, to be defended against an incoming attack (T, h) it suffices to counter-attack a single argument $t \in T$—as this means that the set T is attacked.

Definition 2.5. *Given a SETAF $SF = (A, R)$, an argument $a \in A$ is* defended *(in SF) by a set $S \subseteq A$ if for each $B \subseteq A$, such that $B \mapsto_R a$, also $S \mapsto_R B$. A set $T \subseteq A$ is defended (in SF) by S if each $a \in T$ is defended by S (in SF).*

Moreover, we make use of the *characteristic function* Γ_{SF} of a SETAF $SF = (A, R)$, defined as $\Gamma_{SF}(S) = \{a \in A \mid S \text{ defends } a \text{ in } SF\}$ for $S \subseteq A$.

The semantics we study in this work are admissible, complete, grounded, preferred, stable, naive, stage, semi-stable, ideal, and eager semantics, which we will abbreviate by *adm*, *com*, *grd* *pref*, *stb*, *naive*, *stage*, *sem*, *ideal*, and *eager* respectively (see e.g. [BCD+21]). We denote the set of semantics under our consideration by Σ.

Definition 2.6. *Given a SETAF $SF = (A, R)$ and a conflict-free set $S \in cf(SF)$. Then,*

- $S \in adm(SF)$, *if S defends itself in SF,*
- $S \in com(SF)$, *if $S \in adm(SF)$ and $a \in S$ for all $a \in A$ defended by S,*
- $S \in grd(SF)$, *if $S = \bigcap_{T \in com(SF)} T$,*
- $S \in pref(SF)$, *if $S \in adm(SF)$ and $\nexists T \in adm(SF)$ s.t. $T \supset S$,*
- $S \in stb(SF)$, *if $S \mapsto_R a$ for all $a \in A \setminus S$,*
- $S \in naive(SF)$, *if $\nexists T \in cf(SF)$ with $T \supset S$,*
- $S \in stage(SF)$, *if $\nexists T \in cf(SF)$ with $T_R^{\oplus} \supset S_R^{\oplus}$,*
- $S \in sem(SF)$, *if $S \in adm(SF)$ and $\nexists T \in adm(SF)$ s.t. $T_R^{\oplus} \supset S_R^{\oplus}$,*
- $S \in ideal(SF)$, *if $S \in com(SF)$, $S \subseteq \bigcap_{E \in pref(SF)} E$ and $\nexists T \in com(SF)$ s.t. $T \subseteq \bigcap_{E \in pref(SF)} E$ and $T \supset S$, and*
- $S \in eager(SF)$, *if $S \in com(SF)$, $S \subseteq \bigcap_{E \in sem(SF)} E$ and $\nexists T \in com(SF)$ s.t. $T \subseteq \bigcap_{E \in sem(SF)} E$ and $T \supset S$.*

The relationship between the semantics has been clarified in [NP06b, DGW18, FB19] and matches with the relations between the semantics for AFs, i.e., for any SETAF SF:

$$stb(SF) \subseteq sem(SF) \subseteq pref(SF) \subseteq com(SF) \subseteq adm(SF)$$
$$stb(SF) \subseteq stage(SF) \subseteq naive(SF) \subseteq cf(SF)$$

It has been shown by Nielsen and Parsons that Dung's Fundamental Lemma [Dun95] also holds for SETAFs [NP06b]: intuitively, this means for an admissible set S and two

arguments a, b which are defended by S that adding either argument again yields an admissible set. Moreover, the other argument will again be defended by the resulting set. This "compatibility" result forms the base for many more advanced properties of AFs and SETAFs.

Lemma 2.7 (Fundamental Lemma [NP06b])**.** *Let $SF = (A, R)$ be a SETAF and $S \subseteq A$ be admissible in SF. Let $a, b \in A$ be two arguments that are defended by S in SF, then*

1. *$S' = S \cup \{a\}$ is admissible in SF, and*
2. *b is defended by S'.*

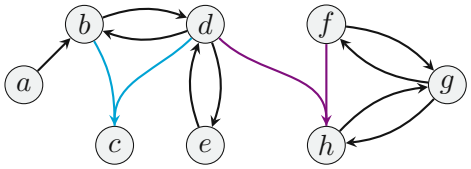
Next, we introduce the notion of the *projection*, which we will revisit and redefine in Chapter 3. The idea is to focus only on a part of the framework, indicated by a set of arguments S. Since the directed hypergraph-structure of SETAFs allows for the case where the head and part of the tail are inside the set S while a non-empty part of the tail is outside S, a special modification is necessary. In this case, we straightforwardly "split" the attack such that for an attack (T, h) in the projection w.r.t. S the remaining attack is $(T \cap S, h)$. Note that the projection is purely syntactical and should not be confused with the *reduct* that we will introduce in Chapter 3.

Definition 2.8. *Let $SF = (A, R)$ be a SETAF and $S \subseteq A$. We define the* projection *$SF{\downarrow_S}$ of SF on S as*

$$SF{\downarrow_S} = (S, \{(T', h) \mid (T, h) \in R, h \in S, T' = T \cap S, T' \neq \emptyset\}).$$

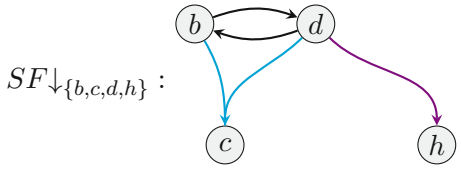
We illustrate the semantics and the concept of projection in the following example.

Example 2.9. *Consider again the SETAF SF from Example 2.2 (left) and its extensions w.r.t. some semantics $\sigma \in \Sigma$ (right).*



$$\begin{aligned} com(SF) &= \{\{a,c\},\{a,c,e\},\{a,c,d\},\{a,c,g\},\\ &\quad\ \{a,c,e,g\},\{a,c,d,g\},\{a,c,f\},\\ &\quad\ \{a,c,d,f\},\{a,c,e,f,h\}\}\\ pref(SF) &= \{\{a,c,e,g\},\{a,c,d,g\},\{a,c,d,f\},\\ &\quad\ \{a,c,e,f,h\}\}\\ grd(SF) &= \{\{a,c\}\} \end{aligned}$$

Intuitively, the projection "hides" parts of the SETAF while we only concentrate on some remaining arguments. Note however, that the extensions do not in general carry over from the "full" SETAF to its part. We project SF to the arguments $\{b, c, d, h\}$ (left) and see that the extensions are incomparable to the original framework.



$com(SF) = \{\emptyset, \{b, c, h\}, \{c, d\}\}$
$pref(SF) = \{\{b, c, h\}, \{c, d\}\}$
$grd(SF) = \{\emptyset\}$

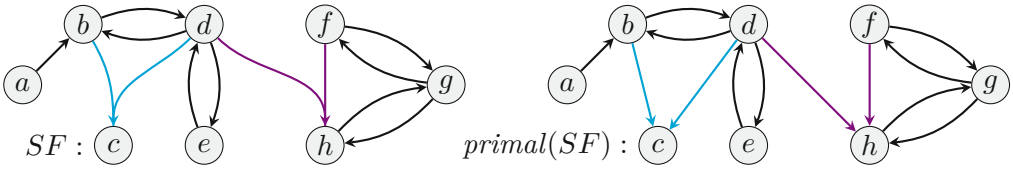
After the projection, the argument b becomes acceptable, and c is no longer in every complete extension. Among others, these issues are formally captured and ultimately fixed in different ways in the next sections.

To investigate the structure of a SETAF, we use the notion of the *primal graph*: a directed graph that resembles the structure of the SETAF's directed hypergraph. We will use this extension of our graph-related terminology to the directed hypergraph structure of SETAFs several times in the remaining part of this thesis as a starting point for structural properties. Intuitively, collective attacks are "split up" in order to obtain a directed graph with a similar structure as the original SETAF.

Definition 2.10 (Primal Graph)**.** *Let $SF = (A, R)$ be a SETAF. Its* primal graph *is defined as $primal(SF) = (A, R')$ with $R' = \{(t, h) \mid (T, h) \in R, t \in T\}$.*

We illustrate the primal graph with the following example.

Example 2.11. *Recall the SETAF SF. Its primal graph $primal(SF)$ looks as follows.*

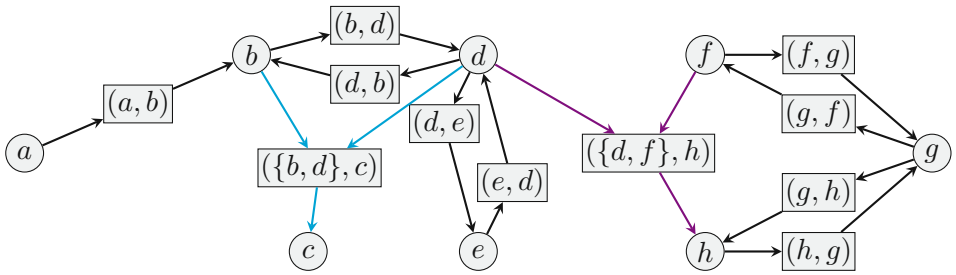


Another technical notion we make use of is the *incidence graph*. As with the primal graph the main idea is to implement the hypergraph structure of a SETAF via a directed graph. However, the incidence graph has both the arguments and the attacks as nodes. Hence, the incidence graph is bipartite by construction, as the argument-nodes only have edges with attack-nodes and vice versa. An edge from an argument-node to an attack node indicates that the argument is in the tail of the attack, an edge from an attack-node to an argument-node indicates that the argument is the head of the attack. While several SETAFs can map to the same primal graph, the incidence graph uniquely characterizes a SETAF.

Definition 2.12. *Let $SF = (A, R)$ be a SETAF. We define the* incidence graph *of SF as $\mathtt{Inc}(SF) = (V, E)$ with $V = A \cup R$ and $E = \{(t, (T, h)), ((T, h), h) \mid (T, h) \in R, t \in T\}$.*

We illustrate the incidence graph with the following example.

Example 2.13. *The incidence graph* $\texttt{Inc}(SF)$ *for* SF *looks as follows.*

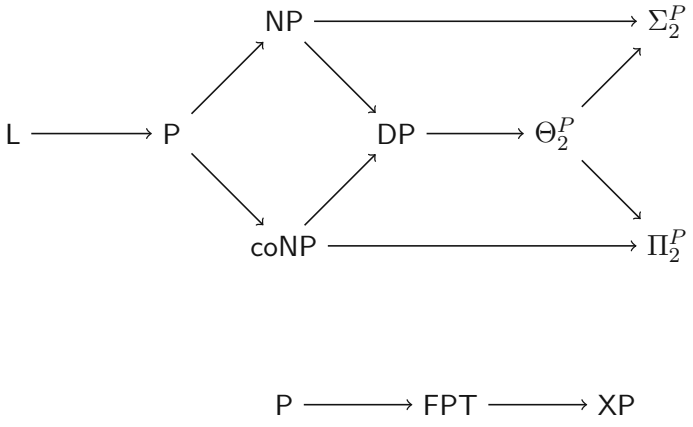


2.2 Complexity of Reasoning in Abstract Argumentation

We assume the reader to have basic knowledge in computational complexity theory[4], in particular we make use of the complexity classes L (logarithmic space), P (polynomial time), NP (non-deterministic polynomial time), coNP, DP ($L_1 \cap L_2$ for $L_1 \in \mathsf{NP}$, $L_2 \in \mathsf{coNP}$), Θ_2^P ($\mathsf{P}^{\mathsf{NP}[log(n)]}$), Σ_2^P (NP^NP), and Π_2^P ($\mathsf{coNP}^\mathsf{NP}$).

For a more fine-grained complexity analysis we make use of the complexity class FPT (fixed-parameter tractability): a problem is fixed-parameter tractable w.r.t. a parameter if there is an algorithm with runtime $O(f(p) \cdot \mathsf{poly}(n))$, where n is the size of the input, p is an integer describing the instance called the *parameter value*, and $f(\cdot)$ is an arbitrary computable function independent of n (typically at least exponential in p). We also make use of the class XP. A problem is in XP w.r.t. a parameter if there is an algorithm with runtime $O(n^{O(p)})$, where again n is the size of the input and p is the parameter value.

We have the following relationships between these classes (an arrow from class $\mathcal{C}$ to $\mathcal{C}'$ means $\mathcal{C} \subseteq \mathcal{C}'$, we omit some arrows that are immediate due to transitivity):



[4]For a gentle introduction to complexity theory in the context of formal argumentation, see [DD18].

For a given SETAF $SF = (A, R)$ and an argument $a \in A$, we consider the following decision problems (under semantics σ):

- Credulous acceptance $Cred_\sigma$: Given $SF = (A, R)$ and $a \in A$, is it true that $a \in E$ for some $E \in \sigma(SF)$?
- Skeptical acceptance $Skept_\sigma$: Given $SF = (A, R)$ and $a \in A$, is it true that $a \in E$ for each $E \in \sigma(SF)$?
- Verification Ver_σ: Given $SF = (A, R)$ and $E \subseteq A$, is it true that $E \in \sigma(SF)$?

In this thesis, we investigate the computational complexity of these problems for SETAFs that are structured in some way. As a starting point for our structural analyses we take the *graph classes* that were investigated for SETAFs [DKW21a]. The graph classes we focus on are acyclicity, even-cycle-freeness, self-attack-free full-symmetry, primal-bipartiteness, and odd-cycle-freeness. We denote the set of SETAFs that belong to these graph classes by ACYC, NOEVEN, SYM, BIP, and NOODD, respectively. These graph classes generalize the respective counterparts on directed graphs, which form tractable graph-fragments for AFs [Dun07, DD18].

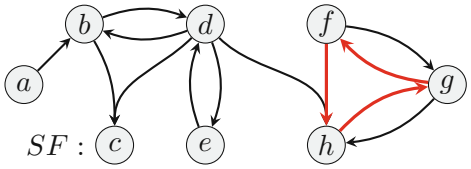
As a fundamental concept towards the graph classes we first define *cycles* for SETAFs.

Definition 2.14. *Let $SF = (A, R)$ be a SETAF. A* cycle *C of length n is a sequence $(a_1, \ldots, a_n, a_1)$ s.t. for $1 \leq i \leq n$ the arguments $a_i \in A$ are pairwise distinct, and there exists attacks $(T_i, a_{i+1}) \in R$ with $a_i \in T_i$, and $(T_n, a_1) \in R$.*

Clearly, the cycles of a SETAF correspond to directed cycles in its primal graph.

Definition 2.15. *Let $SF = (A, R)$ be a SETAF. SF is* acyclic *if it contains no cycle,* even-cycle-free *if it contains no cycle of even length, and* odd-cycle-free *if it contains no cycle of odd length.*

Example 2.16. *Recall the SETAF SF, we highlight its only odd length cycle (f, g, h, f).*



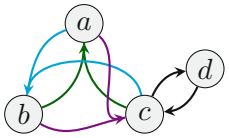
We continue with the notion of *full-symmetry*, which in our context is only relevant for self-attack-free SETAFs[5], i.e., there is no attack (T, h) with $h \in T$.

[5]While this definition goes back to the author's master thesis [Kön20], note that an equivalent definition called 'strong symmetry' was published only few weeks later by Bienvenu and Bourgaux [BB20], who also showed that this class of SETAFs are coherent (i.e., preferred and stable extensions coincide).

Definition 2.17. *Let $SF = (A, R)$ be a self-attack-free SETAF. SF is* fully-symmetric *if for each $(T, h) \in R$ there are attacks $((T \setminus \{t\}) \cup \{h\}, t) \in R$ for each $t \in T$.*

In words, a SETAF is fully-symmetric if for each attack all involved arguments attack every single involved argument. This is illustrated in the following example.

Example 2.18. *The following SETAF is fully-symmetric. Note how e.g. the attacks involving $S = \{a, b, c\}$ are for each $x \in S$ s.t. $(S \setminus \{x\}, x)$. Moreover, also "classical" symmetry as in the case of the arguments c and d is covered by this definition.*



As a final graph-class we consider *primal-bipartiteness*. A SETAF is primal-bipartite if its primal graph is bipartite. This means that within the two partitions, no (parts of) attacks occur.

Definition 2.19. *Let $SF = (A, R)$ be a SETAF. SF is* primal-bipartite *if there is a partitioning of A into two sets (Y, Z) s.t.*

- *$Y \cup Z = A$, $Y \cap Z = \emptyset$, and*
- *for every $(T, h) \in R$ either (1) $T \subseteq Y$ and $h \in Z$, or (2) $T \subseteq Z$ and $h \in Y$.*

The complexity landscape of SETAFs coincides with that of Dung AFs and is depicted in Table 2.1 ("General"). As SETAFs generalize Dung AFs the hardness results for Dung AFs [CMDM05, DT96, DBC02, Dvo12, DW10, DW11, Dun09, CCD12] (for a survey see [DD18]) carry over to SETAFs. Also the same upper bounds hold for SETAFs [DGW18].

In addition to the computational complexity for reasoning in arbitrary SETAFs, Table 2.1 also depicts the known *tractable fragments* of reasoning problems for SETAFs [DKUW24]. If these properties are satisfied for the underlying hypergraph of a SETAF the reasoning problems become (mostly) computationally easy (i.e., polynomial-time). These fragments are acyclicity, even-cycle-freeness, self-attack-free full-symmetry, and primal-bipartiteness. Moreover, while odd-cycle-freeness is not considered a tractable fragment due to the (co)NP-hardness results, a drop to the first level of the polynomial hierarchy can be observed in this class. Moreover, the verification of preferred extensions is tractable for odd-cycle-free SETAFs.

While for acyclicity, even-cycle-freeness, and primal-bipartiteness it was shown that the respective restriction on the primal graph suffices for computational ease, the same is not the case for symmetry: in [DKW21a] it was shown that *primal-symmetry* only admits a

		cf	*grd*	*adm*	*com*	*stb*	*pref*	*naive*	*sem*	*stage*	*ideal*	*eager*
General	$Cred_\sigma$	in P	P-c	NP-c	NP-c	NP-c	NP-c	in P	Σ_2^P-c	Σ_2^P-c	in Θ_2^P	Π_2^P-c
	Ver_σ	in P	P-c	in L	in L	in L	coNP-c	in P	coNP-c	coNP-c	in Θ_2^P	DP-c
	$Skept_\sigma$	in P	P-c	triv.	P-c	coNP-c	Π_2^P-c	in P	Π_2^P-c	Π_2^P-c	in Θ_2^P	Π_2^P-c
ACYC	$Cred_\sigma$	in P	P-c	P-c	P-c	P-c	P-c	in P	P-c	P-c	in P	in P
	Ver_σ	in P	in P	in L	in L	in L	in L	in P	in L	in L	in P	in P
	$Skept_\sigma$	in P	P-c	triv.	P-c	P-c	P-c	in P	P-c	P-c	in P	in P
NOEVEN	$Cred_\sigma$	in P	P-c	P-c	P-c	P-c	P-c	in P	P-c	Σ_2^P-c	in P	in P
	Ver_σ	in P	in P	in L	in P	in P	in P	in P	in P	in coNP	in P	in P
	$Skept_\sigma$	in P	P-c	triv.	P-c	P-c	P-c	in P	P-c	Π_2^P-c	in P	in P
NOODD	$Cred_\sigma$	triv.	P-c	NP-c	NP-c	NP-c	NP-c	in P	NP-c	NP-c	coNP-c	coNP-c
	Ver_σ	in P	in P	in L	in P	in P	in P	in P	in P	in P	coNP-c	coNP-c
	$Skept_\sigma$	in P	P-c	triv.	P-c	coNP-c	coNP-c	in P	coNP-c	coNP-c	coNP-c	coNP-c
SYM	$Cred_\sigma$	triv.	in L	triv.	triv.	triv.	triv.	triv.	triv.	triv.	in P	in P
	Ver_σ	in P	in P	in L	in L	in P	in P	in P	in P	in P	in P	in P
	$Skept_\sigma$	in P	in L	triv.	in L	in L	in L	in L	in L	in L	in P	in P
BIP	$Cred_\sigma$	triv.	P-c	P-c	P-c	P-c	P-c	in P	P-c	P-c	in Θ_2^P	in Π_2^P
	Ver_σ	in P	in P	in L	in L	in L	in L	in P	in L	in L	in Θ_2^P	in DP
	$Skept_\sigma$	in P	P-c	triv.	P-c	P-c	P-c	in P	P-c	P-c	in Θ_2^P	in Π_2^P

Table 2.1: Graph fragments in SETAFs. $\mathcal{C}$-c denotes completeness for class $\mathcal{C}$; "triv." denotes a trivial problem (either all instances are positive or all instances are negative).

shortcut for determining whether an argument is in the grounded extension, but otherwise the hardness-results for each semantics $\sigma \in \Sigma$ still applies.

Many of the above mentioned hardness-results are based on (variations of) the so-called *standard reduction*, see e.g. [DD18, Reduction 3.6]. The idea is to express the complexity of the boolean satisfiability problem in an AF. As we will base some of our complexity results for SETAFs on the standard reduction, we briefly recall the construction and some known results.

Reduction 2.20 (Standard Reduction)**.** *Let φ be a formula in CNF (conjunctive normal form) with clauses C over atoms Y. We construct the AF F_φ as follows:*

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

An example of the standard reduction is depicted in Figure 2.1.

Some of the main results from the literature regarding the semantics under our consideration can be summarized as follows [DD18].

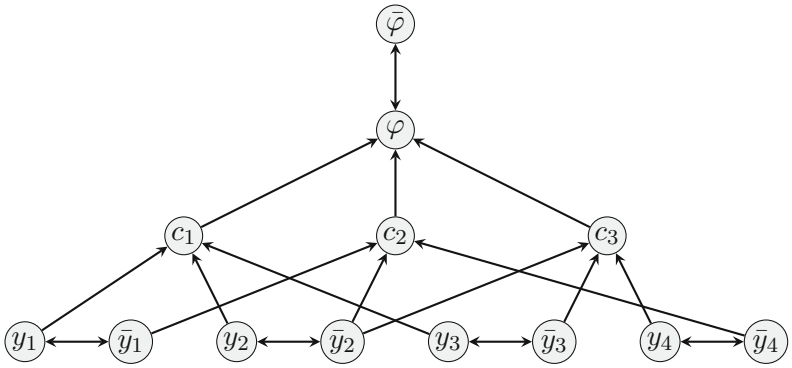


Figure 2.1: The standard reduction applied to φ with atoms $Y = \{y_1, y_2, y_3, y_4\}$, and clauses $C = \{\{y_1, y_2, y_3\}, \{\bar{y}_1, \bar{y}_2, \bar{y}_4)\}, \{\bar{y}_2, \bar{y}_3, y_4\}\}$.

Theorem 2.21. *Let φ be a propositional formula in CNF and F_φ the corresponding AF from the standard reduction. The following statements are equivalent:*

1. *the formula φ is satisfiable,*
2. *the argument φ is credulously accepted in F_φ w.r.t. the semantics $\sigma \in \{adm, com, stb, pref, sem, stage\}$,*
3. *the argument $\bar{\varphi}$ is not skeptically accepted in F_φ w.r.t. the semantics $\sigma \in \{stb, pref, sem, stage\}$, and*
4. *the set $\{\bar{\varphi}\}$ is not the unique ideal/eager extension.*

Based on the notions from this chapter, we will in the following investigate principles and advanced algorithmic techniques for SETAFs. Note that to improve readability we recall the AF-pendants of our techniques in the respective chapters. E.g., the state-of-the-art for *SCC-recursiveness*, *backdoors*, and *treewidth* on AFs (which are the basis for our considerations) are discussed before we introduce our corresponding SETAF notions.

CHAPTER 3

Principle-Based Analysis

In an argumentation process, the selection of consistent sets of arguments (so called extensions) is the final step on the abstract level. Naturally, different approaches for this selection are relevant for different problems, which is why a variety of semantics has been proposed. To classify and distinguish the various semantics the *principle-based analysis* is an established method in formal argumentation research [BG07]. Principle-based investigations have recently been performed e.g. for AFs [vdTV17], ranking semantics [BDKM17], preference-based argumentation frameworks [KvdTV18], quantitative bipolar argumentation frameworks [BRT19], and abstract agent argumentation frameworks [YCQ+21].

In this chapter we consider the principle-based approach for argumentation frameworks with collective attacks (SETAFs). Although we will see that in many cases the behavior generalizes from AFs to the setting with collective attacks, our study also reveals situations where caution is required and thus emphasizes properties we deem natural for AFs. In fact, many AF principles like SCC-recursiveness [BGG05] or the recently introduced modularization property [BBU20a] are concerned with partial evaluation of the given graph and step-wise computation of extensions. We will pay special attention to these kind of principles since (a) they require to establish novel technical foundations when generalizing the underlying structure from simple graphs to hypergraphs and (b) have immediate implications for the design of solvers. Along the way, we will also introduce a SETAF version for the reduct of an AF [BBU20b] which has proven to be a handy tool when investigating argumentation semantics.

Finally, we will utilize prior work regarding *graph-classes*[6], which in conjunction with the formerly established principles provide a framework for efficient computation. We will

[6]While Chapter 3 of this thesis largely comprises of results from [DKUW24], we list the work on graph classes which appeared in the same article as a prior contribution in the background, as these results were obtained during work on the author's master thesis [Kön20]. However, we extend and utilize these results: in Section 3.5.2 we show that the classes remain easy in the presence of *mitigated attacks*

apply these results in the context of SCC-recursiveness to establish the computational speedup, providing novel algorithms for the evaluation of frameworks along the way.

The main takeaway of this chapter is that our natural extensions of the AF principles are well-behaving for SETAFs and can be utilized for efficient computation. We show that basic properties are preserved, as well as their implications in terms of the structure of extensions. More specifically, this chapter is structured as follows.

- First, we generalize and analyze basic principles of abstract argumentation for SETAFs in Section 3.1. Moreover we introduce novel principles that are trivial for standard AFs, but provide additional insights in the case of SETAFs.
- We propose the E-reduct SF^E for a SETAF SF and a set E of arguments and investigate its core properties, including the modularization property (Section 3.2). Moreover, we use the reduct to provide alternative characterizations of SETAF semantics.
- We introduce uninfluenced sets of arguments in SETAFs as the counterpart of unattacked sets in AFs. We then propose and investigate a SETAF version of the directionality and non-interference principles (Section 3.3) and SCC-recursiveness (Section 3.4).
- We discuss the computational implications of modularization, directionality and SCC-recursiveness in Section 3.5. In particular we illustrate the potential for incremental algorithms. We then refine these results in order to be applicable in even more cases. We introduce and analyze graph classes for SETAFs and exemplify their use for efficient computation using the SCC-recursive scheme, generalizing known (parameterized) tractability results from the literature.
- Finally, in Section 3.6 we discuss the results of this chapter.

This chapter is based on the paper [DKUW24], which in turn contains content of the papers [DKUW21], [DKW21a], and [DKUW22].

3.1 Basic Principles

We start our principle-based analysis of SETAF semantics by generalizing basic principles from AFs. Satisfaction (or non-satisfaction) of principles allows us to distinguish semantics with respect to fundamental properties that are crucial in certain applications.

The principles we consider have natural counterparts for Dung-style AFs, simply by applying them to SETAFs where $|T| = 1$ for each tail. Hence, if the AF counterpart of a principle is violated by a semantics, this carries over to the SETAF principle. We therefore formalize the following observation:

(see Definition 3.87) and (with the notable exception of full-symmetry) are applicable in the context of SCC-recursiveness.

	cf	*grd*	*adm*	*com*	*stb*	*pref*	*naive*	*sem*	*stage*	*ideal*	*eager*
Conflict-freeness	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Defense	✗	✓	✓	✓	✓	✓	✗	✓	✗	✓	✓
Admissibility	✗	✓	✓	✓	✓	✓	✗	✓	✗	✓	✓
Reinstatement	✗	✓	✗	✓	✓	✓	✗	✓	✗	✓	✓
CF-reinstatement	✗	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓
Weak reinstatement	✗	✓	✗	✓	✓	✓	✗	✓	✗	✓	✓
Naivety	✗	✗	✗	✗	✓	✗	✓	✗	✓	✗	✗
I-maximality	✗	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓
Allowing abstention	✓	✓	✓	✓	✗	✗	✗	✗	✗	✓	✓
Crash resistance	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
Tightness	*✗*	✓	✗	✗	*✗*	✗	*✗*	✗	*✗*	✓	✓
Modularization	✗	✓	✓	✓	✓	✓	✗	✓	✗	✓	✓
Directionality	✓	✓	✓	✓	✗	✓	✗	✗	✗	✓	✗
Semi-directionality	✓	✓	✓	✓	✗	✓	✓	✗	✗	✓	✗
Weak-directionality	✓	✓	✓	✓	✓	✓	✗	✗	✗	✓	✗
Non-interference	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
SCC-recursiveness	✗	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗
Allowing partial conflicts I	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Allowing partial conflicts II	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓	✗
Allowing partial conflicts III	✓	✗	✗	✗	✓	✗	✓	✗	✗	✗	✗
Tail strengthening	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Attack weakening	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 3.1: An overview of our results regarding SETAF principles. Differences from the respective results for AFs are highlighted (*✗*).

Observation 3.1. *Let P be a SETAF-principle that properly generalizes an AF-principle P^{AF} in the sense that for SETAFs SF with $|T| = 1$ for each $(T, h) \in R(SF)$, every semantics σ satisfies P iff it satisfies P^{AF}. In this case, if a semantics σ does not satisfy P^{AF}, then σ does not satisfy P.*

As all of our principles properly generalize the respective AF-principles, whenever a principle is not satisfied for AFs, this translates to the corresponding SETAF principle as well.

An overview of the results of the principle-based analysis is given in Table 3.1. Note that the SETAF-specific principles Allowing Partial Conflicts I–III, Tail Strengthening, and Attack Weakening trivialize or are not applicable to AFs.

3.1.1 Basic Properties

Now we follow [vdTV17] and introduce analogous principles for SETAFs. Our first set of principles is concerned with basic properties of semantics.

Principle 3.2 (Conflict-freeness [Dun95, NP06b]). *A semantics σ satisfies* conflict-freeness *if and only if for all SETAFs SF, every $E \in \sigma(SF)$ is conflict-free.*

As conflict-freeness is a basic principle that underlies most semantics by definition, it is not surprising that all semantics under our consideration satisfy this principle.

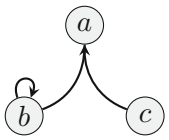
Proposition 3.3. *Each $\sigma \in \Sigma$ satisfies* conflict-freeness.

The concept of *defense* is central to most classical semantics of abstract argumentation. This central notion is the core of Dung's framework [Dun95] and has been adapted by Nielsen and Parsons [NP06b] for SETAFs to take collective attacks into account.

Principle 3.4 (Defense [Dun95, NP06b]). *A semantics σ satisfies* defense *if and only if for all SETAFs SF, we have that $E \in \sigma(SF)$ implies $E \subseteq \Gamma_{SF}(E)$.*

Most semantics that satisfy defense are refinements of *adm*. Thus satisfaction of defense is encoded explicitly within their definition. For stable semantics we recall the well-known relation $stb(SF) \subseteq pref(SF)$ for any SETAF SF. The semantics based only on conflict-freeness but not defense do not satisfy admissibility (as it is the case in AFs), as can be easily seen in the following Example 3.5.

Example 3.5. *Consider the following SETAF SF. We have $\{a, c\} \in cf(SF)$, as well as $\{a, c\} \in naive(SF)$, and $\{a, c\} \in stage(SF)$, but a is not defended by $\{a, c\}$, which means that $\{a, c\}$ is not an extension in any of the admissibility-based semantics.*



Proposition 3.6. *The principle* defense *is satisfied by grd, adm, com, stb, pref, sem, ideal, and eager, and violated by cf, naive, and stage.*

The *admissibility* principle combines the former two, defense and conflict-freeness.

Principle 3.7 (Admissibility [Dun95, NP06b]). *A semantics σ satisfies* admissibility *if and only if for all SETAFs SF, every $E \in \sigma(SF)$ is admissible.*

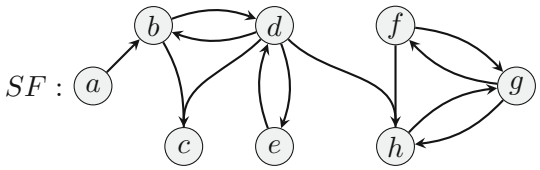
Since our semantics $\sigma \in \Sigma$ all satisfy conflict-freeness by Proposition 3.3, we have that these semantics satisfy the *admissibility* principle if and only if they satisfy defense. Hence, the results from Proposition 3.6 carry over.

Proposition 3.8. *The principle* admissibility *is satisfied by grd, adm, com, stb, pref, sem, ideal, and eager, and violated by cf, naive, and stage.*

In the following, we generalize different versions of *reinstatement*. This principle is concerned the question whether or not all defended arguments are indeed included in a given σ-extension E. The principle is thus inspired by the definition of completeness which requires $a \in E$ whenever E defends a. Speaking in terms of the characteristic function, admissible sets satisfy $E \subseteq \Gamma_{SF}(E)$ whereas complete extensions refine this to $E = \Gamma_{SF}(E)$. The reinstatement principle formalizes the "$\supseteq$"-direction.

Principle 3.9 (Reinstatement [BG07])**.** *A semantics σ satisfies* reinstatement *if and only if for all SETAFs SF, we have that $E \in \sigma(SF)$ implies $E \supseteq \Gamma_{SF}(E)$.*

Example 3.10. *Recall the SETAF SF from Example 2.2. It is easy to check that $\{a\}$ is admissible in SF. Since $\{a\}$ defends c, we have $\Gamma_{SF}(\{a\}) = \{a, c\} \not\subseteq \{a\}$, which means that admissible semantics violates reinstatement.*



Complete semantics satisfies reinstatement by definition, the other results follow from the known relations $stb(SF) \subseteq sem(SF) \subseteq pref(SF) \subseteq com(SF)$, as well as $ideal(SF) \subseteq com(SF)$, and $eager(SF) \subseteq com(SF)$.

Proposition 3.11. *The principle* reinstatement *is satisfied by grd, com, stb, pref, sem, ideal, and eager, and violated by cf, adm, naive, and stage.*

For admissibility-based semantics, the fundamental lemma (originally due to [Dun95], for SETAFs in [NP06b]) ensures conflict-freeness for additional defended arguments. Formally, if $E \in adm(SF)$ and $a \in \Gamma_{SF}(E)$, then $E \cup \{a\} \in cf(SF)$. For semantics based on conflict-freeness such as naive or stage, it might happen that some extension E is in conflict with some argument a, although $a \in \Gamma_{SF}(E)$. However, if $E \cup \{a\} \notin cf(SF)$ is the case, then we do not expect it to be a σ-extension anymore (if σ is cf-based). Therefore, the following refinement of reinstatement has been proposed, which explicitly requires $E \cup \{a\}$ to be conflict-free.

Principle 3.12 (CF-Reinstatement [BG07])**.** *A semantics σ satisfies* CF-reinstatement *if and only if for all SETAFs SF, we have that $E \in \sigma(SF)$, $a \in \Gamma_{SF}(E)$, and $E \cup \{a\} \in cf(SF)$ imply $a \in E$.*

Due to the fundamental lemma, for admissibility-based semantics this notion simply coincides with reinstatement. However, also *naive* and *stage* satisfy CF-reinstatement,

which can be inferred from their respective maximality requirements: assume for $a \notin E$ it holds $E \cup \{a\} \in cf(SF)$, then E cannot be a naive extension as $E \cup \{a\} \supset E$. Finally, recall that $stage(SF) \subseteq naive(SF)$ for every SETAF SF.

Proposition 3.13. *The principle* CF-reinstatement *is satisfied by grd, com, stb, pref, naive, sem, stage, ideal, and eager, and violated by cf and adm.*

Another possible way to refine reinstatement is by restricting our attention to so-called *strongly defended* arguments. Strong defense was initially defined as the underlying defense notion for strong admissibility [BG07, Cam14]. So instead of imposing $E \cup \{a\} \in cf(SF)$ as a premise, we take fewer candidates a into consideration: effectively, a strongly defended argument does not play a role in its own defense. Strong defense for SETAFs is defined as follows[7].

Definition 3.14. *Given a SETAF $SF = (A, R)$, an argument $a \in A$ is* strongly defended *(in SF) by a set $S \subseteq A$ if for each $(B, a) \in R$ there is an argument $b \in B$ and a set $S' \subseteq S$ such that $(S', b) \in R$, and each $s \in S'$ is strongly defended by $S \setminus \{a\}$.*

Naturally, the induced weakening of reinstatement is given as follows.

Principle 3.15 (Weak reinstatement [BG07])**.** *A semantics σ satisfies* weak reinstatement *if and only if for all SETAFs SF, if $E \in \sigma(SF)$ and E strongly defends $a \in A$, then $a \in E$.*

As in the AF case, if a set strongly defends an argument, then it also (classically) defends said argument. Hence, if a semantics satisfies reinstatement, also weak reinstatement is satisfied. The positive results in Table 3.1 are due to this property. The negative cases follow from Observation 3.1 and the respective counter-examples from AFs.

Proposition 3.16. *The principle* weak reinstatement *is satisfied by grd, com, stb, pref, sem, ideal, and eager, and violated by cf, adm, naive, and stage.*

The final two principles we consider in this subsection are concerned with the structure of the σ-extensions. First, naivety checks whether each $E \in \sigma(SF)$ is maximal conflict-free.

Principle 3.17 (Naivety [vdTV17])**.** *A semantics σ satisfies* naivety *if and only if for all SETAFs SF, $E \in \sigma(SF)$ implies that E is $\subseteq$-maximal in $cf(SF)$.*

Again, the negative results are due to Observation 3.1; the positive ones follow from the relation $stb(SF) \subseteq stage(SF) \subseteq naive(SF)$.

[7] Strong admissibility has been generalized to Abstract Dialectical Frameworks (ADFs) [KVV22] and SETAFs can be interpreted as special kind of ADF [Pol16, LPS16]. In fact, our definition of strong defense is compatible with the respective notions on ADFs.

Proposition 3.18. *The principle* naivety *is satisfied by stb, naive, and stage and violated by cf, grd, adm, com, pref, sem, ideal, and eager.*

Second, I-maximality is satisfied iff $\sigma(F)$ forms an anti-chain, i.e. no two extensions are in proper subset relation to each other. Here, I-maximality is due to [BG07].

Principle 3.19 (I-maximality [BG07])**.** *A semantics σ satisfies* I-maximality *if and only if for all SETAFs SF, if $E, E' \in \sigma(SF)$ and $E \subseteq E'$, then $E = E'$.*

Oftentimes, I-maximality is directly implemented in the definition of the σ-extensions (most famously *grd* and *pref*). Further results for SETAFs have been shown in [DFW19].

Proposition 3.20. *The principle* I-maximality *is satisfied by grd, stb, pref, naive, sem, stage, ideal, and eager, and violated by cf, adm, and com.*

3.1.2 Advanced Principles

The next principle we discuss is called *allowing abstention* [BCG11]. As the name suggests, it allows the underlying semantics to be indecisive in certain scenarios. Formally, suppose we have some target argument a and two extensions $E \in \sigma(SF)$ as well as $E' \in \sigma(SF)$ where $a \in E$, but $a \in (E')^+$; that is, E accepts a, but E' rejects it. In this case, since the status of a is not determined, one might argue that σ should also admit an extension where a is neither accepted nor rejected. This idea is formalized by the allowing abstention principle.

Principle 3.21 (Allowing abstention [BCG11])**.** *A semantics σ satisfies* allowing abstention *if and only if for all SETAFs $SF = (A, R)$, for all $a \in A$, if there exist $E, E' \in \sigma(SF)$ with $a \in E$ and $a \in (E')^+$, then there also exists some $E'' \in \sigma(SF)$ such that $a \notin (E'')^{\oplus}$.*

As *grd*, *ideal*, and *eager* always admit a single extension, the principle is trivially satisfied by these semantics. Moreover, allowing abstention is satisfied by complete semantics, since—as in AFs—if there exist $E, E' \in com(SF)$ with $a \in E$ and $a \in E'^+$, this means $a \notin G^{\oplus}$ where $G \in grd(SF)$. For $\sigma \in \{cf, adm\}$, this follows from $\emptyset \in \sigma(SF)$ for all SETAFs SF.

Proposition 3.22. *The principle* allowing abstention *is satisfied by cf, grd, adm, com, ideal, and eager, and violated by stb, pref, naive, sem, and stage.*

The next principle we discuss is called *crash resistance* [CCD12]. It formalizes that it should not be possible to render certain parts of an argumentation framework completely meaningless by adding a particular set of (disjoint) arguments. This idea is formalized in the definition of a contaminating SETAF.

Definition 3.23. *We call a SETAF $SF' = (A', R')$* contaminating *for a semantics σ if for every SETAF $SF = (A, R)$ with $A \cap A' = \emptyset$, it holds that $\sigma(SF \cup SF') = \sigma(SF')$, where $SF \cup SF'$ is the SETAF $(A \cup A', R \cup R')$.*

That is, the semantics of the given SETAF $SF = (A, R)$ are entirely overwritten due to the presence of SF'. Observe that SF' has this influence on *every* conceivable SETAF SF. The crash resistance principle forbids the existence of such a contaminating SETAF.

Principle 3.24 (Crash resistance)**.** *A semantics σ satisfies* crash resistance *if there is no contaminating SETAF for σ.*

As in the case for AFs, *stb* is the only semantics considered in this thesis which is not crash-resistant. The reason is that one can choose SF' to be an isolated odd cycle, yielding $stb(SF \cup SF') = \emptyset$ for any SETAF SF. The other semantics are more robust in this regard and yield $\sigma(SF \cup SF') = \{E \cup E' \mid E \in \sigma(SF), E' \in \sigma(SF')\}$ whenever $A \cap A' = \emptyset$.

Proposition 3.25. *The principle* crash resistance *is satisfied by cf, grd, adm, com, pref, naive, sem, stage, ideal, and eager, and violated by stb.*

The last principle we consider in this subsection is inspired by research on expressiveness in abstract argumentation [DDLW15, DRW20]. In this context, the notion of *tightness* has been introduced. It formalizes that if E is a σ-extension and $a \notin E$, then some $b \in E$ must be the culprit for a not being acceptable. Towards formalizing this, we need the notion of *pairs*, i.e. jointly acceptable arguments.

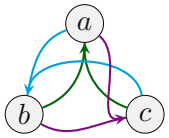
Definition 3.26. *Let SF be a SETAF and σ some semantics. We define the set of* pairs *as $\mathcal{P}airs_\sigma(SF) = \{(a, b) \mid \exists E \in \sigma(SF)$ s.t. $\{a, b\} \subseteq E\}$.*

A semantics σ satisfies the tightness principle if for an argument a that does not belong to an extension $E \in \sigma(SF)$ there is some $b \in E$ such that $\{a, b\}$ is not part of any σ-extension, i.e. there is a single argument b in E which can be considered responsible for excluding a.

Principle 3.27 (Tightness)**.** *A semantics σ is* tight *if for all SETAFs $SF = (A, R)$, for all $E \in \sigma(SF)$ and all credulously accepted $a \in A$, the following implication holds: if $E \cup \{a\} \notin \sigma(SF)$, then there is some $b \in E$ such that $(a, b) \notin \mathcal{P}airs_\sigma(SF)$.*

Clearly, any unique status semantics σ, i.e. $|\sigma(SF)| = 1$ for each SF, is tight. However, while Dunne et al. showed that on AFs tightness holds also for conflict-freeness, naive, stable, and stage semantics, this is not the case for SETAFs, as the following example illustrates (see also [DFW19]).

Example 3.28. *Consider the following SETAF SF.*



We have $naive(SF) = stb(SF) = stage(SF) = \{\{a, b\}, \{b, c\}, \{a, c\}\}$. *Consider for example* $c \notin \{a, b\}$. *Tightness would require* $(a, c) \notin \mathcal{P}airs_\sigma(SF)$ *or* $(b, c) \notin \mathcal{P}airs_\sigma(SF)$, *but both* $\{a, c\}$ *and* $\{b, c\}$ *are* σ*-extensions. Likewise, the same counter-example illustrates that conflict-free sets are not tight.*

Thus, we end up with only the unique status semantics *grd*, *ideal*, and *eager* being tight.

Proposition 3.29. *The principle* tightness *is satisfied by grd, ideal, and eager, and violated by cf, adm, com, pref, naive, sem, stage, and stb.*

3.1.3 SETAF-Specific Principles

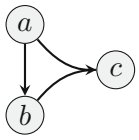
The principles we discussed up until this point were inspired by known AF principles and have been suitably adjusted to SETAFs. In this section we want to introduce *genuine* SETAF principles, i.e., we discuss properties which are not applicable to or trivialize for standard Dung-AFs.

Towards our first SETAF principle, observe that a conflict within some set E of arguments requires the whole tail of a corresponding attack to be contained in E; that is, there has to be some $(T, a) \in R$ with $a \in E$ and $T \subseteq E$. The underlying intuition is that attacks are only "active" if the whole tail is accepted. Semantics which adhere to this intuition should be able to distinguish between attacks that are fully active, i.e., $T \subseteq E$ and attacks which are only partially active, i.e., $T \cap E \neq \emptyset$, but $T \not\subseteq E$. We therefore consider the following notion of a partial conflict.

Definition 3.30. *Let* $SF = (A, R)$ *be a SETAF and* $E \subseteq A$. *We say* E *contains a* partial conflict *whenever there is some* $(T, a) \in R$ *with* $a \in E$ *and* $T \cap E \neq \emptyset$ *as well as* $T \not\subseteq E$.

We illustrate this notion with the following Example 3.31.

Example 3.31. *Consider the following SETAF. The set* $\{a, c\}$ *contains a partial conflict, but is acceptable w.r.t. the conflict-free based semantics, as it is not conflicting. Moreover, in the admissibility-based semantics* $\{a, c\}$ *is accepted, since* a *is unattacked and* c *is defended by* a.



The way SETAF semantics are designed, semantics should usually allow partial conflicts (which we will abbreviate by "APC").

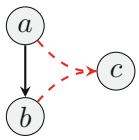
Principle 3.32 (Allowing Partial Conflicts I)**.** *A semantics σ satisfies the principle* allowing partial conflicts I *if there is some SETAF SF and some extension $E \in \sigma(SF)$ s.t. E contains some partial conflict.*

Observe that for Dung-AFs partial conflicts never exist, since the conditions $T \cap E \neq \emptyset$ as well as $T \not\subseteq E$ can never be met simultaneously for a singleton T. Hence this principle is trivially violated for AFs. For SETAFs, it is also easy to see that all semantics under our consideration satisfy APC I.

Proposition 3.33. *Each $\sigma \in \Sigma$ satisfies* allowing partial conflicts I.

We can strengthen this requirement as follows. Intuitively, we say for a given extension E we can add a new attack (T, h) and still have E as an extension in the remaining framework if at least one argument in T is already attacked by E.

Example 3.34. *We continue with the SETAF from Example 3.31. Assume the attack $(\{a, b\}, c)$ is originally not part of the framework, but added later. Consider the set $E = \{a, c\}$: we have $a \in E$ and $b \in E^+$. If the principle APC II is satisfied, adding the attack $(\{a, b\}, c)$ will not render E unacceptable (if it was acceptable before).*



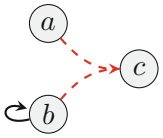
Principle 3.35 (Allowing Partial Conflicts II)**.** *A semantics σ satisfies the principle* allowing partial conflicts II *if for every SETAF $SF = (A, R)$ and every $E \in \sigma(SF)$ it holds for all $h \in E, T_1 \subseteq E, \emptyset \subsetneq T_2 \subseteq E^+$ also $E \in \sigma(SF')$ where $SF' = (A, R \cup \{(T_1 \cup T_2, h)\})$.*

Since in admissibility-based semantics this added attack has no effect (as the tail is attacked), these semantics satisfy the principle. The exception to this rule is semi-stable semantics, as the introduction of the new attack might lead to a different preferred extension with a larger range. Finally, since no conflict is introduced, also *cf* and *naive* satisfy APC II. The counterexamples for *sem*, *stage*, and *eager* are illustrated in Example 3.39.

Proposition 3.36. *The principle* allowing partial conflicts II *is satisfied by cf, grd, adm, com, stb, pref, naive, and ideal, and violated by sem, stage, and eager.*

In APC II we require that for the introduced attack $(T_1 \cup T_2, h)$ there is at least one argument in T_2, i.e., there is at least one argument in $T_1 \cup T_2$ that is *attacked by* E. However, if we only require an argument that is *not in* E (instead of *attacked by* E), we end up with a stronger requirement, as illustrated in Example 3.37.

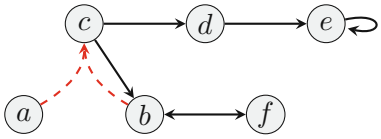
Example 3.37. *As in Example 3.34 assume the attack* $(\{a, b\}, c)$ *is originally not part of the framework, but added later. Consider again the set* $E = \{a, c\}$*: we have* $a \in E$ *and* $b \notin E$ *(but also* $b \notin E^+$*). If the principle APC III is satisfied, adding the attack* $(\{a, b\}, c)$ *will not render* E *unacceptable (if it was acceptable before).*



Principle 3.38 (Allowing Partial Conflicts III)**.** *A semantics* σ *satisfies the principle* allowing partial conflicts III *if for every SETAF* $SF = (A, R)$ *and every* $E \in \sigma(SF)$ *it holds for all* $h \in E, T_1 \subseteq E, \emptyset \subsetneq T_2 \subseteq A \setminus E$ *also* $E \in \sigma(SF')$ *where* $SF' = (A, R \cup \{(T_1 \cup T_2, h)\})$.

First note that for stable semantics APC II and APC III coincide, as for any $E \in stb(SF)$ it holds $E^+ = A \setminus E$ by definition. Similarly, for the conflict-freeness based semantics *cf* and *naive* it plays no role whether an argument is attacked or not, hence, APC III is still satisfied for these semantics. Most admissibility-based semantics under our consideration violate APC III, as the introduction of an attack (T, h) might lead to a situation where h is not defended, as Example 3.37 illustrates: while c was originally trivially acceptable w.r.t. the admissibility-based semantics grounded, admissible, complete, preferred, and ideal (as c was unattacked), after adding the attack $(\{a, b\}, c)$ the argument c cannot be defended. Clearly, we have that if σ satisfies APC III then σ satisfies APC II, and if σ satisfies APC II then σ satisfies APC I. The reverse does not hold, as the results in Table 3.1 illustrate.

Example 3.39. *Consider the SETAF SF (first without the attack* $(\{a, b\}, c)$*). It is easy to check that* $\{a, c, f\}$ *is a semi-stable, stage, and eager extension. However, if we add the attack* $(\{a, b\}, c)$ *the set* $\{a, b, d\}$ *becomes a stable extension, and is in fact the only stable extension of the resulting SETAF. Hence,* $\{a, b, d\}$ *is also the only semi-stable, stage, and eager extension, i.e.,* $\{a, c, f\}$ *is no longer an extension. This violates APC II (and, hence, APC III) for sem, stage, and eager.*



Combining these considerations, we get the following results for APC III.

Proposition 3.40. *The principle* allowing partial conflicts III *is satisfied by cf, stb, and naive, and violated by grd, adm, com, pref, sem, stage, ideal, and eager.*

The underlying idea of a collective attack (T, a) is that *all* arguments in T are required in order to defeat a. Hence, an attack (T', a) is in a certain sense stronger than (T, a) if $T' \subseteq T$. In the same spirit, if $T \subseteq E$ for some extension $E \in \sigma(F)$, then we make E stronger if (T, a) is replaced by some stronger attack.

Principle 3.41 (Tail Strengthening)**.** *A semantics σ satisfies* tail strengthening *if for all SETAFs $SF = (A, R)$ and for all $E \in \sigma(SF)$ the following implication holds: if $(T, a) \in R$ with $T \subseteq E$, then we also have $E \in \sigma(SF')$ where $SF' = (A, R')$ with $R' = (R \setminus \{(T, a)\}) \cup \{(T', a)\}$ for some $T' \subseteq T$.*

Vice versa, suppose we have an argument $a \in E$ and some incoming attack $(T, a) \in R$. If we make this attack weaker, we expect E still to be represent a jointly acceptable point of view. Formally:

Principle 3.42 (Attack Weakening)**.** *A semantics σ satisfies* attack weakening *if for all SETAFs $SF = (A, R)$ and for all $E \in \sigma(SF)$ the following implication holds: if $(T, a) \in R$ with $a \in E$, then we also have $E \in \sigma(SF')$ where $SF' = (A, R')$ with $R' = (R \setminus \{(T, a)\}) \cup \{(T', a)\}$ for some $T \subseteq T'$.*

For the semantics considered in this thesis, it follows by definition that both properties are satisfied.

Proposition 3.43. *Each $\sigma \in \Sigma$ satisfies* tail strengthening *and* attack weakening*.*

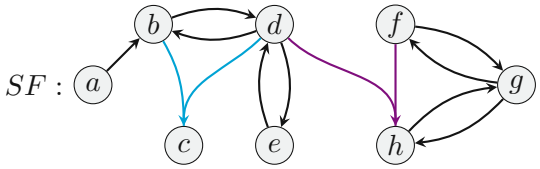
3.2 Reduct and Modularization

In the remainder of this chapter (as well as Chapters 4 and 5), our analysis will put strong emphasis on computational aspects and the partial evaluation of SETAFs. In this section, we will provide the first steps into this direction. First we will introduce the so-called SETAF reduct which corresponds to the resulting SETAF after the status of a certain subset of the arguments is decided. Based on this, we will generalize the modularization property [BBU20a], which formalizes how to compute extensions stepwise by means of the reduct. As an aside, the modularization property yields concise alternative characterizations for the classical semantics.

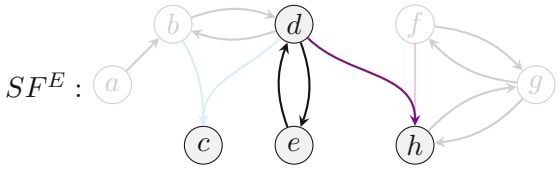
3.2.1 The SETAF Reduct

For many of our subsequent results, the *reduct* of a SETAF w.r.t. a given set E of arguments will play a central role. Intuitively, the reduct w.r.t. E represents the SETAF that result from "accepting" E and rejecting what is defeated by E, while not deciding on the remaining arguments. To illustrate the idea, consider the following example:

Example 3.44. *Recall the following SETAF SF from Example 2.2.*



Consider the singleton $\{a\}$. If we view a as accepted, then b is rejected. This means that the attack from b to d can be disregarded. However, we also observe that c cannot be attacked anymore since attacking it requires both b and d, but b is rejected. Now consider $\{f\}$. Interpreting f as accepted renders g rejected. In order to attack h, only d is still required. Thus, if we let $E = \{a, f\}$, then we expect the SETAF reduct SF^E –with the intuitive meaning that a and f are set to true– to look as follows.



That is, in the reduct SF^E, we only need to consider arguments that are still undecided, i.e. all arguments neither in E nor attacked by E. In contrast to the AF-reduct [BBU20a], it might happen that some attacks are preserved that involve deleted arguments, i.e. the attack is only partially evaluated. In particular, if the arguments in the tail of an attack are "accepted" (i.e. in E), the attack can still play a role in attacking or defending. If the tail of an attack (T, h) is already attacked by E, we can disregard (T, h). By this, we get the following formal definition of the SETAF reduct.

Definition 3.45. *Given a SETAF $SF = (A, R)$ and $E \subseteq A$, the E-*reduct of SF *is the SETAF $SF^E = (A', R')$, with*

$$\begin{aligned} A' &= A \setminus E_R^{\oplus} \\ R' &= \{(T \setminus E, h) \mid (T, h) \in R, T \cap E_R^{+} = \emptyset, T \not\subseteq E, h \in A'\} \end{aligned}$$

Thereby, the condition $T \cap E_R^+ = \emptyset$ captures cases like the attack $(\{b, d\}, c)$ from our example: b is attacked by E, and thus, the whole attack gets removed. The reason why we take $(T \setminus E, h)$ as our novel attacks is the partial evaluation as in the attack $(\{d, f\}, h)$ after setting f to true: when additionally accepting d, we "activate" the attack against h.

Example 3.46. *Given the SETAF SF from Example 3.44 as well as $E = \{a, f\}$ as before, the reduct SF^E is the SETAF depicted above, i.e. $SF^E = \{A', R'\}$ with $A' = \{c, d, e, h\}$ and $R' = \{(d, e), (e, d), (d, h)\}$.*

We start our formal investigation of the reduct with a technical lemma to settle some basic properties.

Lemma 3.47. *Given a SETAF $SF = (A, R)$ and two disjoint sets $E, E' \subseteq A$. Let $SF^E = (A', R')$.*

1. *If there is no $S \subseteq A$ s.t. $S \mapsto_R E'$, then the same is true in SF^E.*
2. *Assume E does not attack $E' \in cf(SF)$. Then, E defends E' in SF iff there is no $S' \subseteq A'$ s.t. $S' \mapsto_{R'} E'$.*
3. *Let $E \in cf(SF)$. If $E \cup E'$ does not attack E in SF and $E' \subseteq A'$, with $E' \in cf\left(SF^E\right)$ then $E \cup E' \in cf(SF)$.*
4. *Let $E \cup E' \in cf(SF)$. If $E' \mapsto_{R'} a$, then $E \cup E' \mapsto_R a$.*
5. *If $E \cup E' \in cf(SF)$, then $SF^{E \cup E'} = \left(SF^E\right)^{E'}$.*

Proof.

1. This is clear since SF^E contains (strictly) less attacks than SF.
2. ($\Rightarrow$) Assume E defends E' in SF. Now suppose there is some attacker in the reduct, i.e. $S' \mapsto_{R'} e'$ for some $e' \in E'$. By definition, there is some $T' \subseteq A \setminus E^{\oplus}$ with $(T', e') \in R'$. By the definition of SF^E, $T' = T \cap A'$ for some $(T, e') \in R$. Now consider an arbitrary $(T, e') \in R$. Since E defends E', $E \mapsto_R T$. Again by definition of SF^E, (T, e') is removed since $T \cap E_R^+ \neq \emptyset$. Hence in R' there is no attack of the form (T', e') with $T' \subseteq T$, contradiction.

   ($\Leftarrow$) Now suppose E does not defend E'. There is thus some $(T, e') \in R$ and E does not attack T, i.e. $T \cap E_R^+ = \emptyset$. Suppose $T \setminus E = \emptyset$. Then $T \subseteq E$ contradicting that E does not attack E'. Thus, $T \setminus E \neq \emptyset$. Since E does not attack E' and $E \cap E' = \emptyset$, we have $e' \in A'$ for each $e' \in E'$. Therefore, in R' we find the attack (T', e') with $T' = T \cap A' \neq \emptyset$, $e' \in A'$, and $T \cap E_R^+ = \emptyset$.
3. We have to show that $E \cup E'$ does not attack E'. Suppose the contrary, i.e. let $T \subseteq E \cup E'$ with $(T, e') \in R$ for some $e' \in E'$. Since E does not attack E or E', $T \cap E_R^+ = \emptyset$. The case $T \subseteq E$ is impossible. Thus, (T, e') induces some attack $(T \setminus E, e')$ in SF^E. We infer $T \setminus E \subseteq E'$ implying $E' \notin cf\left(SF^E\right)$, contradiction.
4. If $E' \mapsto_{R'} a$, then $(T', a) \in R'$ for some $T' \subseteq E'$. Hence $(T, a) \in R$ for some T with $T \setminus E = T'$. The claim follows due to $T \subseteq T' \cup E \subseteq E' \cup E$.
5. We first show that $A(SF^{E \cup E'} = A((SF^E)^{E'})$.

   ($\subseteq$) Let $a \in SF^{E \cup E'}$. Then $a \notin E \cup E'$ and $E \cup E'$ does not attack a. We infer $a \in A\left(SF^E\right)$. Now if $E' \mapsto_{R'} a$, then $E \cup E' \mapsto_R a$ by item 4. Thus $a \in A((SF^E)^{E'})$.

($\supseteq$) Let $a \in A((SF^E)^{E'})$. Hence $a \notin E \cup E'$ and E' does not attack a in SF^E. Assume $(T, a) \in R$ with $T \subseteq E \cup E'$. Since E' does not attack a in SF^E, there cannot be an attack of the form $(T \setminus E, a) \in R'$ satisfying $T \setminus E \neq \emptyset$ and $T \cap E_R^+ = \emptyset$. However, T satisfies $T \cap E_R^+ = \emptyset$ since $E \cup E' \in cf(SF)$. We thus infer $T \setminus E = \emptyset$. This yields $E \mapsto_R a$ contradicting $a \in A((SF^E)^{E'})$.

It remains to show that $R(SF^{E \cup E'}) = R((SF^E)^{E'})$. Let $(T', h) \in R(SF^{E \cup E'})$, this means $T' = T \setminus (E \cup E')$ for some $(T, h) \in R$ and $T \cap (E \cup E')_R^+ = \emptyset$. From $T \cap (E \cup E')_R^+ = \emptyset$ follows $T \cap E_R^+ = \emptyset$, which means $(T \setminus E, h) \in R(SF^E)$. From 4. and the fact that $E \cup E' \not\mapsto_R h$ follows that $E' \not\mapsto_{R'} h$. Likewise, since $E \cup E' \not\mapsto_R t$ for each $t \in T \setminus E$ we get $E_{R'}'^+ \cap T \setminus E = \emptyset$, which means $(T', h) \in R((SF^E)^{E'})$.

Now let $(T', h) \in R((SF^E)^{E'})$. Again there is $(T, h) \in R$ s.t. $T' = T \setminus (E \cup E')$. Clearly $E \cup E' \not\mapsto_R t$ for each $t \in T$, and $E \cup E' \not\mapsto_R h$. Hence, we get $(T', h) \in R(SF^{E \cup E'})$. □

3.2.2 The Modularization Property

Having established the basic properties of the SETAF reduct, we are now ready to introduce the modularization property [BBU20a].

Principle 3.48 (Modularization)**.** *A semantics σ satisfies* modularization *if for all SETAFs SF, for every $E \in \sigma(SF)$ and $E' \in \sigma(SF^E)$, we have $E \cup E' \in \sigma(SF)$.*

Modularization allows us to build extensions iteratively. After finding such a set $E \subseteq A$ we can efficiently compute its reduct SF^E and pause before computing an extension E' for the reduct in order to obtain a larger extension $E \cup E'$ for SF. Hence, this first step can be seen as an intermediate result that enables us to reduce the computational effort of finding extensions in SF, as the arguments whose status is already determined by accepting E do not have to be considered again. Instead, we can reason on the reduct SF^E (see Section 3.5). In the following, we establish the modularization property for admissible and complete semantics.

Theorem 3.49. *Let SF be a SETAF, $\sigma \in \{adm, com\}$ and $E \in \sigma(SF)$.*

1. *If $E' \in \sigma(SF^E)$, then $E \cup E' \in \sigma(SF)$.*
2. *If $E \cap E' = \emptyset$ and $E \cup E' \in \sigma(SF)$, then $E' \in \sigma(SF^E)$.*

Proof. (for $\sigma = adm$) Let $SF^E = (A', R')$.

1) Since E is admissible and $E' \subseteq A'$, E' does not attack E. By Lemma 3.47, item 3, $E \cup E' \in cf(SF)$. Now assume $S \mapsto_R E \cup E'$. If $S \mapsto_R E$, then $E \mapsto_R S$ by admissibility of E. If $S \mapsto_R E'$, there is $T \subseteq S$ s.t. $(T, e') \in R$ for some $e' \in E'$. In case $E \mapsto_R T$, we are done. Otherwise, $(T \setminus E, e') \in R'$ and by admissibility of E' in SF^E, $E' \mapsto_{R'} T \setminus E$. By Lemma 3.47, item 4, $E \cup E' \mapsto_R T \setminus E$.

2) Now assume $E \cup E' \in adm(SF)$. We see $E' \in cf\left(SF^E\right)$ as follows: If $(T', e') \in R'$ for $T' \subseteq E'$ and $e' \in E'$, then there is some $(T, e') \in R$ with $T' = T \setminus E$. Hence $E \cup E' \mapsto_R E'$, contradiction. Now assume E' is not admissible in SF^E, i.e. there is $(T', e') \in R'$ with $e' \in E'$ and E' does not counterattack T' in SF^E. Then there is some $(T, e') \in R$ with $T' = T \setminus E$ and $T \cap E_R^+ = \emptyset$. By admissibility of $E \cup E'$, $E \cup E' \mapsto_R T$, say $(T^*, t) \in R$, $T^* \subseteq E \cup E'$ and $t \in T$. Since $E \cup E'$ is conflict-free, $T^* \cap E_R^+ = \emptyset$ and thus we either have a) $T^* \subseteq E$, contradicting $T \cap E_R^+ = \emptyset$, or b) $(T^* \setminus E, t) \in R'$ and $t \in T'$, i.e. E' counterattacks T' in SF^E contradicting the above assumption.

For *com* semantics we utilize the results for *adm*:

1) We have $E \cup E' \in adm(SF)$. Moreover, E' is complete, i.e. $(SF^E)^{E'}$ does not contain unattacked arguments in the reduct SF^E (see Proposition 3.50). Lemma 3.47, item 5, implies that $SF^{E \cup E'}$ does not contain unattacked arguments, either. Hence $E \cup E' \in com(SF)$.

2) Given $E \cup E' \in com(SF)$ we have $E' \in adm(SF^E)$, as established. Regarding completeness, we again use the fact that $SF^{E \cup E'} = (SF^E)^{E'}$ does not contain unattacked arguments. □

Note that the modularization property also holds for *stb*, *pref*, and *sem* semantics. However, the only admissible set in the reduct w.r.t. a stable/preferred/semi-stable extension is the empty set, rendering the property trivial. The exact relation is captured by the following alternative characterizations of the semantics under our consideration.

Proposition 3.50. *Let $SF = (A, R)$ be a SETAF, $E \in cf(SF)$ and $SF^E = (A', R')$.*

1. *$E \in stb(SF)$ iff $SF^E = (\emptyset, \emptyset)$,*
2. *$E \in adm(SF)$ iff $S \mapsto_R E$ implies $S \setminus E \not\subseteq A'$,*
3. *$E \in pref(SF)$ iff $E \in adm(SF)$ and $adm(SF^E) = \{\emptyset\}$,*
4. *$E \in com(SF)$ iff $E \in adm(SF)$ and $grd(SF^E) = \{\emptyset\}$,*
5. *$E \in sem(SF)$ iff $E \in pref(SF)$ and there is no $E' \in pref(SF)$ s.t. $A(SF^{E'}) \subsetneq A(SF^E)$.*

Proof. The characterizations for *stb* and *adm* are straightforward and *pref* is due to the modularization property of *adm*. For $com(SF)$ we apply Lemma 3.47, item 2, to each singleton E' occurring in SF^E: assume towards contradiction E is complete in SF and there is some $a \in A'$ such that a is unattacked in SF' (and, hence, a is in the grounded extension of SF^E). As $a \in A'$ we know $E \not\mapsto_R \{a\}$. But then we can apply Lemma 3.47, item 2, and get that E defends $\{a\}$ in SF, contradicting $E \in com(SF)$. For *sem* recall that range-maximal preferred extensions are semi-stable. □

From the characterization of complete semantics provided in Proposition 3.50 we infer that for any SETAF SF the complete extensions $E \in com(SF)$ satisfy $grd(SF^E) = \{\emptyset\}$ implying modularization for *grd*. Moreover, as the grounded extension G is the least complete extension, we can utilize modularization of *adm* and obtain G by the following procedure: (1) add the set of unattacked arguments U into G, (2) repeat step (1) on SF^U until there are no unattacked arguments.

We have two cases left to discuss, namely *eager* and *ideal* semantics. Both satisfy the modularization property, because they only admit the empty set as admissible extension in their corresponding reduct SF^E (as in the case of e.g. *sem* semantics). Since this is however not as easy to see, we will give the necessary proofs in detail here. We follow the proof technique of the AF case [FU21]. First we show that the property formalized in Theorem 3.49 also holds for semi-stable semantics. This will be useful later since *eager* semantics build upon semi-stable extensions.

Proposition 3.51. *Let SF be a SETAF and let $E \in sem(SF)$. Suppose $E = E' \cup E''$ with $E' \cap E'' = \emptyset$ for some $E' \in adm(SF)$. Then $E'' \in sem(SF^{E'})$.*

Proof. We already know $E'' \in adm(SF^{E'})$ since $sem(SF) \subseteq adm(SF)$. Now assume E'' is not semi-stable in $F^{E'}$. Then there is some admissible $S \in adm\left(SF^{E'}\right)$ with $(E'')^{\oplus} \subsetneq S^{\oplus}$. Since E'' and S occur in $SF^{E'}$, this immediately yields $E^{\oplus} = (E' \cup E'')^{\oplus} \subsetneq (E' \cup S)^{\oplus}$. Since by modularization we have $E' \cup S \in adm(SF)$, we infer $E \notin sem(SF)$, a contradiction. □

Next we show that the reduct w.r.t. some eager extension admits only $\emptyset$ as admissible set.

Proposition 3.52. *If $E \in eager(SF)$, then $eager\left(SF^E\right) = \{\emptyset\}$.*

Proof. Let $SF = (A, R)$ be a SETAF and let $E \in eager(SF)$. Consider the reduct SF^E and assume $E' \in eager(SF^E)$ is not empty. Let S be a semi-stable extension of SF. By definition of *eager*, we have that $E \subseteq S$. Our goal is to show $E' \subseteq S$ as well, yielding a contradiction since $E \cup E' \in com(SF)$ by modularization of *com*; since S is arbitrary, the eager extension of SF must then contain $E \cup E'$. To this end note that $S = E \cup S'$ for $E \in adm(SF)$ and some set S' of arguments. By the above Proposition 3.51, $S' \in sem\left(SF^E\right)$ and hence $E' \subseteq S' \subseteq S$ and we are done. □

Since $\emptyset$ is thus the only candidate extension in the reduct SF^E, we immediately get satisfaction of the modularization property.

Corollary 3.53. *The eager semantics satisfies modularization.*

In order to lift the above proof technique to *ideal* as well it suffices to note the following adjustment to Proposition 3.51.

Proposition 3.54. *Let SF be a SETAF and let $E \in pref(SF)$. Suppose $E = E' \cup E''$ with $E' \cap E'' = \emptyset$ for some $E' \in adm(SF)$. Then $E'' \in pref(SF^{E'})$.*

Proof. According to Proposition 3.50, we have that $E \in pref(SF)$ if and only if $E \in adm(SF)$ and SF^E does not possess any admissible argument. We already know admissibility of E'' in $SF^{E'}$. Moreover, $SF^E = (SF^{E'})^{E''}$ does not contain admissible arguments; thus we are done. □

This yields the same behavior for *ideal* as well. First, we again infer that the reduct does not tolerate any non-empty extension.

Proposition 3.55. *If $E \in ideal(F)$, then $ideal\left(F^E\right) = \{\emptyset\}$.*

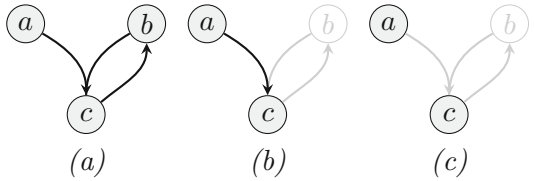
Proof. We reason as in the proof of Proposition 3.52 with $S \in pref(SF)$ instead of $S \in sem(SF)$. □

Corollary 3.56. *The ideal semantics satisfies modularization.*

3.3 Directionality and Non-Interference

In this section we discuss the principles of directionality and non-interference. Intuitively, these principles give information about the behavior of separate parts of a framework. Beside being informative regarding the behavior of semantics, this principles also have computational implications. In order to formalize this separation-property, we start of with the notion of unattacked sets of arguments[8]. For directionality [BG07] we have to carefully consider this notion in order to obtain a natural generalization of the AF case preserving the intended meaning. A naive definition of unattacked sets will lead to nonsensical results: assume a set S is unattacked in a SETAF $SF = (A, R)$ whenever it is not attacked from "outside", i.e. if the condition $A \setminus S \not\mapsto_R S$ holds.

Example 3.57. *Consider now the following SETAF (a) and its projections (b), (c) w.r.t. the "unattacked" set $S = \{a, c\}$.*



[8] While in the previous section we used "unattacked arguments", i.e. arguments that are not the head of any attack, unattacked *sets of arguments* allow for attacks within the set.

Note that $\{a, c\}$ is stable in (a). If we now consider the projection $SF{\downarrow_S}$—see (b)—we find that $\{a, c\}$ is not stable, falsifying directionality. However, one might argue that this is due to the credulous nature of our projection-notion. We could easily consider a different proper generalization of the projection, namely $SF{\downarrow^*_S} = (S, \{(T, h) \mid (T, h) \in R, T \cup \{h\} \subseteq S\})$. In this more skeptical version we delete attacks if any of the arguments in the tail are not in the projected set—see (c). However, we still cannot obtain the desired results: in (a) we find $\{a\}$ to be the unique grounded extension, while in (c) $\{a, c\}$ is grounded, again falsifying directionality. As for the directionality principle we do not want to add additional arguments or attacks and we exhausted all possible reasonable projection notions for this small example, we conclude that the underlying definition of unattacked *sets was improper. We therefore suggest a different definition—and at the same time suggest to think of these sets rather as "uninfluenced" than "unattacked". In AFs, clearly both notions coincide. However, we still argue that the concept of "influence" captures the true nature of directionality in a more intuitive and precise manner. Moreover, note that in the case of uninfluenced sets both notions of projection coincide, as well as the notion of* restriction *(see Definition 3.72) for arbitrary sets $D \subseteq A \setminus S$.*

Towards the formal definition of influence, we utilize the notion of the *primal graph* of a SETAF [DKW21a], cf. Definition 2.10.

Definition 3.58 (Influence)**.** *Let $SF = (A, R)$ be a SETAF. An argument $a \in A$* influences *$b \in A$ if there is a directed path from a to b in primal(SF). A set $U \subseteq A$ is* uninfluenced *in SF if no $a \in A \setminus U$ influences any $b \in U$. We denote the set of uninfluenced sets by $US(SF)$.*

Utilizing this notion, we can properly generalize directionality [BG07].

Principle 3.59 (Directionality)**.** *A semantics σ satisfies* directionality *if for all SETAFs SF and every $U \in US(SF)$ it holds $\sigma(SF{\downarrow_U}) = \{E \cap U \mid E \in \sigma(SF)\}$.*

Moreover, weaker versions of directionality have been proposed which require only a subset relation [vdTV17]:

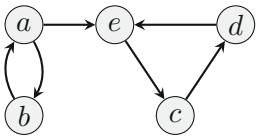
Principle 3.60 (Semi Directionality)**.** *A semantics σ satisfies* semi directionality *if for all SETAFs SF and every $U \in US(SF)$ it holds $\sigma(SF{\downarrow_U}) \subseteq \{E \cap U \mid E \in \sigma(SF)\}$.*

Principle 3.61 (Weak Directionality)**.** *A semantics σ satisfies* weak directionality *if for all SETAFs SF and every $U \in US(SF)$ it holds $\sigma(SF{\downarrow_U}) \supseteq \{E \cap U \mid E \in \sigma(SF)\}$.*

We will revisit directionality at the end of the next section, as we can utilize SCC-recursiveness to show that *grd*, *com*, and *pref* satisfy directionality. In contrast, this is not possible for *eager* and *ideal* semantics, so we investigate these two cases directly.

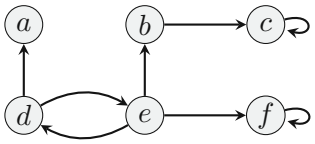
Let us start with *eager* semantics. As the following examples show, *eager* satisfies neither semi directionality nor weak directionality.

Example 3.62. *Let $F = (A, R)$ be the following AF [vdTV17, Figure 3]:*



Let $U = \{a, b\}$. We have $sem(F) = \{\{a, c\}\}$ and thus $eager(F) = \{\{a, c\}\}$ as well. Thus $\{E \cap U \mid E \in eager(F)\} = \{\{a\}\}$. On the other hand, $sem(SF{\downarrow_U}) = \{\{a\}, \{b\}\}$ and thus, $eager(SF{\downarrow_U}) = \{\emptyset\}$, i.e. weak directionality is violated.

Example 3.63. *Now let Let $F = (A, R)$ be the following AF [vdTV17, Figure 8]:*



Let $U = \{d, e, f\}$. We have $sem(F) = \{\{a, e\}, \{d, b\}\}$ and thus $eager(F) = \{\emptyset\}$. On the other hand, $sem(SF{\downarrow_U}) = \{\{e\}\}$ and thus, $eager(SF{\downarrow_U}) = \{\{e\}\}$. Hence semi directionality is violated.

Now let us turn to ideal semantics. We show that directionality is satisfied. Our proof follows the technique from the AF case [BG07]. The required structural properties also hold for SETAFs. Therefore, we only require minor adjustments to reason analogously in our setting.

Lemma 3.64. *Let $SF = (A, R)$ be a SETAF. The the unique ideal extension S satisfies*

$$S = \bigcup_{E \in adm(SF) : \forall P \in pref(SF) : E \subseteq P} E$$

Proof. We let

$$adm_{\subseteq pref}(SF) = \{E \in adm(SF) \mid \forall P \in pref(SF) : E \subseteq P\}.$$

We need to show that (a) S is conflict-free in SF, (b) every argument $a \in S$ is acceptable w.r.t. S, and (c) there is no larger set $S' \supset S$ that satisfies (a) and (b) and is a subset of every preferred extensions of SF. (a) is clear, because if there was a conflict caused by an attack $(T, h) \in R$ with $T \cup \{h\} \subseteq S$, this would mean two sets $E, E' \in adm_{\subseteq pref}(SF)$ attacked each other, which would mean a preferred extension is conflicting, a contradiction. (b) follows from the fact that for all $a \in S$ there is an $E \subseteq S$ with $a \in E, E \in adm(SF)$. (c) is clear since (a) and (b) characterize admissibility—if there was such a larger admissible set $S' \supset S$ with $S' \in adm_{\subseteq pref}(SF)$ by definition we would have $S' \subseteq S$, a contradiction. □

This auxiliary lemma is a convenient characterization of *ideal* in order to infer directionality as follows.

Proposition 3.65. *The semantics ideal satisfies directionality.*

Proof. Let $SF = (A, R)$ be a SETAF and suppose $U \in US(SF)$. We have to show $ideal(SF{\downarrow_U}) = \{E \cap U \mid E \in ideal(SF)\}$. Due to Lemma 3.64 it suffices to show

$$\bigcup_{E \in adm(SF):\forall P \in pref(SF):E \subseteq P} E \cap U = \bigcup_{E \in adm(SF{\downarrow_U}):\forall P \in pref(SF{\downarrow_U}):E \subseteq P} E$$

($\subseteq$) Let E be an arbitrary set in $adm(SF)$. We show the claim for this particular set and thus, the same holds for the union over all extensions in $adm(SF)$ as well. Due to directionality of *adm* semantics, $E \in adm(SF)$ implies $E \cap U \in adm(SF{\downarrow_U})$. Therefore, we have to show that $E \cap U$ is a subset of each preferred extension in $SF{\downarrow_U}$ and thus, $E \cap U$ is part of the union of the right-hand side.

Now, for each $P \in pref(SF)$ we have $E \cap U \subseteq P \cap U$, i.e.

$$\forall P \in pref(SF) : E \cap U \subseteq P \cap U.$$

By directionality of *pref* semantics, $\{P \cap U \mid P \in pref(SF)\} = pref(SF{\downarrow_U})$. This is means $\forall P \in pref(SF{\downarrow_U}) : E \cap U \subseteq P$ which we had to show.

($\supseteq$) Now let $E \in adm(SF{\downarrow_U})$. By definition of admissibility, it is clear that $E \in adm(SF)$ follows. For each $P \in pref(SF{\downarrow_U})$ it follows that $E \subseteq P$, i.e.

$$\forall P \in pref(SF{\downarrow_U}) : E \subseteq P.$$

Again by directionality, we turn this into $\forall P \in pref(SF) : E \subseteq P \cap U \subseteq P$ which proves the claim. □

Similarly, we generalize *non-interference* [CCD12], which has an even stronger requirement. $U \subseteq A$ is *isolated* in $SF = (A, R)$, if U is uninfluenced and $A \setminus U$ is uninfluenced, i.e. there are no edges in $primal(SF)$ between U and $A \setminus U$.

Principle 3.66 (Non-interference)**.** *A semantics σ satisfies* non-interference *iff for all SETAFs SF and all isolated $S \subseteq A(SF)$, it holds $\sigma(SF{\downarrow_U}) = \{E \cap U \mid E \in \sigma(SF)\}$.*

Clearly, directionality implies non-interference. It is easy to see from the respective definitions that also naive, semi-stable, ideal, eager, and stage semantics satisfy non-interference.

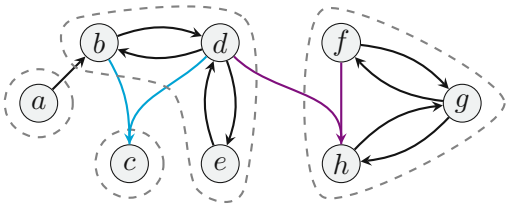
3.4 SCC-Recursiveness

In graph theory, the notion of a strongly connected component (SCC) is a widely known concept. An SCC consists of a set S of nodes s.t. for any $a, b \in S$ there is a directed path from a to b within the given graph. SCC-recursiveness [BGG05] formalizes the intuition that the acceptance status of an argument depends only on its ancestors—i.e., the arguments that feature a directed path to the argument in question. If some semantics satisfies SCC-recursiveness, one can construct all SCCs of a given graph and then compute resp. verify extensions step-wise, by working along the SCCs. This provides theoretical insights as it formalizes the independence of arguments of their child SCCs, but also provides us with computational advantages as we will see in Section 3.5.

In Section 3.3 we considered the concept of *influence*. In a nutshell, an argument a "influences" an argument b in a SETAF SF if there is a directed path from a to b in $primal(SF)$. It is therefore reasonable to investigate SCCs with this idea in mind. In particular, our definition of SCCs captures the equivalence classes w.r.t. the influence relation.

Definition 3.67 (SCCs)**.** *Let SF be a SETAF. By $SCCs(SF)$ we denote the set of strongly connected components of SF, which we define as the sets of arguments contained in the strongly connected components of $primal(SF)$.*

Example 3.68. *Recall our SETAF from before.*



In this SETAF, we have the four SCCs $\{a\}, \{b, d, e\}, \{c\}$, and $\{f, g, h\}$, as depicted in dashed lines.

Analogously to [BGG05], we partition the arguments into defeated, provisionally defeated and undefeated ones. Intuitively, accepting a defeated argument would lead to a conflict, the provisionally defeated cannot be defended and will therefore be rejected (while not being irrelevant for defense of other arguments), and the undefeated form the candidates for extensions. We obtain the following formal definition of the sets we just described.

Definition 3.69. *Let $SF = (A, R)$ be a SETAF. Moreover, let $E \subseteq A$ be a set of arguments and $S \in SCCs(SF)$ be an SCC. We define the set of defeated arguments $D_{SF}(S, E)$, provisionally defeated arguments $P_{SF}(S, E)$, and undefeated arguments $U_{SF}(S, E)$ w.r.t.*

S, E as

$$D_{SF}(S,E) = \{a \in S \mid E \setminus S \mapsto_R a\},$$
$$P_{SF}(S,E) = \{a \in S \mid A \setminus (S \cup E^+) \mapsto_R a\} \setminus D_{SF}(S,E),$$
$$U_{SF}(S,E) = S \setminus (D_{SF}(S,E) \cup P_{SF}(S,E)).$$

Moreover, we set $UP_{SF}(S,E) = U_{SF}(S,E) \cup P_{SF}(S,E)$.

It is important to note that all these sets are calculated w.r.t. a given set candidate E, i.e. the purpose is to verify whether E is some σ-extension.

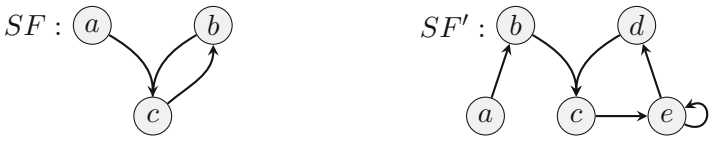
Example 3.70. *Recall the SETAF from above. Let $S = \{b,d,e\}$ be the SCC under consideration.*

Take the admissible extension $E = \{a, e\}$. We have that $D_{SF}(S,E) = \{b\}$ since the argument a from the parent SCC $\{a\}$ defeats b; observe that $d \notin D_{SF}(S,E)$ since d is only defeated by e which is part of the given SCC S. Moreover, $P_{SF}(S,E) = \emptyset$ and hence $U_{SF}(S,E) = \{d, e\}$.

Consider now $E' = \{d\}$. Then $D_{SF}(S,E') = \emptyset$ because no argument within S is defeated from an argument in E' occurring in a parent SCC. However, $P_{SF}(S,E') = \{b\}$ reflecting that b cannot be defended (for this we would have to defeat a, but from within the given SCC S this is impossible). Therefore, $U_{SF}(S,E') = \{b,d,e\} = S$.

In order to formalize SCC-recursiveness, we need the notion of the *restriction*. It will be convenient in order to evaluate our given extension SCC-wise, since in each step we can remove the defeated arguments $D_{SF}(S,E)$ and thus restrict our attention to $U_{SF}(S,E)$. For classical Dung-AFs, the restriction coincides with the *projection* from Definition 2.8. However, in the following we will argue that the projection does not capture the intricacies of this process. Ultimately, we will see that for a reasonable restriction we need semantic tools that are similar to the reduct SF^E. For that, we revisit Example 3.57.

Example 3.71. *Consider the following SETAFs SF and SF'.*



Assume we accept the argument a in SF. Now for the remaining SCC $\{b,c\}$ the projection $SF{\downarrow}_{\{b,c\}}$ contains the attacks (b,c) and (c,b), as one might expect.

Regarding SF', assume we accept a and therefore reject b. The projection $SF'{\downarrow}_{\{c,d,e\}}$ yields an odd cycle where none of the remaining arguments c, d, e can be accepted. However, as b is defeated, the attack $(\{b,d\}, c)$ is counter-attacked and thus, c is defended. Hence we would expect c to be acceptable in the restriction w.r.t. a.

One might argue that this notion of projection is therefore too credulous, i.e., attacks survive that should be discarded. Recall Example 3.57 where we defined the alternative projection

$$SF{\downarrow}^{*}_{S} = (S, \{(T,h) \mid (T,h) \in R,\ T \cup \{h\} \subseteq S\}).$$

Now, one can check that we get the expected results in SF'. However, $SF{\downarrow}^{*}_{\{b,c\}}$ only features the attack (c,b), which incorrectly suggests that we cannot accept b.

We solve this problem by adapting the notion of a *restriction* such that both cases are handled appropriately. We keep track of a set of rejected arguments and discard attacks once an argument in its tail is discarded—these attacks are irrelevant to the further evaluation of the SETAF. This leads to the following notion,

Definition 3.72 (Restriction)**.** *Let $SF = (A,R)$ be a SETAF and let $S, D \subseteq A$. We define the* restriction *of SF w.r.t. S and D as the SETAF $SF{\Downarrow}^{D}_{S} = (S', R')$ where*

$$S' = (A \cap S) \setminus D$$
$$R' = \{(T \cap S', h) \mid (T,h) \in R,\ h \in S',\ T \cap D = \emptyset,\ T \cap S' \neq \emptyset\}.$$

Let us work through the conditions:

- D will be $D_{SF}(S,E)$ later on, i.e. the set of defeated arguments; thus $S =' A \cap S \setminus D$ is the set of non-defeated arguments in the current SCC S.
- The set R' of attacks reduces the tail T of a given attack to the set S' of consideration, i.e. (T,h) is reduced to $(T \cap S', h)$, but only if:
  - the attacked argument h belongs to S',
  - none of the arguments in the tail are defeated, $T \cap D = \emptyset$, and
  - at least one argument in the tail belongs to the current set S', $T \cap S' \neq \emptyset$.

Example 3.73. *The restriction handles both cases of Example 3.71 according to our intuition.*

- *The SETAF $SF{\Downarrow}^{\{a\}}_{\{b,c\}}$ contains b and c, and as we accepted a, i.e. the part tail of $(\{a,b\},c)$ outside $\{b,c\}$, the attack (b,c) is kept.*
- *The restriction $SF'{\Downarrow}^{\{b\}}_{\{c,d,e\}}$ contains the attacks (c,e), (e,d), and (e,e); as $b \in D = \{b\}$ the tail of $(\{b,d\},c)$ is already defeated and we therefore do not include (d,c).*

We want to emphasize that this example illustrates how the notion of projection is akin to the SETAF-reduct: indeed, constructing $SF{\Downarrow}^{D}_{S}$ consists in projecting to a certain set of arguments and then i) removing attacks where defeated arguments are involved as well as ii) partially evaluating the remaining tails. Formally, the connection is as follows.

Lemma 3.74. *Let $SF = (A, R)$ be a SETAF and let $E, S \subseteq A$. Then $SF{\Downarrow}_S^{(E\setminus S)^+} = SF^{(E\setminus S)}{\downarrow_S}$.*

Proof. We have $A(SF{\Downarrow}_S^{(E\setminus S)^+}) = A(SF^{(E\setminus S)}{\downarrow_S})$ because

$$\begin{aligned}(A \cap S) \setminus (E \setminus S)^+ &= (A \setminus (E \setminus S)^+) \cap S \\ &= (A \setminus ((E \setminus S)^+ \cup (E \setminus S))) \cap S \\ &= (A \setminus (E \setminus S)^{\oplus}) \cap S.\end{aligned}$$

Then it holds that $R(SF{\Downarrow}_S^{(E\setminus S)^+}) = R(SF^{(E\setminus S)}{\downarrow_S})$, as for some $(T, h) \in R(SF{\Downarrow}_S^{(E\setminus S)^+})$ with $h \in A'$ and $T \cap (E \setminus S)^+ = \emptyset$ we have $T \cap A' = \emptyset$ if and only if $T \not\subseteq (E \setminus S)$. The claim follows then from $(T \cap A', h) = (T \setminus (E \setminus S), h)$. □

Let us now formally introduce SCC-recursiveness [BGG05] as a SETAF principle. Extensions satisfying this property can be recursively characterized as follows: if the SETAF SF consists of a single SCC, the *base function* $\mathcal{BF}$ of the semantics yields the extensions. For SETAFs that consist of more SCCs, we apply the *generic selection function* $\mathcal{GF}$, where SF is evaluated separately on each SCC by means of our *restriction*, taking into account arguments that are defeated by previous SCCs.

Principle 3.75 (SCC-recursiveness)**.** *A semantics σ satisfies* SCC-recursiveness *if for all SETAFs $SF = (A, R)$, it holds that $\sigma(SF) = \mathcal{GF}(SF)$, where $\mathcal{GF}(SF) \subseteq 2^A$ is defined as follows: $E \subseteq A \in \mathcal{GF}(SF)$ if and only if*

- *if $|SCCs(SF)| = 1$, then $E \in \mathcal{BF}(SF)$;*
- *otherwise, $\forall S \in SCCs(SF)$ it holds that $E \cap S \in \mathcal{GF}\left(SF{\Downarrow}_{UP_{SF}(S,E)}^{(E\setminus S)^+}\right)$,*

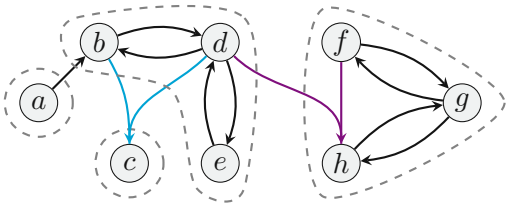
where $\mathcal{BF}$ is a function that maps a SETAF $SF = (A, R)$ with $|SCCs(SF)| = 1$ to a subset of 2^A.

In the following subsections we will investigate and refine SCC-recursiveness for the different semantics under our consideration. For the proofs we loosely follow the structure of [BGG05], incorporating our SETAF-specific notions.

3.4.1 Stable Semantics

We start with stable semantics, as this is the easiest case.

Example 3.76. *Recall Example 3.68.*



We use the base function

$$\mathcal{BF}(SF) = stb(SF).$$

Consider the stable extension $E = \{a, c, d, f\}$ *of* SF. *Let* $S = \{b, d, e\}$. *The projected SETAF* $SF{\Downarrow}_S^{(E\setminus S)^+}$ *is given as*

$$SF{\Downarrow}_S^{(E\setminus S)^+} = SF{\Downarrow}_S^{\{b\}} = (\{d, e\}, \{(d, e), (e, d)\}).$$

This projected SETAF consists of one SCC only, and we apply the base case of $\mathcal{GF}$, *i.e.*

$$\mathcal{GF}\left(SF{\Downarrow}_{UP_{SF}(S,E)}^{(E\setminus S)^+}\right) = \mathcal{BF}\left(SF{\Downarrow}_{UP_{SF}(S,E)}^{(E\setminus S)^+}\right) = stb\left(SF{\Downarrow}_{UP_{SF}(S,E)}^{(E\setminus S)^+}\right).$$

Since $E \cap S = \{d\}$ *is indeed a stable extension of* $SF{\Downarrow}_{UP_{SF}(S,E)}^{(E\setminus S)^+}$, *the required condition w.r.t. the SCC* S *is met.*

In this section we will show that this is no coincidence, i.e. *stb* satisfies SCC-recursiveness (with the base function *stb*). For the investigation of SCC-recursiveness in stable semantics we use the fact that there are no undecided arguments. Thus, in each step we do not have to keep track of as much information from previous SCCs. Formally, we obtain the following auxiliary lemma.

Lemma 3.77. *Let* SF *be a SETAF and* $E \in stb(SF)$, *then for all* $S \in SCCs(SF)$ *it holds* $P_{SF}(S, E) = \emptyset$.

Proof. Assume towards contradiction that for some SCC S there is an argument $a \in P_{SF}(S, E)$. Then, by definition there is an attack $(T, a) \in R(SF)$ such that $T \subseteq A(SF)\setminus S$ and $T \cap E^+ = \emptyset$. Moreover, $a \notin D_{SF}(S, E)$ by definition, i.e. $T \not\subseteq E$. But then there is some $t \in T$ such that neither $t \in E^+$ nor $t \in E$, which is a contradiction to the assumption that E is stable. □

We continue with the main technical underpinning for the SCC-recursive characterization of stable semantics. Intuitively, Proposition 3.78 states that an extension E is "globally" stable in SF if and only if for each of its SCCs S it is "locally" stable in $SF{\Downarrow}_{UP_{SF}(S,E)}^{(E\setminus S)^+}$.

Proposition 3.78. *Let* $SF = (A, R)$ *be a SETAF and let* $E \subseteq A$, *then* $E \in stb(SF)$ *if and only if* $\forall S \in SCCs(SF)$ *it holds* $(E \cap S) \in stb\left(SF{\Downarrow}_{UP_{SF}(S,E)}^{(E\setminus S)^+}\right)$.

Proof. Let $SF' = SF{\Downarrow}^{(E\setminus S)^+}_{UP_{SF}(S,E)}$ for an arbitrary SCC S. We start by assuming $E \in stb(SF)$. We need to show that $(E \cap S) \in stb(SF')$, i.e.:

1. $(E \cap S) \subseteq UP_{SF}(S, E)$,
2. $(E \cap S)$ is conflict-free in SF', and
3. $\forall a \in UP_{SF}(S, E)$ if $a \notin (E \cap S)$ then $(E \cap S)$ attacks a in SF'.

For condition 1. note that $(E \cap X) \cap D_{SF}(X, E) = \emptyset$ holds for any $X \subseteq A$, as otherwise E would not be conflict-free in SF. For condition 2., assume towards contradiction that there is some $(T, h) \in R(SF')$ such that $T \cup \{h\} \subseteq (E \cap S)$. This means there is some $(T', h) \in R$ with $T' \supseteq T$. But by construction we would have $T' \setminus T \subseteq E$, and therefore $T' \cup \{h\} \subseteq E$, a contradiction to conflict-freeness of E. For condition 3. we consider an arbitrary argument $a \in UP_{SF}(S, E) \setminus (E \cap S)$. Since $a \notin E$ and E is stable, there is an attack $(T, a) \in R$ with $T \subseteq E$. Moreover, as $a \in UP_{SF}(S, E)$, it holds $a \notin D_{SF}(S, E)$, i.e. in particular $T \not\subseteq (E \setminus S)$, or in other words $T \cap S \neq \emptyset$. This means by the definition of the restriction and since $T \cap E^+_R = \emptyset$ (otherwise E would not be conflict-free in SF) there is an attack $(T \cap S, a) \in R(SF')$ with $(T \cap S) \subseteq E$.

Now assume $\forall S \in SCCs(SF)$ it holds $(E \cap S) \in stb(SF{\Downarrow}^{(E\setminus S)^+}_{UP_{SF}(S,E)})$. We need to show $E \in stb(SF)$, i.e.

1. E is conflict-free in SF, and
2. E attacks all $a \in A \setminus E$ in SF.

For 1. assume towards contradiction there is some $(T, a) \in R$ such that $T \cup \{a\} \subseteq E$. Let S be the SCC containing a. Clearly $T \cup \{a\} \not\subseteq S$, as this violates our assumed conflict-freeness in $UP_{SF}(S, E)$. Moreover, we do not have $T \subseteq (A \setminus S)$, as this would mean $a \in D_{SF}(S, E)$. Hence, there is an attack $(T \cap S, a) \in R\left(SF{\Downarrow}^{(E\setminus S)^+}_{UP_{SF}(S,E)}\right)$ such that $(T \cap S) \cup \{a\} \subseteq E \cap S$, a contradiction. For condition 2. let us consider an arbitrary argument $a \in A \setminus E$ and let S be the SCC containing a. Then either (i) $a \in D_{SF}(S, E)$ or (ii) $a \in UP_{SF}(S, E)$. In case (i) we immediately get E attacks a. For case (ii), we have $a \notin (E \cap S)$, and by assumption a is attacked in S, i.e. there is an attack $(T, a) \in R\left(SF{\Downarrow}^{(E\setminus S)^+}_{UP_{SF}(S,E)}\right)$. By construction of the restriction, this means there is an attack $(T', a) \in R$ s.t. $T' \supseteq T$ and $T' \setminus T \subseteq E$. Hence, $T \subseteq E$, i.e. E attacks a in SF. □

This leads us to the characterization of stable extensions. As the base function is $stb(SF)$, the base case is immediate. The composite case follows from Proposition 3.78.

Theorem 3.79. *Stable semantics is SCC-recursive.*

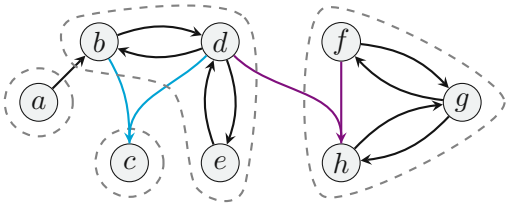
3.4.2 Admissible Sets

As already mentioned, when investigating stable semantics we can use the observation that each argument is either in E or defeated by E, i.e. stable extensions correspond to two-valued models of the AF. For admissibility-based semantics, we might also have "undecided" arguments, i.e. arguments which are not in the range $E^{\oplus}$ of the given extension. These arguments make handling the SCC-recursive procedure more involved.

To this end we add a second component to $\mathcal{GF}$ which intuitively collects all arguments C that can still be defended within the current SCC S. On the other hand, arguments which occur in the restriction $SF{\Downarrow}^{(E\setminus S)^+}_{UP_{SF}(S,E)}$ but not in C cannot be accepted anymore; however, we have to defend our extension against them. We account for this in Definition 3.88 by maintaining a set of candidate arguments C.

While this is all similar in spirit to the AF case, there is however another crucial observation we make. That is, the particular case of SETAFs gives rise to a novel scenario, where certain attacks are present in an SCC, but not applicable.

Example 3.80. *Recall our SETAF from before.*



This time, consider $S = \{f, g, h\}$ with given extension $E = \emptyset$. Then

$$D_{SF}(S,E) = \emptyset \qquad SF{\Downarrow}^{(E\setminus S)^+}_{UP_{SF}(S,E)} = SF{\Downarrow}^{\emptyset}_{S} = (A', R')$$
$$P_{SF}(S,E) = \emptyset \qquad A' = \{f, g, h\}$$
$$U_{SF}(S,E) = \{f, g, h\} \qquad R' = \{(f,h), (g,h), (h,g), (f,g), (g,f)\}$$

We now observe that although there is an attack from f to h in $SF{\Downarrow}^{\emptyset}_{S}$, the argument h can actually not be defeated by f, because this would require d to be present in our extension. Note however that we cannot delete the attack (f,h), as this would mean we could accept h—without defending h against the attack from $\{d, f\}$.

Consequently, we will keep track of these attacks that have to be considered for defense, but cannot themselves be used to defeat an argument. We will call these attacks *mitigated*.

Definition 3.81 (Mitigated Attacks)**.** *Let $SF = (A, R)$ be a SETAF. Moreover, let $E \subseteq A$ and $S \in SCCs(SF)$. The set $M_{SF}(S,E)$ of* mitigated attacks *is given as*

$$M_{SF}(S,E) = \{(T,h) \in R \left(SF{\Downarrow}^{(E\setminus S)^+}_{UP_{SF}(S,E)}\right) \mid \forall (T',h) \in R : T' \supseteq T \Rightarrow (T' \setminus T) \not\subseteq E\}.$$

One can check that with this definition, for Example 3.80 indeed the attack towards h is identified in the SCC $S = \{f, g, h\}$, and the resulting attack (f, h) in $SF{\Downarrow}_S^{\emptyset}$ is mitigated (in particular, we have $M_{SF}(S, \emptyset) = \{(f, h)\}$). The intuition behind the condition

$$\forall (T', h) \in R : T' \supseteq T \Rightarrow (T' \setminus T) \not\subseteq E$$

is that (T, h) might stem from some modified attack (T', h) in the SETAF with $T \subseteq T'$: the attack (T', h) is suitably modified when computing the restriction $SF{\Downarrow}_{UP_{SF}(S,E)}^{(E \setminus S)^+}$ and yields (T, h). Then, $T' \setminus T \not\subseteq E$ ensures that the attack is not active in E, independent of the choice of arguments within the SCC S. Intuitively, this accounts for a scenario where an attack (T, h) that appears in a sub-framework generated from projecting to an SCC has two or more possible origins: at least one attack $(T', h) \in R$ with $T' \supseteq T$ where some argument $t \in T' \setminus T$ is not in $E^{\oplus}$ (i.e., causing the resulting (T, h) to be mitigated), and at least one attack $(T'', h) \in R$ with $T'' \supseteq T$ where $T'' \setminus T \subseteq E$ (i.e., causing the resulting (T, h) to be non-mitigated). In this case the non-mitigated interpretation "overrides" the mitigated interpretation, as this attack can clearly be used to defend other arguments.

To account for the novel scenarios arising from the context of mitigated attacks we adapt the notion of acceptance. We have to assure that the "counter-attacks" used for defense are not mitigated. Recall that in addition our generic selection function also stores some set C of acceptable arguments, with the consequences mentioned above. Putting all of this together yields the following notions:

- If M is the set of mitigated attacks, then some extension E defends $a \in A$ if for each arbitrary attacker $(T, a) \in R$ there is some non-mitigated counter-attack $(X, t) \in R \setminus M$ with $X \subseteq E$ and $t \in T$, i.e. E counters the attack without relying on any mitigated attack;
- Each extension E must be a subset of the set C of acceptable arguments.

Formally, we obtain the following semantics considering C, M.

Definition 3.82 (Semantics Considering C, M)**.** *Let $SF = (A, R)$ be a SETAF, and let $E, C \subseteq A$ and $M \subseteq R$. We say that*

- *E is* conflict-free in C considering M*, denoted by $E \in cf(SF, C, M)$, if $E \subseteq C$ and there is no $(T, h) \in R \setminus M$ s.t. $T \cup \{h\} \subseteq E$;*
- *an argument $a \in A \setminus C$ is* acceptable considering M w.r.t. E *if for all $(T, a) \in R$ there is $(X, t) \in R \setminus M$ s.t. $X \subseteq E$ and $t \in T$;*
- *E is* admissible in C considering M*, denoted by $E \in adm(SF, C, M)$, if $E \subseteq C$, $E \in cf(SF, C, M)$, and each $a \in E$ is acceptable considering M w.r.t. E;*

- *E is* complete in C considering M, *denoted by* $E \in com(SF, C, M)$, *if it holds* $E \in adm(SF, C, M)$ *and* E *contains all* $a \in C$ *acceptable considering* M *w.r.t.* E;
- *E is* preferred in C considering M, *denoted by* $E \in pref(SF, C, M)$, *if it holds* $E \in adm(SF, C, M)$ *and there is no* $E' \in adm(SF, C, M)$ *with* $E \subsetneq E'$.

The characteristic function $F^M_{SF,C}$ of SF in C considering M *is the mapping* $F^M_{SF,C} : 2^C \to 2^C$ *where* $F^M_{SF,C}(E) = \{a \in C \mid a \text{ is acceptable considering } M \text{ w.r.t. } E\}$.

- *E is* grounded in C considering M, *denoted by* $E \in grd(SF, C, M)$, *if* E *is the least fixed point of* $F^M_{SF,C}$.

Setting $C = A$ and $M = \emptyset$ recovers the original semantics, in these cases we will omit writing the respective parameter. Let us discuss some properties of the semantics in C considering M. First, if we deal with admissibility-based semantics, we can actually restrict our attention to the usual notion of conflict-freeness.

Proposition 3.83. *Let* $SF = (A, R)$ *be a SETAF, and let* $E, C \subseteq A$ *and* $M \subseteq R$. *Let* $E \subseteq C$ *be a set of arguments s.t. each* $a \in E$ *is acceptable considering* M *w.r.t.* E. *Then* $E \in cf(SF, C, M)$ *if and only if* $E \in cf(SF)$.

Proof. The ($\Leftarrow$) direction is clear since $E \in cf(SF)$ is a stricter notion. For ($\Rightarrow$) suppose $E \in cf(SF, C, M)$. We have to show that even for mitigated attacks $(T, h) \in M$ it holds that $T \cup \{h\} \not\subseteq E$. Striving for a contradicting suppose otherwise. Then we have in particular that $h \in E$. Since h is acceptable w.r.t. E by assumption, there is some non-mitigated attack $(X, t) \in R \setminus M$ with $X \subseteq E$ and $t \in T$. Since $T \subseteq E$, it follows $t \in E$. Hence, the attack (X, t) causes a conflict (not making use of mitigated attacks), contradiction. □

Since we restrict our attention to admissibility-based semantics, we will for ease of notation in the following assume that $E \in cf(SF)$ instead of $E \in cf(SF, C, M)$. Next we establish that important basic properties of the characteristic function also hold in this generalized setting.

Theorem 3.84. *Let* SF *be a SETAF, and let* $C \subseteq A$ *and* $M \subseteq R$. *Then,*

1. $F^M_{SF,C}$ *is monotonic,*
2. *the fundamental lemma holds, i.e. if* $E \in adm(SF, C, M)$ *and* $a \in A \cap C$ *is acceptable w.r.t.* E *considering* M, *then* $E \cup \{a\} \in adm(SF, C, M)$,
3. $E \in grd(SF, C, M)$ *is the least set in* $com(SF, C, M)$ *w.r.t.* $\subseteq$, *and*
4. $E \in pref(SF, C, M)$ *are the maximal sets in* $com(SF, C, M)$ *w.r.t.* $\subseteq$.

Proof.

1. Monotonicity of the mapping

$$F^M_{SF,C}(E) = \{a \in C \mid a \text{ is acceptable considering } M \text{ w.r.t. } E\}$$

holds by definition of defense.

2. Let $a \in A \setminus C$ be acceptable w.r.t. $E \in adm(SF, C, M)$ considering M. By monotonicity of defense, each argument in $E \cup \{a\}$ is acceptable w.r.t. $E \cup \{a\}$ considering M. As our notion of defense w.r.t. M implies the usual defense, we can apply the standard fundamental lemma for SETAFs [NP06b] and obtain $E \cup \{a\} \in cf(SF)$. Therefore, the conditions for applying Proposition 3.83 are met and we deduce $E \cup \{a\} \in cf(SF, C, M)$. Hence $E \cup \{a\} \in adm(SF, C, M)$ follows.

3. Setting $G = \bigcup_{i \geq 1}(F^M_{SF,C})^i(\emptyset)$ we claim that G is the least set in $com(SF, C, M)$. Due to the fundamental lemma (see 2.) admissibility of $\emptyset$ implies inductively

$$\forall n \in \mathbb{N} : \bigcup_{1 \leq i \leq n} \left(F^M_{SF,C}\right)^i (\emptyset) \in adm(SF, C, M).$$

Since SF is finite and by monotonicity, there is some n s.t.

$$\bigcup_{1 \leq i \leq n} \left(F^M_{SF,C}\right)^i (\emptyset) = \bigcup_{1 \leq i} \left(F^M_{SF,C}\right)^i (\emptyset) = G$$

Thus, G is complete. Now let $E \in com(SF, C, M)$. By monotonicity of $F^M_{SF,C}$ we get $F^M_{SF,C}(\emptyset) \subseteq F^M_{SF,C}(E)$. By induction, $(F^M_{SF,C})^i(\emptyset) \subseteq (F^M_{SF,C})^i(E)$ therefore also holds for any integer $i \geq 1$. Since E is complete, $E = (F^M_{SF,C})^i(E)$ holds for each integer i, i.e. the right-hand side is actually constant. We conclude for each n

$$G = \bigcup_{1 \leq i} \left(F^M_{SF,C}\right)^i (\emptyset) = \bigcup_{1 \leq i \leq n} \left(F^M_{SF,C}\right)^i (\emptyset) \subseteq \bigcup_{1 \leq i \leq n} \left(F^M_{SF,C}\right)^i (E) = E,$$

thus it follows that $G \subseteq E$.

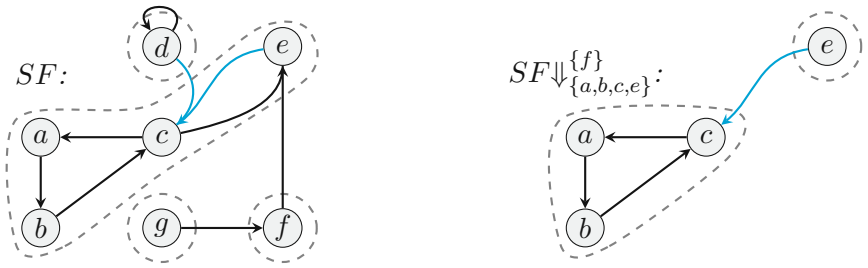
4. By definition $E \in pref(SF, C, M)$ is maximal in $adm(SF, C, M)$. So we show E is maximal in $adm(SF, C, M)$ iff E is maximal in $com(SF, C, M)$.

($\Rightarrow$) Suppose $E \in pref(SF, C, M)$ is not maximal in $com(SF, C, M)$. Then there is a proper complete superset E' of E; since E' is in particular admissible, E is not maximal in $adm(SF, C, M)$.

($\Leftarrow$) Now suppose E is not maximal in $adm(SF, C, M)$. Take $E' \in adm(SF, C, M)$ with $E \subsetneq E'$. By the fundamental lemma and monotonicity of $F^M_{SF,C}$, we find that $\bigcup_{1 \leq i}(F^M_{SF,C})^i(E')$ is a complete proper superset of E (analogous to 2.). Hence E is not maximal in $com(SF, C, M)$. □

To adequately characterize the defeated, provisionally defeated, and undefeated arguments in this setting we now also have to consider mitigated attacks. We illustrate this using the following example.

Example 3.85. *Consider the following SETAF SF (the dashed lines indicate the SCCs).*



The set $\{a, e, g\}$ is not admissible, and should therefore not be characterized by our (yet to be formally defined) notion of SCC-recursiveness. Intuitively, the singleton SCCs $\{g\}, \{f\}, \{d\}$ are unsurprisingly evaluated w.r.t. $\{a, e, g\}$ in the sense that g is accepted, f is defeated, and d is undecided. To characterize the remaining SCC $\{a, b, c, e\}$ we have to take the defeated arguments into account (namely, f), resulting in $SF\Downarrow^{\{f\}}_{\{a,b,c,e\}}$. As f is defeated, we delete the attack towards e and as a result "split" the SCC into two SCCs $\{a, b, c\}, \{e\}$. It is important to see that the remaining attack (e, c) is mitigated, but in contrast to the situation illustrated in Example 3.80 the mitigated attack did not origin in the current recursion step—because we split the original SCC, we will invoke the general function of the SCC-recursive scheme on the sub-framework $SF\Downarrow^{\{f\}}_{\{a,b,c,e\}}$. Consequently, the attack (e, c) is not indicated as mitigated by the set $M_{SF}(E, S)$; to still have this relevant information we generalize this set to also take the mitigated attacks from earlier recursion steps into account. Otherwise, the attack (e, c) is not marked as mitigated. Moreover, for the same reason we have to take the mitigated attacks into account when we calculate the set of defeated arguments: e is not sufficient to defeat c—if we would not account for this "inherited" mitigated attack (e, c), we would conclude that c is defeated and therefore a is acceptable, mistakenly characterizing $\{a, e, g\}$ as admissible.

Formally, we capture this in the following slightly adapted version of Definition 3.69. Note that the only difference to the former definition is that arguments that are only attacked by E via mitigated attacks do not count as defeated, but provisionally defeated.

Definition 3.86. *Let $SF = (A, R)$ be a SETAF. Moreover, let $E \subseteq A$ be a set of arguments, $M \subseteq R$ a set of attacks, and $S \in SCCs(SF)$ be an SCC. We define the set of defeated arguments $D_{SF}(S, E, M)$, provisionally defeated arguments $P_{SF}(S, E, M)$, and*

undefeated arguments $U_{SF}(S,E,M)$ *w.r.t.* S,E,M *as*

$$D_{SF}(S,E,M) = \{a \in S \mid \exists (T,a) \in R \setminus M \text{ s.t. } T \subseteq E \setminus S\},$$
$$P_{SF}(S,E,M) = \{a \in S \mid A \setminus (S \cup E^+) \mapsto_R a\} \setminus D_{SF}(S,E,M),$$
$$U_{SF}(S,E,M) = S \setminus (D_{SF}(S,E,M) \cup P_{SF}(S,E,M)).$$

Moreover, we set $UP_{SF}(S,E,M) = U_{SF}(S,E,M) \cup P_{SF}(S,E,M)$.

We have to make similar adjustments to the notion of mitigated attacks. Due to Definition 3.81, for the computation of mitigated attacks only the ancestor SCCs are relevant. In particular, the set $(T' \setminus T)$ is contained in ancestor SCCs of S for each attack $(T',h) \in R$. However, when we apply the concept of mitigated attacks to characterize SCC-recursiveness in admissibility-based semantics, we will face situations where we already know for the original SETAF that some attacks are mitigated. To account for this set of given mitigated attacks M, we slightly modify the condition for mitigated attacks, s.t. only the non-mitigated attacks $(T',h) \in R$ can "override" the status of a mitigated attack as non-mitigated.

Definition 3.87 (Mitigated Attacks, refined)**.** *Let* $SF = (A,R)$ *be a SETAF. Moreover, let* $E \subseteq A$ *and* $S \in SCCs(SF)$*. The set* $M_{SF}(S,E,M)$ *of* mitigated attacks *is given as*

$$\{(T,h) \in R \left(SF \Downarrow^{(E \setminus S)^+}_{UP_{SF}(S,E,M)}\right) \mid \forall (T',h) \in R \setminus M : T' \supseteq T \Rightarrow (T' \setminus T) \not\subseteq E\}.$$

Indeed, for Example 3.85 we get that the attack (e,c) is mitigated in $SF \Downarrow^{\{f\}}_{\{a,b,c,e\}}$. Next we redefine Definition 3.75 in order to capture the admissibility-based semantics. For this, we need to take into account that in each recursive call of the generic selection function $\mathcal{GF}$ we will also have to pass the current set M of mitigated attacks.

Principle 3.88 (SCC-recursiveness, refined)**.** *A semantics* σ *satisfies* SCC-recursiveness *if and only if for all SETAFs* $SF = (A,R)$ *it holds* $\sigma(SF) = \mathcal{GF}(SF, A, \emptyset)$*, where the* generic selection function $\mathcal{GF}(SF,C,M) \subseteq 2^A$ *is defined as:* $E \subseteq A \in \mathcal{GF}(SF,C,M)$ *if and only if*

- *if* $|SCCs(SF)| = 1$*, then* $E \in \mathcal{BF}(SF,C,M)$*,*
- *otherwise,* $\forall S \in SCCs(SF)$ *it holds that*

$$E \cap S \in \mathcal{GF}\left(SF \Downarrow^{(E \setminus S)^+}_{UP_{SF}(S,E,M)}, U_{SF}(S,E,M) \cap C, M_{SF}(S,E,M)\right),$$

where $\mathcal{BF}$ *maps* $SF = (A,R)$ *with* $|SCCs(SF)| = 1$ *and sets* $C \subseteq A$*,* $M \subseteq R$ *to a subset of* 2^A*.*

Towards an SCC-recursive characterization of admissible sets we discuss the following auxiliary results. Lemma 3.89 shows that global acceptability implies local acceptability, Lemma 3.90 shows the converse direction.

Lemma 3.89. *Let $SF = (A, R)$ be a SETAF, let $M \subseteq R$, $C \subseteq A$, and let $E \in adm(SF, C, M)$ be an admissible set of arguments and let $a \in A \cap C$ be acceptable w.r.t. E considering M in SF, where $a \in S$ for some SCC S. Then*

1. *we have $a \in U_{SF}(S, E, M)$ and a is acceptable w.r.t. $(E \cap S)$ in $SF{\Downarrow}^{(E\setminus S)^+}_{UP_{SF}(S,E,M)}$ considering $M_{SF}(S, E, M)$;*
2. *it holds that $(E \cap S)$ is conflict-free in $SF{\Downarrow}^{(E\setminus S)^+}_{UP_{SF}(S,E,M)}$.*

Proof. Set $SF{\Downarrow}^{(E\setminus S)^+}_{UP_{SF}(S,E,M)} = SF' = (A', R')$.

1. By Theorem 3.84, item 2, we get that $E \cup \{a\} \in adm(SF, C, M)$, i.e. $a \notin D_{SF}(S, E, M)$ by conflict-freeness of E and $a \notin P_{SF}(S, E, M)$ by defense. Consequently, we infer $a \in U_{SF}(S, E, M)$. Likewise, we get $(E \cap S) \subseteq U_{SF}(S, E, M)$, and therefore $(E \cap S) \subseteq A'$.

   To show that a is acceptable in this context we have to consider attacks towards a, i.e. $(T, a) \in R'$, and establish $T \not\subseteq E$ (conflict-freeness) and $(E \cap A')$ attacks T in SF' via non-mitigated attacks (defense). As E is admissible in SF, there is a (non-mitigated) counter-attack $(X, t) \in R \setminus M$ with $t \in T$ and $X \subseteq E$. In particular, this means that $t \notin E$, as (X, t) would otherwise contradict the conflict-freeness of E. Hence, $T \not\subseteq E$. Moreover, because $(T, a) \in R'$, it must be that $X \cap S \neq \emptyset$, as otherwise the attack (T, a) would be deleted when we construct the appropriate restriction to the SCC S. Let $X' = X \cap S$, i.e. there is an attack $(X', t) \in R'$. In other words, $E \cap S$ defends a in SF'. Finally, $X \subseteq E$ and $(X, t) \notin M$ implies $(X', t) \notin M_{SF}(S, E, M)$.

2. Towards contradiction assume there is an attack $(T, h) \in R'$ with $T \cup \{h\} \subseteq (E \cap S)$. This means there is an attack $(T', h) \in R$ with $T \subseteq T'$. As E is admissible in SF there is a counter-attack $(X, t) \in R \setminus M$ with $X \subseteq E$ for some $t \in T'$. If $t \notin S$ then $(T, h) \notin R'$, a contradiction. Therefore $t \in S$ and by assumption $t \in E$. However, this means $X \cup \{t\} \subseteq E$, a contradiction to the conflict-freeness of E in SF. □

Lemma 3.90. *Let $SF = (A, R)$ be a SETAF, let $M \subseteq R$, let $E \subseteq A$ such that*

$$(E \cap S) \in adm\left(SF{\Downarrow}^{(E\setminus S)^+}_{UP_{SF}(S,E,M)}, U_{SF}(S, E, M), M_{SF}(S, E, M)\right)$$

for all $S \in SCCs(SF)$. Moreover, let $S' \in SCCs(SF)$ and let $a \in U_{SF}(S', E, M)$ be acceptable w.r.t. $(E \cap S')$ in $SF{\Downarrow}^{(E\setminus S')^+}_{UP_{SF}(S',E,M)}$ considering $M_{SF}(S', E, M)$. Then a is acceptable w.r.t. E in SF considering M.

Proof. We have to show for each $(T, a) \in R$ that E attacks T in SF with non-mitigated attacks. As before we set $SF{\Downarrow}^{(E\setminus S')^+}_{UP_{SF}(S',E,M)} = SF' = (A', R')$. We distinguish the following three cases:

1. $(T \subseteq S')$ If $T \cap D_{SF}(S', E, M) \neq \emptyset$ we are done, because this means by Definition 3.86 there is an attack in $R \setminus M$ with $T \subseteq E$. Otherwise, all $t \in T$ are in $UP_{SF}(S', E, M)$. Then, $(T, a) \in R'$ and there must be a (not mitigated) counter-attack (X, t) with $t \in T$ and $X \subseteq E \cap S'$ within SF', as we assumed a is acceptable w.r.t. $E \cap S'$ in SF' considering $M_{SF}(S', E, M)$. This means there is an attack $(X', t) \in R \setminus M$ with $X \subseteq X'$, and as (X, t) is not mitigated in SF' we know $X' \subseteq E$. In summary, a is acceptable w.r.t. E in SF considering M.

2. $(T \subseteq A \setminus S')$ Then $T \cap E^+ \neq \emptyset$ by $a \in U_{SF}(S', E, M)$ (otherwise, if E would not attack T in SF, if $(T, a) \in M$ then $a \in P_{SF}(S', E, M)$, and if $(T, a) \notin M$ then $a \in D_{SF}(S', E, M)$).

3. $(T \cap S' \neq \emptyset$ and $T \cap (A \setminus S') \neq \emptyset)$ Assume towards contradiction there is no non-mitigated attack from E to T in SF. Then there is an attack $(T', a) \in R'$ with $T' \subseteq T$ and $(T', a) \notin M_{SF}(S', E, M)$. Now the reasoning proceeds as in case (1).

As we established that there are counter-attacks in all cases (1)-(3), the desired property holds. □

Combining these two results we obtain the SCC-recursive characterization of admissible sets.

Proposition 3.91. *Let $SF = (A, R)$ be a SETAF and let $E \subseteq A$ be a set of arguments. Then for each $C \subseteq A$ and $M \subseteq R$ it holds $E \in adm(SF, C, M)$ if and only if $\forall S \in SCCs(SF)$ it holds $(E \cap S) \in adm(SF{\Downarrow}^{(E \setminus S)^+}_{UP_{SF}(S,E,M)}, U_{SF}(S, E, M) \cap C, M_{SF}(S, E, M))$.*

Proof. Let $SF' = SF{\Downarrow}^{(E \setminus S)^+}_{UP_{SF}(S,E,M)}$.

($\Rightarrow$) Since $E \subseteq C$ and all $a \in E$ are acceptable w.r.t. E in SF considering M, we can apply Lemma 3.89 and get that every $a \in (E \cap S)$ are in $U_{SF}(S, E, M) \cap C$ for any given SCC S. Moreover, we get that a is acceptable w.r.t. $(E \cap S)$ in SF' considering $M_{SF}(S, E, M)$ and that $(E \cap S)$ is conflict-free in SF'. Hence, $(E \cap S)$ is admissible in SF' considering $M_{SF}(E, S, M)$.

($\Leftarrow$) As for all SCCs S we assume $(E \cap S) \subseteq (S \cap C)$ we know $E \subseteq C$, i.e. we only need to show admissibility in SF considering M. Towards contradiction assume E is not conflict-free in SF, i.e. there is an attack $(T, h) \in R$ with $T \cup \{h\} \subseteq E$. Let S' be the SCC containing h.

1. We cannot have (1) $T \subseteq S'$, as this would contradict the assumption that $E \cap S'$ is conflict-free in $SF{\Downarrow}^{(E \setminus S')^+}_{UP_{SF}(S',E)}$ (note that in this case the attack (T, h) is also necessarily in SF').

2. Moreover, it cannot be that (2) $T \subseteq A \setminus S'$, because then $h \in D_{SF}(S', E, M)$ (or $h \in P_{SF}(S', E, M)$ if $(T, h) \in M$) while we assumed $h \in U_{SF}(S', E, M)$.

3. Finally, consider the case (3) where $T \cap S' \neq \emptyset$ and $T \cap (A \setminus S') \neq \emptyset$. Then there is a non-mitigated attack from $(E \setminus S')$ to T, as otherwise there would be $(T \cap S', h) \in SF\Downarrow^{(E\setminus S')^+}_{UP_{SF}(S',E,M)}$, contradicting our assumption of local conflict-freeness. Call this attack $(X, t) \in R \setminus M$ with $X \subseteq (E \setminus S')$ and $t \in T \setminus S'$. Let S'' be the SCC t is in. As before, we cannot have (1) $X \subseteq S''$ or (2) $X \subseteq A \setminus S''$. The only remaining case is again (3) $X \cap S'' \neq \emptyset$ and $X \cap (A \setminus S'') \neq \emptyset$—as this step (3) always takes us to a prior SCC and we assume SF finite, eventually this recursion will stop in case (1) or (2). Now, by induction we get a contradiction for the initial case.

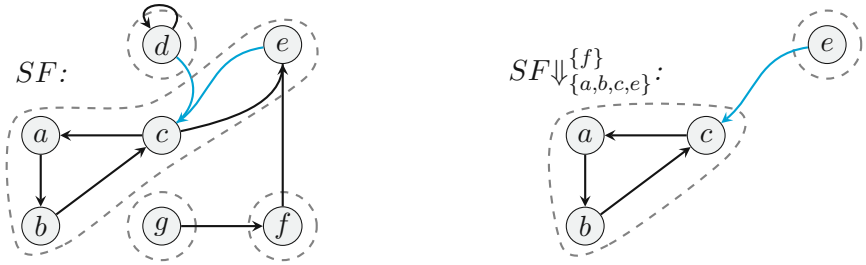
It remains to show that every $a \in E$ is acceptable w.r.t. E in SF considering M. Let S^* be the SCC a is in and let $SF^* = SF\Downarrow^{(E\setminus S^*)^+}_{UP_{SF}(S^*,E,M)}$. By assumption, $(E \cap S^*) \in adm(SF^*, U_{SF}(S^*, E, M), M_{SF}(S^*, E, M))$, i.e. a is acceptable w.r.t. $E \cap S$ in SF^* considering $M_{SF}(S^*, E, M)$. Since we also have $a \in U_{SF}(S^*, E, M)$, we can apply Lemma 3.90 and get that a is acceptable w.r.t. E in SF considering M. □

The base function for admissible sets is $adm(SF, C, M)$. We will utilize this result to obtain the characterizations of the other (admissibility-based) semantics.

Theorem 3.92. *Admissible semantics is SCC-recursive.*

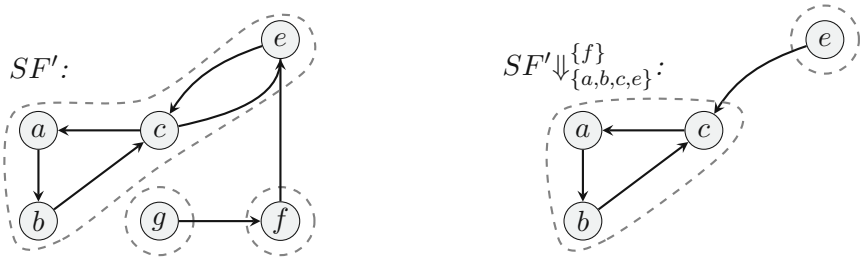
Proof. The base function $\mathcal{BF}(SF)$ is $adm(SF, C, M)$. The case for $|SCCs(SF)| = 1$ is immediate, the composite case follows from Proposition 3.91. □

Example 3.93. *Recall our example with the (not admissible) set $E = \{a, e, g\}$.*



Indeed, in $SF\Downarrow^{\{f\}}_{\{a,b,c,e\}}$ we have that (e, c) is a mitigated attack. Therefore, in the next recursive step, in the SCC $\{a, b, c\}$ the argument c is detected as provisionally defeated. Hence a is not admissible in the corresponding sub-framework and thus, E is rightfully detected as non-admissible.

Let us now consider SF' the same SETAF as SF, but without the self-attacker d.



Let E be as above. Then, the attack (e, c) is not mitigated in $SF'\Downarrow^{\{f\}}_{\{a,b,c,e\}}$ and hence, c is detected as defeated. Hence in order to evaluate $\{a, b, c\}$ we require another recursive step and find acceptance of $\{a\}$ in a sub-SCC consisting only of the single undefeated argument a. Therefore, it is rightfully detected that $E \in adm(SF')$.

3.4.3 Complete Semantics

We already have the tools to characterize complete extensions: Proposition 3.91 proves the desired properties for admissible sets, in addition we can apply Lemma 3.89 and Lemma 3.90 to show that complete extensions contain all arguments they defend (i.e., for an SCC S', an extension E, and a set of mitigated attacks M, exactly those arguments from $U_{SF}(S', E, M)$ that are *acceptable* w.r.t. $(E \cap S')$ in $SF\Downarrow^{(E\setminus S')^+}_{UP_{SF}(S',E,M)}$ considering $M_{SF}(S', E, M)$).

Proposition 3.94. *Let $SF = (A, R)$ be a SETAF, let $M \subseteq R$, and let $E \subseteq A$ be a set of arguments. Then $\forall C \subseteq A$ it holds $E \in com(SF, C, M)$ if and only if $\forall S \in SCCs(SF)$ it holds $(E \cap S) \in com(SF\Downarrow^{(E\setminus S)^+}_{UP_{SF}(S,E,M)}, U_{SF}(S, E, M) \cap C, M_{SF}(S, E, M))$.*

Proof. ($\Rightarrow$) If $E \in com(SF, C, M)$, then in particular $E \in adm(SF, C, M)$. Hence by Proposition 3.91 we get

$$\forall S \in SCCs(SF): \quad (E \cap S) \in adm(SF\Downarrow^{(E\setminus S)^+}_{UP_{SF}(S,E,M)}, UP_{SF}(S, E, M) \cap C, M_{SF}(S, E, M)).$$

For an arbitrary SCC $S' \in SCCs(SF)$, let $a \in U_{SF}(S', E, M)$ be an argument acceptable w.r.t. $(E \cap S')$ in $SF\Downarrow^{(E\setminus S')^+}_{UP_{SF}(S',E,M)}$ considering $M_{SF}(S', E, M)$. By Lemma 3.90, a is acceptable w.r.t. E in SF considering M, and, hence, $a \in E$ and $a \in E \cap S'$ by completeness.

($\Leftarrow$) We get $E \in adm(SF, C, M)$ by Proposition 3.91. For an arbitrary $a \in C$, let S' be the SCC a is in. If a is acceptable w.r.t. E in SF considering M, by Lemma 3.89 we get that a is acceptable w.r.t. $(E \cap S')$ in $SF\Downarrow^{(E\setminus S')^+}_{UP_{SF}(S',E,M)}$ considering $M_{SF}(S', E, M)$. As $(E \cap S')$ is locally complete, we get $a \in E$. □

From this we get the desired result regarding complete extensions. The base function is $com(SF, C, M)$.

Theorem 3.95. *Complete semantics is SCC-recursive.*

3.4.4 Preferred Semantics

The next lemma illustrates that if we already found a globally admissible set E and find a (larger) locally admissible set $E' \supsetneq E \cap S$ in an SCC S, then we can find a globally admissible set incorporating this set E'. This idea underlies the incremental computation of extensions (see Section 3.5).

Lemma 3.96. *Let $SF = (A, R)$, let $M \subseteq R$, and let $E \in adm(SF, A, M)$, let $S \in SCCs(SF)$ be an SCC. Moreover, let $E' \subseteq A$ be a set of arguments such that $(E \cap S) \subseteq E' \subseteq U_{SF}(S, E, M)$, and $E' \in adm(SF{\Downarrow}^{(E\setminus S)^+}_{UP_{SF}(S,E,M)}, U_{SF}(S, E, M), M_{SF}(S, E, M))$. Then $E \cup E'$ is admissible in SF considering M.*

Proof. We first show that $(E \cup E')$ is conflict-free in SF. Again, let $SF' = SF{\Downarrow}^{(E\setminus S)^+}_{UP_{SF}(S,E,M)}$. Assume towards contradiction there is $(T, h) \in R$ with $T \cup \{h\} \subseteq (E \cup E')$. Then we have either (1) $h \in E$ or (2) $h \in E' \setminus E$.

(1) Since $E \in adm(SF, A, M)$ in (1) we have $E \mapsto_R T$ via a non-mitigated attack. We have $E \in cf(SF)$, this means $E \mapsto_R T'$ where $T' = T \setminus E = T \cap E' \neq \emptyset$. But this means $E' \cap D_{SF}(S, E, M) \neq \emptyset$, contradicting our assumption $E' \subseteq U_{SF}(S, E, M)$.

(2) Regarding (2), if $T \subseteq E$, then if $(T, h) \notin M$ we have $h \in D_{SF}(S, E, M)$ (or $h \in P_{SF}(S, E, M)$ if $(T, h) \in M$), a contradiction. Hence, $T \cap (E' \setminus E) \neq \emptyset$. It follows there is $(T', h) \in R(SF')$ with $T' \subseteq T$. However, since we assume E' is conflict-free in SF' it holds $T' \cup \{h\} \not\subseteq E'$, a contradiction because $E \cap S \subseteq E'$.

As both possibilities lead to contradictions, we conclude $(E \cup E') \in cf(SF)$.

It remains to show defense in SF considering M, i.e. for all $(T, h) \in R$ with $h \in E \cup E'$ we show $E \cup E' \mapsto_R T$ via non-mitigated attacks. If $h \in E$, this follows from $E \in adm(SF, A, M)$. For $h \in E' \setminus E$, we distinguish 4 cases:

(1) E attacks T via non-mitigated attacks, then we are done.

(2) $T \subseteq A \setminus S$. But then either $E \mapsto_R T$ via non-mitigated attacks—see (1)—or all attacks from E to T are mitigated, or $E \not\mapsto_R T$, and therefore $h \in P_{SF}(S, E, M)$, a contradiction to $h \in E' \subseteq U_{SF}(S, E, M)$.

(3) $T \subseteq A(SF')$. But this means $(T, h) \in R(SF')$ and therefore since E' is admissible in this context there is a non-mitigated counter-attack $(X, t) \in R(SF')$ with $X \subseteq E'$ and $t \in T$. As (X, t) is not mitigated, there is a non-mitigated "original" attack $(X', t) \in R$ with $X' \supseteq X$ and $X' \setminus X \subseteq E$, i.e. $X' \subseteq (E \cup E')$, contradicting the earlier established conflict-freeness.

(4) $T \cap A(SF') \neq \emptyset$ and $T \cap (A \setminus A(SF')) \neq \emptyset$. If we assume we are not in case (1) then there is an attack $(T', h) \in R(SF')$, and we proceed as in (3). In any case, there is a defense against the attack (T, h), therefore, $(E \cup E') \in adm(SF, A, M)$. □

Given this lemma, we are ready to show SCC-recursiveness for preferred semantics.

Proposition 3.97. *Let $SF = (A, R)$ be a SETAF, let $M \subseteq R$ and let $E \subseteq A$ be a set of arguments. Then $\forall C \subseteq A$ it holds $E \in pref(SF, C, M)$ if and only if $\forall S \in SCCs(SF)$ it holds $(E \cap S) \in pref(SF{\Downarrow}^{(E\setminus S)^+}_{UP_{SF}(S,E,M)}, U_{SF}(S, E, M) \cap C, M_{SF}(S, E, M))$.*

Proof. ($\Rightarrow$) We assume $E \in pref(SF, C, M)$, and can apply Proposition 3.91 and obtain that

$$\forall S \in SCCs(SF): \quad (E \cap S) \in adm(SF{\Downarrow}^{(E\setminus S)^+}_{UP_{SF}(S,E,M)}, U_{SF}(S, E, M) \cap C, M_{SF}(S, E, M)).$$

Assume towards contradiction that for some $S' \in SCCs(SF)$ there is a set

$$E' \in adm(SF{\Downarrow}^{(E\setminus S')^+}_{UP_{SF}(S',E,M)}, U_{SF}(S', E, M) \cap C, M_{SF}(S', E, M))$$

with $E \cap S' \subsetneq E'$. However, by Lemma 3.96 this means the set $E \cup E'$ is in $adm(SF, C, M)$, but since $E \subsetneq E \cup E'$ this contradicts our assumption $E \in pref(SF, C, M)$.

($\Leftarrow$) From Proposition 3.91 we get $E \in adm(SF, C, M)$. Towards contradiction assume there is an $E' \in adm(SF, C; M)$ with $E' \supsetneq E$. This means there is some SCC $S \in SCCs(SF)$ such that $(E \cap S) \subsetneq (E' \cap S)$. W.l.o.g. we choose S such that no ancestor SCC of S has this property. This means that $U_{SF}(S, E, M) = U_{SF}(S, E', M)$ and $P_{SF}(S, E, M) = P_{SF}(S, E', M)$ for S and all of its ancestor SCCs. Consequently, $(E' \cap S) \subseteq U_{SF}(S, E, M) = U_{SF}(S, E', M)$, and by another application of Proposition 3.91 we get $E' \in adm(SF{\Downarrow}^{(E\setminus S)^+}_{UP_{SF}(S,E,M)}, U_{SF}(S, E, M) \cap C, M_{SF}(S, E, M))$. However, this contradicts our assumption $E \in pref(SF{\Downarrow}^{(E\setminus S)^+}_{UP_{SF}(S,E,M)}, U_{SF}(S, E, M) \cap C, M_{SF}(S, E, M))$. □

From this we get the desired result regarding preferred extensions. The base function is $pref(SF, C, M)$.

Theorem 3.98. *Preferred semantics is SCC-recursive.*

3.4.5 Grounded Semantics

For the characterization of grounded semantics we exploit the fact that also in our setting the grounded is the unique minimal complete extension (see Theorem 3.84). Hence, we can apply Proposition 3.94 and utilize the fact that that for the unique grounded extension minimality has to hold for each SCC to prove minimality of the whole extension.

Proposition 3.99. *Let $SF = (A, R)$ be a SETAF, $M \subseteq R$, and let $E \subseteq A$ be a set of arguments. Then $\forall C \subseteq A$ it holds $E \in grd(SF, C, M)$ if and only if $\forall S \in SCCs(SF)$ it holds $(E \cap S) \in grd(SF{\Downarrow}^{(E\setminus S)^+}_{UP_{SF}(S,E,M)}, U_{SF}(S, E, M) \cap C, M_{SF}(S, E, M))$.*

Proof. ($\Rightarrow$) We assume $E \in grd(SF, C)$, and can apply Proposition 3.94 and obtain that

$$\forall S \in SCCs(SF): \quad (E \cap S) \in com(SF\Downarrow^{(E\setminus S)^+}_{UP_{SF}(S,E,M)}, U_{SF}(S,E,M) \cap C, M_{SF}(S,E,M)).$$

Assume towards contradiction that for some SCC S' the set $(E \cap S')$ is not minimal among the locally complete extensions. W.l.o.g. we choose S' such that no ancestor SCC of S' has this property. Let $E' \in grd(SF\Downarrow^{(E\setminus S)^+}_{UP_{SF}(S,E,M)}, U_{SF}(S,E,M) \cap C, M_{SF}(S,E,M))$. We can construct E'' such that for the ancestor SCCs of S' the new set E'' coincides with E, for S' it coincides with E', and for the remaining SCCs S is determined by $grd(SF\Downarrow^{(E'\setminus S)^+}_{U_{SF}(S,E',M)}, UP_{SF}(S,E',M) \cap C, M_{SF}(S,E',M))$ (see Section 3.5 for details). But then $E'' \in com(SF, C, M)$ by Proposition 3.94 and $E \not\subseteq E''$, a contradiction to our assumption $E \in grd(SF, C, M)$.

($\Leftarrow$) We get $E \in com(SF, C, M)$ by Proposition 3.94. Towards contradiction assume there is some $E' \subsetneq E$ with $E' \in grd(SF, C, M)$. This means there is an SCC S where $(E' \cap S) \subsetneq (E \cap S)$. W.l.o.g., we choose S such that no ancestor SCC of S has this property. This means that $U_{SF}(S,E,M) = U_{SF}(S,E',M)$ and $P_{SF}(S,E,M) = P_{SF}(S,E',M)$ for S and its ancestor SCCs. Consequently,

$$(E' \cap S) \in com(SF\Downarrow^{(E\setminus S)^+}_{UP_{SF}(S,E,M)}, UP_{SF}(S,E,M) \cap C, M_{SF}(S,E,M)).$$

However, this contradicts our assumption

$$E \in grd(SF\Downarrow^{(E\setminus S)^+}_{UP_{SF}(S,E,M)}, UP_{SF}(S,E,M) \cap C, M_{SF}(S,E,M)),$$

since $(E' \cap S) \subsetneq (E \cap S)$. □

Theorem 3.100. *Grounded semantics is SCC-recursive.*

3.4.6 Connection to Directionality

As it is the case in AFs, we can obtain results regarding directionality using SCC-recursiveness if the base function always admits at least one extension [BG07]. First note that for an uninfluenced set U any SCC S with $S \cap U \neq \emptyset$ has to be contained in U, as well as all ancestor SCCs of S. Then, by the SCC-recursive characterization we get the following general result, subsuming the semantics under our consideration.

Proposition 3.101. *Let σ be a semantics such that for all SETAFs SF and all $C \subseteq A(SF)$, $M \subseteq R(SF)$ it holds $\mathcal{BF}(SF, C, M) \neq \emptyset$. If σ satisfies SCC-recursiveness then it satisfies directionality.*

Proof. We use the fact that for an uninfluenced set U any SCC S with $S \cap U \neq \emptyset$ has to be contained in U, as well as all ancestor SCCs of S. Let $\mathcal{S}$ be the set of SCCs S with $S \subseteq U$. Considering the SCC-recursive characterization, this yields $\sigma(SF\Downarrow^{\emptyset}_{U}) =$

$\{E \subseteq U \mid \forall S \in \mathcal{S} : (E \cap S) \in \mathcal{GF}(SF\Downarrow^{\emptyset}_{UP_{SF}(S,E,M)}, U_{SF}(S,E,M), M_{SF}(S,E,M))\}$. We have to show that $\sigma(SF\Downarrow^{\emptyset}_{U}) = \{E \cap U \mid E \in \sigma(SF)\}$.

We get the "$\subseteq$" direction from the fact that $U_{SF}(S,E,M) = U_{SF}(S,E\cap U,M)$ and $P_{SF}(S,E,M) = P_{SF}(S,E\cap U,M)$ for all $S \in \mathcal{S}$. The "$\supseteq$" direction is immediate: as we assume that $\mathcal{BF}(SF,C,M)$ always yields at least one extension, we can extend any set $(E \cap U)$ according to the SCC-recursive scheme (see Section 3.5 for details). □

3.5 Incremental Computation

In this section we discuss the computational implications of a semantics satisfying directionality, modularization, or SCC-recursiveness, and how we can improve the asymptotic runtime of the resulting algorithms by utilizing structures in the graph of the SETAFs. To this end, we establish the basic idea of our algorithms exploiting structural properties (Section 3.5.1) and analyze the graph classes acyclicity, even-cycle-freeness, bipartiteness, and full-symmetry in the setting of incremental computation to further refine the relevant structures. These classes form *tractable fragments* for SETAFs [DKUW24]—recall Definitions 2.15–2.19 from the background chapter of this thesis. It is known that the presence of these properties leads to computational ease [DKUW24], we generalize this result to be applicable in the general case in the context of SCCs (Section 3.5.2).

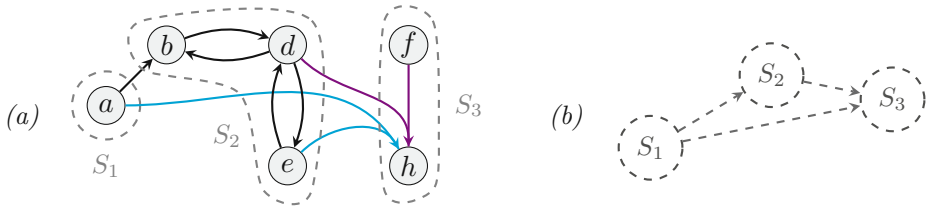
3.5.1 Basic Computational Speedup

First, for a semantics σ satisfying directionality an argument a is in some extension (in all extensions) if and only if it is in some extension (in all extensions) of the framework that is restricted to the arguments that influence a. That is, when deciding credulous or skeptical acceptance of an argument, in a preprocessing step, we can shrink the framework to the relevant part. The property of modularization is closely related to CEGAR style algorithms for preferred semantics that can be implemented via iterative SAT-solving [DJWW14]. In order to compute a preferred extension we can iteratively compute a non-empty admissible set of the current framework, build the reduct w.r.t. this admissible set, and repeat this procedure on the reduct until the empty set is the only admissible set. The preferred extension is then given by the union of the admissible sets.

Finally, for SCC-recursive semantics we can iteratively compute extensions along the SCCs of a given framework (see [Bau11, LJK11, BGL14, CGVZ14] for such approaches for AFs). It is well known that the SCCs of any directed graph form a partial order w.r.t. reachability: in Example 3.102 (b) the SCC S_1 is an initial SCC and precedes S_2 and S_3, and S_2 precedes S_3. In (one of) the initial SCCs we simply compute the extensions and then for each of these extensions we proceed on the preceding SCCs. We then iteratively continue this process on SCCs in their order. To evaluate an SCC that is influenced by other ones we have to take the attacks from earlier SCCs into account and, as we have

already fixed our extension there, we can simply follow the SCC-recursive schema. We next illustrate this for stable semantics.

Example 3.102. *Consider (a) the SETAF SF and (b) the order of its SCCs.*



We can iteratively compute the stable extensions of SF as follows: in the first SCC $S_1 = \{a\}$ we simple compute all the stable extensions, i.e., $stb(SF{\Downarrow}^{\emptyset}_{S_1}) = \{\{a\}\}$. We then proceed with $\{a\}$ as extension E for the part of the SETAF considered so far. Next we consider S_2 and adapt it to take E into account. As $(E \setminus S_2)^+ = \{b\}$ we only have to delete the argument b from S_2 before evaluating the SCC and thus we obtain $SF{\Downarrow}^{\{b\}}_{S_2} = (\{d, e\}, \{(d, e), (e, d)\})$. Combining these with E we obtain two stable extensions $E_1 = \{a, d\}$, $E_2 = \{a, e\}$ for $SF{\Downarrow}^{\emptyset}_{S_1 \cup S_2}$. We proceed with S_3 and first consider E_1. As $(E_1 \setminus S_3)^+ = \{b, e\}$ we do not remove arguments from S_3. However, as $d \in E_1$ we cannot delete the attack $(\{d, f\}, h)$ but have to replace it by the attack (f, h). We then have $stb(SF{\Downarrow}^{\{b,e\}}_{S_3}) = \{\{f\}\}$ and thus obtain the first stable extensions of SF $\{a, d, f\}$. Now consider E_2. We have that E_2 attacks h, i.e., $(E_2 \setminus S_3)^+ = \{b, d, h\}$, and thus we have to remove h before evaluating S_3 and thus obtain $SF{\Downarrow}^{\{b,d,h\}}_{S_3} = (\{f\}, \emptyset)$. We end up with $\{a, e, f\}$ as the second stable extension of SF.

The computational advantage of the incremental computation is that certain computations are performed over single SCCs instead of the whole framework. This is in particular significant for preferred semantics where the $\subseteq$-maximality check can be done within the SCCs. Notice that verifying a preferred extension is in general coNP-complete [DT96, DGW18]. However, given our results regarding the SCC-recursive scheme, the following parameterized tractability result is easy to obtain: it is well known that computing the SCCs of a directed graph can be done efficiently. It then suffices to verify a given extension along the SCCs of the framework, whereby we only need to consider one SCC at the time.

Theorem 3.103. *Let SF be a SETAF where $|S| \leq k$ for all $S \in SCCs(SF)$. Then we can verify a given preferred extensions in $O\left(2^k \cdot poly(|SF|)\right)$ for some polynomial poly.*

Next, we will build on prior work regarding graph classes for SETAFs and illustrate how we can exploit them to reason more efficiently. These classes generalize situations that are known to yield computationally easy fragments in the special case of AFs. Moreover

we will show how we can utilize these graph classes in the context of SCC-recursiveness to achieve a computational speedup even if the framework is heterogeneous, i.e., does not as a whole belong to one of these tractable fragments. Instead, in the spirit of Theorem 3.103 we follow the SCC-recursive scheme and pose the respective restrictions only on the strongly connected components, resulting in a more flexible setting.

3.5.2 Utilizing Tractable Fragments for Efficient Computation Along SCCs

In this section we will show which of our tractable fragments we can exploit in the context of SCC-recursiveness to speed up computation. In particular, we exemplify the speedup with the Ver_{pref} problem. We will show for acyclicity, even-cycle-freeness, and primal-bipartiteness, that we obtain a speedup when every SCC of a SETAF belongs to one of these tractable fragments.

On the other hand, we show that for symmetry this is not the case. Deleting (parts of) attacks from a fully-symmetric SETAF might lead to a situation where the remaining framework is no longer fully symmetric. We will show that therefore this fragment does not allow a speedup in the SCC-recursive scheme. The key idea is that prior SCCs can "disable" arbitrary attacks in a given SCC. In the reduction from the general verification problem we use to prove this (illustrated in Example 3.105) we have at least 3 SCCs: the one containing x, the one containing y, and the ones containing our original framework (if the original framework is not connected, we obtain more than one). Each SCC is fully-symmetric, but the symmetric counter-attacks in the SCCs corresponding to the original framework are irrelevant, as the argument y in the tail is always defeated.[9]

Proposition 3.104. *The problem Ver_{pref} remains* coNP*-complete even for self-attack-free SETAFs SF where all SCCs $S \in SCCs(SF)$ are fully-symmetric, i.e., $SF{\downarrow_S}$ is fully symmetric.*

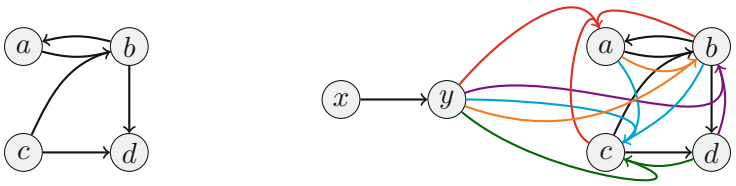
Proof. Let $SF = (A, R)$ be an arbitrary self-attack-free SETAF and let $x, y \notin A$ be new arguments. We define $SF' = (A \cup \{x, y\}, R')$, with

$$R' \;=\; R \cup \{(\{y, h\} \cup T \setminus \{t\}, t) \mid (T, h) \in R, (\{h\} \cup T \setminus \{t\}, t) \notin R\}.$$

In the resulting framework SF' clearly all SCCs are fully-symmetric (note that $\{x\}, \{y\}$ are SCCs of SF', as well as all loosely connected components of SF). Clearly $x \in G$ and $y \in G^+$ for $G \in grd(SF')$, and hence, $x \in E$ and $y \in E^+$ for all $E \in pref(SF')$. By construction we can apply the reduct on $\{x\}$ and obtain $SF'^{\{x\}} = SF$, and by modularization we hence get $pref(SF') = \{E \cup \{x\} \mid E \in pref(SF)\}$, which means $E \in pref(SF)$ if and only if $E \cup \{x\} \in pref(SF')$. □

[9] Note that in the journal version [DKUW24] of this result we erroneously provide a reduction for *primal*-symmetric SCCs (i.e., the primal graph is symmetric) instead of *fully*-symmetric SCCs. However, the very same idea of "disabling" the attack via the defeated argument y applies in both cases.

Example 3.105. *Consider the SETAF in (a) with preferred extensions* $\{\{a, c\}, \{b\}\}$. *By adding the arguments* x, y *we make every SCC fully-symmetric (under projection), while preserving the preferred extensions under projection (i.e., the added arguments and attacks have no practical effect, in particular, if we construct the reduct w.r.t.* $\{x\}$ *we recover the original framework): (b) has preferred extensions* $\{\{x, a, c\}, \{x, b\}\}$. *For example, for the original attack* $(\{a, c\}, b)$ *we add the attacks* $(\{y, a, b\}, c)$ *and* $(\{y, b, c\}, a)$, *which means when we project to the SCC* $\{a, b, c, d\}$ *of the constructed SETAF, we obtain fully-symmetric attacks between* a, b, *and* c. *Finally, note that for our construction it is irrelevant that the SCC structure is altered—while in the original framework, the SCCs are* $\{a, b, c\}$ *and* $\{d\}$, *the SETAF from the reduction contains SCCs* $\{x\}$, $\{y\}$, *and* $\{a, b, c, d\}$.



(a) *(b)*

In contrast to the negative result regarding full-symmetry, it is indeed possible to verify preferred extensions efficiently in primal-bipartite SCCs (under projection). To establish this result, we generalize our notion of the reduct and modularization to the semantics considering a candidate set $C \subseteq A$ and a set of mitigated attacks $M \subseteq R$ (see SCC-recursiveness, Definition 3.88).

Definition 3.106. *Given a SETAF* $SF = (A, R)$, $M \subseteq R$, *and* $E \subseteq A$, *the* E-reduct of SF considering M *is the SETAF* $SF^E_M = (A', R')$, *with*

$$\begin{aligned} A' &= A \setminus E^{\oplus}_{R \setminus M} \\ R' &= \{(T \setminus E, h) \mid (T, h) \in R,\, T \cap E^{+}_{R \setminus M} = \emptyset,\, T \not\subseteq E,\, h \in A'\} \end{aligned}$$

Note the parallels to the definition of the restricted frameworks in the SCC-recursive scheme. We now show that the modularization property also holds in this context. The idea is similar to the special case of $C = A, M = \emptyset$ that we discussed in Theorem 3.49. What we have left to consider is that an admissible set E could attack an argument x in its reduct via mitigated attacks. As in the SCC-recursive scheme, we cannot accept x in the reduct. Hence, we add such arguments to the set C.

Proposition 3.107. *Let* $SF = (A, R)$ *be a SETAF and* $C \subseteq A$, $M \subseteq R$, *and* $E \in adm(SF, C, M)$. *Let* $SF' = SF^E_M = (A', R')$.

1. *If* $E' \in adm(SF', C', M')$ *with*

$$\begin{aligned} C' &= C \cap \{a \in A \mid \nexists (T, a) \in R \cap M : (T \subseteq E)\} \\ M' &= \{(T, h) \in R' \mid \forall (T', h) \in R \setminus M : T' \supseteq T \Rightarrow (T' \setminus T) \not\subseteq E\} \end{aligned}$$

then $E \cup E' \in adm(SF, C, M)$.

2. *If* $E \cap E' \neq \emptyset$ *and* $E \cup E' \in adm(SF, C, M)$, *then* $E' \in adm(SF', C', M')$.

Proof. 1. Since E is admissible in SF considering M, E' does not attack E via non-mitigated attacks in SF. By construction of SF', E does not attack E' via non-mitigated attacks either. Since $E' \subseteq C'$ we know E also does not attack E' via mitigated attacks. Towards contradiction assume E' attacks E via a mitigated attack. By admissibility, then E attacks E' in SF which is not the case (as just established). Hence, $E \cup E' \in cf(SF)$.

Now assume $S \mapsto_R E \cup E'$. If $S \mapsto_R E$, then $S \mapsto_{R \setminus M}$ by admissibility. If $S \mapsto_R E'$, then there is $T \subseteq S$ s.t. $(T, e') \in R$ with $e' \in E'$. If now $E \mapsto_{R \setminus M} T$ we are done. Otherwise, there is $(T \setminus E, e') \in R'$, and $E' \mapsto_{R' \setminus M'} T \setminus E$. This means there is $(X', t) \in R' \setminus M'$ with $t \in T$ and $X' \subseteq E'$, and consequently $(X, t) \in R \setminus M$ with $X \setminus E = X'$. Since $X \mapsto_{R \setminus M} S$ and $E \cup E' \supseteq X$, it also holds $E \cup E' \mapsto_{R \setminus M} S$, i.e., $E \cup E'$ defends itself against S in SF considering M. Hence, we have $E \cup E' \in adm(SF, C, M)$.

2. Assume $E \cup E' \in adm(SF, C, M)$. We see $E' \in cf(SF')$ as follows: if $(T', e') \in R'$ with $T' \subseteq E'$, $e' \in E'$, then there is some $(T, e') \in R$ with $T' = T \setminus E$. Hence, $E \cup E' \mapsto_R E'$, a contradiction.

Now assume $E' \notin adm(SF', C', M')$. This means there is $(T', e') \in R'$ with $e' \in E'$, but there is no non-mitigated counter-attack from E' towards T'. Then there is $(T, e') \in R$ with $T' = T \setminus E$ and $T \cap E^+_{R \setminus M} = \emptyset$. By admissibility we know $E \cup E' \mapsto_{R \setminus M} T$, say $(T^*, t) \in R \setminus M$ with $T^* \subseteq E \cup E'$, $t \in T$. Since $E \cup E'$ is conflict-free in SF, $T^* \cap E^+_R = \emptyset$, and thus we either have a) $T^* \subseteq E$, contradicting $T \cap E^+_{R \setminus M} = \emptyset$, or b) $(T^* \setminus E, t) \in R' \setminus M'$, contradicting the assumption that there is no counter-attack. Finally, note that $E' \subseteq C'$, as otherwise there is $(T, e') \in R \cap M$ with $T \subseteq E, e' \in E'$, contradicting conflict-freeness. In summary, we conclude $E' \in adm(SF', C', M')$. □

It remains to show that we can find non-empty admissible sets in this context in polynomial time. To this end, we slightly adapt Algorithm 1 from [DKUW24] to also account for a candidate set C and mitigated attacks M, see Algorithm 1. The only differences to the original algorithm are in step 2, where we consider C, and in step 6, where for possible counter-attacks from our constructed admissible set to attackers we only take non-admissible attacks into account. We get that the set of credulously accepted arguments considering C and M (i.e., the arguments that are in an admissible set considering C and M) of one part of the partition are exactly the arguments that are returned by the algorithm. The proof of the correctness and completeness is analogous to [DKUW24, Lemma 7.20].

Algorithm 1: Compute the set of credulously accepted arguments w.r.t. *pref* semantics considering a candidate set and mitigated attacks.

Input : A primal-bipartite SETAF $SF = (A, R)$ with a partitioning (Y, Z), sets $C \subseteq A$, $M \subseteq R$

Output : The admissible set Y_i (considering M) of credulously accepted arguments in $Y \cap C$

1 $i := 0$
2 $Y_0 := Y \cap C$
3 $R_0 := R$
4 **repeat**
5 $\quad i := i + 1$
6 $\quad Y_i := Y_{i-1} \setminus \{y \mid y \in Y_{i-1},$ there is some $(Z', y) \in R_{i-1}$ with $Z' \subseteq Z$ such that $\forall z \in Z' \ |\{(Y', z) \mid (Y', z) \in R_{i-1} \setminus M\}| = 0\}$
7 $\quad R_i := R_{i-1} \setminus \{(Y', z) \mid Y' \subseteq Y, z \in Z, Y' \not\subseteq Y_i\}$
8 **until** $Y_i = Y_{i-1}$;

Proposition 3.108. *Let $SF = (A, R)$ be a primal-bipartite SETAF with a partitioning (Y, Z), and let $C \subseteq A$ be a set of arguments and $M \subseteq R$ a set of mitigated attacks. An argument $a \in Y \cap C$ is in $\bigcup adm(SF, C, M)$ iff it is in the set returned by Algorithm 1.*

Proof. "⇒": We show by induction over the iterations of the loop in the algorithm that every argument that is removed in step 6 cannot be defended and the attacks that are removed in step 7 cannot be part of a defending attack. For the first iteration this is the case, as we construct Y_1 by only removing those arguments $y \in Y$ from Y that are attacked by an attack (Z', y) on which no non-mitigated counter-attack exists. Moreover we remove all attacks (Y', z) towards arguments $z \in Z$ such that for one of the arguments $y' \in Y'$ we already showed it is not defensible, as they cannot defend any argument in an admissible set. Likewise, assuming this property holds for the $i-1$-th iteration, in the i-th iteration we only remove arguments that are not defensible via non-mitigated attacks and attacks that cannot play a role for defense in admissible sets.

Assume towards contradiction an argument $y \in Y \cap C$ is credulously accepted, but not in the set that is returned by the algorithm. This means at some iteration i the argument y is removed, but, as established, this means it is not defensible, which is a contradiction to the assumption is it credulously accepted.

"⇐": Let S be the set that is returned by the algorithm. We show that $S \in adm(SF, C, M)$. As we have $S \subseteq Y$, we know S is conflict-free in SF. Moreover we know that S defends every $x \in S$: towards contradiction assume otherwise, i.e. there is an attack (Z', x) towards x such that S does not attack Z' via non-mitigated attacks. But then x would be removed in step 6, which is a contradiction to the assumption that $x \in S$. □

If we apply the algorithm both for the partition (Y, Z) and conversely for (Z, Y), in the

union of the outputs we get the set of all credulously accepted arguments. Moreover, note that the set that is returned by the algorithm is admissible (considering C and M) in SF, as we show in the "$\Leftarrow$" direction of the proof of Proposition 3.108.

This allows us immediately to obtain the following tractability result for preferred semantics.

Proposition 3.109. *Let $SF = (A, R)$ be a primal-bipartite SETAF and $C, E \subseteq A$, $M \subseteq R$. We can decide whether $E \in pref(SF, C, M)$ in polynomial time.*

Proof. We can check in polynomial time whether $E \in adm(SF, C, M)$, and compute SF^E_M. By Proposition 3.107 it suffices to check whether there is a non-empty set $E' \in adm(SF', C', M')$, which we can find out by running Algorithm 1 for both partitions (as per Proposition 3.108). □

Finally, we can see that this result generalizes to odd-cycle-freeness if we restrict ourselves to the case where we only have a single SCC.

Proposition 3.110. *Let $SF = (A, R)$ be an odd-cycle-free SETAF with $|SCCs(SF)| = 1$. Then SF is primal-bipartite.*

Proof. Let x be an arbitrary argument from SF, we say *x reaches argument $y \in A$ in n steps* if there are attacks $(X_1, x_2), (X_2, x_3), \ldots, (X_n, y) \in R$ with $x \in X_1$, $x_i \in X_i$ for $1 \leq i \leq n$. Let $S = \{a \in A \mid a \text{ can be reached from } x \text{ in an even number of steps }\}$. Then $(S, A \setminus S)$ is a partitioning for SF: Clearly, $A \setminus S$ is the set of arguments that can be reached from x in an *odd* number of steps, and because SF is strongly connected at the same time the set of argument from which x can be reached in a odd number of steps. Assume towards contradiction there are two arguments $a, b \in (A \setminus S)$ s.t. a reaches b in 1 step. However, this introduces an odd-length primal-cycle, as x reaches a in an odd number of steps, a reaches b in 1 step, and b reaches x in an odd number of steps, a contradiction. Likewise, there can be no pair a, b of arguments in S where a reaches b in 1 step. □

Clearly, this implies that if $|SCCs(SF)| = 1$ in an odd-cycle-free SETAF we can also verify preferred extensions in polynomial time. Note also that by removing (parts of) attacks from SF we cannot introduce an odd-cycle.

In the following we argue that we can efficiently compute the (unique) preferred extension if the SETAF in question is even-cycle-free. We utilize the fact that an even-cycle-free SETAF has only one preferred extension, namely the grounded—this also holds true considering the candidate set C and mitigated attack M.

Proposition 3.111 (cf. [Dvo12])**.** *Let $SF = (A, R)$ be an even-cycle-free SETAF and $C, E \subseteq A$, $M \subseteq R$. We can decide whether $E \in pref(SF, C, M)$ in polynomial time.*

Proof. An even-cycle-free SETAF admits only one complete extension (see [DKUW24, Proposition 7.7]). By making some of the attacks mitigated, there can be no additional complete extensions. Moreover, by considering the candidate set C there can be no additional complete extensions. Then by Theorem 3.84 item 4 this means that there is only one preferred extension, which by Theorem 3.84 item 3 is the grounded extension, which we can compute in polynomial time. □

Taking these results together, we obtain the following characterization. This generalizes the FPT-result from Theorem 3.103 and illustrates that we can utilize various different graph properties of SCCs at once.

Theorem 3.112. *Let SF be a SETAF where for all SCCs $S \in SCCs(SF)$ it holds either*

- *S is acyclic,*
- *S is even-cycle-free,*
- *S is primal-bipartite,*
- *S odd-cycle-free, or*
- *the size of S is bounded by a parameter k, i.e., $|S| \leq k$.*

Then we can verify a given preferred extensions in $O\left(2^k \cdot poly(|SF|)\right)$ for some polynomial function $poly$.

Proof. Follows from Proposition 3.109, Proposition 3.110, and Proposition 3.111 and the fact that acyclic SETAFs are also even- and odd-cycle-free. □

3.6 Discussion

In this chapter, we systematically analyzed semantics for SETAFs using a principle-based approach (see Table 3.1 for an overview of the investigated properties). Moreover, we introduced and investigated novel principles like allowing partial conflicts (APC) I–III, tail strengthening, and attack weakening, that are trivial on AFs, but desirable non-trivial properties of SETAF semantics. We pointed out interesting concepts that help us to understand the principles more deeply: edge cases that for AFs are hidden behind simple syntactic notions have to be considered explicitly for SETAFs, revealing semantic peculiarities that are already there in the special case. For example, the rich syntax of SETAFs allows us to differentiate between unattacked and uninfluenced sets—these notions trivially coincide in AFs. This distinction allows us to meaningfully generalize directionality to SETAFs and discuss the accompanying phenomena more accurately. We highlight the usefulness of the *reduct* in this context—many seemingly unrelated notions from various concepts boil down to formalizations closely related to the reduct

(recall the modifications we applied for the SCC-recursive scheme). We particularly focused on computational properties like modularization and SCC-recursiveness. The emphasis on the computation of argumentation tasks lead us to our investigations of graph properties in the context of SCCs, during which we introduced and analyzed the computational complexity of reasoning tasks for these restricted cases. Finally, we applied these findings in the context of SCC-recursiveness, which allowed us to push the boundaries of tractability for argumentation tasks even further.

SCC-recursiveness has recently been investigated for Abstract Dialectical Frameworks (ADFs) in a different context by Gaggl et al. [GRS21]. In that work, the acceptance conditions of the statements in an ADFs (that encode the attacks in case the ADF recasts a SETAF) are modified. Note that SETAFs can be seen as a special case of ADFs, where each acceptance condition is a formula in conjunctive normal form with only negative literals [DKZLW20]. The modification of this formula in the approach of Gaggl et al. [GRS21] is indeed closely related to our idea of the SETAF-reduct: attacks (T, h) where in a prior SCC we learn that one argument $t \in T$ is attacked (defeated) in an extension effectively become redundant and are removed. In case an argument $t \in T$ is in an extension, the acceptance condition is modified, in SETAF terms this would correspond to an attack $(T \setminus \{t\}, h)$. However, while we treat the undecided state of an argument that is neither in nor attacked by an extension via mitigated attacks, in the approach of Gaggl et al. self-attacks are introduced to model the resulting effects (akin to the idea of *splitting* [Bau11, Lin14, BDKW24]). While both approaches effectively yield the same results, we expect the introduction of new (self-)attacks to be computationally disadvantageous compared to our approach of labeling an attack as mitigated.

An interesting direction for future works is investigating semantics *cf2* [BGG05] and *stage2* [DG16] for SETAFs. While recently the family of semantics based on weak admissibility (cf. [BBU20b]) has been investigated for SETAFs [BKU24], there is still much work left on a principle-based analysis, as well as computational properties.

CHAPTER 4

Backdoor-Based Evaluation

In this chapter we continue our focus on computational aspects of reasoning in SETAFs. We use the tools of parameterized complexity analysis and focus on the parameter of *deletion-backdoors* (in the following just "backdoors"). The idea of backdoors is used in different contexts such as constraint satisfaction problems (CSP), satisfiability checking (SAT), answer set programming (ASP) (see e.g. [GOS17, FS15, OSS21, GMO+14] for recent work), and has also been investigated for AFs [DOS12], which serves as a starting point for our investigation on SETAFs. In our context, a backdoor is a set of arguments, which is "in the way" of tractable computation. Hence, the removal of these arguments (together with an accompanying change of the attack structure) ensures that the remaining framework is easy to solve. To account for the removed arguments, the general idea is to enumerate all possible evaluations on the backdoor-arguments, and apply the remaining (efficient) computation for the remaining framework for each such possibility. Consequently, instead of enumerating the whole state-space which is exponential in the number of all arguments, we obtain a runtime which is exponential only in the number of *backdoor*-arguments—i.e., in the size of the backdoor.

As mentioned, the notion of the backdoor relies on the fact that the remaining framework after the removal of the backdoor arguments is easy to solve. We ensure this fact by defining different kinds of backdoors, depending on which easy-to-solve property they rely on. In this regard we focus on graph-properties like acyclicity (ACYC), which already served us well when we refined the SCC-recursive approach in the end of Chapter 3. A major advantage of this approach is that we can build upon a solid foundation of research regarding the problem of *finding* appropriate backdoors—as we will show, the problem of finding an acyclicity-backdoor coincides with the (directed) feedback-vertex-set problem, as originally investigated by Karp in 1972 [Kar72]. More recently, an FPT-algorithm has been presented which for a graph of size n either returns a feedback vertex set of size $\leq p$ or correctly returns that such a set does not exist in runtime $O(4^p \cdot p! \cdot n^4 \cdot p^3)$ [CLL+08]. Another backdoor-type we heavily make use of in this chapter is backdoors to even-

cycle-freeness (NOEVEN). While for finding NOEVEN backdoors no FPT algorithm is known, there exists an obvious XP algorithm: since we can check in polynomial time whether a directed graph is even-cycle-free [RST99], we can check for each subset of arguments of size $\leq p$ whether the remaining graph after removing the potential backdoor is even-cycle-free. For other classes we use in this thesis like bipartiteness and symmetry it was already shown for AFs that the standard problems remain NP-hard even for constant backdoor-sizes [DOS12].

Building on this research to *find* suitable backdoors, we will establish algorithms to efficiently *exploit* a given backdoor in the context of abstract argumentation. To this end we present an algorithm to characterize preferred, stable, and semi-stable extensions in time $O(2^p \cdot n^{O(1)})$, where again p is the size of the backdoor, and n is the size of the SETAF. This holds also in the special case of AFs, where the state-of-the art is a $O(3^p \cdot n^{O(1)})$ algorithm. For complete extensions, we provide an algorithm to enumerate all extensions in time $O(3^p \cdot n^{O(1)})$—in this case, the number of extensions might exceed 2^p, as we will show, which makes an algorithm in $O(2^p \cdot n^{O(1)})$ impossible. To answer the respective reasoning problems, we show how the characterization via our algorithm can be exploited, and prove that for all problems under our consideration we either improve or match the asymptotic runtime of the known AF approach. Finally, we show that our approach is optimal unless the Strong Exponential Time Hypothesis (cf. [IP99, IPZ98]) is false.

This chapter is organized as follows.

- In Section 4.1 we introduce the concept of deletion-backdoors for SETAFs for the fragments ACYC and NOEVEN, and discuss why we focus on these fragments. For this we provide arguments illustrating that reasoning for the other tractable fragments we consider remains hard, even when a backdoor of constant size is given. We then investigate how to *find* a suitable backdoor to these fragments.

- Section 4.2 introduces the basis of our backdoor algorithm. We formally establish its correctness, and prove that we can indeed utilize the algorithm to reason efficiently on SETAFs. We show how our algorithm improves the runtime over the state-of-the art in AFs.

- In Section 4.2.2 we show how our novel algorithm can be sped up for the computation of stable extensions by utilizing that stable extensions do not omit undecided arguments. That is, every argument is either in a stable extension or attacked by it, which means we do not have to apply any computational efforts towards arguments that we cannot label as in the extension or attacked by the extension.

- In Section 4.2.3 we generalize the approach of [DOS12] for complete extensions to SETAFs. Note that due to the potential maximal number of complete extensions it is not possible to obtain the same runtime of $O(2^p \cdot \mathsf{poly}(n))$ in this case, instead we provide an $O(3^p \cdot \mathsf{poly}(n))$ time algorithm.

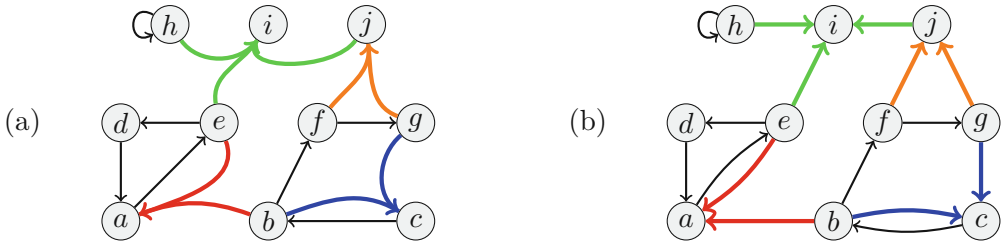


Figure 4.1: (a) The SETAF SF and (b) its primal graph $primal(SF)$. Collective attacks and their corresponding edges in the primal graph are highlighted.

- Finally, we show in Section 4.3 that our approach that depends in $O(2^p)$ on the backdoor size p is optimal, assuming the strong exponential time hypothesis holds.

This chapter is based on [DKW24] (which is currently under review), which in turn contains and extends the content of [DKW22a].

4.1 Towards SETAF Backdoors

The underlying structure of SETAFs is a *directed hypergraph*, which makes it hard to apply certain notions of graph properties. We can avoid these issues by utilizing "standard" directed graphs to describe SETAFs, and analyze graph properties (such as SCC-recursiveness in Chapter 3, backdoors in this chapter, or treewidth in Chapter 5). We investigate both the primal graph and the incidence graph; we focus on the *primal graph*, as it arguably is the most intuitive way to embed SETAFs into directed graphs and fits our purposes best. In the end of this chapter we circle back to the incidence graph, and establish that it yields the same type of backdoors, albeit via a detour. We recall the primal graph (as per Definition 2.10) in Figure 4.1; the corresponding SETAF will serve as a running example for this chapter. We then utilize the primal graph to define *primal-backdoors*.

Recall the tractable fragments ACYC, NOEVEN, SYM, and BIP, as per Definition 2.15, Definition 2.17, and Definition 2.19, respectively. Note that in the special case of AFs these classes coincide with the standard definitions, i.e., they properly generalize the standard case. For AFs, also the classes of bipartite and symmetric frameworks have been investigated. However, while finding suitable backdoors to these classes was shown to be parameterized-tractable, reasoning in AFs with constant backdoor-size to these frameworks remains hard [DOS12]. Even though there are generalizations of symmetry and bipartiteness for SETAFs where reasoning also becomes tractable (cf. Table 2.1), as these classes generalize the respective properties of AFs, the hardness-results for backdoor evaluation carry over to the general SETAFs. Hence, we focus on the fragments ACYC and NOEVEN, i.e., acyclic SETAFs and even-cycle-free SETAFs. On AFs, all of these classes are considered as the *tractable fragments of argumentation*,

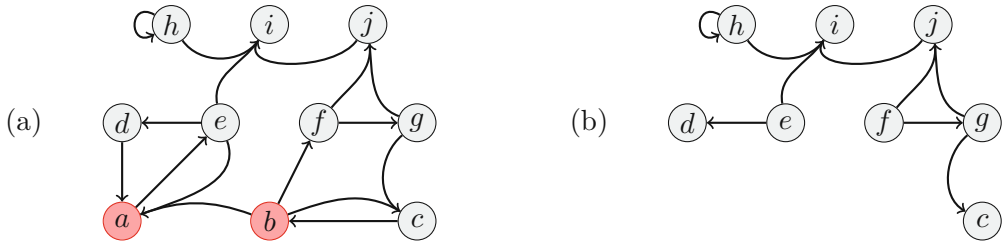


Figure 4.2: (a) The SETAF $SF = (A, R)$ with the highlighted NOEVEN-backdoor $B = \{a, b\}$ and (b) the "remaining" framework $SF{\downarrow}_{A \setminus B}$ after the deletion of B.

as for many semantics there are efficient algorithms to reason in AFs that belong to one of these classes [CMDM05, Dun07, DBC01]. However, these fragments restrict the possible structure of an AF. In order to exploit the speedup while still keeping the full expressiveness, the requirements have been weakened to allow for arbitrary *distance* to these fragments. We generalize this so-called *backdoor*-approach by Dvořák et al. [DOS12] to SETAFs and their hypergraph structure.

For this, we recall the notion of the *projection* as per Definition 2.8. Intuitively, a SETAF $SF = (A, R)$ projected to a set of arguments $S \subseteq A$ (we write $SF{\downarrow}_S$) can be seen as the result of removing the arguments outside of S while retaining as much of the original structure as possible. This means, for projecting on S (i.e., computing $SF{\downarrow}_S$), we remove all arguments $A \setminus S$ and only keep the arguments in S. Consequently, we also remove all attacks (T, h) where the head h is not in S, and where the tail T consists only of arguments outside of S (i.e., where the tail would be empty in $SF{\downarrow}_S$ which is not possible for a SETAF). Moreover, for the remaining attacks we remove from the tail the arguments outside of S, and retain the attack $(T \cap S, h)$. In the context of backdoor sets $B \subseteq A$, we project to the set $S = A \setminus B$, as we are interested in the remaining framework $SF{\downarrow}_{A \setminus B}$ after removing the backdoor arguments. Formally:

Definition 4.1. *Let $SF = (A, R)$ be a SETAF and let C be a class of SETAFs. We call a set $B \subseteq A$ a C-backdoor if $SF{\downarrow}_{A \setminus B}$ belongs to C, where*

$$SF{\downarrow}_{A \setminus B} = (A \setminus B, \{(T \setminus B, h) \mid (T, h) \in R, T \setminus B \neq \emptyset, h \in A \setminus B\}).$$

We denote the size of a smallest C-backdoor by $\mathsf{bd}_{\mathsf{C}}(SF)$.

Figure 4.2 illustrates the idea of backdoors: the set $B = \{a, b\}$ is a NOEVEN-backdoor of size 2 for the SETAF $SF = (A, R)$. It is easy to see in Figure 4.2(b) that in the SETAF SF projected to $A \setminus B$, i.e., the remaining framework $SF{\downarrow}_{A \setminus B}$ after removing the backdoor, has no directed cycles of even length in its primal graph. We want to highlight that in $SF{\downarrow}_{A \setminus B}$ the attack $(\{b, g\}, c)$ from SF is not deleted as a whole, but "reduced" to the attack (g, c). It can easily be verified that indeed, B is a smallest NOEVEN-backdoor. For any ACYC-backdoor, furthermore the argument h has to be included (in general,

every ACYC-backdoor is a NOEVEN-backdoor but not vice versa). Hence, for our example we have $\mathsf{bd}_{\mathsf{ACYC}}(SF) = 3$ and $\mathsf{bd}_{\mathsf{NOEVEN}}(SF) = 2$.

It was shown that reasoning in AFs w.r.t. stable, complete, and preferred semantics is tractable for the fragments $\mathsf{C} \in \{\mathsf{ACYC}, \mathsf{NOEVEN}\}$ if $\mathsf{bd}_{\mathsf{C}}(SF)$ is fixed [DOS12], we generalize these results in Section 4.2. We can efficiently *recognize* these classes for AFs [DOS12] and *find* backdoors of bounded size. As we defined the fragments for SETAFs on the primal graph, the respective results from AFs immediately carry over to our setting. For ACYC-backdoor this boils down to the directed feedback vertex set problem, which has been shown to be in FPT [CLL+08]. Finding a NOEVEN-backdoor is in XP, as checking if a directed graph is even-cycle-free can be done in polynomial time [RST99], and we can check for each subset of arguments up to the size of the backdoor parameter.

Corollary 4.2. *Let SF be a SETAF.*

1. *We can recognize whether SF belongs to ACYC or NOEVEN in polynomial time.*
2. *We can find a NOEVEN-backdoor of size at most p in time $|SF|^{O(p)}$ (or if no such backdoor exists correctly detect so).*
3. *We can find an ACYC-backdoor of size at most p in time $f(p) \cdot \mathsf{poly}(|SF|)$ (or if no such backdoor exists correctly detect so).*

This means, that the fragment ACYC is in principle suitable for FPT algorithms, while for NOEVEN we aim for XP algorithms. Note however that it is principally possible that faster algorithms for finding NOEVEN-backdoors exist.

4.2 Backdoor Evaluation

In this section we tackle the evaluation of backdoors of constant size, i.e., assume we are *given* a backdoor, how can we efficiently compute the extensions. We adapt the results from [DOS12] and generalize the therein mentioned notions such as partial labelings and propagation algorithms to SETAFs. Our approach however differs from the one in [DOS12] as it is tailored to the fragments ACYC and NOEVEN. This allows us to give a tighter upper bound for the runtime: $O(2^p \cdot \mathsf{poly}(n))$ instead of $O(3^p \cdot \mathsf{poly}(n))$ where p is the size of the given backdoor and n is the size of the instance. Our improved algorithm is applicable to SETAFs as well as AFs, as the latter is just a special case of the former.

The basic idea of the backdoor evaluation algorithm is to guess parts of an extension on the arguments in the backdoor, and then cleverly propagate this guess to the rest of the instance. As the remaining instance is even-cycle-free, there are no supportive cycles to consider in the propagation step.

Algorithm 2: Computation of $pref^*(SF, B)$

```
1  pref*(SF, B) ← ∅;
2  foreach I ⊆ B do
3      let λ be a partial labeling on B;
4      set λ(a) = in_𝔞 for a ∈ I;
5      set λ(a) = out_𝔞 for a ∈ B \ I;
6      λ* ← propagate_IO(SF, λ);
7      set λ*(a) = und_𝔞 for a ∈ A \ DEF(λ*);
8      λ† ← propagate_U(SF, λ*);
9      if IN(λ†) ∩ B = I then
10         pref*(SF, B) ← pref*(SF, B) ∪ {IN(λ†)};
```

Algorithm 2 captures the whole process to characterize preferred extensions; in the following we will verify its correctness and give an intuition for each step. If we provide for a SETAF SF a NOEVEN-backdoor B, Algorithm 2 returns a set of admissible candidates $pref^*(SF, B)$ for preferred extensions.

4.2.1 Characterizing Preferred Extensions

We start with an algorithm to characterize preferred extensions, to which end we first introduce the following technical notions. In a nutshell, the algorithm takes as input a SETAF $SF = (A, R)$ and a backdoor set B of arguments to the fragment NOEVEN (i.e., SF projected to $A \setminus B$ contains no even length cycles). We proceed by iterating over every subset of B, representing the arguments of B which we will label as in the extensions candidate. Consequently, the other arguments are *not* part of the extension candidate. The algorithm proceeds by propagating these guessed labels, and ultimately fixing wrong guesses to ensure admissibility. We discuss the steps in detail in the following, starting with some technical notions. We capture the concept of the partial guess by partial labelings (cf. [DOS12]).

Definition 4.3. *Let $SF = (A, R)$ be a SETAF and let $X \subseteq A$ be a set of arguments. A* partial labeling *is a function $\lambda : X \to \{in_{\mathfrak{a}}, out_{\mathfrak{a}}, und_{\mathfrak{a}}\}$. By $IN(\lambda)$ we denote the set $\{a \in X \mid \lambda(a) = in_{\mathfrak{a}}\}$ (similarly, $OUT(\lambda), UND(\lambda)$). We write $DEF(\lambda)$ to identify the set X. For a set $S \subseteq X$ we fix the* corresponding partial labeling on X *as*

$$\lambda_S(a) = \begin{cases} in_{\mathfrak{a}} & \text{if } a \in S, \\ out_{\mathfrak{a}} & \text{if } a \in S_R^+, \\ und_{\mathfrak{a}} & \text{else, i.e. } a \in X \setminus S_R^{\oplus}. \end{cases}$$

A labeling λ is compatible *with a set $S \subseteq A$ if $\forall a \in DEF(\lambda)$ it holds $\lambda(a) = \lambda_S(a)$.*

We consider two "phases" of propagation. First, we propagate the $in_{\mathfrak{a}}/out_{\mathfrak{a}}$ labels of the guess as far as possible by utilizing the function $\mathsf{propagate}_{\mathsf{IO}}$.

Definition 4.4. *Let $SF = (A, R)$ be a SETAF and λ be a partial labeling on $X \subseteq A$. Consider the following propagation rules for λ:*

1. *set $\lambda^*(a) = out_{\mathfrak{a}}$ if $\exists (T, a) \in R$ with $T \subseteq IN(\lambda^*)$,*
2. *set $\lambda^*(a) = in_{\mathfrak{a}}$ if $\forall (T, a) \in R$ there is a $t \in T$ with $\lambda^*(t) = out_{\mathfrak{a}}$.*

We define $\mathsf{propagate}_{\mathsf{IO}}(SF, \lambda)$ as the result of initializing λ^* with λ on $DEF(\lambda)$, and then exhaustively applying the rules (1) and (2) to each argument $a \in A \setminus DEF(\lambda)$.

The second phase fixes incorrectly labeled arguments and assures admissibility in the generated labeling by effectively propagating the $und_{\mathfrak{a}}$ label utilizing the function $\mathsf{propagate}_{\mathsf{U}}$ (see Definition 4.5). We will show that with this approach we can capture all preferred extensions. Following Algorithm 2 we initially label these arguments by $und_{\mathfrak{a}}$ that did not get any label during the first phase of applying $\mathsf{propagate}_{\mathsf{IO}}$.

Definition 4.5. *Let $SF = (A, R)$ be a SETAF and λ be a partial labeling on $X \subseteq A$. Consider the following propagation rules for λ:*

(3) set $\lambda^\dagger(a) = und_{\mathfrak{a}}$ if $\lambda^\dagger(a) = in_{\mathfrak{a}}$ and there is $(T, a) \in R$ s.t. $\nexists t \in T : \lambda^\dagger(t) = out_{\mathfrak{a}}$,

(4) set $\lambda^\dagger(a) = und_{\mathfrak{a}}$ if $\lambda^\dagger(a) = out_{\mathfrak{a}}$ and there is no $(T, a) \in R$ s.t. $T \subseteq IN(\lambda^\dagger)$.

We define $\mathsf{propagate}_{\mathsf{U}}(SF, \lambda)$ as the result of initializing $\lambda^\dagger$ with λ on $DEF(\lambda)$, and then exhaustively applying the rules (3) and (4) to each argument $a \in IN(\lambda) \cup OUT(\lambda)$.

In the following Lemma 4.6 we formalize the intuition of $\mathsf{propagate}_{\mathsf{IO}}$ in the context of Algorithm 2. By applying the function $\mathsf{propagate}_{\mathsf{IO}}$ in step 6 we propagate the labels we guessed on B (steps 3–5) and treat them at this stage as if they are "confirmed", i.e., whenever an argument is defended according to the current partial labeling, we add it (by labeling it $in_{\mathfrak{a}}$), whenever an argument is defeated by the partial labeling, we keep track of this fact (by labeling it $out_{\mathfrak{a}}$). In this stage, no argument will be labeled $und_{\mathfrak{a}}$. We revisit our running example from Figure 4.2. Assume we guess $\lambda_1(a) = \lambda_1(b) = out_{\mathfrak{a}}$ for the backdoor $\{a, b\}$. The labeling λ_1^* after exhaustively applying rules (1) and (2) is depicted in Figure 4.3(a). In general, we will incorrectly label arguments: as can be seen, the argument a is labeled $out_{\mathfrak{a}}$, but there is no attack towards a that verifies this label. Now assume we guess $\lambda_2(a) = \lambda_2(b) = in_{\mathfrak{a}}$. One can see in Figure 4.3(b) that there is a problem with the propagated labeling λ_2^*: the arguments d and a are both labeled $in_{\mathfrak{a}}$, effectively causing a conflict. Though we will correct this problem in the second step of the propagation algorithm, we will have to change the label of a from $in_{\mathfrak{a}}$ to $und_{\mathfrak{a}}$. We will see that we can actually already stop at this point, as we will then compute (a subset of) the extension we get from the guess $\lambda_3(a) = out_{\mathfrak{a}}, \lambda_3(b) = in_{\mathfrak{a}}$. In the following lemma we establish that for a "correct" guess on the backdoor arguments (i.e., a guess that actually corresponds to a preferred extension E) we label all arguments $a \in E_R^{\oplus}$ correctly (i.e., $in_{\mathfrak{a}}$ and $out_{\mathfrak{a}}$, resp.).

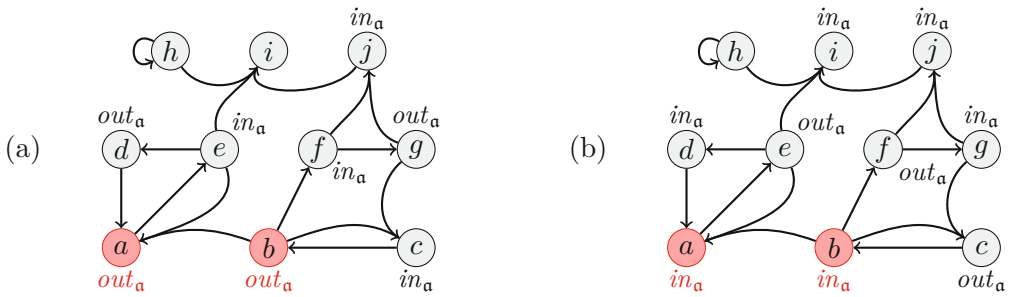


Figure 4.3: The first propagation $\mathsf{propagate}_{\mathsf{IO}}$ (a) λ_1^*, (b) λ_2^*, on the guessed labels for the backdoor arguments a and b (highlighted).

Lemma 4.6. *Let $SF = (A, R)$ be a SETAF, let $E \in pref(SF)$, and $B \subseteq A$ a NOEVEN-backdoor for SF. For the input (SF, B) to Algorithm 2, assume in step 2 we choose $I = E \cap B$, let λ be the corresponding partial labeling from steps 4 and 5. Set $\lambda^* = \mathsf{propagate}_{IO}(SF, \lambda)$. Then for each $a \in A$:*

(a) if $\lambda^*(a) = in_{\mathfrak{a}}$ then $a \notin E_R^+$,

(b) if $\lambda^*(a) = out_{\mathfrak{a}}$ then $a \notin E$,

(c) if $a \notin DEF(\lambda^*)$ then $a \notin E_R^{\oplus}$,

(d) $E \subseteq IN(\lambda^*)$, and

(e) $E_R^+ \subseteq OUT(\lambda^*)$.

Proof. We show (a) and (b) by induction on the number of labeled arguments in the construction of λ^*. For the base case $\lambda^* = \lambda$ it is easy to see that all conditions (a) and (b) hold (by assumption we have $OUT(\lambda) = B \setminus IN(\lambda) = (A \cap B) \setminus E$). For the step we consider the rules (1) and (2):

Assume a is labeled via rule (1), i.e. we set $\lambda^*(a) = out_{\mathfrak{a}}$ for some $a \in A \setminus DEF(\lambda)$. Clearly, (a) is not violated by labeling a as $out_{\mathfrak{a}}$, for (b) we show $a \notin E$. Since we invoked rule (1), there is an attack $(T, a) \in R$ with $T \subseteq IN(\lambda^*)$. By our induction hypothesis we know that (a) holds for each $t \in T$, i.e. $T \cap E_R^+ = \emptyset$. This means E does not defend a against the attack (T, a), and since $E \in pref(SF)$, we get $a \notin E$.

Now assume a is labeled via rule (2), i.e. we set $\lambda^*(a) = in_{\mathfrak{a}}$ for some $a \in A \setminus DEF(\lambda)$. Clearly, (b) is not violated by labeling a as $in_{\mathfrak{a}}$, for (a) we show $a \notin E_R^+$. Since we invoked rule (2), for all attacks $(T, a) \in R$ there is some $t \in T$ with $\lambda^*(t) = out_{\mathfrak{a}}$. By our induction hypothesis we know that (b) holds for each such t, i.e. $t \notin E$. Hence, there can be no attack (T, a) with $T \subseteq E$, i.e., $a \notin E_R^+$.

For (c) assume towards contradiction there is an argument $a_1 \in E$ such that $a_1 \notin DEF(\lambda^*)$. If there is no attack $(T, a_1) \in R$ towards a_1, we would have $\lambda^*(a_1) = in_{\mathfrak{a}}$, hence, there is such an attack. Moreover, there is a $(T, a_1) \in R$ s.t. for no $t \in T$ we have $\lambda^*(t) = out_{\mathfrak{a}}$, otherwise we would have $\lambda^*(a_1) = in_{\mathfrak{a}}$. However, by admissibility of E there is at least one $t_1 \in T \cap E_R^+$. For this t_1 we have $t_1 \in A \setminus DEF(\lambda^*)$, i.e. t_1 is unlabeled (the only other option, the label $in_{\mathfrak{a}}$, violates (b)). Since $t_1 \in E_R^+$, there is an attack $(S, t_1) \in R$, but since t_1 is unlabeled, $S \not\subseteq IN(\lambda^*)$, i.e., there is an $a_2 \in S$ such that a_2 is unlabeled. We have $a_1 \neq a_2$, as this would imply an even-length cycle (a_2, t_1). As for a_2 we can reason in the same way as for a_1, we obtain another unlabeled argument $t_2 \in E_R^+$, and eventually a sequence $a_1, t_1, a_2, t_2, \ldots$ of arguments. However, as SF is finite and all a_i are different, eventually there is either (i) an unlabeled argument a_k where no attack (T_k, a_k) towards a_k has an unlabeled $t_k \in T_k$, but then $\lambda^*(a_k) = in_{\mathfrak{a}}$, a contradiction, or (ii) an unlabeled argument t_k where there is no counter-attack (S_k, t_k) with an unlabeled $a_{k+1} \in S_k$, but then $\lambda^*(t_k) = out_{\mathfrak{a}}$, a contradiction. Hence, no such a_1 can exist. Assuming there is an unlabeled argument $t_1 \in E_R^+$ analogously leads to a contradiction.

Finally, from (a) and (c) follows (d) and from (b) and (c) follows (e). □

Next we illustrate illustrates the idea of the function $\mathsf{propagate}_{\mathsf{U}}(SF, \lambda)$. In particular, it is easy to see that any partial labeling λ^* with $DEF(\lambda^*) = A$ that we give as input to $\mathsf{propagate}_{\mathsf{U}}$ will give us an admissible set (rule (3) removes conflicts and both rules (3) and (4) resolve undefended arguments).

Lemma 4.7. *Let $SF = (A, R)$ be a SETAF, and let λ^* be an arbitrary labeling s.t $DEF(\lambda^*) = A$. Set $\lambda^\dagger = \mathsf{propagate}_{\mathsf{U}}(SF, \lambda^*)$. Then $E = IN(\lambda^\dagger)$ is admissible in SF.*

Proof. We first show that $E \in cf(SF)$: assume towards contradiction that for some $(T, h) \in R$ it holds $T \cup h \subseteq E$. This means $\lambda^\dagger(h) = in_{\mathfrak{a}}$ and $\lambda^\dagger(t) = in_{\mathfrak{a}}$ for each $t \in T$. However, by rule (3) this means that we set $\lambda^\dagger(h) = und_{\mathfrak{a}}$, a contradiction. Hence, E is conflict-free in SF. Again towards contradiction assume there is an undefended $a \in E$, i.e., there is an attack $(T, a) \in R$ s.t. $T \cap E^+ = \emptyset$. Since $a \in E$ we know $\lambda^\dagger(a) = in_{\mathfrak{a}}$, and since we exhaustively applied rule (3) we know for each $(T, a) \in R$ it holds that $\exists t \in T : \lambda^\dagger(t) = out_{\mathfrak{a}}$. However, since we also applied rule (4) exhaustively, this means for each such t that there is some $(T', t) \in R$ with $T' \subseteq IN(\lambda^\dagger)$. Hence, E attacks each $\{t\}$ and therefore E also attacks each T that attacks a, which in turn means that a is indeed defended by E, a contradiction. Since E is conflict-free and defends every $a \in E$ in SF it is admissible. □

Finally, we show in Lemma 4.8 that by applying both functions $\mathsf{propagate}_{\mathsf{IO}}$ and $\mathsf{propagate}_{\mathsf{U}}$ consecutively, we indeed obtain a preferred extension that corresponds to the guess on B. An argument is incorrectly labeled $in_{\mathfrak{a}}$ if it is not defended against each attack; it is incorrectly labeled $out_{\mathfrak{a}}$ if it is not attacked from within the set $IN(\lambda)$. By exhaustively applying the propagation rules (3) and (4) we fix these incorrect labels. To illustrate this, we revisit our running example. In Figure 4.3 we see the resulting labelings λ_1^*, λ_2^* for the

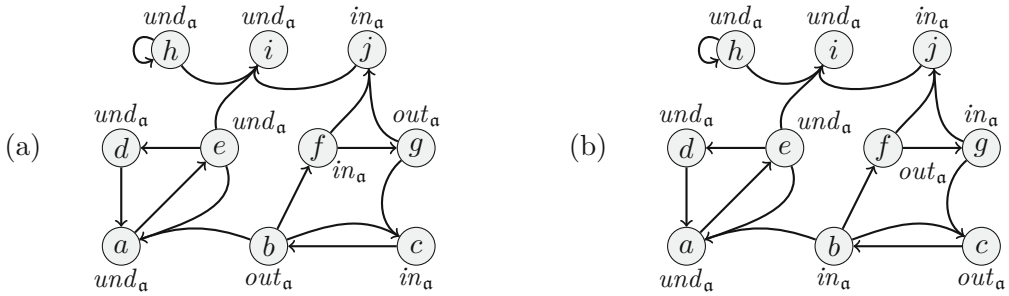


Figure 4.4: The second propagation $\mathsf{propagate}_{\mathsf{U}}$ (a) $\lambda_1^\dagger$, (b) $\lambda_2^\dagger$, computed on the labels from Figure 4.3(a) and (b), respectively.

backdoor $B = \{a, b\}$ for the guesses (a) $\lambda_1(a) = \lambda_1(b) = out_{\mathfrak{a}}$, and (b) $\lambda_2(a) = \lambda_2(b) = in_{\mathfrak{a}}$. Following Algorithm 2, in step 7 we set the arguments without a label to $und_{\mathfrak{a}}$ and compute $\mathsf{propagate}_{\mathsf{U}}$ for both labelings. In Figure 4.4 we see the resulting labelings $\lambda_1^\dagger, \lambda_2^\dagger$. Our algorithm outputs the sets $\{c, f, j\}, \{b, g, j\}$ respectively, and it can be indeed verified that these sets are preferred in SF. Now consider the guess $\lambda_3(a) = out_{\mathfrak{a}}, \lambda_3(b) = in_{\mathfrak{a}}$. It turns out that $\lambda_3^\dagger = \lambda_2^\dagger$: the fact that both labelings result in the same propagation means in particular that there is no preferred extension E where $\{a, b\} \subseteq E$, as by trying to construct such an extension in (b) we had to correct the label of a. In general, whenever we re-label one of the guessed $in_{\mathfrak{a}}$-labels, we can abort the run of the algorithm on this guess and proceed with a different guess, as we will always compute (a superset of) the thereby "missed" set when we start with the corresponding "correct" labeling on the backdoor arguments.

We show in the following Lemma 4.8 that for every argument a where we fix the label $und_{\mathfrak{a}}$, a is indeed neither in the corresponding extension nor attacked by it, i.e. $a \notin E_R^{\oplus}$. This together with the results of Lemma 4.6 suffices to show that if we guess correctly, the algorithm computes every preferred extension.

Lemma 4.8. *Let $SF = (A, R)$ be a SETAF, let $E \in pref(SF)$, and $B \subseteq A$ a NOEVEN-backdoor for SF. For the input (SF, B) to Algorithm 2, assume in step 2 we choose $I = E \cap B$, let λ^* be the corresponding propagated partial labeling from step 7 with $und_{\mathfrak{a}}$ labels. Set $\lambda^\dagger = \mathsf{propagate}_{\mathsf{U}}(SF, \lambda^*)$. Then $E = IN(\lambda^\dagger)$.*

Proof. We first show by induction on the number of re-labeled arguments (i.e., arguments that are labeled $und_{\mathfrak{a}}$ during the construction of $\lambda^\dagger$) that for each $a \in A$ it holds if $\lambda^\dagger(a) = und_{\mathfrak{a}}$ then $a \in A \setminus E_R^{\oplus}$. The base case where $\lambda^\dagger = \lambda^*$ is covered by Lemma 4.6(c). For the inductive step consider the rules (3) and (4):

Assume a is re-labeled by rule (3), i.e. we set $\lambda^\dagger(a) = und_{\mathfrak{a}}$ for some $a \in IN(\lambda^*)$. By Lemma 4.6(a) we know $a \notin E_R^+$, we show $a \notin E$. Since we invoked rule (3), there is $(T, a) \in R$ s.t. $\nexists t \in T$ with $\lambda^\dagger(t) = out_{\mathfrak{a}}$. By induction hypothesis and Lemma 4.6(a) and

(c), this means $E_R^+ \cap T = \emptyset$, i.e. a is not defended by E against (T, a). By admissibility this means $a \notin E$.

Assume a is re-labeled by rule (4), i.e. we set $\lambda^\dagger(a) = \mathit{und}_\mathfrak{a}$ for some $a \in \mathit{OUT}(\lambda^*)$. By Lemma 4.6(b) we know $a \notin E$, we show $a \notin E_R^+$. Since we invoked rule (4), there is no $(T, a) \in R$ s.t. $T \subseteq \mathit{IN}(\lambda^*)$. By induction hypothesis and Lemma 4.6(c), this means $T \not\subseteq E$, i.e. a is not attacked by E.

Next, we show $\mathit{IN}(\lambda^\dagger) \in \mathit{adm}(SF)$. By E' we identify the set $\mathit{IN}(\lambda^\dagger)$. Assume towards contradiction there is a conflicting attack $(T, h) \in R$ with $T \cup \{h\} \subseteq E'$. However, this means we would re-label h by rule (3), as there is no $t \in T$ with $\lambda^\dagger(t) = \mathit{out}_\mathfrak{a}$, a contradiction. Hence, $E' \in \mathit{cf}(SF)$. Now assume towards contradiction there is an undefended argument $a \in E'$, i.e., there is an attack $(T, a) \in R$ s.t. for no $t \in T$ there is an attack $(S, t) \in R$ with $S \subseteq E'$. As $a \in E'$, there is some $t \in T$ where either (i) we set $\lambda^*(t) = \mathit{out}_\mathfrak{a}$ during the computation of λ^* and did not change the label later, in which case we did not invoke propagation rule (4), and there is indeed an attack $(S, t) \in R$ towards t with $S \subseteq E'$ and a is defended, or (ii) we set $\lambda^*(t) = \mathit{out}_\mathfrak{a}$ during the computation of λ^* and during the computation of $\lambda^\dagger$ update it to $\lambda^\dagger(t) = \mathit{und}_\mathfrak{a}$, but then if no $t' \in T$ with $\lambda^\dagger(t') = \mathit{out}_\mathfrak{a}$ is left, we would have invoked propagation rule (3) for a and set it to $\mathit{und}_\mathfrak{a}$, and if such a t' exists where we did not change the label, then a is also defended as in case (i). In all cases we see that indeed a is defended by E' and can conclude $E' \in \mathit{adm}(SF)$.

Finally, by Lemma 4.6(d) and the formerly established fact that for each $a \in A$ it holds if $\lambda^\dagger(a) = \mathit{und}_\mathfrak{a}$ then $a \in A \setminus E_R^\oplus$ we know also $E \subseteq E'$. Since $E' \in \mathit{adm}(SF)$ and we assumed E is preferred, we get $E = E'$. □

These correctness results for the propagation procedures are the basis for the main result of this section, namely the efficient computation of preferred extensions if a suitable backdoor is given.

Theorem 4.9. *Let $\mathsf{C} \in \{\mathsf{NOEVEN}, \mathsf{ACYC}\}$ be a SETAF class, let $SF = (A, R)$ be a SETAF, and $B \subseteq A$ a C-backdoor for SF with $|B| \leq p$. With Algorithm 2 we can characterize all σ-extensions in time $2^p \cdot \mathsf{poly}(|SF|)$ for $\sigma \in \{\mathit{pref}, \mathit{stb}, \mathit{sem}\}$ on input (SF, B).*

Proof. Let $E \in \mathit{pref}(SF)$, we show that E is in the output of Algorithm 2, i.e., $E \in \mathit{pref}^*(SF, B)$. Since in step 2 we try all subsets of B, we will try $I = E \cap B$. Lemma 4.8 ensures $E \in \mathit{pref}^*(SF, B)$. It remains to show that all steps 3–10 can be done in polynomial time w.r.t. $|SF|$. It is easy to see that (assuming we use reasonable data structures) this is the case for steps 3–5, 7, 9, and 10. For step 6 and 8 note that each argument is (re)-labeled at most once, and the check for each propagation rule can clearly be carried out in polynomial time. Hence, the overall runtime is $2^p \cdot \mathsf{poly}(|SF|)$. Since $\mathit{stb}(SF) \subseteq \mathit{sem}(SF) \subseteq \mathit{pref}(SF)$, we indeed characterize all stable, semi-stable, and preferred extensions. □

Finally, we can then check whether the extensions are (range)-subset-maximal or attack every argument not in them, as all candidates are available. Hence, we can remove those sets that are not preferred/stable/semi-stable. "Filtering out" the stable extensions amounts to checking for each candidate whether the range covers all arguments, which can be done in polynomial time. To actually enumerate all preferred and semi-stable extensions and decide skeptical acceptance, we still have to filter out the solution candidates that are not (range-)subset-maximal. To this end, we can pairwise compare the solution candidates in quadratic runtime w.r.t. the number of solution candidates, which for enumerating and skeptical acceptance gives us a worst-case runtime of $O(4^p \cdot \mathsf{poly}(|SF|))$.

That is, from the fact that we can compute the candidate set in FPT we immediately get that also preferred and semi-stable extensions can be enumerated in FPT. A similar observation has been made in the AF-case [DOS12]. However, no details for the exact runtime bounds are provided. We next provide details on how to enumerate preferred extensions in $O(3^p \cdot \mathsf{poly}(|SF|))$, which is a significant improvement over the above $O(4^p \cdot \mathsf{poly}(|SF|))$ bound and the $O(9^p \cdot \mathsf{poly}(|SF|))$ bound one would get with the naive approach in the AF setting [DOS12]. The key observation here is that when comparing candidate labelings in order to decide whether one induces a superset of the other it suffices to consider the corresponding partial labelings on the backdoor set B. That is, a labeling λ_1 is a subset of λ_2 if the corresponding partial labelings λ'_1, λ'_2 over B satisfy the following:

- there is no $a \in B$ s.t. $\lambda'_1(a) \in \{in_{\mathfrak{a}}, out_{\mathfrak{a}}\}$, $\lambda'_2(a) \in \{in_{\mathfrak{a}}, out_{\mathfrak{a}}\}$, and $\lambda'_1(a) \neq \lambda'_2(a)$,
- there is some $a \in B$ s.t. $\lambda'_1(a) = und_{\mathfrak{a}}$ and $\lambda'_2(a) \in \{in_{\mathfrak{a}}, out_{\mathfrak{a}}\}$, and
- there is no $a \in B$ s.t. $\lambda'_2(a) = und_{\mathfrak{a}}$ and $\lambda'_1(a) \in \{in_{\mathfrak{a}}, out_{\mathfrak{a}}\}$.

That is, if one of the candidate labelings sets an argument of the backdoor to $out_{\mathfrak{a}}$ and a different candidate labeling sets it to $in_{\mathfrak{a}}$ then these two labelings correspond to incomparable sets. We can use this observation to enumerate preferred extensions as follows. First of all, we keep track over all 3-valued labelings over the backdoor and indicate its status of whether it is (1) a preferred extension, (2) admissible but a subset of a larger admissible set, (3) no admissible set at all, or (4) not yet considered. Starting with the labelings with no undecided labels, we then iterate over them, test whether they induce a complete extension, and update the corresponding status of the processed candidate, as well as the candidates which have one additional undecided argument and are compatible with the processed candidate (w.r.t. the conditions stated above). If the labeling is complete it is indeed also preferred (case 1) and we add it to the solution set and mark all the successors as superseded by it (case 2). If the set is not admissible at all we simply proceed with the next one (case 3). When done with the labelings with no undecided labels, we apply the same procedure to labelings with one undecided label and then continue by increasing the number of undecided arguments until we end up with the labeling that labels the whole backdoor undecided. The only difference is that we

additionally have to check whether the current labeling is already superseded by some other labeling and if so we propagate this status to its successors. Notice that we have $O(3^p)$ such labelings in total, and each solution candidate has at most p successors. That is, the above procedure is in $O(3^p \cdot \mathsf{poly}(|SF|))$.

Regarding semi-stable extensions, notice that the above approach does not work correctly, as it does not suffice to compare the partial labelings on B to compare the ranges of the corresponding extensions. So far, we do not see a way to overcome this even in the AF case, and hence, we only obtain the bound of $O(4^p \cdot \mathsf{poly}(|SF|))$ from the naive approach.

As we can efficiently characterize all preferred extensions, credulous reasoning also becomes efficient in this case. This result also carries over to admissible and complete semantics, as the respective credulous acceptance problems coincide with preferred semantics, $Skept_{com}$ is already possible in polynomial time and $Skept_{adm}$ is trivial. We want to highlight that even if we have a NOEVEN-backdoor of size p, there can be up to $O(3^p)$ complete extensions. Clearly we cannot enumerate them with our approach, which is why we capture the preferred extensions (in contrast to the AF approach [DOS12]).

Corollary 4.10. *Let $\mathsf{C} \in \{\mathsf{NOEVEN}, \mathsf{ACYC}\}$ be a SETAF class, let $SF = (A, R)$ be a SETAF, and $B \subseteq A$ a C-backdoor for SF with $|B| \leq p$. Then we can decide the problems $Skept_{stb}$ and $Cred_\sigma$ in time $2^p \cdot \mathsf{poly}(|SF|)$ for $\sigma \in \{adm, com, pref, stb\}$. Moreover, we can decide the problems $Cred_{sem}$ and $Skept_\sigma$ in time $4^p \cdot \mathsf{poly}(|SF|)$ for $\sigma \in \{pref, sem\}$.*

Combining Corollary 4.2 and Corollary 4.10, we immediately obtain the following result that captures finding and exploiting minimal backdoors for SETAFs.

Theorem 4.11. *Let SF be a SETAF and let $\sigma \in \{adm, com, pref, stb, sem\}$ be a semantics.*

1. *If $\mathsf{bd}_{\mathsf{ACYC}}(SF) \leq p$, we can solve $Cred_\sigma$ and $Skept_\sigma$ in FPT w.r.t. parameter p.*
2. *If $\mathsf{bd}_{\mathsf{NOEVEN}}(SF) \leq p$, we can solve $Cred_\sigma$ and $Skept_\sigma$ in XP w.r.t. parameter p.*

In fact, our algorithm for exploiting a small backdoor that is already given is only in 2^p for the parameter value p, which is an improvement over the existing 3^p approach [DOS12]. This also has implications in practice, as it turns out that finding the minimal backdoor-size often comes with high computational cost: for example, one known FPT algorithm for the computation of minimal NOEVEN-backdoors of size p for directed graphs G has runtime $4^p \cdot p! \cdot \mathsf{poly}(|G|)$ [CLL+08], which is often impractical even for rather small parameter values. Instead, recent implementations focus on finding backdoors *heuristically*, cf. [DHK+22]. Note that since our algorithm is directly applicable for AFs and dominates the state-of-the art 3^p approach for AFs, our results are also relevant for the development of fast algorithms for AFs.

Algorithm 3: Computation of $stb(SF)$ given B

```
1  stb(SF) ← ∅;
2  foreach I ⊆ B do
3      let λ be a partial labeling on B;
4      set λ(a) = in_𝔞 for a ∈ I;
5      set λ(a) = out_𝔞 for a ∈ B \ I;
6      λ* ← propagate_IO(SF, λ);
7      if DEF(λ*) = A then
8          λ† ← propagate*_U(SF, λ*);
9          if UND(λ†) = ∅ then
10             stb(SF) ← stb(SF) ∪ {IN(λ†)};
```

4.2.2 Speedup for Stable Semantics

If our goal is to compute the stable extensions or reason in stable semantics, we can in fact apply a shortcut to the computation. Algorithm 3 depicts the steps we perform to compute the stable extensions given a backdoor set B. Since for a stable extension E each argument is either in E or attacked by E, we do not need to compute undecided arguments. While we still need to call a function $\mathsf{propagate}_{\mathsf{U}}$ to detect conflicts or undefended arguments, we can stop as soon as we have to assign the label $und_{\mathfrak{a}}$ for the first time. For this reason we can use the updated function $\mathsf{propagate}^*_{\mathsf{U}}$ which we define as follows:

Definition 4.12. *Consider the following propagation rules for λ:*

(3) set $\lambda^\dagger(a) = und_{\mathfrak{a}}$ if $\lambda^\dagger(a) = in_{\mathfrak{a}}$ and there is $(T, a) \in R$ s.t. $\nexists\, t \in T : \lambda^\dagger(t) = out_{\mathfrak{a}}$,

(4) set $\lambda^\dagger(a) = und_{\mathfrak{a}}$ if $\lambda^\dagger(a) = out_{\mathfrak{a}}$ and there is no $(T, a) \in R$ s.t. $T \subseteq IN(\lambda^\dagger)$.

We define $\mathsf{propagate}^*_{\mathsf{U}}(SF, \lambda)$ as the result of initializing $\lambda^\dagger$ with λ on $DEF(\lambda)$, and if rule (3) or (4) is applicable to any argument, we apply the rule once *and return the resulting labeling.*

One can see that this function $\mathsf{propagate}^*_{\mathsf{U}}$ in Definition 4.12 differs from the $\mathsf{propagate}_{\mathsf{U}}$ of Definition 4.5 only in that the application of rules (3) and (4) happens at most once.

Theorem 4.13. *Let $\mathsf{C} \in \{\mathsf{NOEVEN}, \mathsf{ACYC}\}$ be a SETAF class, let $SF = (A, R)$ be a SETAF, and $B \subseteq A$ a C-backdoor for SF with $|B| \leq p$. With Algorithm 3 we can enumerate all stb-extensions in time $2^p \cdot \mathsf{poly}(|SF|)$.*

Effectively, Algorithm 3 will compute all sets that Algorithm 2 also computes that contain no undecided arguments. While the asymptotic runtime of this improved version is still

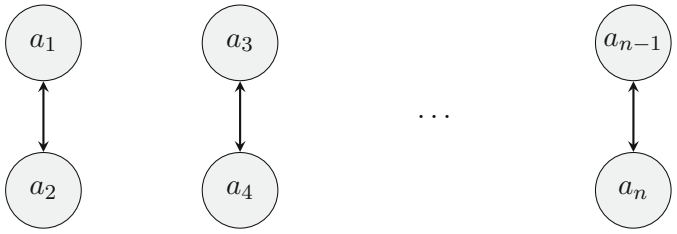


Figure 4.5: SETAF with n arguments and a minimal ACYC/NOEVEN-backdoor of size $p = n/2$. For each of the p pairs of mutually attacking arguments a_{2k}, a_{2k+1} a complete extension can contain either a_{2k} or a_{2k+1} or neither of the two, resulting in 3^p complete extensions.

$O(2^p \cdot \mathsf{poly}(|SF|))$, we obtain a speedup for the function $\mathsf{propagate}^*_{\mathsf{U}}$ (the runtime of which is hidden in the term $\mathsf{poly}(|SF|)$).

4.2.3 Enumerating Complete Extensions

In the following we present an algorithm to enumerate the complete extensions of a SETAF. Note that we cannot hope to enumerate the complete extensions in time $2^p \cdot \mathsf{poly}(|SF|)$ w.r.t. the NOEVEN- or ACYC backdoor of size p, as the example in Figure 4.5 illustrates: while the smallest backdoor is of size $p = n/2$, there are 3^p complete extensions. Intuitively, this is due to the fact that in contrast to preferred extensions, in a "choice" like the even-length cycles in Figure 4.5 complete extensions have the three possible outcomes for arguments that correspond to the argument being in, out, or undecided. Crucially, the arguments can be undecided even though via the efficient backdoor algorithm for preferred extensions it would be possible to find supersets of admissible arguments.

Hence, we generalize the "standard" backdoor evaluation approach of AFs by Dvořák et al. [DOS12] with a three valued guess on the backdoor arguments. The revised Algorithm 4 contains the same basic steps as the algorithm for preferred extensions. This means we can use similar strategies to show the correctness and completeness of the algorithm. To this end, we use $com(SF, B)$ to compute the complete extensions of a SETAF SF via backdoor B.

Again, we show that the two propagation algorithms $\mathsf{propagate}_{\mathsf{IO}}$ and $\mathsf{propagate}_{\mathsf{U}}$ perform as expected. First, for the application of $\mathsf{propagate}_{\mathsf{IO}}$ note that no $und_{\mathfrak{a}}$ label will be propagated (these labels will effectively be ignored). The only effect of these labels is "blocking" the application of a propagation rule that would "fire" in the algorithm for preferred semantics: in the example from Figure 4.5 if a_1 is a backdoor argument and is set to $out_{\mathfrak{a}}$, the argument a_2 is labeled as $in_{\mathfrak{a}}$ via $\mathsf{propagate}_{\mathsf{IO}}$. If on the other hand we set a_1 to $und_{\mathfrak{a}}$, no label will be assigned to a_2 via the exhaustive application of $\mathsf{propagate}_{\mathsf{IO}}$. An example of the execution of algorithm 4 is shown in Figure 4.6.

Algorithm 4: Computation of $com(SF, B)$

```
1  com(SF, B) ← ∅;
2  foreach partition (I, O, U) of B do
3      let λ be a partial labeling on B;
4      set λ(a) = in_𝔞 for a ∈ I;
5      set λ(a) = out_𝔞 for a ∈ O;
6      set λ(a) = und_𝔞 for a ∈ U;
7      λ* ← propagate_IO(SF, λ);
8      set λ*(a) = und_𝔞 for a ∈ A \ DEF(λ*);
9      λ† ← propagate_U(SF, λ*);
10     if IN(λ†) ∩ B = I and OUT(λ†) ∩ B = O then
11         com(SF, B) ← com(SF, B) ∪ {IN(λ†)};
```

Lemma 4.14. *Let $SF = (A, R)$ be a SETAF, let $E \in com(SF)$, and $B \subseteq A$ a* NOEVEN*-backdoor for SF. For the input (SF, B) to Algorithm 4, assume in step 2 we choose $I = E \cap B$, $O = E_R^+ \cap B$, and $U = (E \cap B) \setminus (I \cup O)$. Let λ be the corresponding partial labeling from steps 4, 5 and 6. Set $\lambda^* = \mathsf{propagate}_{\mathsf{IO}}(SF, \lambda)$. Then for each $a \in A$:*

(a) if $\lambda^*(a) = in_{\mathfrak{a}}$ then $a \notin E_R^+$,

(b) if $\lambda^*(a) = out_{\mathfrak{a}}$ then $a \notin E$,

(c) if $a \notin DEF(\lambda^*)$ then $a \notin E_R^{\oplus}$,

(d) $E \subseteq IN(\lambda^*)$, and

(e) $E_R^+ \subseteq OUT(\lambda^*)$.

Proof. We show (a) and (b) by induction on the number of labeled arguments in the construction of λ^*. For the base case $\lambda^* = \lambda$ it is easy to see that all conditions (a) and (b) hold (by assumption we have $IN(\lambda) = E \cap B$ and $OUT(\lambda) = E_R^+ \cap B$). For the step we consider the rules (1) and (2):

Assume a is labeled via rule (1), i.e. we set $\lambda^*(a) = out_{\mathfrak{a}}$ for some $a \in A \setminus DEF(\lambda)$. Clearly, (a) is not violated by labeling a as $out_{\mathfrak{a}}$, for (b) we show $a \notin E$. Since we invoked rule (1), there is an attack $(T, a) \in R$ with $T \subseteq IN(\lambda^*)$. By our induction hypothesis we know that (a) holds for each $t \in T$, i.e. $T \cap E_R^+ = \emptyset$. This means E does not defend a against the attack (T, a), and since $E \in com(SF)$, we get $a \notin E$.

Now assume a is labeled via rule (2), i.e. we set $\lambda^*(a) = in_{\mathfrak{a}}$ for some $a \in A \setminus DEF(\lambda)$. Clearly, (b) is not violated by labeling a as $in_{\mathfrak{a}}$, for (a) we show $a \notin E_R^+$. Since we invoked rule (2), for all attacks $(T, a) \in R$ there is some $t \in T$ with $\lambda^*(t) = out_{\mathfrak{a}}$. By our

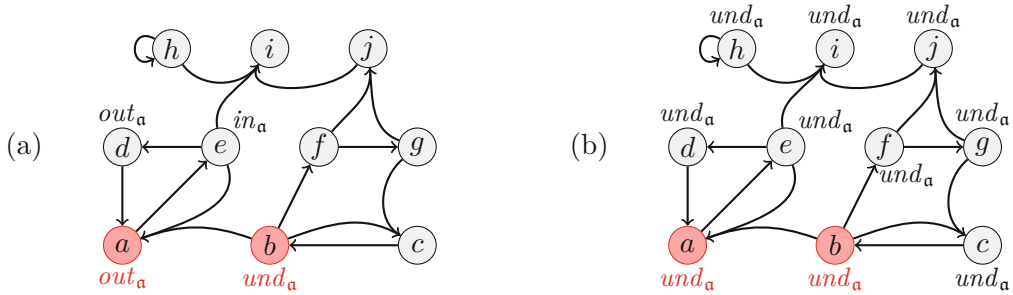


Figure 4.6: Example for the backdoor-algorithm for complete extensions. The backdoor $B = \{a, b\}$ and the guessed partition on B are highlighted. (a) shows the outcome of the application of $\mathsf{propagate}_{\mathsf{IO}}$, (b) shows the outcome of the application of $\mathsf{propagate}_{\mathsf{U}}$ and the characterized complete extension $\emptyset$.

induction hypothesis we know that (b) holds for each such t, i.e. $t \notin E$. Hence, there can be no attack (T, a) with $T \subseteq E$, i.e., $a \notin E_R^+$.

For (c) assume towards contradiction there is an argument $a_1 \in E$ such that $a_1 \notin DEF(\lambda^*)$. If there is no attack $(T, a_1) \in R$ towards a_1, we would have $\lambda^*(a_1) = in_{\mathfrak{a}}$, hence, there is such an attack. Moreover, there is a $(T, a_1) \in R$ s.t. for no $t \in T$ we have $\lambda^*(t) = out_{\mathfrak{a}}$, otherwise we would have $\lambda^*(a_1) = in_{\mathfrak{a}}$. However, by admissibility of E there is at least one $t_1 \in T \cap E_R^+$. For this t_1 we have $t_1 \in (A \setminus DEF(\lambda^*)) \cup UND(\lambda^*)$, i.e. t_1 is unlabeled or labeled $und_{\mathfrak{a}}$ (the only other option, the label $in_{\mathfrak{a}}$, violates (b)). Since $t_1 \in E_R^+$, there is an attack $(S, t_1) \in R$, but since t_1 is unlabeled or labeled $und_{\mathfrak{a}}$, $S \not\subseteq IN(\lambda^*)$, i.e., there is an $a_2 \in S$ such that a_2 is unlabeled or labeled $und_{\mathfrak{a}}$. We have $a_1 \neq a_2$, as this would imply an even-length cycle (a_2, t_1): if $a_1 = a_2$ then either a_2 or t_1 are in B and by assumption we have that these arguments are labeled in accordance to E (and are not re-labeled by $\mathsf{propagate}_{\mathsf{IO}}$), a contradiction. As for a_2 we can reason in the same way as for a_1, we obtain another unlabeled or $und_{\mathfrak{a}}$-labeled argument $t_2 \in E_R^+$, and eventually a sequence $a_1, t_1, a_2, t_2, \ldots$ of arguments. However, as SF is finite and all a_i are different, eventually there is either (i) an unlabeled or $und_{\mathfrak{a}}$-labeled argument a_k where no attack (T_k, a_k) towards a_k has an unlabeled (or $und_{\mathfrak{a}}$-labeled) $t_k \in T_k$, but then $\lambda^*(a_k) = in_{\mathfrak{a}}$, a contradiction, or (ii) an unlabeled argument t_k where there is no counter-attack (S_k, t_k) with an unlabeled or $und_{\mathfrak{a}}$-labeled $a_{k+1} \in S_k$, but then $\lambda^*(t_k) = out_{\mathfrak{a}}$, a contradiction. Hence, no such a_1 can exist. Assuming there is an unlabeled argument $t_1 \in E_R^+$ analogously leads to a contradiction.

Finally, from (a) and (c) follows (d) and from (b) and (c) follows (e) (recall that the only arguments that are labeled $und_{\mathfrak{a}}$ at this point are arguments in B, which are labeled in accordance with E by assumption). □

Note that after step 8 the only arguments with an $und_{\mathfrak{a}}$ label are the ones in B we

initially labeled this way due to our three valued guess. In step 9 we apply the $und_{\mathfrak{a}}$ label to the remaining (unlabeled) arguments and again fix the mislabeled arguments.

Lemma 4.15. *Let $SF = (A, R)$ be a SETAF, let $E \in com(SF)$, and $B \subseteq A$ a NOEVEN-backdoor for SF. For the input (SF, B) to Algorithm 4, assume in step 2 we choose $I = E \cap B$, $O = E_R^+ \cap B$, and $U = (E \cap B) \setminus (I \cup O)$. Let λ^* be the corresponding propagated partial labeling from step 8 with $und_{\mathfrak{a}}$ labels. Set $\lambda^\dagger = \mathsf{propagate}_U(SF, \lambda^*)$. Then $E = IN(\lambda^\dagger)$.*

Proof. We first show by induction on the number of re-labeled arguments (i.e., arguments that are labeled $und_{\mathfrak{a}}$ during the construction of $\lambda^\dagger$) that for each $a \in A$ it holds if $\lambda^\dagger(a) = und_{\mathfrak{a}}$ then $a \in A \setminus E_R^\oplus$. The base case where $\lambda^\dagger = \lambda^*$ is covered by Lemma 4.14(c) and our assumption that $U = (E \cap B) \setminus (I \cup O)$, i.e., the backdoor arguments that are undecided w.r.t. E are labeled correctly (note that these are never relabeled). For the inductive step consider the rules (3) and (4):

Assume a is re-labeled by rule (3), i.e. we set $\lambda^\dagger(a) = und_{\mathfrak{a}}$ for some $a \in IN(\lambda^*)$. By Lemma 4.14(a) we know $a \notin E_R^+$, we show $a \notin E$. Since we invoked rule (3), there is $(T, a) \in R$ s.t. $\nexists\, t \in T$ with $\lambda^\dagger(t) = out_{\mathfrak{a}}$. By induction hypothesis and Lemma 4.14(a) and (c), this means $E_R^+ \cap T = \emptyset$, i.e. a is not defended by E against (T, a). By admissibility this means $a \notin E$.

Assume a is re-labeled by rule (4), i.e. we set $\lambda^\dagger(a) = und_{\mathfrak{a}}$ for some $a \in OUT(\lambda^*)$. By Lemma 4.14(b) we know $a \notin E$, we show $a \notin E_R^+$. Since we invoked rule (4), there is no $(T, a) \in R$ s.t. $T \subseteq IN(\lambda^*)$. By induction hypothesis and Lemma 4.14(c), this means $T \not\subseteq E$, i.e. a is not attacked by E.

Next, we show $IN(\lambda^\dagger) \in adm(SF)$. By E' we identify the set $IN(\lambda^\dagger)$. Assume towards contradiction there is a conflicting attack $(T, h) \in R$ with $T \cup \{h\} \subseteq E'$. However, this means we would re-label h by rule (3), as there is no $t \in T$ with $\lambda^\dagger(t) = out_{\mathfrak{a}}$, a contradiction. Hence, $E' \in cf(SF)$. Now assume towards contradiction there is an undefended argument $a \in E'$, i.e., there is an attack $(T, a) \in R$ s.t. for no $t \in T$ there is an attack $(S, t) \in R$ with $S \subseteq E'$. As $a \in E'$, there is some $t \in T$ where either (i) we set $\lambda^*(t) = out_{\mathfrak{a}}$ during the computation of λ^* and did not change the label later, in which case we did not invoke propagation rule (4), and there is indeed an attack $(S, t) \in R$ towards t with $S \subseteq E'$ and a is defended, or (ii) we set $\lambda^*(t) = out_{\mathfrak{a}}$ during the computation of λ^* and during the computation of $\lambda^\dagger$ update it to $\lambda^\dagger(t) = und_{\mathfrak{a}}$, but then if no $t' \in T$ with $\lambda^\dagger(t') = out_{\mathfrak{a}}$ is left, we would have invoked propagation rule (3) for a and set it to $und_{\mathfrak{a}}$, and if such a t' exists where we did not change the label, then a is also defended as in case (i). In all cases we see that indeed a is defended by E' and can conclude $E' \in adm(SF)$.

Finally, by Lemma 4.14(d) and (e) and since we do not falsely relabel these arguments the formerly established fact that for each $a \in A$ it holds if $\lambda^\dagger(a) = und_{\mathfrak{a}}$ then $a \in A \setminus E_R^\oplus$ we get (1) $E \subseteq IN(\lambda^\dagger) = E'$, (2) $E^+ \subseteq OUT(\lambda^\dagger)$, and (3) $A \setminus E^\oplus \supseteq UND(\lambda^\dagger)$. From this we obtain $E = E'$. □

Finally, combining Lemma 4.14 and Lemma 4.15 we obtain similar to Theorem 4.9 the following enumeration result for complete extensions. The final check in step 11 ensures that we only keep the complete extensions that correspond to the initially guessed partition of B. Note that due to the 3-valued guess we obtain the factor 3^p.

Theorem 4.16. *Let $C \in \{\mathsf{NOEVEN}, \mathsf{ACYC}\}$ be a SETAF class, let $SF = (A, R)$ be a SETAF, and $B \subseteq A$ a C-backdoor for SF with $|B| \leq p$. With Algorithm 4 we can enumerate all com-extensions in time $3^p \cdot \mathsf{poly}(|SF|)$ on input (SF, B).*

Proof. Let $E \in com(SF)$, we show that E is in the output of Algorithm 4, i.e., $E \in com(SF, B)$. Since in step 2 we try all partitions of B, we will try $I = E \cap B$, $O = E^+ \cap B$, $U = B \setminus (I \cup O)$. Lemma 4.15 ensures $E \in com(SF, B)$. It remains to show that all steps 3–11 can be done in polynomial time w.r.t. $|SF|$. It is easy to see that (assuming we use reasonable data structures) this is the case for steps 3–6, 8, 10, and 11. For step 7 and 9 note that each argument is (re)-labeled at most once, and the check for each propagation rule can clearly be carried out in polynomial time. Finally, checking whether $E \in com(SF, B)$ is complete can be done in polynomial time. Hence, the overall runtime is $3^p \cdot \mathsf{poly}(|SF|)$. □

4.3 Conditional Lower Bounds for Backdoor Evaluation

In this section we show a so-called conditional lower bound [AW14] for NOEVEN/ACYC-backdoor evaluation, i.e. we show that our algorithm is basically optimal based on some well-known conjecture. The conjecture we are going to use is the *Strong exponential-time hypothesis (SETH)* [IP99, IPZ98]. We show this for AFs; the result carries over to SETAFs.

Conjecture 4.17 (Strong Exponential Time Hypothesis (SETH))**.** *For each $\epsilon > 0$ there is a k such that k-CNF-SAT on n variables and m clauses cannot be solved in $O(2^{(1-\epsilon)n} \cdot \mathsf{poly}(n + m))$ time.*

Let p be the parameter for the backdoor size. We will show that any NOEVEN-backdoor evaluation that runs in time $O(2^{(1-\epsilon)p} \cdot \mathsf{poly}(|F|))$ for AFs F would violate SETH and thus imply a major break-through in the development of SAT algorithms.

Theorem 4.18. *Let $F = (A, R)$ be an AF and let $C \in \{\mathsf{NOEVEN}, \mathsf{ACYC}\}$ and let p the size of the given backdoor. There is no $O(2^{(1-\epsilon)p} \cdot \mathsf{poly}(|A|))$ C-backdoor evaluation algorithm for $Cred_\sigma$ for $\sigma \in \{adm, com, pref, stb, sem\}$ unless SETH is false.*

Proof. Given an instance from SETH, i.e. a CNF formula φ with n variables and m clauses. Let $X = \{x_1, \ldots, x_n\}$ be the set of variables and $C = \{c_1, \ldots, c_m\}$ be the set of clauses appearing in φ. Consider the standard translation [DD18] from a CNF formula φ to an AF $F_\varphi = (A, R)$ (cf. Reduction 2.20). We know that the formula φ is satisfiable

iff the argument φ is credulously accepted w.r.t. σ [DD18]. Moreover, we have that the set X is a C-backdoor of F_φ and has size n.

Towards, a contradiction let us assume we have a $O(2^{(1-\epsilon)p} \cdot \mathsf{poly}(|A|))$ C-backdoor evaluation algorithm. Then we can decide whether a CNF formula φ is satisfiable as follows: We first construct the AF F_φ (which is polynomial in $n+m$) and then run the C-backdoor evaluation with backdoor X and return the answer for the credulous decision problem as answer to the satisfiability problem. By assumption the latter step has a running time of $O(2^{(1-\epsilon)n} \cdot \mathsf{poly}(n+m))$. That is, we have a $O(2^{(1-\epsilon)n} \cdot \mathsf{poly}(n+m))$ algorithm for k-CNF-SAT for arbitrary $k > 1$, which contradicts SETH. □

4.4 Discussion

In this chapter we studied the applicability of the backdoor concept for SETAFs. We investigated this on the primal-graph of a SETAF, which is arguably the most natural starting place for such investigations. However, in this thesis we also investigate the incidence-graph (cf. Definition 2.12), and in the following we will briefly argue why it suffices to analyze the primal graph. First note that the incidence graph is trivially bipartite by construction, and symmetric edges in the incidence graph correspond to self-attacks, which means backdoors to these classes *on* the incidence graph are nonsensical. Moreover, every (even-length) cycle in a SETAF SF corresponds to an equivalent cycle in its incidence graph $\mathtt{Inc}(SF)$ with the same arguments and (parts of) attacks involved. Finally, note that for backdoors to ACYC and NOEVEN on the incidence-graph we do not need to consider removing nodes corresponding to *attacks*: it is always at least as good to remove e.g. the head of an attack. This means, given a backdoor on the incidence-graph that contains a node that corresponds to an attack (T, h), clearly instead having the node corresponding to the argument h in the backdoor "breaks" all cycles the original backdoor does, as well as potentially more. Hence, it suffices even on the incidence-graph to focus on backdoors containing only arguments. But then we know that (since the cycles in SF and $\mathtt{Inc}(SF)$ coincide) the backdoors coincide as well. Hence, if we allow our notation of $\mathsf{bd}()$ to be applicable for arbitrary directed (hyper-)graphs, we obtain the following result.

Proposition 4.19. *Let SF be a SETAF. It holds $\mathsf{bd}_\mathsf{C}(SF) = \mathsf{bd}_\mathsf{C}(\mathtt{Inc}(SF))$ with $\mathsf{C} \in \{\mathsf{NOEVEN}, \mathsf{ACYC}\}$.*

Given this result, it is clear that it suffices to focus on backdoors in the primal graph.

In this chapter, we established a new approach for enumerating the extensions and for reasoning on a SETAF, which improves the runtime of the best known approaches for AFs. Moreover, unless the Strong Exponential Time Hypothesis is false, our approach is optimal in this regard. In the case of enumerating complete extensions we match the runtime of the state-of-the-art for AFs and argue why a similar speedup as for the other semantics is impossible. We focus in this chapter on ways to exploit the structure of

the primal graph, as this boils down to finding backdoors in a directed graph. Since for directed hypergraphs much less literature on this topic is available, this approach seems most promising regarding implementations. For example, finding an ACYC-backdoor boils down to the directed feedback vertex set problem, which has been studied in great detail since it was first shown to be NP-complete by Karp in 1972 [Kar72].

In summary, we have shown that the idea of backdoors is indeed applicable to SETAFs in all cases that are known to work for AFs. Moreover, we have found another instance where the closer look on the more detailed structure of a SETAF (compared to an AF) payed off, as our new algorithm improves the state-of-the-art, in that with our novel technique preferred extensions can be characterized in time $O(2^p \cdot \mathsf{poly}(|SF|))$ instead of $O(3^p \cdot \mathsf{poly}(|SF|))$ (as in the approach of Dvořák et al. [DOS12]) for a backdoor of size p in a SETAF SF.

CHAPTER 5

Treewidth-Based Evaluation

In this chapter we discuss another approach for efficient reasoning on SETAFs. Here, our focus lies on the parameter *treewidth*. Intuitively, low treewidth indicates a certain "tree-likeness" of a graph, and since many problems become easy on trees we can exploit this tree-like structure. The key idea is to compute sub-problems for a given task, and carry out the sub-problems iteratively. The sub-problems correspond to evaluating a part of the SETAF and are organized in a tree-structure, the so called *tree-decomposition*. This way, we do not have a solution space that is exponential in the size of the whole framework, but only in the size of the largest sub-problem (note also that the number of sub-problems, i.e., *bags* of the tree-decomposition, is at most linear in the size of the SETAF). Finally, we can efficiently combine the solutions of all sub-problems and obtain answers for the original task on the whole SETAF.

In AFs, it has been shown that reasoning is fixed-parameter tractable w.r.t. the parameter treewidth [Dun07, DPW12, DSW12]. Also in the field of structured argumentation, the parameter treewidth has recently been investigated w.r.t. assumption-based argumentation by Popescu and Wallner [PW23], who show fixed-parameter tractability for several reasoning problems via monadic second-order logic and tailored algorithms, similarly to the work we present in Section 5.1.2. We investigate how this notion of treewidth is applicable to the *directed hypergraph*-structure of SETAFs and show that certain generalizations admit FPT algorithms, while other reasonable attempts do not. In particular, we show that the approach via the primal-graph that served as a starting point for SCC-recursiveness as well as our backdoor algorithms does not work for treewidth, i.e., we still obtain NP-hardness for a constant primal-treewidth. However, based on the *incidence-graph* of a SETAF we can indeed apply the treewidth approach to obtain FPT-algorithms for reasoning. The incidence-graph is the bipartite graph with all arguments and all attacks as nodes, and edges that indicate an argument's involvement in an attack (be it as the head or an argument of the tail of the attack). While several SETAFs can map to the same primal-graph, the incidence-graph fully captures the structure of the

SETAF, which is why it should not come as a surprise that for this task the primal-graph is insufficient.

In this chapter, we establish that low (incidence-)treewidth can indeed be exploited to reason in FPT first via generic results. We then provide detailed algorithms tailored to stable, admissible, and complete semantics which provide an improved runtime over the generic results. These cases exemplify the concepts we face also for other semantics and can be seen as a showcase for the remaining cases, which comprise of the same concepts. During this analysis we show that our tailor-made algorithms in fact yield a theoretical runtime improvement over the state-of-the-art in AFs, once again pointing out how beneficial it can be to take a close look at the structure of the frameworks by investigating the syntactically richer SETAFs (compared to AFs).

This chapter is organized as follows.

- We start in Section 5.1 by introducing our notions of treewidth for SETAFs, based on the primal-graph and the incidence-graph, respectively. We proceed by first presenting negative results regarding primal-treewidth, namely that reasoning remains intractable even for small parameter values (Section 5.1.1). Section 5.1.2 establishes FPT results for incidence-treewidth via a generic argument utilizing Monadic Second Order logic; this theoretical result gets refined and improved in Section 5.2 where we discuss a dynamic programming algorithm tailored to SETAFs.
- In Section 5.2 we present the foundation for our tailor-made algorithms for stable, admissible, and complete semantics.
- In Section 5.3 we illustrate how to characterize stable extensions via dynamic programming, and illustrate corner cases and interesting novel ideas with extended examples and detailed proofs.
- Section 5.4 extends the ideas we use for stable extensions by considering undecidedness to provide an algorithm to characterize admissible extensions. Moreover, we discuss how our DP algorithms can be used for counting extensions.
- Finally, in Section 5.5 we combine the ideas of the approach for stable and admissible extensions to characterize complete extensions. We achieve this by incorporating the concept of provisional colors for undecidedness as well, and ultimately provide a DP algorithm for complete semantics.
- In Section 5.6 we discuss our findings regarding the concept of treewidth for SETAFs.

This chapter is based on [DKW24] (which is currently under review), which in turn contains and extends the content of [DKW22b].

5.1 Towards SETAF Treewidth

In this section we discuss approaches to apply the notion of *treewidth* [RS86] to SETAFs. Intuitively, the treewidth of a graph measures the "tree-likeness" of a graph. Since most reasoning problems remain tractable on trees, we can exploit this structure to construct efficient algorithms—given the graph is sufficiently "tree-like", i.e., has a low *treewidth*. On AFs, it has been shown that treewidth is indeed a parameter that allows for FPT-algorithms for reasoning [DPW12], i.e., the runtime is exponential only in the parameter value, but polynomial in the size of the framework. We will show that these results carry over to SETAFs, when we apply treewidth to the "right" underlying graph that adequately captures the intricate structure of SETAFs.

Definition 5.1 (Treewidth)**.** *Let $G = (V, E)$ be an undirected graph. A* tree decomposition (TD) *of G is a pair $(\mathcal{T}, \mathcal{X})$, where $\mathcal{T} = (V_\mathcal{T}, E_\mathcal{T})$ is a tree and $\mathcal{X} = (X_n)_{n \in V_\mathcal{T}}$ is a set of bags (a bag is a subset of V) s.t.*

1. *$\bigcup_{n \in V_\mathcal{T}} X_n = V$;*
2. *for each $v \in V$, the subgraph induced by v in $\mathcal{T}$ is connected; and*
3. *for each $\{v, w\} \in E$, $\{v, w\} \subseteq X_n$ for some $n \in V_\mathcal{T}$.*

The width *of a TD is $\max\{|X_n| \mid n \in V_\mathcal{T}\} - 1$, the* treewidth *of G is the minimum width of all TDs for G.*

In words, a tree decomposition captures the structure of a given graph $G = (V, E)$ by constructing a tree $\mathcal{T} = (V_\mathcal{T}, E_\mathcal{T})$. To each node $n \in V_\mathcal{T}$ we assign a bag X_n containing vertices of the original graph s.t.

1. all original vertices are in at least one bag;
2. whenever a vertex $v \in V$ appears in two bags X_n and $X_{n'}$, the corresponding nodes $n, n'' \in V_\mathcal{T}$ are connected in $\mathcal{T}$ via $E_\mathcal{T}$ such that all nodes in between also have v in their corresponding bags; and
3. for each edge $(v, w) \in E$ of the original graph, for at least one node $n \in V_\mathcal{T}$ its corresponding bag X_n contains both v and w.

As the treewidth is the smallest size of the largest bag minus 1 over all possible tree decompositions, trivially each graph $G = (V, E)$ has treewidth of at most $|V| - 1$ (since it is always possible to put all vertices in a single bag of a tree decomposition). However, the lower the treewidth, the more tree-like a graph is by this measure. In particular, for trees we obtain a treewidth of 1.

For fixed k it can be decided in linear time whether a graph has treewidth at most k; moreover an according tree decomposition can be computed in linear time [Bod96]. For

practical applications there are heuristic approaches available that will return decompositions of reasonable width very efficiently [AMW17]. However, as the underlying structure of SETAFs is a *directed hypergraph*, this notion is not directly applicable in our context. We can use "standard" directed graphs to describe SETAFs, and apply treewidth by simply ignoring the direction of the involved arcs. For SETAFs there is the *primal graph* and the *incidence graph* as such notions, each of which leads to its own treewidth notion for SETAFs. First, we utilize the primal graph to define *primal-treewidth* (recall Definition 2.10).

Definition 5.2 (Primal Treewidth)**.** *Let SF be a SETAF and primal(SF) its primal graph. The* primal-treewidth $\texttt{ptw}(SF)$ *is defined as the treewidth of primal(SF).*

As mentioned earlier, several SETAFs can map to the same primal graph. However, restrictions on the primal graph are often useful to obtain computational speedups for otherwise hard problems [DKW21a, DKW21b]. In contrast, the *incidence graph* uniquely describes a SETAF, as attacks are explicitly modeled in this notion. Again, we utilize the incidence graph to define *incidence-treewidth*.

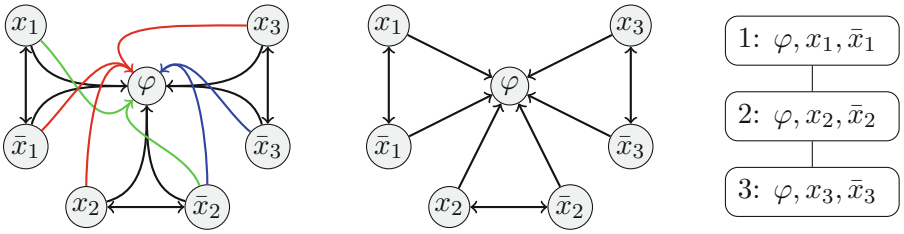
Definition 5.3 (Incidence Treewidth)**.** *Let SF be a SETAF and $\texttt{Inc}(SF)$ its incidence-graph. The* incidence-treewidth $\texttt{itw}(SF)$ *is defined as the treewidth of $\texttt{Inc}(SF)$.*

We want to highlight that (a) both of these notions properly generalize the classical notion of treewidth commonly applied to Dung-style AFs, and (b) these measures coincide on AFs. Formally:

Proposition 5.4. *The "standard" treewidth of AFs F coincides with $\texttt{ptw}(F)$ and $\texttt{itw}(F)$.*

Proof. The case of primal-treewidth is immediate, as $\textit{primal}(F) = F$. For incidence-treewidth note that we can construct $\texttt{Inc}(F)$ from F by substituting each edge $r = (a, b) \in R$ by a fresh vertex r and two edges (a, r), (r, b). It is well known that this operation preserves treewidth. □

We will first show that reasoning on SETAFs with fixed primal-treewidth remains hard (Section 5.1.1). Incidence-treewidth on the other hand admits FPT algorithms—we will initially establish this by characterizing the SETAF semantics via Monadic Second Order logic (MSO) [Dun07, DSW12] (Section 5.1.2). We utilize this characterization to obtain the desired upper bounds, as in this context we can apply a meta-theorem due to Courcelle [Cou87, Cou90]. In a nutshell, it states that every graph property that can be characterized in MSO can be checked in polynomial time. However, this generic method typically produces infeasible constants in practice, which is why in Section 5.2 we will refine these results and provide a prototypical algorithm with feasible constants for stable semantics, later we refine our methods for admissible sets and complete extensions. (cf. [DPW12]).



(a) (b) (c)

Figure 5.1: (a) The framework SF_φ from the proof of Theorem 5.5 for $\varphi = (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_2) \wedge (x_2 \vee x_3)$, (b) $primal(SF_\varphi)$, and (c) a tree-decomposition of $primal(SF_\varphi)$ of width 2.

5.1.1 Decomposing the Primal Graph

We start with an investigation of the treewidth of the primal graph. It has been shown that various restrictions on the primal graph can lead to computational ease [DKW21a, DKW21b]. However, we will see that reasoning remains hard for SETAFs with constant primal-treewidth (in contrast to the AF-case, where we observe FPT results). We establish this via reductions from (QBF-)SAT, illustrated in Figures 5.1 and 5.2. Intuitively, the attacks between the dual literals x and $\bar{x}$ represent the choice between assigning x to *true* or *false*. The collective attack $(\{x, \bar{x}\}, \varphi)$ ensures that we take at least one of x and $\bar{x}$ into any extension in order to defend φ, making sure we only construct proper truth assignments. Finally, the remaining attacks towards φ correspond to the clauses and make sure that we cannot defend φ if for any clause we set all duals of its literals *true*—as this means the clause is not satisfied.

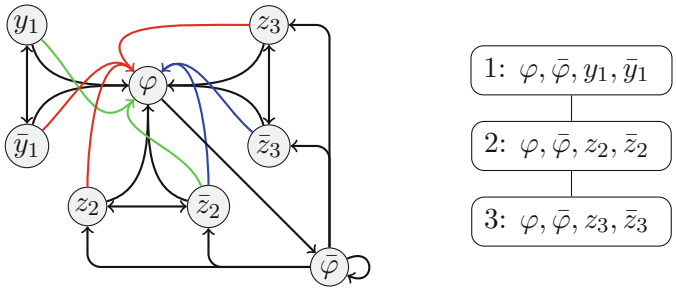
Theorem 5.5. *The problems $Cred_\sigma$ for $\sigma \in \{adm, com, stb, pref\}$ are* NP*-complete, and $Skept_{stb}$ is* coNP*-complete for SETAFs SF with* $\mathtt{ptw}(SF) \geq 2$.

Proof. The membership coincides with the general case. For the respective hardness results, consider the following reduction from SAT (see Figure 5.1). Let X be the set of atoms and C be the set of clauses of a boolean formula φ (given in CNF). We denote a clause $c \in C$ as the set of literals in the clause, e.g. the clause $x_1 \vee \bar{x}_2 \vee \bar{x}_3$ correspond to the set of literals (arguments, resp.) $\{x_1, \bar{x}_2, \bar{x}_3\}$. By x^d we denote the dual of a literal (e.g. $x^d = \bar{x}$ and $\bar{x}^d = x$). Let $SF_\varphi = (A, R)$, where $A = \{x, \bar{x} \mid x \in X\} \cup \{\varphi\}$, and

$$R = \{(\{x^d \mid x \in c\}, \varphi) \mid c \in C\} \cup \{(\{x, \bar{x}\}, \varphi), (x, \bar{x}), (\bar{x}, x) \mid x \in X\}$$

Now it holds that φ is in at least one σ extension for $\sigma \in \{adm, com, stb, pref\}$ if and only if φ is satisfiable.

($\Rightarrow$) An admissible set E containing φ contains exactly one of each pair $x, \bar{x}$, as otherwise φ would not be defended against the attack $(\{x, \bar{x}\}, \varphi)$. Moreover, as $\bar{c} \not\subseteq E$ for each attack $(\bar{c}, \varphi)$—$\bar{c}$ consists of the *duals* of the literals in c—this means at least one argument



(a) (b)

Figure 5.2: (a) SF_Φ from the proof of Theorem 5.6 for $\Phi = \forall\{y_1\}\exists\{z_2, z_3\}(y_1 \vee \bar{z}_2 \vee \bar{z}_3) \wedge (\bar{y}_1 \vee z_2) \wedge (z_2 \vee z_3)$, and (b) a tree-decomposition of $primal(SF_\Phi)$ of width 3.

corresponding to a literal of each clause $c \in C$ is in E. Hence, E corresponds to a satisfying assignment of φ.

($\Leftarrow$) Every satisfying assignment of φ corresponds to a stable extension of SF_φ: let I be the interpretation satisfying φ, the corresponding set $E = \{\varphi\} \cup \{x \mid I(x) = \mathit{true}\} \cup \{\bar{x} \mid I(x) = \mathit{false}\}$ is then stable (admissible, complete, preferred): As I satisfies φ, for each attack corresponding to a clause not all tail-arguments are in E, and hence φ is defended.

For the coNP completeness result, we add an argument $\bar{\varphi}$ and an attack $(\varphi, \bar{\varphi})$. If φ is unsatisfiable, φ will be attacked and $\bar{\varphi}$ will be contained in every stable extension. The constant primal-treewidth is immediate, as illustrated in the example of Figure 5.1(c). □

Also for preferred semantics reasoning remains intractable for SETAFs with fixed primal-treewidth. For this result, we extend the construction from Theorem 5.5 by an additional argument $\bar{\varphi}$ that attacks the existentially quantified variables (see Figure 5.2).

Theorem 5.6. *$Skept_{pref}$ is Π_2^P-complete for SETAFs SF with* $\mathtt{ptw}(SF) \geq 3$.

Proof. We show this by a reduction from the Π_2^P-complete $QBF_\forall^2$ problem. Let $\Phi = \forall Y \exists Z \varphi$ be a $QBF_\forall^2$-formula with φ in CNF. We construct the SETAF SF_Φ by extending F_φ from Theorem 5.5 in the following way (for an example see Figure 5.2): First, we set $X = Y \cup Z$ and add all arguments and attacks according to the construction of F_φ. Moreover we add an argument $\bar{\varphi}$ and attacks $(\bar{\varphi}, \bar{\varphi}), (\varphi, \bar{\varphi})$. The last step is to add attacks $(\bar{\varphi}, z), (\bar{\varphi}, \bar{z})$ for each $z \in Z$. Now φ is in every preferred extension of SF_Φ if and only if Φ is valid. We start with some general observations: $\bar{\varphi}$ cannot be in any admissible set, and φ can only be in an admissible set S if for each $x \in Y \cup Z$ exactly one of x and $\bar{x}$ is in S. Moreover, in order to have $S \cap (Z \cup \bar{Z}) \neq \emptyset$, the argument $\bar{\varphi}$ has to be attacked by S, and consequently $\varphi \in S$. This means for every admissible set S that $S \cap (Y \cup \bar{Y} \cup Z \cup \bar{Z})$ corresponds to a satisfying assignment of the formula φ. In summary, every assignment on the variables Y corresponds to an admissible set, and every other admissible set in SF_Φ contains φ and represents a satisfying assignment for φ.

($\Rightarrow$) Assume φ is in every preferred extension. Since every set $S \subseteq 2^Y$ is admissible and in order to accept φ for each $x \in Y \cup Z$ either x or $\bar{x}$ have to be accepted, we know that for every assignment of Y variables there is an assignment satisfying φ.

($\Leftarrow$) Now assume Φ is valid, i.e. for each assignment I_Y on the variables Y, there is an assignment I_Z on Z such that $I_Y \cup I_Z$ satisfies φ. From this and our observations above it follows that φ is in every preferred extensions.

It is easy to see that the primal-treewidth of F_Φ is bounded by 3 (see Figure 5.2(b)). □

Hence, under standard complexity-theoretical assumptions, these problems do not become tractable when parameterized by the primal-treewidth.

5.1.2 Parameterized Tractability via Incidence-Treewidth

In this section, we establish tractability for reasoning in SETAFs with constant incidence treewidth by utilizing a meta-theorem due to Courcelle [Cou87, Cou90]. In particular, we use the tools of Monadic Second Order logic (MSO) to characterize the semantics of SETAFs (similarly, this has been done for AFs [Dun07, DSW12]). MSO generalizes first-order logic in the sense that it is also allowed to quantify over sets. Domain elements in our settings are vertices of an (incidence)-graph, i.e., arguments or attacks. MSO in our context consists of *variables* corresponding to domain elements (indicated by lower case letters), and *set-variables* corresponding to sets of domain elements (indicated by uppercase letters). Moreover, we use the standard logical connectives $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$, as well as quantifiers $\exists, \forall$ for both types of variables. Hence, given a graph $G = (V, E)$, an MSO formula depends on open variables in the form of vertices $x_1, \ldots, x_m \in V$, and sets of vertices $X_1, \ldots, X_n \subseteq V$ (while we could also have open variables in the form of edges and sets of edges, we do not make use of this in the following characterizations). We recall Courcelle's theorem in the style of [DSW12].

Theorem 5.7. *For every fixed MSO formula $\phi(x_1, \ldots, x_m, X_1, \ldots, X_n)$ and fixed integer c there is a linear-time algorithm that, given a graph $G = (V, E)$ of treewidth $\leq c$, decides whether given $G = (V, E)$, $v_1, \ldots, v_m \in V$, $B_1, \ldots, B_n \subseteq V$ satisfy $\phi(v_1, \ldots, v_m, B_1, \ldots, B_n)$.*

We use the unary predicates $A(\cdot)$ and $R(\cdot)$ to indicate an element being an argument or an attack of our SETAF, respectively. Moreover, we use the binary predicate $E(x, y)$ to indicate an edge in the incidence graph between incidence-vertices x and y. Alternatively, we write $a \in A$, $r \in R$, $(x, y) \in E$ for $A(a), R(r), E(x, y)$, respectively. Based on these basic definitions, we define notational shortcuts to conveniently characterize SETAF-properties. Let $SF{=}(A, R)$ be a SETAF and $\texttt{Inc}(SF){=}(V, E)$ its incidence graph. We define the following notion for $T \subseteq V$ and $h \in V$: let $(T, h) \in R$ be short-hand notation for $\exists r\,(r \in R \wedge (r, h) \in E \wedge \forall t(t \in T \leftrightarrow (t, r) \in E))$. This notion consists of four parts: (1) vertex r corresponds to an attack, (2) h is the head of the attack r, (3) the arguments in T constitute the tail of r. We utilize this to avoid dealing with the attack-vertices

of the incidence graph in our semantics characterizations. We borrow the following "building blocks" from [DSW12] (slightly adapted for our setting).

$$\begin{array}{rclcrcl} X \subseteq Y & = & \forall x\,(x \in X \rightarrow x \in Y) & & X \not\subset Y & = & \neg(X \subset Y) \\ X \subset Y & = & X \subseteq Y \wedge \neg(Y \subseteq X) & & x \notin X & = & \neg(x \in X) \\ X \not\subseteq Y & = & \neg(X \subseteq Y) & & x \in X^{\oplus}_R & = & x \in X \vee \exists Y(Y \subseteq X \wedge (Y,x) \in R) \\ X \subseteq^{\oplus}_R Y & = & \forall x\,(x \in X^{\oplus}_R \rightarrow x \in Y^{\oplus}_R) & & X \subset^{\oplus}_R Y & = & X \subseteq^{\oplus}_R Y \wedge \neg(Y \subseteq^{\oplus}_R X) \end{array}$$

We can express (subset-)maximality and -minimality for any expressible property $P(\cdot)$ as follows [DSW12]:

$$\max{}_{A,P(.),\subseteq}(X) = P(X) \wedge \neg\exists Y(Y \subseteq A \wedge P(Y) \wedge X \subset Y)$$
$$\min{}_{A,P(.),\subseteq}(X) = \max{}_{A,P(.),\supseteq}(X)$$

Having these tools at hand, we can characterize the SETAF semantics in an intuitive way. It is easy to verify that these exactly correspond to the respective notions from Definition 2.6. Utilizing these building blocks, we can encode the semantics *cf*, *adm*, *com*, *grd*, *stb*, *pref*, *naive*, *stage*, and *sem* exactly as in AFs [Dun07, DSW12].

Definition 5.8. *Let $SF = (A, R)$ be a SETAF and let $\mathtt{Inc}(SF) = (V, E)$ be its incidence graph. For a set $X \subseteq V$ where $\forall x \in X(x \in A)$:*

$$\begin{array}{rcl} cf(X) & = & \forall T,h\,((T,h) \in R \rightarrow (T \not\subseteq X \vee h \notin X)) \\ adm(X) & = & cf(X) \wedge \forall T,h(((T,h) \in R \wedge h \in X) \rightarrow \exists S,t(S \subseteq X \wedge t \in T \wedge (S,t) \in R)) \\ com(X) & = & adm(X) \wedge \forall x((x \in A \wedge x \notin X) \rightarrow \\ & & \qquad \exists S((S,x) \in R \wedge \neg\exists T(T \subseteq X \wedge (X,s) \in R \wedge s \in S))) \\ grd(X) & = & \min{}_{A,com(\cdot),\subseteq}(X) \\ stb(X) & = & cf(X) \wedge \forall x(x \in A \rightarrow x \in X^{\oplus}_R) \\ pref(X) & = & \max{}_{A,adm(\cdot),\subseteq}(X) \\ naive(X) & = & \max{}_{A,cf(\cdot),\subseteq}(X) \\ stage(X) & = & \max{}_{A,cf(\cdot),\subseteq^{\oplus}_R}(X) \\ sem(X) & = & \max{}_{A,adm(\cdot),\subseteq^{\oplus}_R}(X) \end{array}$$

Reasoning and verification can then be done in an intuitive way, see [DSW12]. For example, to find out whether an argument a is credulously accepted w.r.t. stable semantics we can use the formula $\exists X(stb(X) \wedge a \in X)$. We can immediately apply Courcelle's theorem [Cou87, Cou90] to obtain the desired result.

Theorem 5.9. *Let SF be a SETAF. For the semantics under our consideration, reasoning is fixed-parameter tractable w.r.t. the incidence-treewidth $\mathtt{itw}(SF)$.*

5.2 Dynamic Programming on SETAFs

In the following, we specify three dynamic programming algorithms utilizing incidence-treewidth to reason in stable, admissible, and complete semantics. Ultimately, we

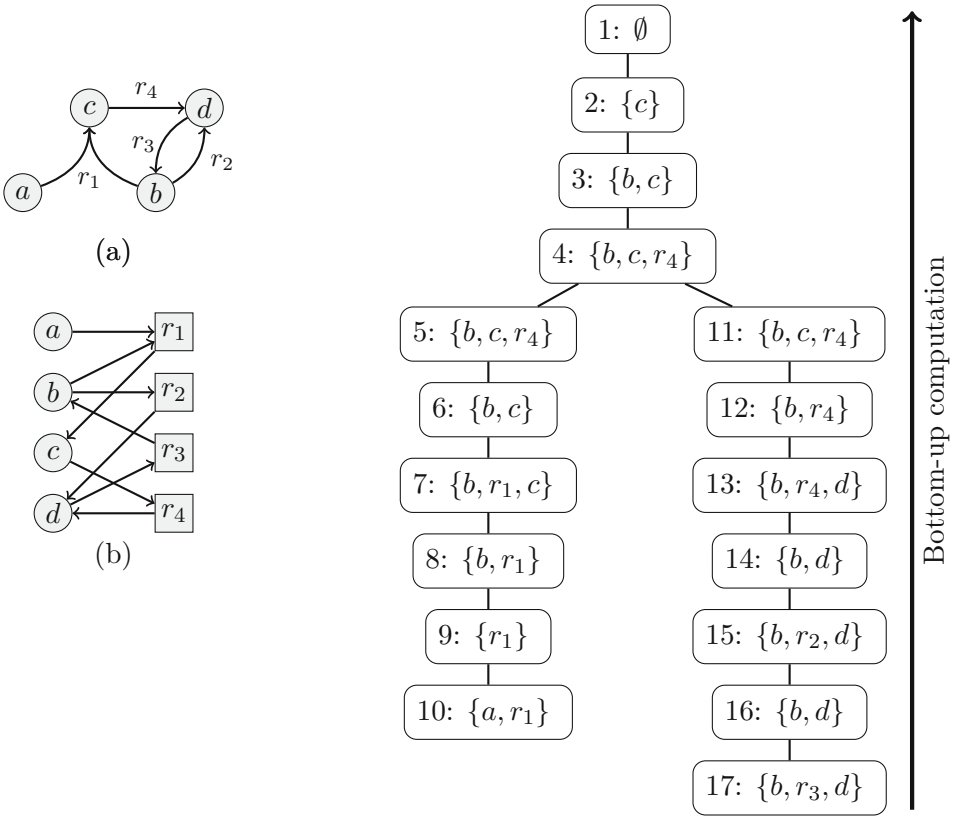


Figure 5.3: Running example for Section 5.2: (a) SETAF SF; (b) incidence graph $\texttt{Inc}(SF)$; (c) nice tree decomposition of $\texttt{Inc}(SF)$.

will show that these algorithms allow us to reason efficiently in SETAFs with fixed incidence-treewidth.

To illustrate the underlying idea of all three of these algorithms, we restrict the tree-decompositions of the incidence graph to *nice tree-decompositions*.

Definition 5.10. *A tree-decomposition $(\mathcal{T}, \mathcal{X})$ is called* nice *if $\mathcal{T} = (V_\mathcal{T}, E_\mathcal{T})$ is a rooted tree with an empty bag in the root node, and each node $t \in \mathcal{T}$ (shorthand notation for $n \in V_\mathcal{T}$) is of one of the following types:*

1. Leaf*: n has no children in $\mathcal{T}$,*
2. Forget*: n has one child n', and $X_n = X_{n'} \setminus \{v\}$ for some $v \in X_{n'}$,*
3. Insert*: n has one child n', and $X_n = X_{n'} \cup \{v\}$ for some $v \notin X_{n'}$,*
4. Join*: n has two children n', n'', with $X_n = X_{n'} = X_{n''}$.*

Any tree-decomposition can be transformed into a nice tree decomposition with the same width in linear time [Klo94]. Let $SF = (A, R)$ be a SETAF and $\texttt{Inc}(SF) = (V, E)$. For sets $U \subseteq V$, by U^A, U^R we identify the sets $(U \cap A)$, $(U \cap R)$, respectively. By $X_{\geq n}$ we denote the union of all bags X_m where $m \in V_{\mathcal{T}}$ appears in the subtree of $\mathcal{T}$ rooted in n. Likewise, by $X_{>n}$ we denote the set $X_{\geq n} \setminus X_n$.

Example 5.11. *We exemplify the notation with our running example, illustrated in Figure 5.3. If we focus on node* 5*, we have* $X_5 = \{b, c, r_4\} = X_5^A \cup X_5^R$*, with* $X_5^A = \{b, c\}$*, and* $X_5^R = \{r_4\}$*. The subtree rooted in node* 5 *contains the nodes* 5-10*. Consequently, we have* $X_{\geq 5} = \{a, b, c, r_1, r_4\}$ *and* $X_{>5} = \{a, r_1\}$*. We illustrate the relevant parts of the SETAF for each node in Figure 5.4.*

We use *colors* to keep track of the arguments and attacks that appear in the bag X_n of node $n \in V_{\mathcal{T}}$. We call an assignment of colors to arguments and attacks a *coloring*—they are akin to the idea of *labelings* for SETAFs, which is an alternative though (mostly) equivalent approach for SETAF semantics [FB19]. However, note that in our case we also color (or "label") the attacks in addition to the arguments, similar in spirit to *attacks semantics* [VBvdT11, CKRU24]. In our case, these colorings characterize extension candidates w.r.t. nodes in the tree-decomposition that are consistent with the sub-framework rooted in the node in question. Consequently, as the framework rooted in the root node of the decomposition is the entire framework, the colorings in the root node characterize the extensions of the SETAF. Let $SF = (A, R)$ be a SETAF. For an argument $a \in A$, we use different colors to indicate its relation to an extension $E \subseteq A$. The color $in_{\mathfrak{a}}$ indicates $a \in E$ (a is "accepted"), $out_{\mathfrak{a}}/pout_{\mathfrak{a}}$ indicate $a \in E^+$ (a is "defeated"), and $und_{\mathfrak{a}}/pund_{\mathfrak{a}}$ indicate $a \in A \setminus (E \cup E^+)$ (a is "undecided"). We use two colors each for defeated and undecided arguments, respectively, to ensure the correctness of our algorithm. The colors $pout_{\mathfrak{a}}$ and $pund_{\mathfrak{a}}$ correspond are in a sense "provisional" in that the "responsible" attack has not yet appeared in the bottom-up computation along the tree decomposition.

Likewise, for attacks (T, h) we use the colors $in_{\mathfrak{r}}$ if all $t \in T$ are $in_{\mathfrak{a}}$; $out_{\mathfrak{r}}$ if one $t \in T$ is $out_{\mathfrak{a}}$ or $pout_{\mathfrak{a}}$; $pout_{\mathfrak{r}}$ for the case where we expect at a later stage in the algorithm to add some $t \in T$ that is $out_{\mathfrak{a}}$ or $pout_{\mathfrak{a}}$; $und_{\mathfrak{r}}$ if one $t \in T$ is $und_{\mathfrak{a}}$ or $pund_{\mathfrak{a}}$ (and no $t \in T$ is $out_{\mathfrak{a}}$ or $pout_{\mathfrak{a}}$'); and $pund_{\mathfrak{r}}$ for the case where we expect at a later stage in the algorithm to add some $t \in T$ that is $und_{\mathfrak{a}}$ or $pund_{\mathfrak{a}}$ (and again no $t \in T$ is $out_{\mathfrak{a}}$ or $pout_{\mathfrak{a}}$'). Note that the exact interpretation of the colorings will be discussed in detail when we investigate each semantics, as we will be able to optimize the colorings tailored to each semantics. For example, we do not need to consider undecided arguments or attacks in stable semantics, as no extension can contain these.

5.3 Characterizing Stable Extensions

In the following we give a detailed account on how to characterize stable extensions via dynamic programming on the incidence-tree decomposition. In this regard, we define so

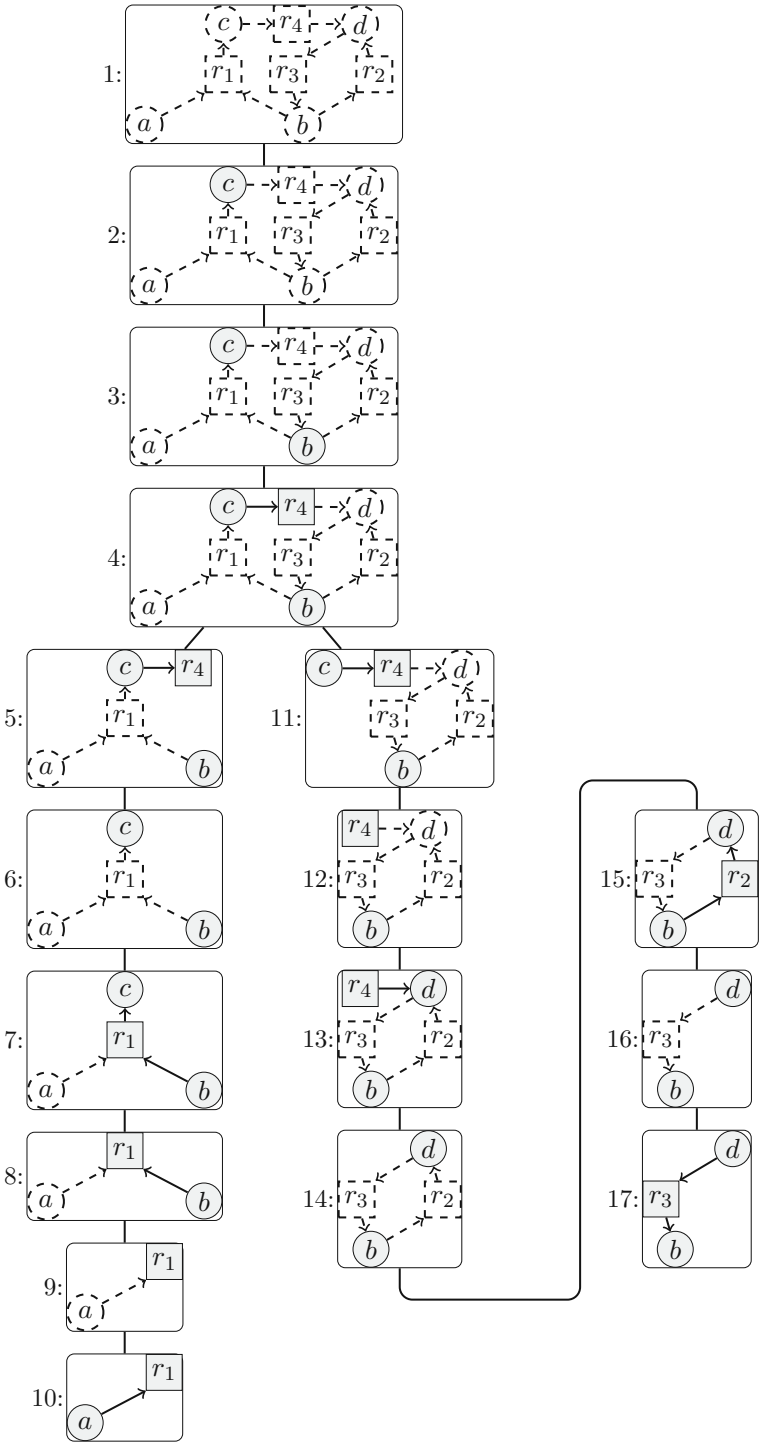


Figure 5.4: Running example ctd.: for each node we illustrate the parts of the SETAF rooted in the subtree (dashed+non-dashed) and the arguments/attacks of the current node (non-dashed).

called st-colorings for each node of the tree-decomposition in a way that can easily be implemented in a system. Moreover, we define *stable* colorings, i.e., the desired colorings for each node. Ultimately, we show that st-colorings and stable colorings coincide and thereby show the correctness and completeness of our algorithm.

As in a stable extension E we only have arguments either in E or attacked by E, we do not need to consider the undecided colors. After all, no extension candidate with undecided arguments or attacks in it can be extended to a stable extension. Hence, the colors $\mathcal{C}_{stb}$ we use for the dynamic programming algorithm for stable semantics are given as follows.

$$\mathcal{C}_{stb} = \{in_{\mathfrak{a}}, out_{\mathfrak{a}}, pout_{\mathfrak{a}}, in_{\mathfrak{r}}, out_{\mathfrak{r}}, pout_{\mathfrak{r}}\}$$

In the following we assume we are given an arbitrary but fixed SETAF $SF = (A, R)$, as well as a corresponding nice tree decomposition $\mathcal{T} = (V_{\mathcal{T}}, E_{\mathcal{T}})$ of $\mathtt{Inc}(SF)$.

We next characterize the *stable* colorings for each node of the tree decomposition. The stable colorings of a node characterize the candidates for stable extensions, as per the information that is available up to the current node. That means, in the root node (where information about the whole SETAF is available) the stable colorings correspond exactly to the stable extensions of the SETAF. We will later define st-colorings as the output of our algorithm and show that st-colorings and stable colorings coincide.

We relate a coloring X_n of a specific node n to a (hypothetical, not actually computed) generally larger coloring C' on all the arguments and attacks of the subtree rooted in node n (i.e., on all elements in $X_{\geq n}$). Ultimately, for the root node this subtree contains *all* arguments and attacks of the SETAF, which ensures that the colorings in this node in fact correspond only to stable extensions, as we will show in Proposition 5.13.

Definition 5.12. *A coloring $C : X_n \to \mathcal{C}_{stb}$ for a node $n \in V_{\mathcal{T}}$ is* stable *if we can extend C to a coloring $C' : X_{\geq n} \to \mathcal{C}_{stb}$ such that the following conditions are met for each $a \in X^A_{\geq n}$ and each $(T, h) \in X^R_{\geq n}$:*

1. *if $a \in X^A_{>n}$ then $C'(a) \in \{in_{\mathfrak{a}}, out_{\mathfrak{a}}\}$,*
2. *if $r \in X^R_{>n}$ then $C'((T, h)) \in \{in_{\mathfrak{r}}, out_{\mathfrak{r}}\}$,*
3. *if $C'(a) = in_{\mathfrak{a}}$ then $\forall (S, a) \in X^R_{\geq n} : C'((S, a)) \in \{out_{\mathfrak{r}}, pout_{\mathfrak{r}}\}$,*
4. *$C'(a) = out_{\mathfrak{a}}$ if and only if $\exists (S, a) \in X^R_{\geq n} : C'((S, a)) = in_{\mathfrak{r}}$,*
5. *if $C'((T, h)) = in_{\mathfrak{r}}$ then $\forall t \in T \cap X^A_{\geq n} : C'(t) = in_{\mathfrak{a}}$, and*
6. *$C'((T, h)) = out_{\mathfrak{r}}$ if and only if $\exists t \in T \cap X^A_{\geq n} : C'(t) \in \{out_{\mathfrak{a}}, pout_{\mathfrak{a}}\}$.*

For a coloring C in node n we define by $e_n(C)$ the set of such extended colorings C' on $X_{\geq n}$ s.t. (1)-(6) are met. These are the characterized colorings.

We now give some intuition for this definition. For the extended coloring C' we require six conditions (1)-(6) to be satisfied. (1) and (2) ensure that for all the arguments/attacks that only appear below a current node (i.e., in the subtree rooted in the node, excluding the node itself) only non-provisional colors are used. This is because by the properties of tree decompositions, we will not "encounter" these arguments/attacks again in the remaining computation; hence, provisional colors could not be "confirmed" to become non-provisional, and could therefore never be extended to an actual stable extension of the SETAF. (3) and (4) capture the requirements of stable extensions for arguments, i.e., if an argument is *in* then all attacks towards the argument must be *out*, and if an argument is *out* then at least one attack towards it must be *in*. Note that these conditions effectively characterize admissibility, but since we only allow two options $in_{\mathfrak{a}}$ and $out_{\mathfrak{a}}$ for arguments and every argument has to be colored, this suffices to characterize stable extensions. Similarly, (5) ensures that for each attack that is *in* all of the arguments in the tail of the attack are *in*, and (6) makes sure that for each attack that is *put* at least one tail-argument is *out*. These last two requirements make sure that the colors of the attacks correspond to the desired intuition, namely: (5) an attack (T, h) is colored $in_{\mathfrak{r}}$ if and only if it is "effective", i.e., all of its tail-arguments $t \in T$ are in the characterized stable extensions and the head of the attack t is indeed attacked, and (6) an attack (T, h) is colored $out_{\mathfrak{r}}$ or $pout_{\mathfrak{r}}$ if at least one of the tail arguments $t \in T$ is attacked by the characterized extensions E which renders the attack "ineffective" and h is defended against the attack (by whatever attack is responsible for attacking t).

We now illustrate the purpose of the colorings, namely that the characterized colorings of the *root node* correspond to the stable extensions of the SETAF.

Proposition 5.13. *Let $\mathcal{T} = (V_{\mathcal{T}}, E_{\mathcal{T}})$ be a nice tree-decomposition of a SETAF $SF = (A, R)$ and let $r \in V_{\mathcal{T}}$ be the root node of $\mathcal{T}$. Moreover, let $\mathcal{C}_r$ be the set of stable colorings for r. Then*

$$stb(SF) = \{\{a \mid C'(a) = in_{\mathfrak{a}}\} \mid C \in \mathcal{C}_r, C' \in e_r(C)\}.$$

Proof. Since the root node is empty, we have that $X^A_{>n} = A$ and $X^R_{>n} = R$, which by (1) and (2) means that we have no provisional colors in each extended coloring $C' \in e_r(C)$ (in case there is a stable coloring $C \in \mathcal{C}_r$, i.e., $\mathcal{C}_r \neq \emptyset$). Finally, condition (3) and (6) give us the following property (a) and (4) and (5) give us the following property (b):

(a) If $C'(a) = in_{\mathfrak{a}}$ then $\forall (T, a) \in R\ \exists t \in T : C'(t) = out_{\mathfrak{a}}$, and

(b) $C'(a) = out_{\mathfrak{a}}$ if and only if $\exists (T, a) \in R\ \forall t \in T : C'(t) = in_{\mathfrak{a}}$.

(a) exactly characterizes conflict-freeness, i.e., for each attack (T, a) towards an argument a that is in the extensions (i.e., in the set $\{a \mid C'(a) = in_{\mathfrak{a}}\}$) there is at least one argument in T that is *not* in the extension. (b) characterizes the fact that every argument that is not in the extension is attacked by the extension (since for stable semantics the only options for arguments are the colors $in_{\mathfrak{a}}$ and $out_{\mathfrak{a}}$). (a) and (b) together exactly define the stable extensions. □

We want to point out that since the root node of a nice tree decomposition is always empty, we have either $\mathcal{C}_r = \emptyset$ or $\mathcal{C}_r = \{\epsilon\}$, i.e., either there are no colorings in r or the empty coloring $C = \epsilon$. In the first case, Proposition 5.13 tells us that there are no stable extensions; in the second case the characterized extended colorings $e_r(C)$ characterize the stable extensions. Intuitively, the actual extensions can be computed by straightforwardly combining the corresponding colorings of one branch of $\mathcal{T}$, wherever a coloring "persists" throughout the whole branch. All extensions can be enumerated this way with linear delay (for details see [JPW09]).

We will now continue to discuss the four node types and show that in each node we indeed capture exactly the stable colorings. In particular, we present the algorithm itself (i.e., define the st-colorings), and show that we characterize *all the* stable extensions and *only* stable extensions (i.e., that each intermediate coloring in each node is stable as per Definition 5.12).

5.3.1 Leaf Nodes

Intuitively, in leaf nodes for each argument and each attack we guess one of two possibilities: *in* or *out* (in the second case we use either *out* or *pout*, as we will explain below). We then keep every "consistent" coloring, e.g., if an argument is colored $in_{\mathfrak{a}}$ then there cannot be an attack towards it that is colored $in_{\mathfrak{r}}$. Whether (i) an argument or (ii) an attack is colored *out* or *pout* depends only on whether the color is "confirmed", i.e., (i) for arguments whether there is an attack colored $in_{\mathfrak{r}}$ towards the argument, and (ii) for attacks whether there is an argument colored $out_{\mathfrak{a}}$ in the tail of the attack.

Definition 5.14 (st-colorings: Leaf)**.** *An* st-coloring *for a* leaf node $n \in V_{\mathcal{T}}$ *is each coloring* $C : X_n \to \mathcal{C}_{stb}$ *that satisfies the following conditions. For each argument* $a \in X_n^A$*:*

$$C(a) = in_{\mathfrak{a}} \Rightarrow \forall (T, a) \in X_n^R \colon C((T, a)) \in \{pout_{\mathfrak{r}}, out_{\mathfrak{r}}\}$$
$$C(a) = out_{\mathfrak{a}} \Leftrightarrow \exists (T, a) \in X_n^R \colon C((T, a)) = in_{\mathfrak{r}}$$

For every attack $r = (T, h) \in X_n^R$*:*

$$C(r) = in_{\mathfrak{r}} \Rightarrow \forall t \in T \cap X_n^A \colon C(t) = in_{\mathfrak{a}}$$
$$C(r) = out_{\mathfrak{r}} \Leftrightarrow \exists t \in T \cap X_n^A \colon C(t) \in \{pout_{\mathfrak{a}}, out_{\mathfrak{a}}\}$$

In our running example in Figure 5.5, the nodes 10 and 17 are leaf nodes. Note that there is no explicit condition for the colors $pout_{\mathfrak{a}}$ and $pout_{\mathfrak{r}}$. This reflects the fact that we can color any argument/attack provisionally out (as long as it does not violate any of the conditions for the other colors) which captures the situation where an argument is *going to be* attacked by an attack that we have not yet considered, i.e., that is still going to be handled in a later (i.e., upper) node in the tree decomposition.

For the following lemmas note that for leaves it holds $X_{>n}^A = X_{>n}^R = \emptyset$. Then the following two results directly follow from Definition 5.12 of stable colorings and Definition 5.14 of

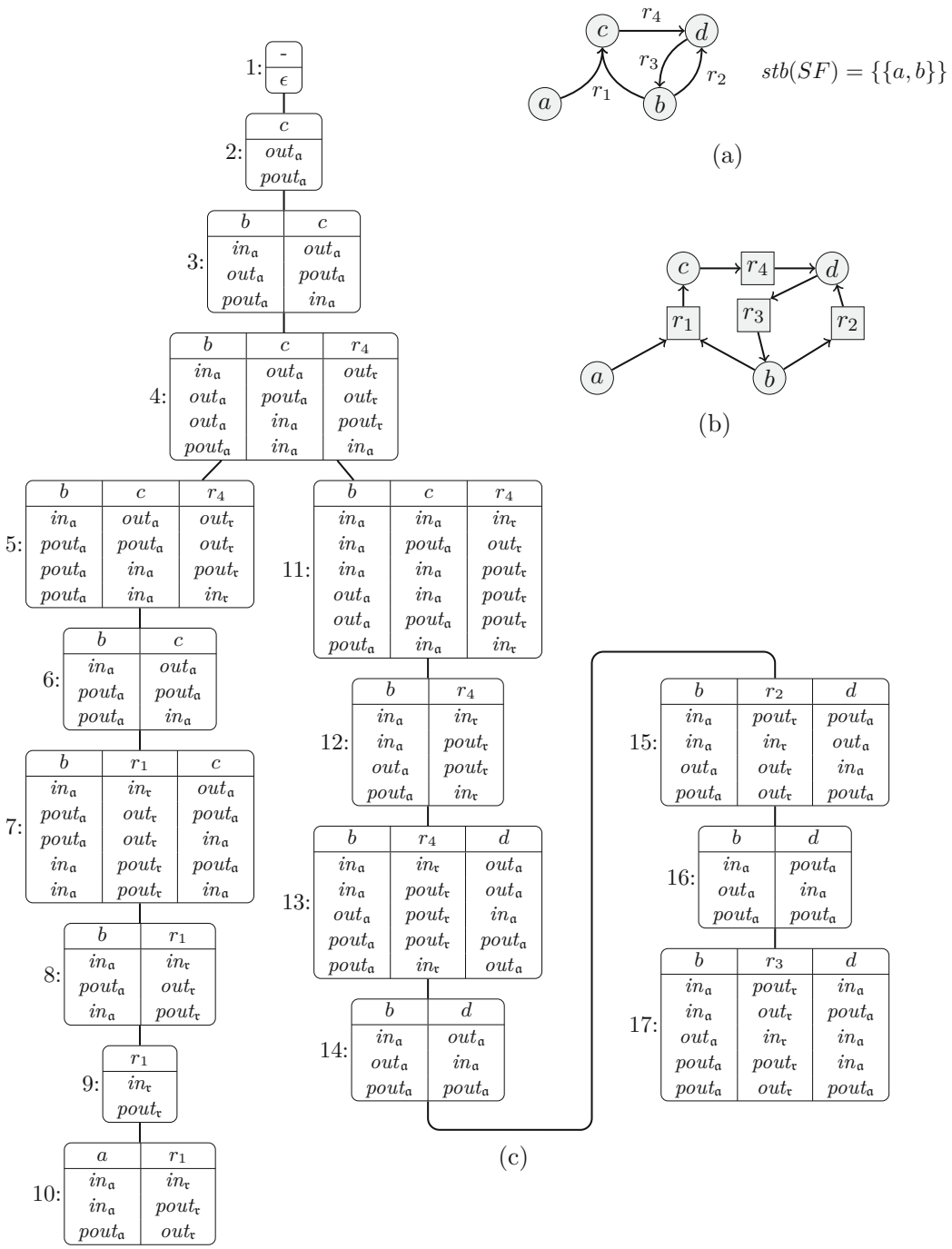


Figure 5.5: All st-colorings for our running example. (a) Our running example SETAF SF with its stable extensions, (b) its incidence-graph $\texttt{Inc}(SF)$, and (c) all st-colorings for SF w.r.t. the tree decomposition given in Figure 5.3.

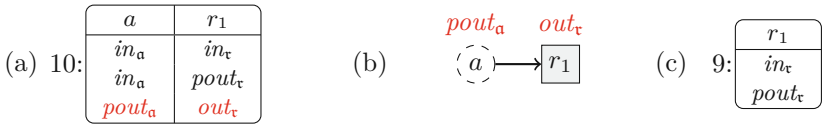


Figure 5.6: St-colorings for leaf nodes and forget argument nodes. Example for st-colorings for (a) the leaf node 10 from Figure 5.5 and (c) the "Forget a" node 9 for argument a. Subfigure (b) illustrates the subgraph of the incidence graph that corresponds to the leaf node together with the coloring that is discarded by the forget node.

st-colorings in leaves. The following lemma captures the soundness of our algorithm and holds because the conditions for st-colorings imply the properties of stable colorings.

Lemma 5.15. *Each st-coloring C for a leaf node $n \in V_{\mathcal{T}}$ is stable.*

Proof. We need to show that each of the conditions (1)-(6) of Definition 5.12 is met. (1) and (2) are trivially true, as for leaves it holds $X^A_{>n} = X^R_{>n} = \emptyset$. (3)-(6) follow directly from the conditions in Definition 5.14. □

The next result captures the completeness of our algorithm and holds because the properties of stable colorings imply the conditions for st-colorings for leaves.

Lemma 5.16. *Each stable coloring C for a leaf node $n \in V_{\mathcal{T}}$ is an st-coloring.*

Proof. It follows directly from the definition of stable colorings that C adheres to the conditions in Definition 5.14. □

5.3.2 Forget Nodes

We examine forget argument nodes and forget attack nodes separately. Let n be a *forget argument node* with child n' such that $X^A_n = X^A_{n'} \setminus \{a\}$. We have to discard all colorings C where $C(a) = \mathit{pout}_{\mathfrak{a}}$, as in these colorings a is supposed to be attacked by the extensions that C potentially characterizes. However, as we forget a in the current node by the definition of a tree-decomposition, this cannot happen: consider again the running example from Figure 5.5. Node 9 is a forget node, where the argument a is forgotten (we illustrate this situation in Figure 5.6). This means, in the "upper" part of the tree decomposition, no attacks towards a can be added. Hence, there cannot be an attack colored $\mathit{in}_{\mathfrak{r}}$ towards a that confirms a being attacked, and the provisional color $\mathit{pout}_{\mathfrak{a}}$ cannot be updated to $\mathit{out}_{\mathfrak{a}}$. One can clearly see that this behavior is correct, as there is indeed no attack towards a in SF.

Definition 5.17 (st-coloring: Forget Argument)**.** *Let n be a forget argument node with child n' such that $X^A_n = X^A_{n'} \setminus \{a\}$, and let C be an st-coloring for n'. If $C(a) \neq \mathit{pout}_{\mathfrak{a}}$, then $C - a$ is an st-coloring for n, where $C - a$ for each $b \in X_n$ is defined as follows:*

$$(C - a)(b) = C(b)$$

(a) 6:

b	c
$in_{\mathfrak{a}}$	$out_{\mathfrak{a}}$
$pout_{\mathfrak{a}}$	$pout_{\mathfrak{a}}$
$pout_{\mathfrak{a}}$	$in_{\mathfrak{a}}$

(b) 7:

b	r_1	c
$in_{\mathfrak{a}}$	$in_{\mathfrak{r}}$	$out_{\mathfrak{a}}$
$pout_{\mathfrak{a}}$	$out_{\mathfrak{r}}$	$pout_{\mathfrak{a}}$
$pout_{\mathfrak{a}}$	$out_{\mathfrak{r}}$	$in_{\mathfrak{a}}$
~~$in_{\mathfrak{a}}$~~	~~$pout_{\mathfrak{r}}$~~	~~$pout_{\mathfrak{a}}$~~
~~$in_{\mathfrak{a}}$~~	~~$pout_{\mathfrak{r}}$~~	~~$in_{\mathfrak{a}}$~~

(c)

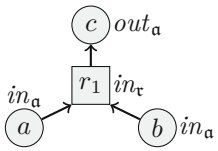


Figure 5.7: St-colorings forget attack nodes. Example of (a) the forget node 6 and (b) its child node 7 from the running example (the discarded colorings are struck out). (c) illustrates the sub-framework rooted in 6 (i.e., containing $X_{\geq 6}$) and an extended coloring C' that extends the highlighted colorings of (a) and (b).

We handle *forget attack nodes* in a similar way, i.e., we discard colorings where the attack we forget is provisionally out. In this case, we colored the attack $pout_{\mathfrak{r}}$ at an earlier stage in the algorithm, anticipating the possibility for an insert argument node where a tail-argument is colored $out_{\mathfrak{a}}$ or $pout_{\mathfrak{r}}$. Due to the properties of tree-decompositions, such an insert argument node cannot appear above the forget attack node, which is why we can discard colorings where the attack is colored provisionally.

Definition 5.18 (st-coloring: Forget Attack)**.** *Let n be a forget attack node with child n' such that $X_n^R = X_{n'}^R \setminus \{r\}$, and let C be an st-coloring for n'. If $C(r) \neq pout_{\mathfrak{r}}$, then $C - r$ is an st-coloring for n, where $C - r$ for each $b \in X_n$ is defined as follows:*

$$(C - r)(b) = C(b)$$

Next we show the soundness of our algorithm for forget nodes. This result carries over from the child node, as we discard colorings where the forgotten argument/attack is provisional. Let us revisit a part of our running example (see Figure 5.7), we see in node 6 (where we "forget" the attack r_1) that we "copy" each st-coloring of the child node 7 except for the last two colorings where r_1 has the provisional color $pout_{\mathfrak{r}}$ (Figure 5.7(b)). It is then clear that since each st-coloring of node 7 can be extended to a coloring on all of $X_{\geq 7}$ (since we assume these colorings are stable) that the same holds for the st-colorings of node 6. For example, the st-coloring C^* of node 7 with $C^*(b) = in_{\mathfrak{a}}$, $C^*(r_1) = in_{\mathfrak{r}}$, and $C^*(c) = out_{\mathfrak{a}}$ can be extended to the coloring C' on $X_{\geq 7} = \{a, b, c, r_1\}$ with $C'(a) = in_{\mathfrak{a}}$ (and of course $C'(x) = C^*(x)$ for $x \in \{b, c, r_1\}$); analogously the st-coloring C of 6 with $C(b) = in_{\mathfrak{a}}$ and $C(c) = out_{\mathfrak{a}}$ can also be extended to C' on $X_{\geq 6}$, and it can easily be checked that C is stable.

Lemma 5.19. *If for the child node n' of a forget node n each st-coloring is stable, then each coloring C of n is stable.*

Proof. We need to show that each of the conditions (1)-(6) of Definition 5.12 is met. (1) and (2) are true because they hold for n', and if the forgotten argument/attack is provisional, then it is discarded. The satisfaction of (3)-(6) immediately carries over from n'. □

To show the completeness of our algorithm, we can show that for each stable coloring C for node n there has to be a corresponding st-colorings C^* in the child node n' such that $C = C^* - x$ for the forgotten argument/attack x. This means that n contains every stable coloring. The key idea is that C^* in the child node n' coincides with C and does not color x in a provisional color, as otherwise C would not be a stable coloring for n. Again consider node the forget node 6 of our running example (Figure 5.7). Even without knowing that coloring C with $C(b) = in_{\mathfrak{a}}$ and $C(c) = out_{\mathfrak{a}}$ is an st-coloring of node 6, we can check that this coloring is stable for node 6. This means we can extend it to a coloring C' on $X_{\geq 6} = \{a, b, c, r_1\}$, and we can infer that there has to be a stable coloring C^* for the child node 7 that can also be extended to C'. As with the other direction, we then see that indeed $C = C^* - r_1$.

Lemma 5.20. *Let C be a stable coloring of a forget node n. If in the child node n' of n stable colorings and st-colorings coincide then C is an st-coloring for n.*

Proof. Let C' be the extended coloring of C on $X_{\geq n}$ s.t. the conditions of Definition 5.12 are satisfied.

We start with *forget argument* nodes. We show that C^* is an st-coloring for n' where C^* coincides with C on each $a, (T, h) \in X_n$ and for the forgotten argument $b \in X_{n'} \setminus X_n$ it holds $C^*(b) \neq pout_{\mathfrak{a}}$.

- If $C'(b) = in_{\mathfrak{a}}$ then clearly C^* is stable for n' with $C^*(b) = in_{\mathfrak{a}}$: in this case each condition (1)-(6) carries over from C. Then by assumption this means C^* is an st-coloring for n'.
- If $C'(b) = out_{\mathfrak{a}}$ then C^* is stable for n' with $C^*(b) = out_{\mathfrak{a}}$: this is because by condition (4) we have for node n that $\exists (S, b) \in X^R_{\geq n} : C'((S, b)) = in_{\mathfrak{r}}$, and since $X^R_{\geq n} = X^R_{\geq n'}$ this carries over to node n'. The other conditions clearly carry over, and by assumption this means C^* is an st-coloring for n'.

In both cases we see that there is a corresponding st-coloring C^* in n' that gives us C as an st-coloring in n, and since there is no other possibility, this means that $C = C^* - b$ is an st-coloring of n.

We continue with *forget attack* nodes. We show that C^* is an st-coloring for n' where C^* coincides with C on each $a, (T, h) \in X_n$ and for the forgotten attack $(S, t) \in X_{n'} \setminus X_n$ it holds $C^*((S, t)) \neq pout_{\mathfrak{r}}$.

- If $C'((S, t)) = in_{\mathfrak{r}}$ then clearly C^* is stable for n' with $C^*(b) = in_{\mathfrak{r}}$: in this case each condition (1)-(6) carries over from C. Then by assumption this means C^* is an st-coloring for n'.
- If $C'((S, t)) = out_{\mathfrak{r}}$ then C^* is stable for n' with $C^*((S, t)) = out_{\mathfrak{r}}$: this is because by condition (6) we have for node n that $\exists s \in S \cap X^A_{\geq n} : C'(s) \in \{out_{\mathfrak{a}}, pout_{\mathfrak{a}}\}$, and

since $X^A_{\geq n} = X^A_{\geq n'}$ this carries over to node n'. Again the other conditions clearly carry over, and by assumption this means C^* is an st-coloring for n'.

In both cases we see that there is a corresponding st-coloring C^* in the child node n' that gives us C as an st-coloring in n, and since there is no other possibility, this means that $C = C^* - (S, t)$ is an st-coloring of n. □

5.3.3 Insert Nodes

We distinguish the two cases where we insert an argument and insert an attack. Whenever we insert an argument a, we have to consider up to two different scenarios. Scenario 1, $(C + a)$: the added argument is attacked by the characterized extension. In this case the added argument is colored $pout_{\mathfrak{a}}$ or $out_{\mathfrak{a}}$, depending on whether the "responsible" attack is already in the current bag. In case a is in the tail of an attack, we can color this attack $out_{\mathfrak{r}}$. Scenario 2, $(C \dot{+} a)$: the added argument is in the extension. In both scenarios we have to check whether the result will be consistent with the existing colors, i.e., for $(C + a)$ the added argument must not be in the tail of an attack that is colored $in_{\mathfrak{r}}$, and for $(C \dot{+} a)$ there must not be an attack colored $in_{\mathfrak{r}}$ towards the added argument. Assume we would color the inserted argument b as $out_{\mathfrak{a}}/pout_{\mathfrak{a}}$ while b is in the tail of attack r, which we already colored $in_{\mathfrak{r}}$ in a previous step. Of course, this is not consistent with our intended meaning of the attack color $in_{\mathfrak{r}}$. On the other hand, assume we color b as $in_{\mathfrak{a}}$ while it is attacked by r which we already colored $in_{\mathfrak{r}}$ in a previous step. This would introduce a conflict in the constructed extension.

Definition 5.21 (st-coloring: Insert Argument)**.** *Let n be an insert argument node with child n' such that $X^A_{n'} = X^A_n \setminus \{a\}$, and let C be an st-coloring for n'.*

- *If $\nexists (T, h) \in X^R_{n'}$: $(C((T, h)) = in_{\mathfrak{r}} \wedge a \in T)$, then $C + a$ is an st-coloring for n;*
- *If $\nexists (T, a) \in X^R_{n'}$: $(C((T, a)) = in_{\mathfrak{r}})$, then $C \dot{+} a$ is an st-coloring for n.*

The operations $C + a$ and $C \dot{+} a$ are defined as follows for each $b \in X_n$:

$$(C + a)(b) = \begin{cases} out_{\mathfrak{a}} & \text{if } b = a \wedge \exists (T, a) \in X^R_{n'}\colon C((T, a)) = in_{\mathfrak{r}} \\ pout_{\mathfrak{a}} & \text{if } b = a \wedge \nexists (T, a) \in X^R_{n'}\colon C((T, a)) = in_{\mathfrak{r}} \\ out_{\mathfrak{r}} & \text{if } b = (T, h) \wedge a \in T \wedge C(b) = pout_{\mathfrak{r}} \\ C(b) & \text{otherwise} \end{cases}$$

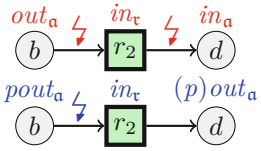
$$(C \dot{+} a)(b) = \begin{cases} in_{\mathfrak{a}} & \text{if } b = a \\ C(b) & \text{otherwise} \end{cases}$$

For *insert attack nodes*, we also have to consider two cases for an attack $r = (T, h)$: $(C + r)$: the extension attacks T, either in the current bag (in which case we color the attack $out_{\mathfrak{r}}$), or possibly in the "upper" parts of $\mathcal{T}$, then we color the attack $pout_{\mathfrak{r}}$. $(C \dot{+} r)$: this case indicates $T \subseteq E$ for the extension E. In this case the head of the attack can be

(a) 16:

b	d
$in_{\mathfrak{a}}$	$pout_{\mathfrak{a}}$
$out_{\mathfrak{a}}$	$in_{\mathfrak{a}}$
$pout_{\mathfrak{a}}$	$pout_{\mathfrak{a}}$

(b) (c)



(d) 15:

b	r_2	d
$in_{\mathfrak{a}}$	$pout_{\mathfrak{r}}$	$pout_{\mathfrak{a}}$
$in_{\mathfrak{a}}$	$in_{\mathfrak{r}}$	$out_{\mathfrak{a}}$
$out_{\mathfrak{a}}$	$out_{\mathfrak{r}}$	$in_{\mathfrak{a}}$
$pout_{\mathfrak{a}}$	$out_{\mathfrak{r}}$	$pout_{\mathfrak{a}}$

Figure 5.8: St-colorings for insert attack nodes. (a) shows the st-colorings for node 16 and (d) the st-colorings for "Insert attack r_2" node 15 from Figure 5.5. Subfigures (b) and (c) show inconsistent colorings.

set to $out_{\mathfrak{a}}$. Again, we can only apply this coloring if it is consistent with the previous colors. The idea is very similar to the insert argument nodes; we exemplify the issues in Figure 5.8: in (b) we cannot color the added attack r_2 with $in_{\mathfrak{r}}$ due to two issues, namely (1) the argument b in the tail of r_2 is not colored $in_{\mathfrak{a}}$, and (2) the head of r_2 (argument d) is already colored $in_{\mathfrak{a}}$. In (c) we cannot color r_2 with $in_{\mathfrak{r}}$ even though the head is matching, because the tail argument b is not colored $in_{\mathfrak{a}}$. We will use the operations $C + r$ and $C \dot{+} r$ as given in Definition 5.22.

Definition 5.22 (st-coloring: Insert Attack)**.** *Let n be an insert attack node with child n' such that $X^R_{n'} = X^R_n \setminus \{r\}$, and let C be an st-coloring for n', and $r = (T, h)$.*

- *$C + r$ is an st-coloring for n;*
- *If $(h \notin X^A_{n'} \vee C(h) \neq in_{\mathfrak{a}}) \wedge \forall t \in T \cap X^A_{n'} : C(t) = in_{\mathfrak{a}}$, then $C \dot{+} r$ is an st-coloring for n.*

The operations $C + r$ and $C \dot{+} r$ are defined as follows for each $b \in X_n$:

$$(C + r)(b) = \begin{cases} out_{\mathfrak{r}} & \text{if } b = r \wedge \exists t \in T \cap X^A_{n'}\colon C(t) \in \{pout_{\mathfrak{a}}, out_{\mathfrak{a}}\} \\ pout_{\mathfrak{r}} & \text{if } b = r \wedge \nexists t \in T \cap X^A_{n'}\colon C(t) \in \{pout_{\mathfrak{a}}, out_{\mathfrak{a}}\} \\ C(b) & \text{otherwise} \end{cases}$$

$$(C \dot{+} r)(b) = \begin{cases} in_{\mathfrak{r}} & \text{if } b = r \\ out_{\mathfrak{a}} & \text{if } r = (T, h) \wedge b = h \\ C(b) & \text{otherwise} \end{cases}$$

Again, we show that the insert nodes exactly characterize the stable colorings. For the following lemma (soundness) we need to show that each of the colorings of an insert node n is stable. It can be shown that in both cases, i.e., when the st-coloring C in node n is constructed either from an st-coloring C' of child node n' via $C = C' + x$ or $C = C' \dot{+} x$ (for an argument/attack x) all conditions (1)-(6) of stable colorings hold. This is ensured either via the requirements for adding $C' + x$ or $C' \dot{+} x$, or by construction by the assigned colors. In particular, the colors $out_{\mathfrak{a}}/pout_{\mathfrak{a}}$ and $out_{\mathfrak{r}}/pout_{\mathfrak{r}}$ are assigned according to whether the "confirming" argument/attack is already present in the current

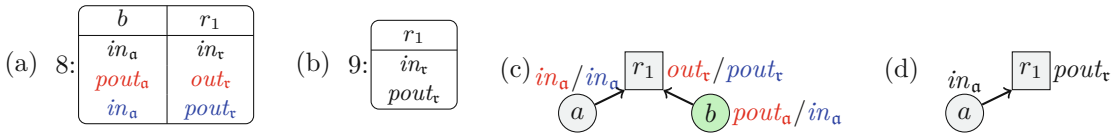


Figure 5.9: St-colorings for insert argument nodes. Example of (a) the insert node 8 with highlighted colorings C_1 (red) and C_2 (blue), and (b) its child node 9 from the running example. (c) illustrates the sub-framework rooted in 8 (i.e., containing $X_{\geq 8}$) and an extended colorings C'_1 (red) and C'_2 (blue), (d) illustrates the sub-framework rooted in 9 with the extended coloring C''.

node n. This together with the fact that the original st-coloring C' of the child node n' is stable—the basis for the constructed coloring C in n—ensures that C is stable as well. For example, consider in our running example the insert node 8 (illustrated in Figure 5.9). Its child node 9 contains an st-coloring C^* with $C^*(r_1) = pout_{\mathfrak{r}}$. For 8 we get $C_1 = C^* + b$ and $C_2 = C^* \dot{+} b$ (Figure 5.9(a)). In C_1 we set $C_1(b) = pout_{\mathfrak{a}}$ which "confirms" the provisional color of r_1 and "upgrades" it to $out_{\mathfrak{r}}$, the corresponding extended coloring C'_1 on $X_{\geq 8}$ is illustrated in Figure 5.9(c). In C_2 on the other hand we set $C_2(b) = in_{\mathfrak{a}}$ and keep $C_2(r_1) = pout_{\mathfrak{r}}$, the corresponding extended coloring C'_2 is also illustrated in Figure 5.9(c). Since we update the color of r_1 only because we have "proof" to do so, we can again check that these st-colorings are stable.

Lemma 5.23. *If for the child node n' of an insert node n each st-coloring is stable, then each st-coloring C of n is stable.*

Proof. We need to show that each of the conditions (1)-(6) of Definition 5.12 is met. (1) and (2) immediately carry over from n' both for insert argument and insert attack nodes.

For (3)-(6) we discuss first the *insert argument* nodes. Note that we construct C either from a stable coloring C' of n' via $C = C' + a$ or via $C = C' \dot{+} a$. Clearly, (3) and (4) hold both in the cases $C = C' + a$ and $C = C' \dot{+} a$ w.r.t. C for all $b \in X^A_{n'}$ (i.e., the "old" arguments that have not been added in this node) because they hold for n'.

- For the added argument a regarding (3) only the coloring $C' \dot{+} a$ is relevant (as $(C' + a)(a) \neq in_{\mathfrak{a}}$). Since we can add the coloring $C' \dot{+} a$ only if $\nexists (T, a) \in X^R_{n'} : (C'((T, a)) = in_{\mathfrak{r}})$ and the only possible colors for attacks are $in_{\mathfrak{r}}$, $out_{\mathfrak{r}}$, and $pout_{\mathfrak{r}}$, we know that in this case (3) is satisfied.

- For the added argument a regarding (4) only the coloring $C = C' + a$ is relevant. In this case, if $C(a) = out_{\mathfrak{a}}$ then we know $\exists (T, a) \in X^R_{n'} : (C((T, a)) = in_{\mathfrak{r}} \wedge a \notin T)$ which means condition (4) is satisfied; if otherwise $C(a) = pout_{\mathfrak{a}}$ then (4) is trivially true for a.

- Conditions (5) and (6) immediately carry over from n' for all attacks $(T,h) \in X_n^R$ where for the added argument a it holds $a \notin T$. Let $(T,h) \in X_n^R$ be such that for the added argument a it holds $a \in T$. If $C'((T,h)) = in_{\mathfrak{r}}$ then we do not add $C' + a$ as a coloring to n, which means the satisfaction of condition (5) carries over from n' in this case. For $C' \dot{+} a$ condition (5) also carries over from n' for each attack $(T,h) \in X_{\geq n}^R$ with $a \notin T$, and as $(C' \dot{+} a)(a) = in_{\mathfrak{a}}$ the condition is also satisfied for the attacks with $a \in T$.

- Moreover, in both cases $C' + a$ and $C' \dot{+} a$ condition (6) carries over for each attack $(T,h) \in X_{\geq n}^R$ with $a \notin T$. Moreover, for the case where we set $(C'+a)((T,h)) = out_{\mathfrak{r}}$ where we had $C'((T,h)) = pout_{\mathfrak{r}}$, condition (6) is satisfied as the $a \in T$ and $(C' + a)(a) \in \{out_{\mathfrak{a}}, pout_{\mathfrak{a}}\}$.

Hence, conditions (1)-(6) are met for insert argument nodes.

For (3)-(6) we now discuss the *insert attack* nodes.

- Regarding (3), it suffices to discuss for the added attack $r = (T,h)$ the case $(C' \dot{+} r)$ (for the other attacks condition(3) carries over from n'). However, the coloring $C' \dot{+} r$ is only added to n if $C'(h) \neq in_{\mathfrak{a}}$, which also ensures that condition (3) is met for when we set the added attack to $in_{\mathfrak{r}}$.

- Condition (4) carries over from n', and the case where we set $(C' \dot{+} r)(a) = out_{\mathfrak{a}}$ when we previously had $C'(a) = pout_{\mathfrak{a}}$ is only applied if we set $(C' \dot{+} (T,a))((T,a)) = in_{\mathfrak{r}}$, i.e., (4) is satisfied.

- For (5), only the case $C' \dot{+} (T,h)$ is relevant, the other cases follow from n'. However, we only add $C' \dot{+} (T,h)$ to n if $\forall t \in T \cap X_n^A$ it holds $C(t) = in_{\mathfrak{a}}$, and because of the properties of tree decompositions we know that $X_{>n}^A \cap T = \emptyset$, i.e., there are no tail arguments strictly below the current node. Hence, (5) is satisfied.

- As to (6), the only case that is not clear from n' is in $C' + (T,h)$, but in this case we set $C((T,h))$ to $out_{\mathfrak{r}}$ only if $\exists t \in T \cap X_n^A$: $C(t) \in \{pout_{\mathfrak{a}}, out_{\mathfrak{a}}\}$, which means (6) is also satisfied.

To summarize, all of (1)-(6) is satisfied, which means that each coloring in n is stable. □

For the following result (completeness) we can show that for each stable coloring C in the insert node n there has to be a corresponding st-coloring C^* in the child node n' on all of $X_{\geq n}$ such that $C = C^* + x$ or $C = C^* \dot{+} x$ for the added argument/attack x. The key idea here is that C^* differs from C exactly in those cases where the added argument/attack leads to non-provisional colors in C, these colors we set to $pout_{\mathfrak{a}}/pout_{\mathfrak{r}}$ in C^*. We can then check that all conditions of stable colorings are satisfied for C^*, and since we assume that in the child node n' stable colorings and st-colorings coincide, we know that C^* is an st-coloring of n'. Finally it is easy to see that in both cases the

requirements for adding $C = C^* + x$ or $C = C^* \dot{+} x$ as an st-coloring for n are satisfied. Revisiting the example of Figure 5.9 we can see that for both st-colorings C_1 and C_2 the corresponding st-coloring in node 9 has $C^*(r_1) = pout_{\mathfrak{r}}$, which is due to the fact that there is no "confirming" argument in the tail of r_1 in $X^A_{\leq 9}$. Since this coloring has to be an st-coloring of 9 and we can construct both C_1 and C_2 from C^*, we can see that these two stable colorings are indeed st-colorings of node 8.

Lemma 5.24. *Let C be a stable coloring for an insert node n. If in the child node n' of n stable colorings and st-colorings coincide then C is an st-coloring for n.*

Proof. Let C' be the extended coloring of C on $X_{\geq n}$ s.t. the conditions of Definition 5.12 are satisfied.

We start with *insert argument* nodes. We show that there is an st-coloring C^* in node n' s.t. for the added argument a we have $C = C^* + a$ if $C'(a) \in \{out_{\mathfrak{a}}, pout_{\mathfrak{a}}\}$, and $C = C^* \dot{+} a$ if $C'(a) = in_{\mathfrak{a}}$.

- Let $C'(a) \in \{out_{\mathfrak{a}}, pout_{\mathfrak{a}}\}$, then C^* is an st-coloring of n' where C^* coincides with C on each $b, (T,h) \in X_{n'}$ with the exception of attacks (T,h) with $a \in T$, $C((T,h)) = out_{\mathfrak{r}}$, and $\nexists t \in T \cap X^A_{n'} : C'(t) \in \{out_{\mathfrak{a}}, pout_{\mathfrak{a}}\}$, for these attacks we set $C^*((T,h)) = pout_{\mathfrak{r}}$. To show that C^* is an st-coloring of n' we need to show that the conditions (1)-(6) of stable colorings are satisfied. (1) and (2) carry over from C, as $X_{>n} = X_{>n'}$. Note for (3) that C and C^* coincide on the color $in_{\mathfrak{a}}$, and on the set of attacks colored either $out_{\mathfrak{r}}$ or $pout_{\mathfrak{r}}$, which means (3) carries over from C. (4) and (5) immediately carry over as well, as C and C^* coincide on all relevant colors on $X_{n'}$. Regarding (6) note that for each attack $(T,h) \in X^R_{n'}$ with $a \in T$ this holds by definition of C^*, and for the other attacks the property carries over from C. Hence, in this case C^* is stable for n' and therefore by assumption an st-coloring for n'. Then since C^* is stable from condition (5) it follows $\nexists (T,h) \in X^{*R}_{n'} : (C((T,h)) = in_{\mathfrak{r}} \wedge a \in T)$, hence, $C = C^* + a$ is an st-coloring for n (the correct color $out_{\mathfrak{a}}/pout_{\mathfrak{a}}$ directly follows from the fact that C is stable and that due to the properties of tree decompositions all attacks towards a in $X^R_{\geq n}$ have to be in X^R_n).
- Now on the other side let $C'(a) = in_{\mathfrak{a}}$, then C^* is an st-coloring of n' where C^* coincides with C on each $b, (T,h) \in X_{n'}$. Again we show that the conditions (1)-(6) are satisfied. (1) and (2) carry over from C, as $X_{>n} = X_{>n'}$. (3)-(6) immediately carry over as well, as C and C^* coincide on all relevant colors on $X_{n'}$. Then since C^* is stable from condition (3) it follows $\nexists (T,a) \in X^R_{n'} : (C((T,a)) = in_{\mathfrak{r}})$, hence, $C = C^* \dot{+} a$ is an st-coloring for n.

We showed that in both cases C is an st-coloring of n.

We continue with the *insert attack* nodes. Again we show that there is an st-coloring C^* in node n' s.t. for the added attack (T,h) we have $C = C^* + (T,h)$ if $C'((T,h)) \in \{out_{\mathfrak{r}}, pout_{\mathfrak{r}}\}$, and $C = C^* \dot{+} (T,h)$ if $C'((T,h)) = in_{\mathfrak{r}}$.

- Let $C'((T,h)) \in \{out_{\mathfrak{r}}, pout_{\mathfrak{r}}\}$, then C^* is an st-coloring of n' where C^* coincides with C on each $b, (S,t) \in X_{n'}$. To show that C^* is an st-coloring of n' we need to show that the conditions (1)-(6) of stable colorings are satisfied. (1) and (2) carry over from C, as $X_{>n} = X_{>n'}$. (3)-(6) immediately carry over from C as C and C^* coincide on all relevant colors on $X_{n'}$. Hence, C^* is stable for n' and therefore an st-coloring, and $C = C^* + (T,h)$ is an st-coloring for n (as with insert argument nodes, the correct color $out_{\mathfrak{r}}, pout_{\mathfrak{r}}$ follows from C being stable and the properties of tree decompositions).

- Now on the other side let $C'((T,h)) = in_{\mathfrak{r}}$, then C^* is an st-coloring of n' where C^* coincides with C on each $b, (T,h) \in X_{n'}$ with the exception of the argument h (given that $h \in X^A_{n'}$) if $\nexists (S,h) \in X^R_{\geq n'} : C'((S,h)) = in_{\mathfrak{r}}$, in this case we set $C^*(h) = pout_{\mathfrak{a}}$. Again we show that the conditions (1)-(6) are satisfied. (1) and (2) carry over from C, as $X_{>n} = X_{>n'}$. (3), (5), and (6) immediately carry from C, as C and C^* coincide on all relevant colors on $X_{n'}$. Regarding (4), note that this clearly carries over from C for all arguments $a \in X^A_{n'} \setminus \{h\}$, and for the argument h it holds by definition of C^*. Then since C^* is stable from condition (3) it follows $(h \notin X^A_n \vee C(h) \neq in_{\mathfrak{a}})$ and from condition (5) it follows $\forall t \in T \cap X^A_{n'} : C(t) = in_{\mathfrak{a}}$, hence, $C = C^* \dot{+} (T,h)$ is an st-coloring for n.

We showed that in both cases C is an st-coloring of n. □

5.3.4 Join Nodes

In these nodes we combine the colorings of immediate child nodes. The $in_{\mathfrak{a}}$ colored arguments and the $in_{\mathfrak{r}}$ colored attacks of the (to be) combined colorings have to coincide to yield an st-coloring for the join node. Whenever at least one of the child node colorings has a non-provisional *out* color (on arguments or attacks), the non-provisional color carries over to the resulting coloring in the join node. This is because it means in at least one of the sub-trees the color has been "confirmed".

Definition 5.25 (st-coloring: Join)**.** *Let n be a join node with children n', n'', let C be an st-coloring for n', and let D be an st-coloring for n''. If $\{a \mid C(a) = in_{\mathfrak{a}}\} = \{a \mid D(a) = in_{\mathfrak{a}}\}$ and $\{r \mid C(r) = in_{\mathfrak{r}}\} = \{r \mid D(r) = in_{\mathfrak{r}}\}$ then $C \bowtie D$ is an st-coloring for n, where*

$$(C \bowtie D)(b) = \begin{cases} in_{\mathfrak{a}} & \text{if } C(b) = D(b) = in_{\mathfrak{a}} \\ out_{\mathfrak{a}} & \text{if } C(b) = out_{\mathfrak{a}} \vee D(b) = out_{\mathfrak{a}} \\ pout_{\mathfrak{a}} & \text{if } C(b) = D(b) = pout_{\mathfrak{a}} \\ in_{\mathfrak{r}} & \text{if } C(b) = D(b) = in_{\mathfrak{r}} \\ out_{\mathfrak{r}} & \text{if } C(b) = out_{\mathfrak{r}} \vee D(b) = out_{\mathfrak{r}} \\ pout_{\mathfrak{a}} & \text{if } C(b) = D(b) = pout_{\mathfrak{r}} \end{cases}$$

Again we show that our algorithm for st-colorings is correct, i.e., that the conditions (1)-(6) of Definition 5.12 are satisfied. Since we assume that the colorings C and D in the

(a) 4:

b	c	r_4
$in_{\mathfrak{a}}$	$out_{\mathfrak{a}}$	$out_{\mathfrak{r}}$
$out_{\mathfrak{a}}$	$pout_{\mathfrak{a}}$	$out_{\mathfrak{r}}$
$out_{\mathfrak{a}}$	$in_{\mathfrak{a}}$	$pout_{\mathfrak{r}}$
$pout_{\mathfrak{a}}$	$in_{\mathfrak{a}}$	$in_{\mathfrak{a}}$

(b) 5:

b	c	r_4
$in_{\mathfrak{a}}$	$out_{\mathfrak{a}}$	$out_{\mathfrak{r}}$
$pout_{\mathfrak{a}}$	$pout_{\mathfrak{a}}$	$out_{\mathfrak{r}}$
$pout_{\mathfrak{a}}$	$in_{\mathfrak{a}}$	$pout_{\mathfrak{r}}$
$pout_{\mathfrak{a}}$	$in_{\mathfrak{a}}$	$in_{\mathfrak{r}}$

(c) 11:

b	c	r_4
$in_{\mathfrak{a}}$	$in_{\mathfrak{a}}$	$in_{\mathfrak{r}}$
$in_{\mathfrak{a}}$	$pout_{\mathfrak{a}}$	$out_{\mathfrak{r}}$
$in_{\mathfrak{a}}$	$in_{\mathfrak{a}}$	$pout_{\mathfrak{r}}$
$out_{\mathfrak{a}}$	$in_{\mathfrak{a}}$	$pout_{\mathfrak{r}}$
$out_{\mathfrak{a}}$	$pout_{\mathfrak{a}}$	$pout_{\mathfrak{r}}$
$pout_{\mathfrak{a}}$	$in_{\mathfrak{a}}$	$in_{\mathfrak{r}}$

Figure 5.10: St-colorings for join nodes. Example of (a) the join node 4 with highlighted coloring $C \bowtie D$, (b) its child node 5 with highlighted coloring C, and (b) its child node 11 with highlighted coloring D.

child nodes n' and n'' are stable, the conditions for the join node n are satisfied by $C \bowtie D$: the sub-frameworks rooted in nodes n' and n'' are parts of the sub-framework rooted in n which means all conditions carry over. This situation is illustrated in Figure 5.10. Note how in Figure 5.10(a) the provisional color $pout_{\mathfrak{a}}$ for b of C (Figure 5.10(b)) and $pout_{\mathfrak{r}}$ for r_4 of D (Figure 5.10(c)) are "upgraded" to non-provisional colors $out_{\mathfrak{a}}$ and $out_{\mathfrak{r}}$, respectively. This is because b is colored $out_{\mathfrak{a}}$ in D and r_4 is colored $out_{\mathfrak{r}}$ in C, which means for the join node 4 that the "proof" for the non-provisional colors occurs in the sub-framework.

Lemma 5.26. *If for the child nodes n', n'' of a join node n each st-coloring is stable, then each st-coloring C of n is stable.*

Proof. We need to show that each of the conditions (1)-(6) of Definition 5.12 is met. (1) and (2) immediately carry over from n' and n'' since $X^A_n = X^A_{n'} = X^A_{n''}$ and $X^R_n = X^R_{n'} = X^R_{n''}$. (3) holds because it has to hold both for all (S, a) in $X^R_{n'}$ and in $X^R_{n''}$ by assumption. (4) is satisfied, as we set an argument to $out_{\mathfrak{a}}$ only if it is colored $out_{\mathfrak{a}}$ in at least one of the two colorings of n' and n''. Similarly, (5) and (6) carry over from n' and n''. □

To show the other direction, i.e., completeness of our algorithm for join nodes, we have to show that for each stable coloring C in join node n there are st-colorings C^* and D^* in the child nodes of n s.t. $C = C^* \bowtie D^*$ and therefore C is an st-coloring of the join node. If we assume that C is stable, we can easily construct such C^* and D^* for the child nodes—for the construction we have to take into account that the non-provisional colors of C might be provisional in C^* or D^* (but not both), then we can easily check that the thereby constructed colorings are stable for the child nodes. For example, we can check that the highlighted coloring C is stable for node 4 in the running example, illustrated in Figure 5.10. When constructing the corresponding coloring C^* for child node 5 we see that there is no possible attack towards b that is colored $in_{\mathfrak{a}}$ and we have to set $C^*(b) = pout_{\mathfrak{a}}$. Similarly, we set $D^*(r_4) = pout_{\mathfrak{r}}$ since we can extend D^* to a coloring on $X_{\geq 11}$ such that no argument in the tail of the attack r_4 is colored $pout_{\mathfrak{a}}$ or $out_{\mathfrak{a}}$. C^* and D^* are stable for their respective nodes 5 and 11, and in fact we see in Figure 5.10(b) and Figure 5.10(c), respectively, that the constructed colorings are indeed st-colorings.

Lemma 5.27. *Let C be a stable coloring for a join node n. If in the child nodes n', n'' of n stable colorings and st-colorings coincide then C is an st-coloring for n.*

Proof. Let C' be the extended coloring of C on $X_{\geq n}$ s.t. the conditions of Definition 5.12 are satisfied.

We will show that there are st-colorings C^* for n' and D^* for n'' such that $C = C^* \bowtie D^*$. We define C^* such that it coincides with C on each $a, (T,h) \in X_n$ with the exception of arguments b s.t. $C(b) = out_{\mathfrak{a}}$ and $\nexists (S,b) \in X^R_{\geq n'} : C'((S,b)) = in_{\mathfrak{r}}$, for these arguments we set $C^*(b) = pout_{\mathfrak{a}}$, and the second exception of attacks (S,b) s.t. $C((S,b)) = out_{\mathfrak{r}}$ and $\nexists s \in S \cap X^A_{\geq n'} : C'(s) \in \{out_{\mathfrak{a}}, pout_{\mathfrak{a}}\}$, for these attacks we set $C^*((S,b)) = pout_{\mathfrak{r}}$. We show that properties (1)-(6) of stable colorings are satisfied for C^*. (1) and (2) carry over from C, as $X_{>n} = X_{>n'} = X_{>n''}$. (3) is satisfied as for C and C^* it holds

$$\{(T,h) \in X^R_n \mid C((T,h)) \in \{out_{\mathfrak{r}}, pout_{\mathfrak{r}}\})\} = \{(T,h) \in X^R_{n'} \mid C^*((T,h)) \in \{out_{\mathfrak{r}}, pout_{\mathfrak{r}}\})\}.$$

(4) and (6) are satisfied by definition of C^*. (5) immediately carries over from C. Hence, C^* is stable (and D^* as well due to symmetry). Finally, note that by construction of C^* and D^* it holds

- $\{a \mid C'(a) = in_{\mathfrak{a}}\} \cap X^A_n = \{a \mid C(a) = in_{\mathfrak{a}}\} = \{a \mid C^*(a) = in_{\mathfrak{a}}\} = \{a \mid D(a) = in_{\mathfrak{a}}\}$ and
- $\{r \mid C'(r) = in_{\mathfrak{r}}\} \cap X^R_n = \{r \mid C(r) = in_{\mathfrak{r}}\} = \{r \mid C^*(r) = in_{\mathfrak{r}}\} = \{r \mid D^*(r) = in_{\mathfrak{r}}\}$.

Hence, $C = C^* \bowtie D^*$ is an st-coloring for n. □

5.3.5 Final Steps for Stable Semantics

Towards characterizing the stable extensions via our st-colorings, we can now utilize the soundness and completeness results that we established for each node type. We obtain the following results that underpin the adequacy of the algorithm.

Proposition 5.28. *Let $\mathcal{T} = (V_{\mathcal{T}}, E_{\mathcal{T}})$ be a nice tree-decomposition of a SETAF SF. Then in each node $n \in V_{\mathcal{T}}$ st-colorings and stable colorings coincide.*

Proof. ($\subseteq$) Follows by structural induction over the tree-decomposition structure with the leaves as a base (Lemma 5.15) and the forget, insert, and join nodes as steps (Lemma 5.19, Lemma 5.23, Lemma 5.26).

($\supseteq$) First note that by ($\subseteq$) all st-colorings in each node are stable. Then the statement follows by structural induction over the tree-decomposition structure with the leaves as a base (Lemma 5.16) and the forget, insert, and join nodes as steps (Lemma 5.20, Lemma 5.24, Lemma 5.27). □

Effectively, we established that the root node indeed characterizes the stable extensions. We can decide $Cred_{stb}/Skept_{stb}$ for an argument $a \in A$ by flagging colorings that contain/do not contain a. In each node we update the flag accordingly; the flag in the root node indicates credulous/skeptical acceptance. We can keep the count of extensions w.r.t. each coloring, and to enumerate the extensions once the dynamic programming algorithm is done we can traverse the tree top-down and output the extensions with linear delay (cf. [JPW09, DPW12]). Hence, we obtain the following upper bound for the problems of stable semantics.

Theorem 5.29. *With the presented algorithm, $Cred_{stb}$, $Skept_{stb}$ as well as counting the number of stable extensions can be done in time $O(5^k \cdot k \cdot (|A| + |R|))$. Moreover, we can enumerate all stable extensions with linear delay.*

Proof. The correctness of the algorithm (i.e., we account *only* for stable colorings in the root node) and the completeness of the algorithm (i.e., we account for *all* the stable colorings in the root node) is shown in Proposition 5.28. Then, from Proposition 5.13 we get that we exactly characterize $stb(SF)$.

We can assume the number of nodes to be bounded by $O(|A| + |R|)$ and that we can find and access rows in linear time w.r.t. k. For each node, the number of (valid) colorings (i.e., rows in our tables of colorings) is bounded by 3^k. In leaf nodes, we can check the colorings in time $O(k^2)$ for each of the $O(3^k)$ possible colorings, resulting in $O(3^k \cdot k^2)$. In forget nodes, we can check the conditions and compute eventually resulting colorings in time $O(k)$ for each of the $O(3^k)$ colorings of the child node, resulting in $O(3^k \cdot k)$. In insert nodes, we can check the conditions and compute eventually resulting colorings in time $O(k^2)$ for each of the $O(3^k)$ colorings of the child node, resulting in $O(3^k \cdot k^2)$. Finally, for join nodes we have to consider $3^k \cdot 3^k = 9^k$ pairs. However, we only need to consider 5^k pairs if we assume the data structure to be properly sorted, e.g. lexicographically by treating the colors $in_{\mathfrak{a}}/in_{\mathfrak{r}}$ as 0 and $pout_{\mathfrak{a}}/out_{\mathfrak{a}}/pout_{\mathfrak{r}}/out_{\mathfrak{r}}$ as 1. As each table has $O(3^k)$ rows, sorting is in $O(3^k \cdot k)$. Let C be a coloring such that $m \leq k$ arguments/attacks are colored as $in_{\mathfrak{a}}/in_{\mathfrak{r}}$. There exist at most 2^{k-m} distinct colorings C' with $\forall x : (C(x) \in \{in_{\mathfrak{a}}, in_{\mathfrak{r}}\} \Leftrightarrow C'(x) \in \{in_{\mathfrak{a}}, in_{\mathfrak{r}}\})$. There are $\binom{k}{m}$ possibilities resulting from the choice of m, resulting in $\sum_{m=0}^{k} \binom{k}{m} \cdot 2^{k-m} \cdot 2^{k-m} = 5^k$ join pairs. We can then compute $C \bowtie D$ in $O(k)$, resulting in $O(5^k \cdot k)$ for join nodes, dominating the runtime of the other node types. The resulting runtime for the algorithm is $O(5^k \cdot k \cdot (|A| + |R|))$. □

5.4 Characterizing Admissible Sets

For admissible sets we have to account for arguments that are neither in an extension nor attacked by it. Hence, we extend colorings the colorings we used to characterize stable extensions to reflect these "undecided" arguments. In addition to the colors $in_{\mathfrak{a}}, out_{\mathfrak{a}}, pout_{\mathfrak{a}}$ for arguments and $in_{\mathfrak{r}}, out_{\mathfrak{r}}, pout_{\mathfrak{r}}$ for attacks, we consider the color $und_{\mathfrak{a}}$ for arguments, and the two colors $und_{\mathfrak{r}}, pund_{\mathfrak{r}}$ for attacks. Intuitively, with the $und_{\mathfrak{a}}$ color we indicate arguments that are neither in an admissible set E nor attacked by E

(this situation is not allowed in stable extensions). The color $und_\mathfrak{r}$ indicates attacks (T, h) that in relation to an admissible set E neither attack E (i.e., have $T \subseteq E$) nor have a defeated argument in their tail (i.e., all arguments $t \in T$ are either in the extension or undecided, and at least one argument is not in the extension). Analogous to the $out_\mathfrak{r}$ color, we have to account for *provisionally undecided* attacks. That is, if all arguments $t \in T$ we encountered *so far* in the bottom-up computation of the algorithm have been colored $in_\mathfrak{a}$, but we could encounter an additional tail-argument that we will color $und_\mathfrak{a}$. In insert argument nodes, the $pund_\mathfrak{r}$ color can be "upgraded" to $und_\mathfrak{r}$ if we color the inserted argument $und_\mathfrak{a}$ and it appears in the tail of the attack. As with the $out_\mathfrak{r}/pout_\mathfrak{r}$ colors we have to discard attacks colored $pund_\mathfrak{r}$ in forget attack nodes, as we cannot update this color to $und_\mathfrak{r}$ anymore. Hence, for admissible sets we get the following set of colors.

$$\mathcal{C}_{adm} = \{in_\mathfrak{a}, out_\mathfrak{a}, pout_\mathfrak{a}, und_\mathfrak{a}, in_\mathfrak{r}, out_\mathfrak{r}, pout_\mathfrak{r}, und_\mathfrak{r}, pund_\mathfrak{r}\}$$

We next characterize the *admissible* colorings for each node of the tree decomposition, similar to the stable colorings of Section 5.3 Again we will utilize this notion to establish the correctness of our algorithm: as we will show, if in each node we compute as ad-colorings exactly the admissible colorings then we ultimately characterize exactly the admissible sets with the colorings for SF in the root node.

Definition 5.30. *A coloring $C : X_n \to \mathcal{C}_{adm}$ for a node $n \in V_\mathcal{T}$ is* admissible *if we can extend C to a coloring $C' : X_{\geq n} \to \mathcal{C}_{adm}$ such that the following conditions are met for each $a \in X^A_{\geq n}$ and each $(T, h) \in X^R_{\geq n}$:*

1. *if $a \in X^A_{>n}$ then $C'(a) \in \{in_\mathfrak{a}, out_\mathfrak{a}, und_\mathfrak{a}\}$,*
2. *if $r \in X^R_{>n}$ then $C'((T, h)) \in \{in_\mathfrak{r}, out_\mathfrak{r}, und_\mathfrak{r}\}$,*
3. *if $C'(a) = in_\mathfrak{a}$ then $\forall (S, a) \in X^R_{\geq n} : C'((S, a)) \in \{out_\mathfrak{r}, pout_\mathfrak{r}\}$,*
4. *$C'(a) = out_\mathfrak{a}$ if and only if $\exists (S, a) \in X^R_{\geq n} : C'((S, a)) = in_\mathfrak{r}$,*
5. *if $C'((T, h)) = in_\mathfrak{r}$ then $\forall t \in T \cap X^A_{\geq n} : C'(t) = in_\mathfrak{a}$,*
6. *$C'((T, h)) = out_\mathfrak{r}$ if and only if $\exists t \in T \cap X^A_{\geq n} : C'(t) \in \{out_\mathfrak{a}, pout_\mathfrak{a}\}$, and*
7. *$C'((T, h)) = und_\mathfrak{r}$ if and only if $\exists t \in T \cap X^A_{\geq n} : C'(t) \in \{und_\mathfrak{a}, pund_\mathfrak{a}\}$ and $\nexists t \in T \cap X^A_{\geq n} : C'(t) \in \{out_\mathfrak{a}, pout_\mathfrak{a}\}$.*

For a coloring C in node n we define by $e_n(C)$ the set of such extended colorings C' s.t. (1)-(7) are met. These are the characterized colorings.

First note that we give no explicit condition for arguments w.r.t. the color $und_\mathfrak{a}$. This is due to the fact that we can color an argument undecided "at will", only with the exception that an $in_\mathfrak{r}$-colored attack towards the argument mandates we color it $out_\mathfrak{a}$

(reflected by condition (4)). Next we point out that the conditions (1)-(6) are almost identical to stable colorings as per Definition 5.30, with the only exception that in (1) and (2) we also allow undecided colors for the arguments and attacks that only appear strictly below the current node. Conditions (3)-(6) are exactly as we had them in stable colorings and guarantee admissibility. Finally, the novel condition (7) is similar to (6) and characterizes a "confirmed" undecided color for attacks where it is required that at least one tail-argument is undecided. Moreover, (7) captures the "priority" of the $out_{\mathfrak{r}}/pout_{\mathfrak{r}}$ colors over the $und_{\mathfrak{r}}/pund_{\mathfrak{r}}$ colors, in that the undecided colors can only be applied if the attack in question is not necessarily out due to an argument in its tail. We want to point out that the conditions (6) and (7) imply the following additional property for provisionally undecided attacks:

$$\text{If } C'((T,h)) = pund_{\mathfrak{r}} \text{ then } \forall t \in T \cap X^A_{\geq n} : C'(t) = in_{\mathfrak{a}}.$$

This is due to the fact that if the attack (T,h) had a tail-argument that is colored $out_{\mathfrak{a}}$ or $pout_{\mathfrak{a}}$, the attack would have to be colored $out_{\mathfrak{r}}$ by (6), and if there was a tail argument colored $und_{\mathfrak{a}}$ the attack would have to be colored $und_{\mathfrak{r}}$ by (7). Since the only colors for arguments for admissible sets are $in_{\mathfrak{a}}$, $out_{\mathfrak{a}}$, $pout_{\mathfrak{a}}$, and $und_{\mathfrak{a}}$, this property is implied.

Proposition 5.31. *Let $\mathcal{T} = (V_{\mathcal{T}}, E_{\mathcal{T}})$ be a nice tree-decomposition of a SETAF $SF = (A,R)$ and let $r \in V_{\mathcal{T}}$ be the root node of $\mathcal{T}$. Moreover, let $\mathcal{C}_r$ be the set of admissible colorings for r. Then*

$$adm(SF) = \{\{a \mid C'(a) = in_{\mathfrak{a}}\} \mid C \in \mathcal{C}_r, C' \in e_r(C)\}.$$

Proof. Since the root node is empty, we have that $X^A_{>n} = A$ and $X^R_{>n} = R$, which by (1) and (2) means that we have no provisional colors in each extended coloring $C' \in e_r(C)$. Condition (3) and (6) give us the following property (a) and (4) and (5) give us the following property (b):

(a) If $C'(a) = in_{\mathfrak{a}}$ then $\forall (T,a) \in R\ \exists t \in T : C'(t) = out_{\mathfrak{a}}$, and

(b) $C'(a) = out_{\mathfrak{a}}$ if and only if $\exists (T,a) \in R\ \forall t \in T : C'(t) = in_{\mathfrak{a}}$.

(a) exactly characterizes conflict-freeness, i.e., for each attack (T,a) towards an argument a that is in the extensions (i.e., in the set $\{a \mid C'(a) = in_{\mathfrak{a}}\}$) there is at least one argument in T that is *not* in the extension. (a) and (b) together characterize defense, i.e., for each argument a that has the color $in_{\mathfrak{a}}$ for each attack (T,a) towards a, there is a counter-attack (S,t) for some $t \in T$ such that $S \subseteq \{b \mid C'(b) = in_{\mathfrak{a}}\}$. Conflict-freeness and defense exactly characterizes the admissible sets. □

We adapt our running example for admissible semantics (i.e., we compute the ad-colorings w.r.t. the tree decomposition in Figure 5.3) and along the way we introduce extension-counting. This concept is illustrated together with the ad-colorings in Figure 5.11.

Say one is interested only in extensions that contain a certain argument. Then we can use the "filter-method" to (1) only obtain the desired results and (2) speed up the computation by omitting to compute irrelevant colorings. In a nutshell, we only care about the colorings that contain (or do not contain) a certain label for an argument/attack and discard colorings that do not match. Note that for nodes n where $X_{\geq n}$ does not contain the argument/attack in question we cannot omit any colorings. It is easy to see that the (adapted) soundness and completeness results carry over to this setting. We will illustrate this method in our running example (see Figure 5.11) by filtering for colorings that set argument c to $und_{\mathfrak{a}}$. Note that we can also filter for multiple arguments/attacks, i.e., if we are only interested in colorings (and, ultimately, extensions) that set argument a to $und_{\mathfrak{a}}$ and at the same time attack r_2 to $in_{\mathfrak{r}}$ the same principles apply. Finally, on the account of filtering we want to point out optimization potential for implementations: in our running example we filter for colorings that set c to $und_{\mathfrak{a}}$, this has implications for the incoming and outgoing attacks. In particular, in this case all attacks *towards* c cannot be $in_{\mathfrak{r}}$ (as this would mean c has to be $out_{\mathfrak{a}}$). Similarly, outgoing attacks from c cannot be colored $in_{\mathfrak{r}}$ either. Moreover, we do not consider colorings where outgoing attacks from c are colored $pund_{\mathfrak{r}}$ (as we know that every node containing the argument c will have these attacks set to the non-provisional $und_{\mathfrak{r}}$ color). These optimizations depend on the semantics in question and the filtered colors. Note that in our running example for simplicity we do *not* apply these optimization strategies.

We next formally introduce the concept of *extension counting* for our algorithm. In that, we can without much additional effort utilize our tree-decomposition based computation to *count* the number of extended colorings $e_n(C)$ on the (partial) framework $SF_{\geq n}$ w.r.t. a given node n and a coloring C. Ultimately, this means we can count the number of extensions w.r.t. the semantics in question. To this end, we define the function $\#_n(C)$ and show that this coincides with $|e_n(C)|$.

Definition 5.32 (Counting)**.** *Let $\mathcal{T} = (V_{\mathcal{T}}, E_{\mathcal{T}})$ be a nice tree-decomposition of a SETAF $SF = (A, R)$ and let $n \in V_{\mathcal{T}}$ be a node of $\mathcal{T}$. Moreover let C be an (st/ad/co)-coloring for n. We define the number of extended colorings $\#_n(C)$ as follows:*

- *If n is a leaf node, we set $\#_n(C) = 1$ for all C.*
- *If n is a forget node with child n' with colorings $\mathcal{C}_{n'}$ s.t. $X_n = X_{n'} \setminus \{x\}$, we set*

$$\#_n(C) = \sum_{C' \in \mathcal{C}_{n'}, C = C' - x} \#_{n'}(C'),$$

  i.e., $\#_n(C)$ is the sum of all $\#_{n'}(C')$ where $C = C' - x$.
- *If n is an insert node with child n' with colorings $\mathcal{C}_{n'}$ s.t. $X_{n'} = X_n \setminus \{x\}$, we set*

$$\#_n(C) = \sum_{C' \in \mathcal{C}_{n'}, C = C' \circ x, \circ \in \{+, \dot{+}, \ddot{+}\}} \#_{n'}(C'),$$

  i.e., $\#_n(C)$ is the sum of all $\#_{n'}(C')$ where $C = C' + x$, $C = C' \dot{+} x$, or $C = C' \ddot{+} x$.

- *If n is a join node with child nodes n', n'' with colorings $\mathcal{C}_{n'}, \mathcal{C}_{n''}$, respectively, we as set*

$$\#_n(C) = \sum_{C' \in \mathcal{C}_{n'} C'' \in \mathcal{C}_{n''}, C = C' \bowtie C''} \#_{n'}(C') \cdot \#_{n''}(C''),$$

i.e., $\#_n(C)$ is the sum of all products $\#_{n'}(C') \cdot \#_{n''}(C'')$ where $C = C' \bowtie D'$.

With this method, we can count the number of extension candidates, and for the root node, the number of extensions. We want to highlight that this result not only holds for stable semantics, but also for the other semantics we consider in this work, i.e., admissible and complete semantics. We will present the respective definitions for ad- and co-colorings in Sections 5.4 and 5.5, respectively.

Proposition 5.33. *Let $\mathcal{T} = (V_\mathcal{T}, E_\mathcal{T})$ be a nice tree-decomposition of a SETAF $SF = (A, R)$ and let $n \in V_\mathcal{T}$ be a node of $\mathcal{T}$. Moreover let C be an (st/ad/co)-coloring for n. Then $\#_n(C) = |e_n(C)|$.*

It is easy to see why this holds; similar ideas have also been applied in previous work for AFs [DPW12, Cha12, FHM19]. For leaf nodes this is clearly the case. For forget nodes, the extended colorings of n and the extended colorings of its child node exactly coincide. For insert nodes, even though some of the colors may be changed to non-provisional by the construction, note that there is still a clear correspondence of the colorings of n and its child node. In this case, the number of extended colorings that correspond to a coloring of the child node depends on how many of the operations $+, \dot{+}, \ddot{+}$ are applicable. Finally, for join nodes we have to account for all the combinations of colorings.

Clearly, we can combine the ideas of filtering and counting, and count the number of filtered colorings (extensions)—as we illustrate with our running example. Finally note that both filtering and counting can analogously be done for all the semantics discussed in this chapter, and the additional effort is polynomial for each node.

5.4.1 Leaf Nodes

We start by characterizing *leaf nodes*. Intuitively, in leaf nodes for each argument and each attack we guess one of three possibilities (in contrast to the two possibilities of stable semantics): *in*, *out*, or *und* (in the second and third case we use either the confirmed or provisional color, as we will explain below). We then again keep every "consistent" coloring. Again whether an argument/attack is colored *out* or *pout* depends only on whether the color is "confirmed", this distinction is analogous to stable semantics. For admissible semantics, we have no provisionally undecided arguments. Since there is no "maximality"-requirement as in e.g. complete semantics, we freely apply the $und_\mathfrak{a}$ color. However, the color $out_\mathfrak{a}$ has "priority" if it is applicable. In summary, whenever an argument is the head of an attack we color $in_\mathfrak{r}$ we have to apply the color $out_\mathfrak{a}$; for the color $in_\mathfrak{a}$ every attack towards the argument has to be colored $out_\mathfrak{a}/pout_\mathfrak{a}$, and the colors $und_\mathfrak{a}$ and $pout_\mathfrak{a}$ can always be applied if the color $out_\mathfrak{a}$ is not mandated. For attacks, we

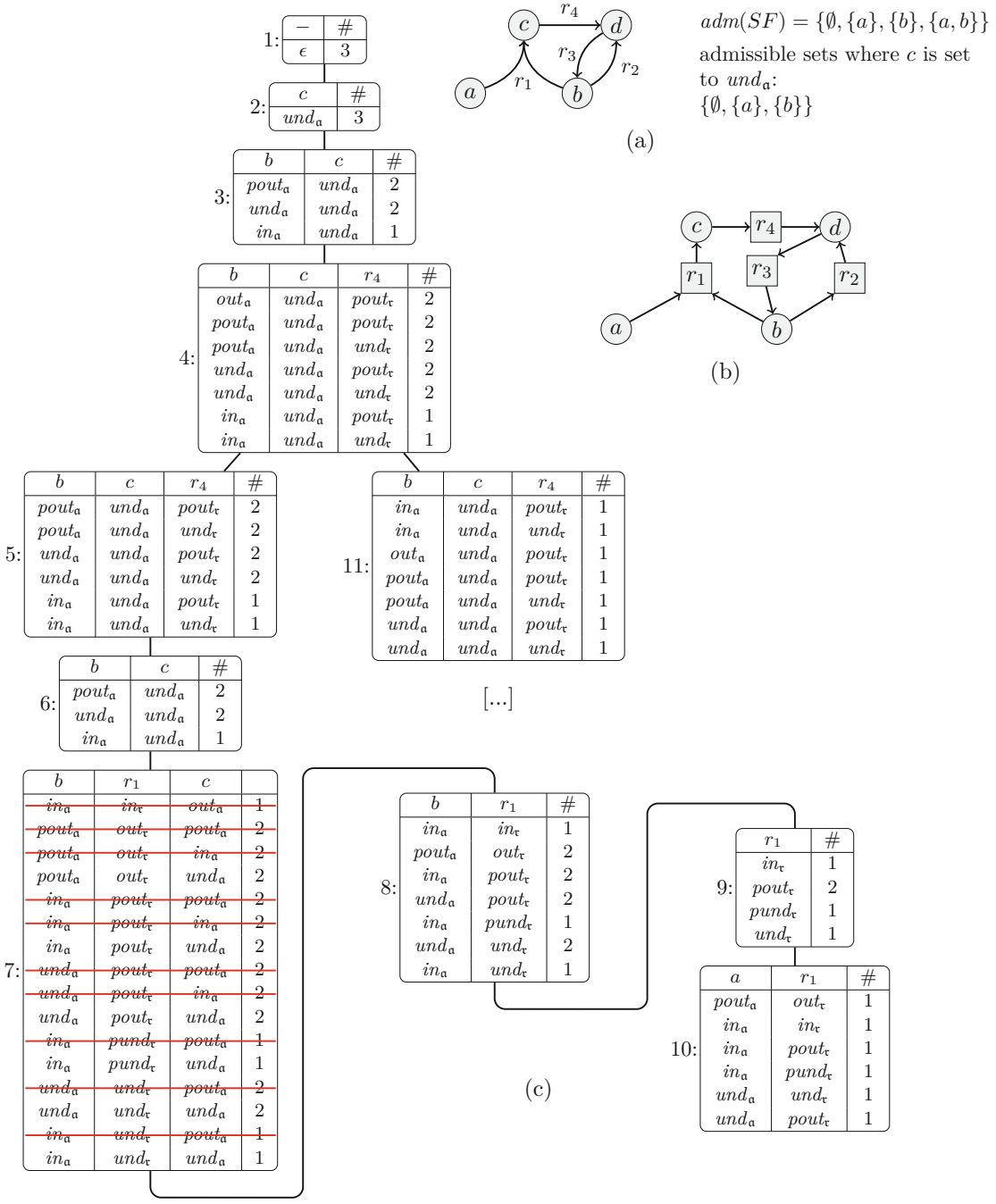


Figure 5.11: Selected ad-colorings for our running example. (a) Our running example SETAF SF with its admissible sets, (b) its incidence-graph $\texttt{Inc}(SF)$, and (c) the ad-colorings for SF w.r.t. the tree decomposition given in Figure 5.3 "filtered" for the argument c set to $und_{\mathfrak{a}}$ (discarded colorings are indicated in node 7). The "#" columns show the number of colorings in $e_n(C)$ for the coloring C of each line (cf. Definition 5.32). We omit the branch below node 11.

do distinguish the cases $und_{\mathfrak{r}}$ and $pund_{\mathfrak{r}}$. This is very similar to the distinction between $out_{\mathfrak{r}}$ and $pout_{\mathfrak{r}}$: an undecided attack gets a provisional color unless there is a "proof", i.e., an undecided argument in its tail.

Definition 5.34 (ad-colorings: Leaf)**.** *An* ad-coloring *for a* leaf node $n \in V_{\mathcal{T}}$ *is each coloring that satisfies the following conditions. For each argument* $a \in X_n^A$*:*

$$C(a) = in_{\mathfrak{a}} \Rightarrow \forall (T, a) \in X_n^R \colon C((T, a)) \in \{pout_{\mathfrak{r}}, out_{\mathfrak{r}}\}$$
$$C(a) = out_{\mathfrak{a}} \Leftrightarrow \exists (T, a) \in X_n^R \colon C((T, a)) = in_{\mathfrak{r}}$$

For every attack $r = (T, h) \in X_n^R$*:*

$$C(r) = in_{\mathfrak{r}} \Rightarrow \forall t \in T \cap X_n^A \colon C(t) = in_{\mathfrak{a}}$$
$$C(r) = out_{\mathfrak{r}} \Leftrightarrow \exists t \in T \cap X_n^A \colon C(t) \in \{pout_{\mathfrak{a}}, out_{\mathfrak{a}}\}$$
$$C(r) = und_{\mathfrak{r}} \Leftrightarrow \exists t \in T \cap X_n^A \colon C(t) = und_{\mathfrak{a}} \wedge \nexists t \in T \cap X_n^A \colon C(t) \in \{pout_{\mathfrak{a}}, out_{\mathfrak{a}}\}$$

The new $und_{\mathfrak{a}}$ color is similar in spirit to the $pout_{\mathfrak{a}}$ and $in_{\mathfrak{a}}$ color, in that it can be assigned to an argument without reason. The only situation where an argument cannot be $und_{\mathfrak{a}}$ (or $pout_{\mathfrak{a}}$, $in_{\mathfrak{a}}$) is when there is already an attack towards the argument that is colored $in_{\mathfrak{r}}$. The colors for attacks also follow the same intuitive ideas. We want to highlight that the $out_{\mathfrak{r}}$ color has "priority" over the $und_{\mathfrak{r}}$ color: if in the tail T of an attack (T, h) we color some arguments $und_{\mathfrak{a}}$ and others $out_{\mathfrak{a}}/pout_{\mathfrak{a}}$, the color of the attack (T, h) has to be $out_{\mathfrak{r}}$. This corresponds to the fact that the argument h is indeed defended against this attack (which would not be captured if the attack was colored $und_{\mathfrak{r}}$).

We now establish that all ad-colorings in the leaf nodes coincide with the admissible colorings.

Lemma 5.35. *Each ad-coloring* C *for a leaf node* $n \in V_{\mathcal{T}}$ *is admissible.*

Proof. We need to show that each of the conditions (1)-(7) of Definition 5.30 is met. (1) and (2) are trivially true, as for leaves it holds $X_{>n}^A = X_{>n}^R = \emptyset$. (3)-(7) follow directly from the conditions in Definition 5.34. □

In our running example in Figure 5.11 the node 10 is a leaf node. Regarding the counting feature we recall that in leaf nodes all colorings characterize exactly one extended coloring (as $X_{\geq n} = X_n$).

Lemma 5.36. *Each admissible coloring* C *for a leaf node* $n \in V_{\mathcal{T}}$ *is an ad-coloring.*

Proof. It follows directly from the definition of admissible colorings that C adheres to the conditions in Definition 5.34. □

5.4.2 Forget Nodes

We again examine forget argument nodes and forget attack nodes separately. Let n be a *forget argument node* with child n' such that $X_n^A = X_{n'}^A \setminus \{a\}$. We have to discard all colorings C where $C(a) = \mathit{pout}_{\mathfrak{a}}$, as in these colorings a is supposed to be attacked by the characterized admissible set. As we forget a in the current node, this cannot happen.

Definition 5.37 (ad-coloring: Forget Argument)**.** *Let n be a forget argument node with child n' such that $X_n^A = X_{n'}^A \setminus \{a\}$, and let C be an ad-coloring for n'. If $C(a) \neq \mathit{pout}_{\mathfrak{a}}$, then $C - a$ is an ad-coloring for n, where $C - x$ for each $b \in X_n$ is defined as follows:*

$$(C - x)(b) = C(b)$$

We handle *forget attack nodes* in the same way. However, this time we also have to account for provisionally undecided attacks in addition to the provisionally out attacks.

Definition 5.38 (ad-coloring: Forget Attack)**.** *Let n be a forget attack node with child n' such that $X_n^R = X_{n'}^R \setminus \{r\}$, and let C be an ad-coloring for n'. If $C(r) \notin \{\mathit{pout}_{\mathfrak{r}}, \mathit{pund}_{\mathfrak{r}}\}$, then $C - r$ is an ad-coloring for n, where $C - x$ for each $b \in X_n$ is defined as follows:*

$$(C - x)(b) = C(b)$$

We next establish that in forget nodes admissible colorings and ad-colorings coincide.

Lemma 5.39. *If for the child node n' of a forget node n each ad-coloring is admissible, then each coloring C of n is admissible.*

Proof. We need to show that each of the conditions (1)-(7) of Definition 5.30 is met. (1) and (2) are true because they hold for n', and if the forgotten argument/attack is provisional, then it is discarded. The satisfaction of (3)-(7) immediately carries over from n'. □

In Figure 5.11 node 9 is a forget argument node where $X_9 = X_{10} \setminus \{a\}$. As expected, we discard the colorings C where $C(a) = \mathit{pout}_{\mathfrak{a}}$. Note that both colorings C_1 and C_2 of node 10 with $C_1(a) = \mathit{in}_{\mathfrak{a}}$, $C_2(a) = \mathit{und}_{\mathfrak{a}}$, $C_1(r_1) = C_2(r_2) = \mathit{pout}_{\mathfrak{r}}$ are kept for node 9, which means the coloring C_3 of node 9 characterizes two extended colorings on $X_{\geq 9}$, as indicated in the # column.

Lemma 5.40. *Let C be an admissible coloring of a forget node n. If in the child node n' of n all admissible colorings and ad-colorings coincide then C is an ad-coloring for n.*

Proof. Let C' be the extended coloring of C on $X_{\geq n}$ s.t. the conditions of Definition 5.30 are satisfied.

We start with *forget argument* nodes. We show that C^* is an ad-coloring for n' where C^* coincides with C on each $a, (T, h) \in X_n$ and for the forgotten argument $b \in X_{n'} \setminus X_n$ it holds $C^*(b) \neq \mathit{pout}_{\mathfrak{a}}$.

- If $C'(b) = in_{\mathfrak{a}}$ then clearly C^* is admissible for n' with $C^*(b) = in_{\mathfrak{a}}$: in this case each condition (1)-(7) carries over from C. Then by assumption this means C^* is an ad-coloring for n'.
- If $C'(b) = out_{\mathfrak{a}}$ then C^* is admissible for n' with $C^*(b) = out_{\mathfrak{a}}$: this is because by condition (4) we have for node n that $\exists (S,a) \in X^R_{\geq n} : C'((S,a)) \in \{out_{\mathfrak{r}}, pout_{\mathfrak{r}}\}$, and since $X^R_{\geq n} = X^R_{\geq n'}$ this carries over to node n'. Again the other conditions clearly carry over, and by assumption this means C^* is an ad-coloring for n'.
- Finally, if $C'(b) = und_{\mathfrak{a}}$ then C^* is admissible for n' with $C^*(b) = und_{\mathfrak{a}}$: in this case each condition (1)-(7) carries over from C. Then by assumption this means C^* is an ad-coloring for n'.

In all three cases we see that there is a corresponding ad-coloring C^* in the child node n' that gives us C as an ad-coloring in n, and since there is no other possibility, this means that $C = C^* - b$ is an ad-coloring of n.

We continue with *forget attack* nodes. We show that C^* is an ad-coloring for n' where C^* coincides with C on each $a, (T,h) \in X_n$ and for the forgotten attack $(S,t) \in X_{n'} \setminus X_n$ it holds $C^*((S,t)) \neq pout_{\mathfrak{r}}$ and $C^*((S,t)) \neq pund_{\mathfrak{r}}$.

- If $C'((S,t)) = in_{\mathfrak{r}}$ then clearly C^* is admissible for n' with $C^*(b) = in_{\mathfrak{r}}$: in this case each condition (1)-(7) carries over from C. Then by assumption this means C^* is an ad-coloring for n'.
- If $C'((S,t)) = out_{\mathfrak{r}}$ then C^* is admissible for n' with $C^*((S,t)) = out_{\mathfrak{r}}$: this is because by condition (6) we have for node n that $\exists s \in S \cap X^A_{\geq n} : C'(s) \in \{out_{\mathfrak{a}}, pout_{\mathfrak{a}}\}$, and since $X^A_{\geq n} = X^A_{\geq n'}$ this carries over to node n'. Again the other conditions clearly carry over, and by assumption this means C^* is an ad-coloring for n'.
- Finally, if $C'((S,t)) = und_{\mathfrak{r}}$ then clearly C^* is admissible for n' with $C^*(b) = und_{\mathfrak{r}}$: this is because by condition (7) we have for node n that $\exists s \in S \cap X^A_{\geq n} : C'(s) = und_{\mathfrak{a}}$, and since $X^A_{\geq n} = X^A_{\geq n'}$ this carries over to node n'. Again the other conditions clearly carry over, and by assumption this means C^* is an ad-coloring for n'.

In all three cases we see that there is a corresponding ad-coloring C^* in the child node n' that gives us C as an ad-coloring in n, and since there is no other possibility, this means that $C = C^* - b$ is an ad-coloring of n. □

5.4.3 Insert Nodes

As with forget nodes, we handle the case of inserting an argument and inserting an attack separately. In Section 5.3.3 we introduced two operations $(C + x, C \dot{+} x)$ to compute the st-colorings when aiming at characterizing the stable extensions—these two operations corresponded to the cases where the inserted argument/attack is *in* or *out*. For admissible

sets we consider similar scenarios but also need to consider the case where the added argument/attack is *undecided*, which is reflected by the third operation $C\ddot{+}x$.

Regarding insert argument nodes, the intuition for the operations $C + a$ and $C\dot{+}a$ are the same as it was for stable semantics: $C + a$ corresponds to the case where the added argument a is colored $out_{\mathfrak{a}}/pout_{\mathfrak{a}}$ and can be applied whenever a is not in the tail of an attack that is colored $in_{\mathfrak{r}}$. $C\dot{+}a$ characterized the case where a is colored $in_{\mathfrak{a}}$ and can be applied whenever there is no attack colored $in_{\mathfrak{r}}$, $und_{\mathfrak{r}}$, or $pund_{\mathfrak{r}}$ towards a (or equivalently, if all attacks towards a are colored $out_{\mathfrak{r}}$ or $pout_{\mathfrak{r}}$). $C\ddot{+}a$ captures the case where a is colored $und_{\mathfrak{a}}$ and can be applied whenever a is not in the head or tail of an attack that is colored $in_{\mathfrak{r}}$. In the first case $(C+a)$, the added argument a is either colored provisionally out ($pout_{\mathfrak{a}}$) if the responsible attack is not in the current bag, or "confirmed" $out_{\mathfrak{a}}$ otherwise. If a is in the tail of an attack that is colored $pout_{\mathfrak{r}}$ in the original coloring of the child node, then this color is "upgraded " to $out_{\mathfrak{r}}$. Similarly, in the third case $(C\ddot{+}a)$ a provisional color $pund_{\mathfrak{r}}$ of attacks that have the newly added argument a in its tail the color is "upgraded" to $und_{\mathfrak{r}}$, as the color $und_{\mathfrak{a}}$ of a "confirms" the provisional color of the attack. Formally:

Definition 5.41 (ad-coloring: Insert Argument)**.** *Let n be an insert argument node with child n' such that $X_{n'}^A = X_n^A \setminus \{a\}$, and let C be an ad-coloring for n'.*

- *if $\nexists (T,h) \in X_{n'}^R : (C((T,h)) \in \{in_{\mathfrak{r}}, und_{\mathfrak{r}}, pund_{\mathfrak{r}}\} \wedge a \in T)$, then $C + a$ is an ad-coloring for n;*
- *if $\nexists (T,a) \in X_{n'}^R : (C((T,a)) \in \{in_{\mathfrak{r}}, und_{\mathfrak{r}}, pund_{\mathfrak{r}}\})$, then $C\dot{+}a$ is an ad-coloring for n;*
- *if $\nexists (T,h) \in X_{n'}^R : (C((T,h)) = in_{\mathfrak{r}} \wedge (a = h \vee a \in T))$, then $C\ddot{+}a$ is an ad-coloring for n.*

The operations $C + a$, $C\dot{+}a$, and $C\ddot{+}a$ are defined as follows for each $b \in X_n$:

$$(C+a)(b) = \begin{cases} out_{\mathfrak{a}} & \text{if } b = a \wedge \exists (T,a) \in X_{n'}^R : C((T,a)) = in_{\mathfrak{r}} \\ pout_{\mathfrak{a}} & \text{if } b = a \wedge \nexists (T,a) \in X_{n'}^R : C((T,a)) = in_{\mathfrak{r}} \\ out_{\mathfrak{r}} & \text{if } b = (T,h) \wedge a \in T \wedge C(b) = pout_{\mathfrak{r}} \\ C(b) & \text{otherwise} \end{cases}$$

$$(C\dot{+}a)(b) = \begin{cases} in_{\mathfrak{a}} & \text{if } b = a \\ C(b) & \text{otherwise} \end{cases}$$

$$(C\ddot{+}a)(b) = \begin{cases} und_{\mathfrak{a}} & \text{if } b = a \\ und_{\mathfrak{r}} & \text{if } b = (T,h) \wedge a \in T \wedge C(b) = pund_{\mathfrak{r}} \\ C(b) & \text{otherwise} \end{cases}$$

We can apply the first case $C+a$ where we set $(C+a)(a) = out_{\mathfrak{a}}$ or $(C+a)(a) = pout_{\mathfrak{a}}$ if a is not part of an attack that is either $in_{\mathfrak{r}}$, $und_{\mathfrak{r}}$, or $pund_{\mathfrak{r}}$ (as the color of a only allows

for the color $out_{\mathfrak{r}}$ for the attack). Similarly, if no attack *towards* a is $in_{\mathfrak{r}}$, $und_{\mathfrak{r}}$, or $pund_{\mathfrak{r}}$, we can apply the case $(C \dot{+} a)(a) = in_{\mathfrak{a}}$ (otherwise we would violate conflict-freeness or admissibility). Finally, for the case $(C \ddot{+} a)(a) = und_{\mathfrak{a}}$ we have to check that a is in the head or tail of an attack that is $in_{\mathfrak{r}}$. For *insert attack nodes*, we also have to consider the three cases of the added attack being $out_{\mathfrak{r}}/pout_{\mathfrak{r}}$, $in_{\mathfrak{r}}$, or $und_{\mathfrak{r}}/pund_{\mathfrak{r}}$.

Definition 5.42 (ad-coloring: Insert Attack)**.** *Let n be an insert attack node with child n' such that $X^R_{n'} = X^R_n \setminus \{r\}$, and let C be an ad-coloring for n', and $r = (T, h)$.*

- *$C + r$ is an ad-coloring for n;*
- *If $(h \notin X^A_n \vee C(h) \in \{out_{\mathfrak{a}}, pout_{\mathfrak{a}}\}) \wedge \forall t \in T \cap X^A_{n'} : C(t) = in_{\mathfrak{a}}$, then $C \dot{+} r$ is an ad-coloring for n;*
- *If $C(h) \neq in_{\mathfrak{a}} \wedge \nexists t \in T \cap X^A_{n'} : C(t) \in \{out_{\mathfrak{a}}, pout_{\mathfrak{a}}\}$, then $C \ddot{+} r$ is an ad-coloring for n.*

The operations $C + r$, $C \dot{+} r$, and $C \ddot{+} r$ are defined as follows for each $b \in X_n$:

$$(C + r)(b) = \begin{cases} out_{\mathfrak{r}} & \text{if } b = r \wedge \exists t \in T \cap X^A_{n'} : C(t) \in \{out_{\mathfrak{a}}, pout_{\mathfrak{a}}\} \\ pout_{\mathfrak{r}} & \text{if } b = r \wedge \nexists t \in T \cap X^A_{n'} : C(t) \in \{out_{\mathfrak{a}}, pout_{\mathfrak{a}}\} \\ C(b) & \text{otherwise} \end{cases}$$

$$(C \dot{+} r)(b) = \begin{cases} in_{\mathfrak{r}} & \text{if } b = r \\ out_{\mathfrak{a}} & \text{if } r = (T, h) \wedge b = h \\ C(b) & \text{otherwise} \end{cases}$$

$$(C \ddot{+} r)(b) = \begin{cases} und_{\mathfrak{r}} & \text{if } b = r \wedge \exists t \in T \cap X^A_{n'} : C(t) = und_{\mathfrak{a}} \\ pund_{\mathfrak{r}} & \text{if } b = r \wedge \nexists t \in T \cap X^A_{n'} : C(t) = und_{\mathfrak{a}} \\ C(b) & \text{otherwise} \end{cases}$$

The condition for the applicability of $C \ddot{+} r$ reflects the "priority" of the $out_{\mathfrak{r}}$ color over the $pund_{\mathfrak{r}}/und_{\mathfrak{r}}$ color: we can only apply this case if none of the arguments in the tail of the attack are colored $out_{\mathfrak{a}}/pout_{\mathfrak{a}}$. Again, we show that the insert nodes exactly characterize the admissible colorings.

Lemma 5.43. *If for the child node n' of an insert node n each ad-coloring is admissible, then each ad-coloring C of n is admissible.*

Proof. We need to show that each of the conditions (1)-(7) of Definition 5.30 is met. (1) and (2) immediately carry over from n' both for insert argument and insert attack nodes.

For (3)-(7) we discuss first the *insert argument* nodes. Note that we construct C either from an admissible coloring C' of n' via $C = C' + a$, via $C = C' \dot{+} a$, or via $C = C' \ddot{+} a$.

- Clearly, (3) and (4) hold in all three cases w.r.t. C for all $b \in X_{n'}^A$ (i.e., the "old" arguments that have not been added in this node) because they hold for n'. For the added argument a regarding (3) only the coloring $C' \dot{+} a$ is relevant (as $(C' + a)(a) \neq in_{\mathfrak{a}}$ and $(C' \ddot{+} a)(a) \neq in_{\mathfrak{a}}$). Since we can add the coloring $C' \dot{+} a$ only if $\nexists (T, a) \in X_{n'}^R$: $(C'((T, a)) \in \{in_{\mathfrak{r}}, und_{\mathfrak{r}}, pund_{\mathfrak{r}}\})$ and the only possible colors for attacks are $in_{\mathfrak{r}}$, $out_{\mathfrak{r}}$, $pout_{\mathfrak{r}}$, $und_{\mathfrak{r}}$, and $pund_{\mathfrak{r}}$, we know that in this case (3) is satisfied.

- For the added argument a regarding (4) only the coloring $C = C' + a$ is relevant. In this case, if $C(a) = out_{\mathfrak{a}}$ then we know $\exists (T, a) \in X_n^R$: $(C((T, a)) = in_{\mathfrak{r}} \wedge a \notin T)$ which means condition (4) is satisfied; if otherwise $C(a) = pout_{\mathfrak{a}}$ then (4) is trivially true for a.

- Conditions (5), (6), and (7) carry over from n' for all attacks $(T, h) \in X_n^R$ where for the added argument a it holds $a \notin T$. Let $(T, h) \in X_n^R$ be such that for the added argument a it holds $a \in T$. If $C'((T, h)) = in_{\mathfrak{r}}$ then we do not add $C' + a$ or $C' \ddot{+} a$ as a coloring to n, which means the satisfaction of condition (5) carries over from n' in these cases. For $C' \dot{+} a$ condition (5) also carries over from n' for each attack $(T, h) \in X_{\leq n}^R$ with $a \in T$ as $(C' \dot{+} a)(a) = in_{\mathfrak{a}}$.

- Moreover, in all three cases $C' + a$, $C' \dot{+} a$, and $C' \ddot{+} a$ condition (6) carries over for each attack $(T, h) \in X_{\leq n}^R$ with $a \notin T$. Moreover, for the case where we set $(C' + a)((T, h)) = out_{\mathfrak{r}}$, condition (6) is satisfied as we have $a \in T$ and $(C' + a)(a) \in \{out_{\mathfrak{a}}, pout_{\mathfrak{a}}\}$.

- Finally, for the case where we set $(C' + a)((T, h)) = und_{\mathfrak{r}}$, condition (7) is satisfied as we have $a \in T$ and $(C' + a)(a) = und_{\mathfrak{a}}$.

Hence, conditions (1)-(7) are met for insert argument nodes.

We now discuss the *insert attack* nodes w.r.t. the conditions (3)-(7).

- Regarding (3), it suffices to discuss for the added attack $r = (T, h)$ the cases $(C' \dot{+} r)$ and $(C' \ddot{+} r)$ (for the other attacks (3) carries over from n'). However, the coloring $C' \dot{+} r$ is only added to n if $C'(h) \notin \{in_{\mathfrak{a}}, und_{\mathfrak{a}}, pund_{\mathfrak{a}}\}$, which also ensures that condition (3) is met for when we set the added attack to $in_{\mathfrak{r}}$. Likewise, $C' \ddot{+} r$ is only added if $C'(h) \neq in_{\mathfrak{a}}$.

- Condition (4) carries over from n', and the case where we set $(C' \dot{+} r)(a) = out_{\mathfrak{a}}$ when we previously had $C'(a) = pout_{\mathfrak{a}}$ is only applied if we set $(C' \dot{+} r)(r) = in_{\mathfrak{r}}$, i.e., (4) is satisfied.

- For (5), only the case $C' \dot{+} r$ is relevant, the other cases follow from n'. However, we only add $C' \dot{+} r$ to n if $\forall t \in T \cap X_{n'}^A$ it holds $C'(t) = in_{\mathfrak{a}}$, and because of the properties of tree decompositions we know that $X_{>n}^A \cap T = \emptyset$, i.e., there are no tail arguments strictly below the current node. Hence, (5) is satisfied.

- As to (6), the only case that is not clear from n' is in $C' + (T,h)$, but in this case we set $C((T,h))$ to $out_{\mathfrak{r}}$ only if $\exists t \in T \cap X_n^A : C(t) \in \{pout_{\mathfrak{a}}, out_{\mathfrak{a}}\}$, which means (6) is also satisfied.
- Finally, for (7) we need to consider the case $C \ddot{+} r$, and in this case we set $C((T,h))$ to $und_{\mathfrak{r}}$ only if $\exists t \in T \cap X_n^A : C(t) = und_{\mathfrak{a}}$, which means (7) is also satisfied.

To summarize, all of (1)-(7) is satisfied, which means that each coloring in n is admissible. □

In our running example in Figure 5.11 node 8 is an insert argument node. Node 9 contains two colorings C_1 and C_2 with $C_1(r_1) = pund_{\mathfrak{r}}$, $C_2(r_1) = und_{\mathfrak{r}}$. We can in both cases apply the operation $\ddot{+}$ and obtain $C_3 = C_1 \ddot{+} b = C_2 \ddot{+} b$, as in the case of C_1 the provisional color of r_1 is "upgraded" to confirmed $und_{\mathfrak{r}}$. The fact that the resulting coloring C_3 of node 8 corresponds to two extended colorings on $X_{\geq 8}$ is reflected by the # column. The corresponding extended colorings of C_3 (i.e., $e_8(C_3) = \{C_1', C_2'\}$) defined on $X_{\geq 8}$ are as follows:

$$\begin{array}{lll} C_1'(a) = in_{\mathfrak{a}} & C_1'(b) = und_{\mathfrak{a}} & C_1'(r_1) = und_{\mathfrak{r}} \\ C_2'(a) = und_{\mathfrak{a}} & C_2'(b) = und_{\mathfrak{a}} & C_2'(r_1) = und_{\mathfrak{r}} \end{array}$$

It can be checked that these extended colorings have corresponding pendants in each of the nodes above (towards the root node), i.e., in each parent node at least one coloring is preserved that extends to an extension of C_1' and C_2'. Indeed, these extended colorings correspond to the admissible sets $\emptyset$ and $\{a\}$.

Lemma 5.44. *Let C be an admissible coloring for an insert node n. If in the child node n' of n admissible colorings and ad-colorings coincide then C is an ad-coloring for n.*

Proof. Let C' be the extended coloring of C on $X_{\geq n}$ s.t. the conditions of Definition 5.30 are satisfied.

We start with *insert argument* nodes. We show that there is an ad-coloring C^* in node n' s.t. for the added argument a we have $C = C^* + a$ if $C'(a) \in \{out_{\mathfrak{a}}, pout_{\mathfrak{a}}\}$, $C = C^* \dot{+} a$ if $C'(a) = in_{\mathfrak{a}}$, and $C = C^* \ddot{+} a$ if $C'(a) = und_{\mathfrak{a}}$.

- Let $C'(a) \in \{out_{\mathfrak{a}}, pout_{\mathfrak{a}}\}$, then C^* is an ad-coloring of n' where C^* coincides with C on each $b, (T,h) \in X_{n'}$ with the exception of attacks (T,h) with $a \in T$, $C((T,h)) = out_{\mathfrak{r}}$, and $\nexists t \in T \cap X_{n'}^A : C'(t) \in \{out_{\mathfrak{a}}, pout_{\mathfrak{a}}\}$, for these attacks we set $C^*((T,h)) = pout_{\mathfrak{r}}$. To show that C^* is an ad-coloring of n' we need to show that the conditions (1)-(7) of admissible colorings are satisfied. (1) and (2) carry over from C, as $X_{>n} = X_{>n'}$. Note for (3) that C and C^* coincide on the colors $in_{\mathfrak{a}}$ and $und_{\mathfrak{a}}$, and on the set of attacks colored either $out_{\mathfrak{r}}$ or $pout_{\mathfrak{r}}$, which means (3) carries over from C. (4) and (5) immediately carry over as well, as C and C^* coincide on

all relevant colors on the arguments/attacks $X_{n'}$. Regarding (6) note that for each attack $(T,h) \in X^R_{n'}$ with $a \in T$, $C((T,h)) = out_{\mathfrak{r}}$ property (6) holds by definition of C^*, and for the other attacks (with $C(T,h) = out_{\mathfrak{r}}$ and $a \notin T$) the property carries over from C. Finally, regarding (7) we have that for each attack $(T,h) \in X^R_{n'}$ with $a \in T$, $C((T,h)) = und_{\mathfrak{r}}$ property (7) carries over from C. Hence, in this case C^* is admissible for n' and therefore by assumption an ad-coloring for n'. Then since C^* is admissible from condition (5) it follows $\nexists (T,h) \in X^R_{n'} : (C^*((T,h)) = in_{\mathfrak{r}} \wedge a \in T)$, hence, $C = C^* + a$ is an ad-coloring for n (the correct color $out_{\mathfrak{a}}/pout_{\mathfrak{a}}$ directly follows from the fact that C is admissible and that due to the properties of tree decompositions all attacks towards a in $X^R_{\leq n}$ have to be in X^R_n).

- Now on the other side let $C'(a) = in_{\mathfrak{a}}$, then C^* is an ad-coloring of n' where C^* coincides with C on each $b, (T,h) \in X_{n'}$. Again we show that the conditions (1)-(7) are satisfied. (1) and (2) again carry over from C, as $X_{>n} = X_{>n'}$. (3)-(7) immediately carry over as well, as C and C^* coincide on all relevant colors on $X_{n'}$. Then since C^* is admissible it is an ad-coloring for n' and from condition (3) it follows $\nexists (T,a) \in X^R_{n'} : (C((T,a)) = in_{\mathfrak{r}})$, hence, $C = C^* \dot{+} a$ is an ad-coloring for n.

- Finally, let $C'(a) = und_{\mathfrak{a}}$, then C^* is an ad-coloring of n' where C^* coincides with C on each $b, (T,h) \in X_{n'}$, with the exception of attacks (T,h) with $a \in T$, $C((T,h)) = und_{\mathfrak{r}}$, and $\nexists t \in T \cap X^A_{n'} : C'(t) = und_{\mathfrak{a}}$, for these attacks we set $C^*((T,h)) = pund_{\mathfrak{r}}$. Again we show that the conditions (1)-(7) are satisfied. (1) and (2) carry over from C, as $X_{>n} = X_{>n'}$. (3)-(7) immediately carry over as well, as C and C^* coincide on all relevant colors on $X_{n'}$ (note for (7) that similarly to the first case with $C'(a) \in \{out_{\mathfrak{a}}, pout_{\mathfrak{a}}\}$ regarding property (6) we have $\{(T,h) \in X^R_{n'} \mid C^*((T,h)) = und_{\mathfrak{r}}\} \subseteq \{(T,h) \in X^R_n \mid C((T,h)) = und_{\mathfrak{r}}\}$). Then since C^* is admissible it is an ad-coloring for n' and from conditions (3) and (5) it follows $\nexists (T,h) \in X^R_{n'} : (C((T,h)) = in_{\mathfrak{r}} \wedge (a = h \vee a \in T))$, hence, $C = C^* \ddot{+} a$ is an ad-coloring for n.

We showed that in all three cases C is an ad-coloring of n.

We continue with the *insert attack* nodes. Again we show that there is an ad-coloring C^* in node n' s.t. for the added attack (T,h) we have $C = C^* + (T,h)$ if $C'((T,h)) \in \{out_{\mathfrak{r}}, pout_{\mathfrak{r}}\}$, $C = C^* \dot{+} (T,h)$ if $C'((T,h)) = in_{\mathfrak{r}}$, and $C = C^* \ddot{+} (T,h)$ if $C'((T,h)) \in \{und_{\mathfrak{r}}, pund_{\mathfrak{r}}\}$.

- Let $C'((T,h)) \in \{out_{\mathfrak{r}}, pout_{\mathfrak{r}}\}$, then C^* is an ad-coloring of n' where C^* coincides with C on each $b, (S,t) \in X_{n'}$. To show that C^* is an ad-coloring of n' we need to show that the conditions (1)-(7) of admissible colorings are satisfied. (1) and (2) carry over from C, as $X_{>n} = X_{>n'}$. (3)-(7) immediately carry over from C as C and C^* coincide on all relevant colors on $X_{n'}$. Hence, C^* is admissible for n' and therefore an ad-coloring, and $C = C^* + (T,h)$ is an ad-coloring for n (as with insert argument nodes, the correct color $out_{\mathfrak{r}}, pout_{\mathfrak{r}}$ follows from C being admissible and the properties of tree decompositions).

- Now on the other side let $C'((T,h)) = in_{\mathfrak{r}}$, then C^* is an ad-coloring of n' where C^* coincides with C on each $b, (T,h) \in X_{n'}$ with the exception the argument h (given that $h \in X^A_{n'}$) if $\nexists (S,h) \in X^R_{\geq n'} : C'((S,h)) = in_{\mathfrak{r}}$, in this case we set $C^*(h) = pout_{\mathfrak{a}}$. Again we show that the conditions (1)-(7) are satisfied. (1) and (2) carry over from C, as $X_{>n} = X_{>n'}$. (3), (5), (6), and (7) immediately carry from C, as C and C^* coincide on all relevant colors on $X_{n'}$. Regarding (4), note that this clearly carries over from C for all arguments $a \in X^A_{n'} \setminus \{h\}$, and for the argument h it holds by definition of C^*. Then since C^* is admissible from condition (3) it follows $(h \notin X^A_{n'} \vee C(h) \neq in_{\mathfrak{a}})$ and from condition (5) it follows $\forall t \in T \cap X^A_{n'} : C(t) = in_{\mathfrak{a}}$, hence, $C = C^* \dot{+} (T,h)$ is an ad-coloring for n.

- Finally, let $C'((T,h)) \in \{und_{\mathfrak{r}}, pund_{\mathfrak{r}}\}$, then C^* is an ad-coloring of n' where C^* coincides with C on each $b, (S,t) \in X_{n'}$. To show that C^* is an ad-coloring of n' we need to show that the conditions (1)-(7) of admissible colorings are satisfied. (1) and (2) carry over from C, as $X_{>n} = X_{>n'}$. (3)-(7) immediately carry over from C as C and C^* coincide on all relevant colors on $X_{n'}$. Hence, C^* is admissible for n' and therefore an ad-coloring, and $C = C^* + (T,h)$ is an ad-coloring for n (the correct color $und_{\mathfrak{r}}, pund_{\mathfrak{r}}$ follows from C being admissible and the properties of tree decompositions).

We showed that in all three cases C is an ad-coloring of n. □

5.4.4 Join Nodes

We now consider the join nodes. In these nodes we combine matching colorings. A coloring matches another if all arguments and attacks have the same color *ignoring provisional colors*. If two colorings are combined where an argument/attack is provisional in one, but non-provisional in the other, the non-provisional color is kept. This is due to the fact that the "proof" for the non-provisional color already appeared in the branch where the coloring originates. Formally:

Definition 5.45 (ad-coloring: Join)**.** *Let n be a join node with children n', n'', let C be an ad-coloring for n', and let D be an ad-coloring for n''. If C and D are compatible, i.e.,*

1. $\{a \in X^A_{n'} \mid C(a) = in_{\mathfrak{a}}\} = \{a \in X^A_{n''} \mid D(a) = in_{\mathfrak{a}}\}$,
2. $\{r \in X^R_{n'} \mid C(r) = in_{\mathfrak{r}}\} = \{r \in X^R_{n''} \mid D(r) = in_{\mathfrak{r}}\}$,
3. $\{a \in X^A_{n'} \mid C(a) = und_{\mathfrak{a}}\} = \{a \in X^A_{n''} \mid D(a) = und_{\mathfrak{a}}\}$, *and*
4. $\{r \in X^R_{n'} \mid C(r) \in \{und_{\mathfrak{r}}, pund_{\mathfrak{r}}\}\} = \{r \in X^R_{n''} \mid D(r) \in \{und_{\mathfrak{r}}, pund_{\mathfrak{r}}\}\}$

then $C \bowtie D$ is an ad-coloring for n, where

$$(C \bowtie D)(b) = \begin{cases} in_{\mathfrak{a}} & \text{if } C(b) = D(b) = in_{\mathfrak{a}} \\ und_{\mathfrak{a}} & \text{if } C(b) = D(b) = und_{\mathfrak{a}} \\ out_{\mathfrak{a}} & \text{if } C(b) = out_{\mathfrak{a}} \vee D(b) = out_{\mathfrak{a}} \\ pout_{\mathfrak{a}} & \text{if } C(b) = D(b) = pout_{\mathfrak{a}} \\ in_{\mathfrak{r}} & \text{if } C(b) = D(b) = in_{\mathfrak{r}} \\ und_{\mathfrak{r}} & \text{if } C(b) = und_{\mathfrak{r}} \vee D(b) = und_{\mathfrak{r}} \\ pund_{\mathfrak{r}} & \text{if } C(b) = D(b) = pund_{\mathfrak{r}} \\ out_{\mathfrak{r}} & \text{if } C(b) = out_{\mathfrak{r}} \vee D(b) = out_{\mathfrak{r}} \\ pout_{\mathfrak{r}} & \text{if } C(b) = D(b) = pout_{\mathfrak{r}} \end{cases}$$

Lemma 5.46. *If for the child nodes n', n'' of a join node n each ad-coloring is admissible, then each ad-coloring C of n is admissible.*

Proof. We need to show that each of the conditions (1)-(7) of Definition 5.30 is met. (1) and (2) immediately carry over from n' and n'' since $X^A_n = X^A_{n'} = X^A_{n''}$ and $X^R_n = X^R_{n'} = X^R_{n''}$. (3) holds because it has to hold both for all (S, a) in $X^R_{n'}$ and in $X^R_{n''}$ by assumption. (4) is satisfied, as we set an argument to $out_{\mathfrak{a}}$ only if it is colored $out_{\mathfrak{a}}$ in at least one of the two colorings of n' and n''. Similarly, (5) and (6) carry over from n' and n''. (7) is satisfied, similarly to (4), because we set an attack to $und_{\mathfrak{r}}$ only if it is colored $und_{\mathfrak{r}}$ in at least one of the two colorings of n' and n''. □

Node 4 in Figure 5.11 is our illustrated join node. If we look at the coloring C_1 of node 5 with $C_1(b) = pout_{\mathfrak{a}}$, $C_1(c) = und_{\mathfrak{a}}$, $C_1(r_4) = pout_{\mathfrak{r}}$ we see two matching colorings D_1 and D_2 in node 11: $D_1(b) = out_{\mathfrak{a}}$, $D_1(c) = und_{\mathfrak{a}}$, $D_1(r_4) = pout_{\mathfrak{r}}$ and $D_2(b) = pout_{\mathfrak{a}}$, $D_2(c) = und_{\mathfrak{a}}$, $D_2(r_4) = pout_{\mathfrak{r}}$. We get the resulting ad-colorings for node 4: $(C_1 \bowtie D_1) = D_1$ and $(C_1 \bowtie D_2) = D_2$.

Lemma 5.47. *Let C be an admissible coloring for a join node n. If in the child nodes n', n'' of n admissible colorings and ad-colorings coincide then C is an ad-coloring for n.*

Proof. Let C' be the extended coloring of C on $X_{\geq n}$ s.t. the conditions of Definition 5.30 are satisfied.

We will show that there are ad-colorings C^* for n' and D^* for n'' such that $C = C^* \bowtie D^*$. We define C^* such that it coincides with C on each $a, (T, h) \in X_n$ with the following three exceptions, each accounting for one of the provisional colors $pout_{\mathfrak{a}}$, $pout_{\mathfrak{r}}$, and $pund_{\mathfrak{r}}$:

1. arguments b s.t. $C(b) = out_{\mathfrak{a}}$ and $\nexists (S, b) \in X^R_{\geq n'} : C'((S, b)) = in_{\mathfrak{r}}$, for these arguments we set $C^*(b) = pout_{\mathfrak{a}}$,
2. attacks (S, b) s.t. $C((S, b)) = out_{\mathfrak{r}}$ and $\nexists s \in S \cap X^A_{\geq n'} : C'(s) \in \{out_{\mathfrak{a}}, pout_{\mathfrak{a}}\}$, for these attacks we set $C^*((S, b)) = pout_{\mathfrak{r}}$, and

3. attacks (S, b) s.t. $C((S, b)) = und_{\mathfrak{r}}$ and $\nexists s \in S \cap X^A_{\geq n'} : C'(s) = und_{\mathfrak{a}}$, for these attacks we set $C^*((S, b)) = pund_{\mathfrak{r}}$.

We show that properties (1)-(7) of admissible colorings are satisfied for C^*. (1) and (2) carry over from C, as $X_{>n} = X_{>n'} = X_{>n''}$. (3) is satisfied as for C and C^* it holds

$$\{(T, h) \in X^R_n \mid C((T, h)) \in \{out_{\mathfrak{r}}, pout_{\mathfrak{r}}\})\} = \{(T, h) \in X^R_{n'} \mid C^*((T, h)) \in \{out_{\mathfrak{r}}, pout_{\mathfrak{r}}\})\}.$$

(4), (6) and (7) are satisfied by definition of C^* for the "updated" colors, for the others the property carries over from C. (5) immediately carries over from C. Hence, C^* is admissible (and D^* as well due to symmetry). Finally, note that by construction of C^* and D^* it holds

1. $\{a \in X^A_{n'} \mid C^*(a) = in_{\mathfrak{a}}\} = \{a \in X^A_{n''} \mid D^*(a) = in_{\mathfrak{a}}\}$,
2. $\{r \in X^R_{n'} \mid C^*(r) = in_{\mathfrak{r}}\} = \{r \in X^R_{n''} \mid D^*(r) = in_{\mathfrak{r}}\}$,
3. $\{a \in X^A_{n'} \mid C^*(a) = und_{\mathfrak{a}}\} = \{a \in X^A_{n''} \mid D^*(a) = und_{\mathfrak{a}}\}$, and
4. $\{r \in X^R_{n'} \mid C^*(r) \in \{und_{\mathfrak{r}}, pund_{\mathfrak{r}}\}\} = \{r \in X^R_{n''} \mid D^*(r) \in \{und_{\mathfrak{r}}, pund_{\mathfrak{r}}\}\}$

Hence, $C = C^* \bowtie D^*$ is an ad-coloring for n. □

5.4.5 Final Steps for Admissible Semantics

Again we can combine the results we obtained for each of the node types to prove the soundness and completeness of our algorithm as a whole. We show that our ad-colorings coincide with the admissible colorings in each node.

Proposition 5.48. *Let $\mathcal{T} = (V_{\mathcal{T}}, E_{\mathcal{T}})$ be a nice tree-decomposition of a SETAF SF. Then in each node $n \in V_{\mathcal{T}}$ ad-colorings and admissible colorings coincide.*

Proof. ($\subseteq$) Follows by structural induction over the tree-decomposition structure with the leaves as a base (Lemma 5.35) and the forget, insert, and join nodes as steps (Lemma 5.39, Lemma 5.43, Lemma 5.46).

($\supseteq$) First note that by ($\subseteq$) all ad-colorings in each node are admissible. Then the statement follows by structural induction over the tree-decomposition structure with the leaves as a base (Lemma 5.36) and the forget, insert, and join nodes as steps (Lemma 5.40, Lemma 5.44, Lemma 5.47). □

Now we are ready for the main result regarding admissible semantics. We obtain a runtime of $O(9^k \cdot k \cdot (|A| + |R|))$ which is mainly due to the high complexity of join nodes. The key idea is to sort the ad-colorings in the child nodes such that the combination of colorings can be done more efficiently.

Theorem 5.49. *The problems $Cred_{adm} = Cred_{com} = Cred_{pref}$ as well as counting the number of admissible sets can be done in time $O(9^k \cdot k \cdot (|A| + |R|))$. Moreover, we can enumerate all admissible sets with linear delay.*

Proof. The correctness of the algorithm (i.e., we account *only* for admissible colorings in the root node) and the completeness of the algorithm (i.e., we account for *all* the admissible colorings in the root node) is shown in Proposition 5.48. Then, from the fact that in the root node n we have $X_n^A = X_n^R = \emptyset$, we trivially get $X_{>n}^A = A$ and $X_{>n}^R = R$. Hence, by property (1) and (2) of admissible colorings we know that the colorings in the root node can be extended to SF such that no provisional colors are used. By Proposition 5.31 we then get that the colorings in the root node exactly characterize $adm(SF)$.

We can assume the number of nodes to be bounded by $O(|A| + |R|)$ and that we can find and access rows in linear time w.r.t. k. For each node, the number of valid colorings (i.e., rows in our tables of colorings) is bounded by 5^k. In leaf nodes, we can check the conditions for leaves in time $O(k^2)$ for each of the $O(5^k)$ possible colorings, resulting in $O(5^k \cdot k^2)$. In forget nodes, we can check whether the condition is satisfied and compute eventually resulting colorings in time $O(k)$ for each of the $O(5^k)$ colorings of the child node, resulting in $O(5^k \cdot k)$. In insert nodes, we can check whether the condition is satisfied and compute eventually resulting colorings in time $O(k^2)$ for each of the $O(5^k)$ colorings of the child node, resulting in $O(5^k \cdot k^2)$. Finally, for join nodes we have to consider $5^k \cdot 5^k = 25^k$ pairs. However, we only need to consider 9^k pairs if we assume the data structure to be properly sorted, e.g. lexicographically by treating the colors $in_{\mathfrak{a}}/in_{\mathfrak{r}}$ as 0, $und_{\mathfrak{a}}/und_{\mathfrak{r}}/pund_{\mathfrak{r}}$ as 1, and $pout_{\mathfrak{a}}/out_{\mathfrak{a}}/pout_{\mathfrak{r}}/out_{\mathfrak{r}}$ as 2. As each table has $O(5^k)$ rows, sorting is in $O(5^k \cdot k)$. Let C be a coloring such that $m \leq k$ arguments/attacks are colored as $in_{\mathfrak{a}}/in_{\mathfrak{r}}$. There exist at most 2^{k-m} "distinct" colorings C' (distinct within their group when we group colorings as follows: $\{in_{\mathfrak{a}}, in_{\mathfrak{r}}\}, \{und_{\mathfrak{a}}, und_{\mathfrak{r}}, pund_{\mathfrak{r}}\}, \{out_{\mathfrak{a}}, pout_{\mathfrak{a}}, out_{\mathfrak{r}}, pout_{\mathfrak{r}}\}$) with $\forall x : (C(x) \in \{in_{\mathfrak{a}}, in_{\mathfrak{r}}\} \Leftrightarrow C'(x) \in \{in_{\mathfrak{a}}, in_{\mathfrak{r}}\})$. There are $\binom{k}{m}$ possibilities resulting from the choice of m. Finally, we have to distinguish 2^{k-m} combinations of sets $\{x \mid C(x) \in \{und_{\mathfrak{a}}, und_{\mathfrak{r}}, pund_{\mathfrak{r}}\}\}$ and $\{x \mid C(x) \in \{out_{\mathfrak{a}}, pout_{\mathfrak{a}}, out_{\mathfrak{r}}, pout_{\mathfrak{r}}\}\}$, resulting in $\sum_{m=0}^{k} \binom{k}{m} \cdot 2^{k-m} \cdot 2^{k-m} \cdot 2^{k-m} = 9^k$ join pairs. We can then compute $C \bowtie D$ in $O(k)$, resulting in $O(9^k \cdot k)$ for join nodes, dominating the runtime of the other node types. The resulting runtime for the algorithm is $O(9^k \cdot k \cdot (|A| + |R|))$.

We can decide credulous acceptance by flagging those colorings that contain the argument in question. In each node we update this flag accordingly, and the flag in the root node indicates whether the argument is credulously accepted. We can keep the count of admissible sets corresponding to each coloring (cf. [DPW12]). Finally, to enumerate the admissible sets once the dynamic programming algorithm is done we can traverse the tree top-down and output the admissible sets with linear delay (cf. [DPW12]). □

Since in the special case of AFs we have $|R| = O(|A|^2)$, this algorithm has runtime $O(9^k \cdot k \cdot |A|^2)$ for AFs (compared to $O(10^k \cdot k \cdot |A|)$ in the "standard" algorithm [DPW12]). However, since in this case it will always be the case that the color of an attack (a, b) coincides with the color of a, it is not necessary to compute these two separately and we can achieve a speedup obtaining a runtime of $O(9^k \cdot k \cdot |A|)$.

5.5 Characterizing Complete Extensions

For complete extensions we can extend our ideas from admissible sets. The main difference is that for a complete extension, there has to be a "proof" for an undecided argument, i.e., an argument can only be undecided if there is an undecided attack towards it. The effect is that whenever all attacks towards an argument are colored $out_{\mathfrak{r}}$ the argument has to be colored $in_{\mathfrak{a}}$ (in contrast to admissible sets where both $in_{\mathfrak{a}}$ and $und_{\mathfrak{a}}$ were options in this case). This corresponds to the requirement of complete extensions to contain each argument that is defended. Hence, we extend the colorings we used to characterize admissible sets by the color $pund_{\mathfrak{a}}$ to reflect "provisionally undecided" arguments, i.e., arguments towards which in the sub-tree originating in the current node there is no undecided attack. Hence, the colors for the complete algorithm are as follows:

$$C_{com} = \{in_{\mathfrak{a}}, out_{\mathfrak{a}}, pout_{\mathfrak{a}}, und_{\mathfrak{a}}, pund_{\mathfrak{a}}, in_{\mathfrak{r}}, out_{\mathfrak{r}}, pout_{\mathfrak{r}}, und_{\mathfrak{r}}, pund_{\mathfrak{r}}\}$$

As with the previous semantics we characterize our target that we want to capture with our algorithm. We will define co-colorings for each node in a tree decomposition of SF and will ultimately show that these coincide with *complete coloring*.

Definition 5.50. *A coloring $C : X_n \to \mathcal{C}_{com}$ for a node $n \in V_{\mathcal{T}}$ is* complete *if we can extend C to a coloring $C' : X_{\geq n} \to \mathcal{C}_{com}$ such that the following conditions are met for each $a \in X^A_{\geq n}$ and each $(T, h) \in X^R_{\geq n}$:*

1. *if $a \in X^A_{>n}$ then $C'(a) \in \{in_{\mathfrak{a}}, out_{\mathfrak{a}}, und_{\mathfrak{a}}\}$,*
2. *if $r \in X^R_{>n}$ then $C'((T, h)) \in \{in_{\mathfrak{r}}, out_{\mathfrak{r}}, und_{\mathfrak{r}}\}$,*
3. *if $C'(a) = in_{\mathfrak{a}}$ then $\forall (S, a) \in X^R_{\geq n} : C'((S, a)) \in \{out_{\mathfrak{r}}, pout_{\mathfrak{r}}\}$,*
4. *$C'(a) = out_{\mathfrak{a}}$ if and only if $\exists (S, a) \in X^R_{\geq n} : C'((S, a)) = in_{\mathfrak{r}}$,*
5. *if $C'((T, h)) = in_{\mathfrak{r}}$ then $\forall t \in T \cap X^A_{\geq n} : C'(t) = in_{\mathfrak{a}}$,*
6. *$C'((T, h)) = out_{\mathfrak{r}}$ if and only if $\exists t \in T \cap X^A_{\geq n} : C'(t) \in \{out_{\mathfrak{a}}, pout_{\mathfrak{a}}\}$,*
7. *$C'((T, h)) = und_{\mathfrak{r}}$ if and only if $\exists t \in T \cap X^A_{\geq n} : C'(t) \in \{und_{\mathfrak{a}}, pund_{\mathfrak{a}}\}$ and $\nexists t \in T \cap X^A_{\geq n} : C'(t) \in \{out_{\mathfrak{a}}, pout_{\mathfrak{a}}\}$, and*
8. *$C'(a) = und_{\mathfrak{a}}$ if and only if $\exists (S, a) \in X^R_{\geq n} : C'((S, a)) \in \{und_{\mathfrak{r}}, pund_{\mathfrak{r}}\}$ and $\nexists (S, a) \in X^R_{\geq n} : C'((S, a)) = in_{\mathfrak{r}}$.*

For a coloring C in node n we define by $e_n(C)$ the set of such extended colorings C' s.t. (1)-(8) are met. These are the characterized colorings.

Note that conditions (1)-(6) exactly coincide with Definition 5.30 for admissible colorings, and (7) mostly coincides as well, with the minor adjustment to account for the provisionally undecided arguments (recall that we did not consider the color $pund_{\mathfrak{a}}$ for admissible sets). The added condition (8) parallels condition (4) and ensures that an undecided argument is non-provisional if and only if there is a "witnessing" attack towards it, in a similar spirit as with the non-provisional $out_{\mathfrak{a}}$ color in condition (4). The second part of condition (8) reflects the "priority" of the color $out_{\mathfrak{a}}$ over $und_{\mathfrak{a}}$, i.e., whenever we can only use the color $und_{\mathfrak{a}}$ if we do not instead have to use the color $out_{\mathfrak{a}}$. This addition, together with the introduction of the new color $pund_{\mathfrak{a}}$, leads to a situation where we have to have a "proof" for an argument being undecided. In particular, in admissible sets it was possible for an argument a where each attack (T, a) towards it was colored $out_{\mathfrak{r}}$ (or $pout_{\mathfrak{r}}$) to still color the argument $und_{\mathfrak{a}}$. Now with the complete extensions on the other hand we can set a the color $pund_{\mathfrak{a}}$, but in order to have a non-provisional color $und_{\mathfrak{a}}$ we need an attack that is colored $und_{\mathfrak{r}}/pund_{\mathfrak{r}}$ towards a (note also that there cannot be an attack colored $in_{\mathfrak{r}}$ towards a, as this would imply that a is colored $out_{\mathfrak{a}}$). This has the effect that in the end whenever all attacks towards argument a are colored $out_{\mathfrak{a}}$, the only possible (non-provisional) color for a is $in_{\mathfrak{a}}$, reflecting the "maximality"-condition for complete extensions.

Proposition 5.51. *Let $\mathcal{T} = (V_{\mathcal{T}}, E_{\mathcal{T}})$ be a nice tree-decomposition of a SETAF $SF = (A, R)$ and let $r \in V_{\mathcal{T}}$ be the root node of $\mathcal{T}$. Moreover, let $\mathcal{C}_r$ be the set of complete colorings for r. Then*

$$com(SF) = \{\{a \mid C'(a) = in_{\mathfrak{a}}\} \mid C \in \mathcal{C}_r, C' \in e_r(C)\}.$$

Proof. Since the root node is empty, we have that $X^A_{>n} = A$ and $X^R_{>n} = R$, which by (1) and (2) means that we have no provisional colors in each extended coloring $C' \in e_r(C)$. Conditions (3) and (6) give us the following property (a), (4) and (5) give us the following property (b), and (8) gives us the following property (c):

(a) $C'(a) = in_{\mathfrak{a}}$ if and only if $\forall (T, a) \in R\ \exists t \in T : C'(t) = out_{\mathfrak{a}}$,

(b) $C'(a) = out_{\mathfrak{a}}$ if and only if $\exists (T, a) \in R\ \forall t \in T : C'(t) = in_{\mathfrak{a}}$, and

(c) $C'(a) = und_{\mathfrak{a}}$ if and only if $\nexists (T, a) \in R\ \forall t \in T : C'(t) = in_{\mathfrak{a}}$ and $\not\forall (T, a) \in R\ \exists t \in T : C'(t) = out_{\mathfrak{a}}$.

(a) exactly characterizes conflict-freeness, i.e., for each attack (T, a) towards an argument a that is in the extensions (i.e., in the set $\{a \mid C'(a) = in_{\mathfrak{a}}\}$) there is at least one argument in T that is *not* in the extension. (a) and (b) together characterize defense, i.e., for each argument a that has the color $in_{\mathfrak{a}}$ for each attack (T, a) towards a, there is

a counter-attack (S, t) for some $t \in T$ such that $S \subseteq \{b \mid C'(b) = in_{\mathfrak{a}}\}$. (c) characterizes completeness, i.e., if an argument is undecided, then it is not defended against each incoming attack (as in that case it would have to be colored $in_{\mathfrak{a}}$). (a), (b), and (c) exactly characterize the complete extensions. □

5.5.1 Leaf Nodes

Intuitively, as with admissible semantics, in leaf nodes we guess for each argument and each attack one of three possibilities *in*, *out*, or *und*, and keep every "consistent" coloring. The distinction between $out_{\mathfrak{a}}$ and $pout_{\mathfrak{a}}$ (or between $out_{\mathfrak{r}}$ and $pout_{\mathfrak{r}}$) is analogous to admissible semantics. However, in contrast to admissible semantics we now make use of the color $pund_{\mathfrak{a}}$. We color an argument $pund_{\mathfrak{a}}$ instead of $und_{\mathfrak{a}}$ if there is not yet "proof" for the argument being undecided: for complete semantics, not all attacks towards an undecided argument can be defended against (i.e., not all attacks towards an argument can be colored $out_{\mathfrak{r}}/pout_{\mathfrak{r}}$), as then the argument would have to be *in* the extension. For the case where such an attack that is not colored $out_{\mathfrak{r}}/pout_{\mathfrak{r}}$ is not yet encountered in the current leaf node, we assign the color $pund_{\mathfrak{a}}$. Effectively, this means for an argument to be colored $und_{\mathfrak{a}}$ there needs to be an attack towards it colored $und_{\mathfrak{r}}/pund_{\mathfrak{r}}$. Note that again for both arguments and attacks the *out* colors have "priority" over the *und* colors.

Definition 5.52 (co-colorings: Leaf)**.** *A* co-coloring *for a* leaf node $n \in V_{\mathcal{T}}$ *is each coloring that satisfies the following conditions. For each argument* $a \in X_n^A$*:*

$$C(a) = in_{\mathfrak{a}} \Rightarrow \forall (T, a) \in X_n^R\colon C((T, a)) \in \{pout_{\mathfrak{r}}, out_{\mathfrak{r}}\}$$
$$C(a) = out_{\mathfrak{a}} \Leftrightarrow \exists (T, a) \in X_n^R\colon C((T, a)) = in_{\mathfrak{r}}$$
$$C(a) = und_{\mathfrak{a}} \Leftrightarrow \exists (T, a) \in X_n^R\colon C((T, a)) \in \{pund_{\mathfrak{r}}, und_{\mathfrak{r}}\} \wedge \nexists (T, a) \in X_n^R\colon C((T, a)) = in_{\mathfrak{r}}$$

For every attack $r = (T, h) \in X_n^R$*:*

$$C(r) = in_{\mathfrak{r}} \Rightarrow C(h) \neq in_{\mathfrak{a}} \wedge \forall t \in T \cap X_n^A\colon C(t) = in_{\mathfrak{a}}$$
$$C(r) = out_{\mathfrak{r}} \Leftrightarrow \exists t \in T \cap X_n^A\colon C(t) \in \{pout_{\mathfrak{a}}, out_{\mathfrak{a}}\}$$
$$C(r) = und_{\mathfrak{r}} \Leftrightarrow \exists t \in T \cap X_n^A\colon C(t) \in \{pund_{\mathfrak{a}}, und_{\mathfrak{a}}\} \wedge \nexists t \in T \cap X_n^A\colon C(t) \in \{pout_{\mathfrak{a}}, out_{\mathfrak{a}}\}$$

For co-colorings we have to consider both for arguments and attacks the confirmed undecided color. In both cases the out color has priority, which is why we have to account for the case where the out color should be applied. We now establish that all co-colorings in the leaf nodes coincide with the complete colorings.

Lemma 5.53. *Each co-coloring* C *for a leaf node* $n \in V_{\mathcal{T}}$ *is complete.*

Proof. We need to show that each of the conditions (1)-(8) of Definition 5.50 is met. (1) and (2) are trivially true, as for leaves it holds $X_{>n}^A = X_{>n}^R = \emptyset$. (3)-(8) follow directly from the conditions in Definition 5.52. □

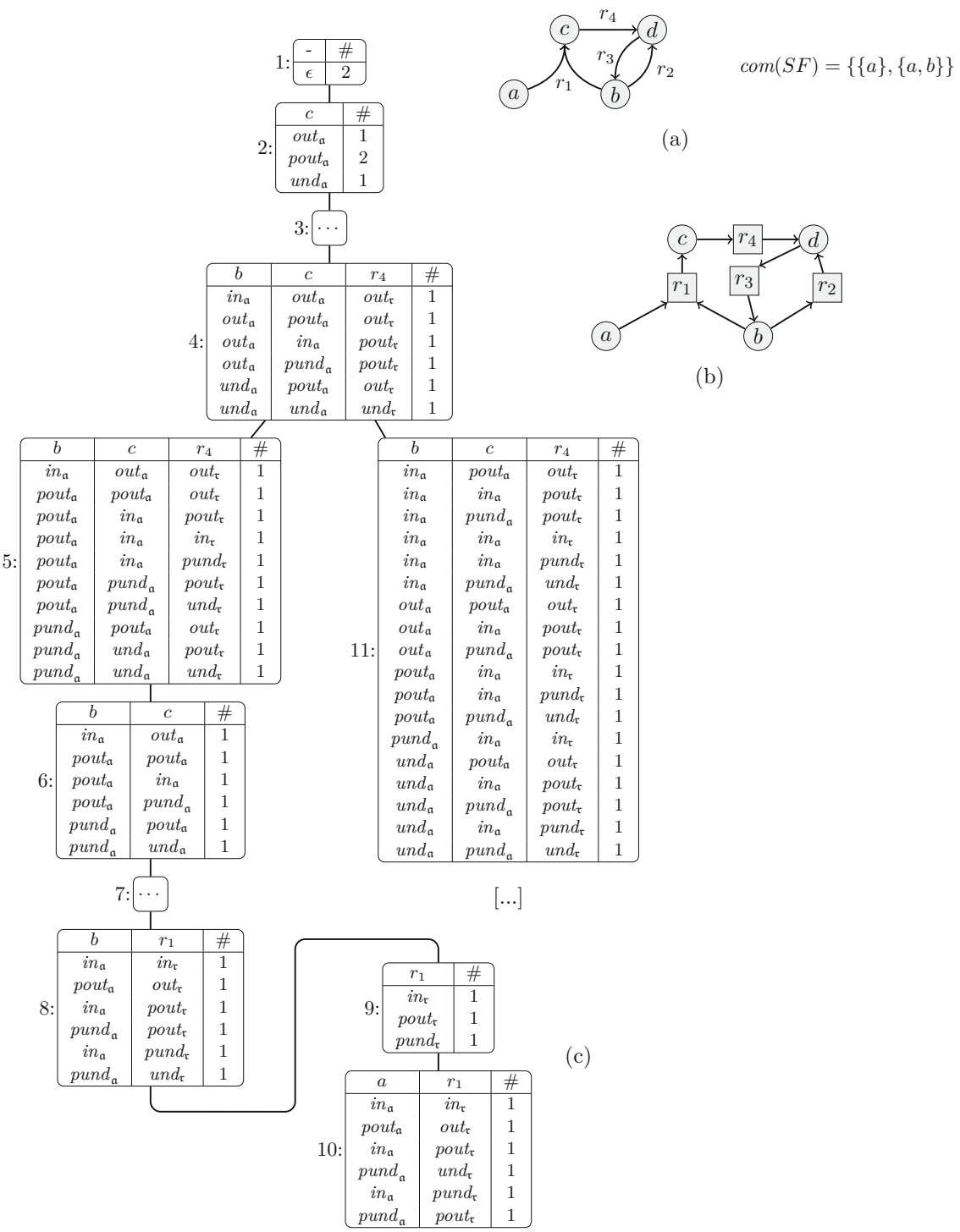


Figure 5.12: Selected co-colorings for our running example. (a) Our running example SETAF SF with its complete extensions, (b) its incidence-graph $\texttt{Inc}(SF)$, and (c) the co-colorings for SF w.r.t. the tree decomposition given in Figure 5.3. We omit parts of the tree (co-colorings for nodes 3 and 7 as well as the sub-tree rooted in node 11).

In our example from Figure 5.12 we have the leaf node 10. We see that we have similar co-colorings as we had ad-colorings in Figure 5.11, but instead of $und_{\mathfrak{a}}$ colors for a we have $pund_{\mathfrak{a}}$ colors. Since in this particular example we have no arguments that are in the head of an attack in the present node, this behavior is of course to be expected.

Lemma 5.54. *Each complete coloring C for a leaf node $n \in V_{\mathcal{T}}$ is a co-coloring.*

Proof. It follows directly from the definition of complete colorings that C adheres to the conditions in Definition 5.52. □

5.5.2 Forget Nodes

We examine forget argument nodes and forget attack nodes separately. Let n be a *forget argument node* with child n' such that $X_n^A = X_{n'}^A \setminus \{a\}$. We have to discard all colorings C where $C(a) \in \{pout_{\mathfrak{a}}, pund_{\mathfrak{a}}\}$, as in these colorings a is supposed to be attacked by the characterized extension or undecided. As we forget a in the current node, this cannot happen.

Definition 5.55 (co-coloring: Forget Argument)**.** *Let n be a forget argument node with child n' such that $X_n^A = X_{n'}^A \setminus \{a\}$, and let C be a co-coloring for n'. If $C(a) \notin \{pout_{\mathfrak{a}}, pund_{\mathfrak{a}}\}$, then $C - a$ is a co-coloring for n, where $C - x$ for each $b \in X_n$ is defined as follows:*

$$(C - x)(b) = C(b)$$

We handle *forget attack nodes* in a similar way—in fact, the steps for ad-colorings and co-colorings are exactly the same.

Definition 5.56 (co-coloring: Forget Attack)**.** *Let n be a forget attack node with child n' such that $X_n^R = X_{n'}^R \setminus \{r\}$, and let C be a co-coloring for n'. If $C(r) \notin \{pout_{\mathfrak{r}}, pund_{\mathfrak{r}}\}$, then $C - r$ is a co-coloring for n, where $C - x$ for each $b \in X_n$ is defined as follows:*

$$(C - x)(b) = C(b)$$

We next establish that in forget nodes complete colorings and co-colorings coincide.

Lemma 5.57. *If for the child node n' of a forget node n each co-coloring is complete, then each coloring C of n is complete.*

Proof. We need to show that each of the conditions (1)-(8) of Definition 5.50 is met. (1) and (2) are true because they hold for n', and if the forgotten argument/attack is provisional, then it is discarded. The satisfaction of (3)-(8) immediately carries over from n'. □

We see in Figure 5.12 in node 9 that all colorings of node 10 where a is undecided (or out) are discarded. This corresponds to the fact that a is unattacked and has to be in every complete extension. We are left with three co-colorings in node 9 that each extend to an extended coloring on $X_{\geq 9}$ where a is set to $in_{\mathfrak{a}}$.

Lemma 5.58. *Let C be a complete coloring of a forget node n. If in the child node n' of n complete colorings and co-colorings coincide then C is a co-coloring for n.*

Proof. Let C' be the extended coloring of C on $X_{\geq n}$ s.t. the conditions of Definition 5.50 are satisfied.

We start with *forget argument* nodes. We show that C^* is a co-coloring for n' where C^* coincides with C on each $a, (T, h) \in X_n$ and for the forgotten argument $b \in X_{n'} \setminus X_n$ it holds $C^*(b) \notin \{pout_{\mathfrak{a}}, pund_{\mathfrak{a}}\}$.

- If $C'(b) = in_{\mathfrak{a}}$ then clearly C^* is complete for n' with $C^*(b) = in_{\mathfrak{a}}$: in this case each condition (1)-(8) carries over from C. Then by assumption this means C^* is a co-coloring for n'.
- If $C'(b) = out_{\mathfrak{a}}$ then C^* is complete for n' with $C^*(b) = out_{\mathfrak{a}}$: this is because by condition (4) we have for node n that $\exists (S, a) \in X^R_{\geq n} : C'((S, a)) \in \{out_{\mathfrak{r}}, pout_{\mathfrak{r}}\}$, and since $X^R_{\geq n} = X^R_{\geq n'}$ this carries over to node n'. Again the other conditions clearly carry over, and by assumption this means C^* is a co-coloring for n'.
- Finally, if $C'(b) = und_{\mathfrak{a}}$ then C^* is complete for n' with $C^*(b) = und_{\mathfrak{a}}$: this is because by condition (8) we have for node n that $\exists (S, a) \in X^R_{\geq n} : C'((S, a)) \in \{und_{\mathfrak{r}}, pund_{\mathfrak{r}}\}$, and since $X^R_{\geq n} = X^R_{\geq n'}$ this carries over to node n'. Again the other conditions clearly carry over, and by assumption this means C^* is a co-coloring for n'.

In all three cases we see that there is a corresponding co-coloring C^* in the child node n' that gives us C as a co-coloring in n, and since there is no other possibility, this means that $C = C^* - b$ is a co-coloring of n.

We continue with *forget attack* nodes. We show that C^* is a co-coloring for n' where C^* coincides with C on each $a, (T, h) \in X_n$ and for the forgotten attack $(S, t) \in X_{n'} \setminus X_n$ it holds $C^*((S, t)) \neq pout_{\mathfrak{r}}$ and $C^*((S, t)) \neq pund_{\mathfrak{r}}$.

- If $C'((S, t)) = in_{\mathfrak{r}}$ then clearly C^* is complete for n' with $C^*(b) = in_{\mathfrak{r}}$: in this case each condition (1)-(8) carries over from C. Then by assumption this means C^* is a co-coloring for n'.
- If $C'((S, t)) = out_{\mathfrak{r}}$ then C^* is complete for n' with $C^*((S, t)) = out_{\mathfrak{r}}$: this is because by condition (6) we have for node n that $\exists s \in S \cap X^A_{\geq n} : C'(s) \in \{out_{\mathfrak{a}}, pout_{\mathfrak{a}}\}$, and since $X^A_{\geq n} = X^A_{\geq n'}$ this carries over to node n'. Again the other conditions clearly carry over, and by assumption this means C^* is a co-coloring for n'.

- Finally, if $C'((S,t)) = und_{\mathfrak{r}}$ then clearly C^* is complete for n' with $C^*(b) = und_{\mathfrak{r}}$: this is because by condition (7) we have for node n that $\exists s \in S \cap X^A_{\geq n} : C'(s) \in \{und_{\mathfrak{a}}, pund_{\mathfrak{a}}\}$, and since $X^A_{\geq n} = X^A_{\geq n'}$ this carries over to node n'. Again the other conditions clearly carry over, and by assumption this means C^* is a co-coloring for n'.

In all three cases we see that there is a corresponding co-coloring C^* in the child node n' that gives us C as a co-coloring in n, and since there is no other possibility, this means that $C = C^* - b$ is a co-coloring of n. □

5.5.3 Insert Nodes

As with forget nodes, we handle the case of inserting an argument and inserting an attack separately. For insert argument nodes, we will use the operations $C + a$, $C\dot{+}a$, $C\ddot{+}a$, similar to the case of admissible sets. The key difference to characterizing admissible sets is that for $C\ddot{+}a$ where we account for the case that the added argument a is undecided, we have to distinguish the case where a is "confirmed" undecided (i.e., if there is already an undecided attack towards a present in the current node), and the case where a is provisionally undecided. This operation is now mostly symmetrical to $C + a$ where we handle the case where the added argument a is out.

Definition 5.59 (co-coloring: Insert Argument)**.** *Let n be an insert argument node with child n' such that $X^A_{n'} = X^A_n \setminus \{a\}$, and let C be a co-coloring for n'.*

- *If $\nexists (T,h) \in X^R_n : (C((T,h)) \in \{in_{\mathfrak{r}}, und_{\mathfrak{r}}, pund_{\mathfrak{r}}\} \wedge a \in T)$, then $C+a$ is a co-coloring for n;*
- *If $\nexists (T,a) \in X^R_n : (C((T,a)) \in \{in_{\mathfrak{r}}, und_{\mathfrak{r}}, pund_{\mathfrak{r}}\})$, then $C\dot{+}a$ is a co-coloring for n;*
- *If $\nexists (T,h) \in X^R_n : (C((T,h)) = in_{\mathfrak{r}} \wedge (a = h \vee a \in T))$, then $C\ddot{+}a$ is a co-coloring for n.*

The operations $C + a$, $C\dot{+}a$, and $C\ddot{+}a$ are defined as follows for each $b \in X_n$:

$$(C+a)(b) = \begin{cases} out_{\mathfrak{a}} & \text{if } b = a \wedge \exists (T,a) \in X^R_n : C((T,a)) = in_{\mathfrak{r}} \\ pout_{\mathfrak{a}} & \text{if } b = a \wedge \nexists (T,a) \in X^R_n : C((T,a)) = in_{\mathfrak{r}} \\ out_{\mathfrak{r}} & \text{if } b = (T,h) \wedge a \in T \wedge C(b) = pout_{\mathfrak{r}} \\ C(b) & \text{otherwise} \end{cases}$$

$$(C\dot{+}a)(b) = \begin{cases} in_{\mathfrak{a}} & \text{if } b = a \\ C(b) & \text{otherwise} \end{cases}$$

$$(C\ddot{+}a)(b) = \begin{cases} und_{\mathfrak{a}} & \text{if } b = a \wedge \exists (T,a) \in X^R_n : (C((T,a)) \in \{und_{\mathfrak{r}}, pund_{\mathfrak{r}}\}) \\ pund_{\mathfrak{a}} & \text{if } b = a \wedge \nexists (T,a) \in X^R_n : (C((T,a)) \in \{und_{\mathfrak{r}}, pund_{\mathfrak{r}}\}) \\ und_{\mathfrak{r}} & \text{if } b = (T,h) \wedge a \in T \wedge C(b) = pund_{\mathfrak{r}} \\ C(b) & \text{otherwise} \end{cases}$$

With *insert attack nodes* we apply the same intuition as for admissible sets (cf. Definition 5.42).

Definition 5.60 (co-coloring: Insert Attack)**.** *Let n be an insert attack node with child n' such that $X_{n'}^R = X_n^R \setminus \{r\}$, and let C be a co-coloring for n', and $r = (T, h)$.*

- *$C + r$ is a co-coloring for n;*
- *If $(h \notin X_n^A \vee C(h) \in \{out_{\mathfrak{a}}, pout_{\mathfrak{a}}\}) \wedge \forall t \in T \cap X_n^A\colon C(t) = in_{\mathfrak{a}}$ then $C\dot{+}r$ is a co-coloring for n;*
- *if $C(h) \neq in_{\mathfrak{a}} \wedge \nexists t \in T \cap X_n^A\colon C(t) \in \{out_{\mathfrak{a}}, pout_{\mathfrak{a}}\}$ then $C\ddot{+}r$ is a co-coloring for n.*

The operations $C + r$, $C\dot{+}r$, and $C\dot{+}r$ are defined as follows for each $b \in X_n$:

$$(C + r)(b) = \begin{cases} out_{\mathfrak{r}} & \text{if } b = r \wedge \exists t \in T \cap X_n^A\colon C(t) \in \{out_{\mathfrak{a}}, pout_{\mathfrak{a}}\} \\ pout_{\mathfrak{r}} & \text{if } b = r \wedge \nexists t \in T \cap X_n^A\colon C(t) \in \{out_{\mathfrak{a}}, pout_{\mathfrak{a}}\} \\ C(b) & \text{otherwise} \end{cases}$$

$$(C\dot{+}r)(b) = \begin{cases} in_{\mathfrak{r}} & \text{if } b = r \\ out_{\mathfrak{a}} & \text{if } r = (T, h) \wedge b = h \\ C(b) & \text{otherwise} \end{cases}$$

$$(C\ddot{+}r)(b) = \begin{cases} und_{\mathfrak{r}} & \text{if } b = r \wedge \exists t \in T \cap X_n^A\colon C(t) \in \{und_{\mathfrak{a}}, pund_{\mathfrak{a}}\} \\ pund_{\mathfrak{r}} & \text{if } b = r \wedge \nexists t \in T \cap X_n^A\colon C(t) \in \{und_{\mathfrak{a}}, pund_{\mathfrak{a}}\} \\ und_{\mathfrak{a}} & \text{if } r = (T, h) \wedge b = h \\ C(b) & \text{otherwise} \end{cases}$$

Again, we show that the insert nodes exactly characterize the complete colorings.

Lemma 5.61. *If for the child node n' of an insert node n each co-coloring is complete, then each co-coloring C of n is complete.*

Proof. We need to show that each of the conditions (1)-(8) of Definition 5.50 is met. (1) and (2) immediately carry over from n' both for insert argument and insert attack nodes.

For (3)-(8) we discuss first the *insert argument* nodes. Note that we construct C either from a complete coloring C' of n' via $C = C' + a$, via $C = C'\dot{+}a$, or via $C = C'\ddot{+}a$.

- Clearly, (3) and (4) hold in all three cases for w.r.t. C for all $b \in X_{n'}^A$ (i.e., the "old" arguments that have not been added in this node) because they hold for n'. For the added argument a regarding (3) only the coloring $C'\dot{+}a$ is relevant (as $(C' + a)(a) \neq in_{\mathfrak{a}}$ and $(C'\ddot{+}a)(a) \neq in_{\mathfrak{a}}$). Since we can add the coloring $C'\dot{+}a$ only if $\nexists (T, a) \in X_{n'}^R\colon (C'((T, a)) \in \{in_{\mathfrak{r}}, und_{\mathfrak{r}}, pund_{\mathfrak{r}}\})$ and the only possible colors for attacks are $in_{\mathfrak{r}}$, $out_{\mathfrak{r}}$, $pout_{\mathfrak{r}}$, $und_{\mathfrak{r}}$, and $pund_{\mathfrak{r}}$, we know that in this case (3) is satisfied.

- For the added argument a regarding (4) only the coloring $C = C' + a$ is relevant. In this case, if $C(a) = out_{\mathfrak{a}}$ then we know $\exists (T, a) \in X_n^R$: $(C((T, a)) = in_{\mathfrak{r}} \wedge a \notin T)$ which means condition (4) is satisfied; if otherwise $C(a) = pout_{\mathfrak{a}}$ then (4) is trivially true for a.
- Conditions (5), (6), and (7) carry over from n' for all attacks $(T, h) \in X_n^R$ where for the added argument a it holds $a \notin T$. Let $(T, h) \in X_n^R$ be such that for the added argument a it holds $a \in T$. If $C'((T, h)) = in_{\mathfrak{r}}$ then we do not add $C' + a$ or $C' \ddot{+}$ as a coloring to n, which means the satisfaction of condition (5) carries over from n' in these cases. For $C' \dot{+} a$ condition (5) also carries over from n' for each attack $(T, h) \in X_{\geq n}^R$ with $a \in T$ as $(C' \dot{+} a)(a) = in_{\mathfrak{a}}$.
- Moreover, in all three cases $C' + a$, $C' \dot{+} a$, and $C' \ddot{+} a$ condition (6) carries over for each attack $(T, h) \in X_{\geq n}^R$ with $a \notin T$. Moreover, for the case where we set $(C' + a)((T, h)) = out_{\mathfrak{r}}$, condition (6) is satisfied as we have $a \in T$ and $(C' + a)(a) \in \{out_{\mathfrak{a}}, pout_{\mathfrak{a}}\}$.
- For the case where we set $(C' + a)((T, h)) = und_{\mathfrak{r}}$, condition (7) is satisfied as we have $a \in T$ and $(C' + a)(a) \in \{und_{\mathfrak{a}}, pund_{\mathfrak{a}}\}$.
- For the added argument a regarding (8) only the coloring $C = C' \ddot{+} a$ is relevant. In this case, if $C(a) = und_{\mathfrak{a}}$ then we know $\exists (T, a) \in X_n^R$: $(C((T, a)) \in \{und_{\mathfrak{r}}, pund_{\mathfrak{r}}\})$ which means condition (8) is satisfied; if otherwise $C(a) = pund_{\mathfrak{a}}$ then (8) is trivially true for a.

Hence, conditions (1)-(8) are met for insert argument nodes.

We now discuss the *insert attack* nodes w.r.t. the conditions (3)-(8).

- Regarding (3), it suffices to discuss for the added attack $r = (T, h)$ the cases $(C' \dot{+} r)$ and $(C' \ddot{+} r)$ (otherwise, the condition carries over from the child node or the added attack is set to either $und_{\mathfrak{r}}$ or $pund_{\mathfrak{r}}$). However, the coloring $C' \dot{+} r$ is only added to n if $C'(h) \notin \{in_{\mathfrak{a}}, und_{\mathfrak{a}}, pund_{\mathfrak{a}}\}$, which also ensures that condition (3) is met for when we set the added attack to $in_{\mathfrak{r}}$. Likewise, $C' \ddot{+} r$ is only added if $C'(h) \neq in_{\mathfrak{a}}$.
- Condition (4) carries over from n', and the case where we set $(C' \dot{+} r)(a) = out_{\mathfrak{a}}$ when we previously had $C'(a) = pout_{\mathfrak{a}}$ is only applied if we set $(C' \dot{+} r)(r) = in_{\mathfrak{r}}$, i.e., (4) is satisfied.
- For (5), only the case $C' \dot{+} r$ is relevant, the other cases follow from n'. However, we only add $C' \dot{+} r$ to n if $\forall t \in T \cap X_n^A$ it holds $C(t) = in_{\mathfrak{a}}$, and because of the properties of tree decompositions we know that $X_{>n}^A \cap T = \emptyset$, i.e., there are no tail arguments strictly below the current node. Hence, (5) is satisfied.
- As to (6), the only case that is not clear from n' is in $C' + (T, h)$, but in this case we set $C((T, h))$ to $out_{\mathfrak{r}}$ only if $\exists t \in T \cap X_n^A$: $C'(t) \in \{pout_{\mathfrak{a}}, out_{\mathfrak{a}}\}$, which means (6) is also satisfied.

- For (7) we need to consider the case $C \ddot{+} r$, and in this case we set $C((T,h))$ to $und_{\mathfrak{r}}$ only if $\exists t \in T \cap X_n^A$: $C(t) \in \{und_{\mathfrak{a}}, pund_{\mathfrak{a}}\}$, which means (7) is also satisfied.
- Condition (8) carries over from n', and the case where we set $(C' \ddot{+} r)(a) = und_{\mathfrak{a}}$ when we previously had $C'(a) = pund_{\mathfrak{a}}$ is only applied if we set $(C' \ddot{+} r)(r) = und_{\mathfrak{r}}$ or $(C' \ddot{+} r)(r) = pund_{\mathfrak{r}}$, i.e., (8) is satisfied.

To summarize, all of (1)-(8) is satisfied, which means that each coloring in n is stable. □

In Figure 5.12 node 5 is an insert attack node. Consider the coloring C_1 in node 6 with $C_1(b) = pout_{\mathfrak{a}}$ and $C_1(c) = in_{\mathfrak{a}}$. Here we can apply all three cases $C_1 + r_4$, $C_1 \dot{+} r_4$, and $C_1 \ddot{+} r_4$. It is easy to see that in general for each coloring C in the child node of an insert attack node n for an attack (T,h) where for each $t \in T \cap X_n^A$ it holds $C(t) = in_{\mathfrak{a}}$ all three possibilities are applicable, since (1) the attack can still be set to *out* if another $t \in T \setminus X_n^A$ is inserted and set to $out_{\mathfrak{a}}/pout_{\mathfrak{a}}$, (2) the attack can be set to *in* since since all $t \in T$ are *in*, and (3) the attack can be *undecided* if no $t \in T \setminus X_n^A$ is inserted and set to $out_{\mathfrak{a}}/pout_{\mathfrak{a}}$ and some $t \in T \setminus X_n^A$ is inserted and set to $und_{\mathfrak{a}}/pund_{\mathfrak{a}}$.

Lemma 5.62. *Let C be a complete coloring for an insert node n. If in the child node n' of n complete colorings and co-colorings coincide then C is a co-coloring for n.*

Proof. Let C' be the extended coloring of C on $X_{\geq n}$ s.t. the conditions of Definition 5.50 are satisfied.

We start with *insert argument* nodes. We show that there is a co-coloring C^* in node n' s.t. for the added argument a we have $C = C^* + a$ if $C'(a) \in \{out_{\mathfrak{a}}, pout_{\mathfrak{a}}\}$, $C = C^* \dot{+} a$ if $C'(a) = in_{\mathfrak{a}}$, and $C = C^* \ddot{+} a$ if $C'(a) \in \{und_{\mathfrak{a}}, pund_{\mathfrak{a}}\}$.

- Let $C'(a) \in \{out_{\mathfrak{a}}, pout_{\mathfrak{a}}\}$, then C^* is a co-coloring of n' where C^* coincides with C on each $b, (T,h) \in X_{n'}$ with the exception of attacks (T,h) with $a \in T$, $C((T,h)) = out_{\mathfrak{r}}$, and $\nexists t \in T \cap X_{n'}^A : C'(t) \in \{out_{\mathfrak{a}}, pout_{\mathfrak{a}}\}$, for these attacks we set $C^*((T,h)) = pout_{\mathfrak{r}}$. To show that C^* is a co-coloring of n' we need to show that the conditions (1)-(8) of complete colorings are satisfied. (1) and (2) carry over from C, as $X_{>n} = X_{>n'}$. Note for (3) that C and C^* coincide on the colors $in_{\mathfrak{a}}$, $und_{\mathfrak{a}}$, and $pund_{\mathfrak{a}}$, and on the set of attacks colored either $out_{\mathfrak{r}}$ or $pout_{\mathfrak{r}}$, which means (3) carries over from C. (4) and (5) immediately carry over as well, as C and C^* coincide on all relevant colors on the arguments/attacks $X_{n'}$. Regarding (6) note that for each attack $(T,h) \in X_{n'}^R$ with $a \in T$, $C((T,h)) = out_{\mathfrak{r}}$ property (6) holds by definition of C^*, and for the other attacks (with $C(T,h) = out_{\mathfrak{r}}$ and $a \notin T$) the property carries over from C. Regarding (7) we have that for each attack $(T,h) \in X_{n'}^R$ with $a \in T$, $C((T,h)) = und_{\mathfrak{r}}$ property (7) carries over from C. Finally, (8) immediately carries over as well, as the arguments and attacks with the relevant colors coincide in C and C'. Hence, in this case C^* is complete for n' and therefore by assumption an co-coloring for n'. Then since C^* is complete from

condition (5) it follows $\nexists(T,h) \in X_n^R$: $(C^*((T,h)) = in_{\mathfrak{r}} \wedge a \in T)$, hence, $C = C^* + a$ is a co-coloring for n (the correct color $out_{\mathfrak{a}}/pout_{\mathfrak{a}}$ directly follows from the fact that C is complete and that due to the properties of tree decompositions all attacks towards a in $X_{\geq n}^R$ have to be in X_n^R).

- Now on the other side let $C'(a) = in_{\mathfrak{a}}$, then C^* is a co-coloring of n' where C^* coincides with C on each $b, (T,h) \in X_{n'}$. Again we show that the conditions (1)-(8) are satisfied. (1) and (2) again carry over from C, as $X_{>n} = X_{>n'}$. (3)-(8) immediately carry over as well, as C and C^* coincide on all relevant colors on $X_{n'}$. Then since C^* is complete it is a co-coloring for n' and from condition (3) it follows $\nexists(T,a) \in X_{n'}^R$: $(C((T,a)) = in_{\mathfrak{r}})$, hence, $C = C^* \dot{+} a$ is a cod-coloring for n.

- Finally, let $C'(a) \in \{und_{\mathfrak{a}}, pund_{\mathfrak{a}}\}$, then C^* is a co-coloring of n' where C^* coincides with C on each $b, (T,h) \in X_{n'}$, with the exception of attacks (T,h) with $a \in T$, $C((T,h)) = und_{\mathfrak{r}}$, and $\nexists t \in T \cap X_{n'}^A : C'(t) \in \{und_{\mathfrak{r}}, pund_{\mathfrak{r}}\}$, for these attacks we set $C^*((T,h)) = pund_{\mathfrak{r}}$. Again we show that the conditions (1)-(8) are satisfied. (1) and (2) carry over from C, as $X_{>n} = X_{>n'}$. Note for (3) that C and C^* coincide on the colors $in_{\mathfrak{a}}$, $und_{\mathfrak{a}}$, and $pund_{\mathfrak{a}}$, and on the set of attacks colored either $out_{\mathfrak{r}}$ or $pout_{\mathfrak{r}}$, which means (3) carries over from C. (4) and (5) immediately carry over as well, as C and C^* coincide on all relevant colors on the arguments/attacks $X_{n'}$. Regarding (6) we have that for each attack $(T,h) \in X_{n'}^R$ with $C((T,h)) = out_{\mathfrak{r}}$ property (6) carries over from C. Regarding (7) note that for each attack $(T,h) \in X_{n'}^R$ with $a \in T$, $C((T,h)) = und_{\mathfrak{r}}$ property (7) holds by the definition of C^*, and for the other attacks (with $C(T,h) = out_{\mathfrak{r}}$ and $a \notin T$) the property carries over from C. Finally, (8) immediately carries over as well, as the arguments and attacks with the relevant colors coincide in C and C'. Hence, in this case C^* is complete for n' and therefore by assumption a co-coloring for n'. Then since C^* is complete from conditions (4) and (5) it follows $\nexists(T,h) \in X_{n'}^R : (C^*((T,h)) = in_{\mathfrak{r}} \wedge (a = h \vee a \in T))$, hence, $C = C^* \ddot{+} a$ is a co-coloring for n (the correct color $und_{\mathfrak{a}}/pund_{\mathfrak{a}}$ directly follows from the fact that C is complete and that due to the properties of tree decompositions all attacks towards a in $X_{\geq n}^R$ have to be in X_n^R).

We showed that in all three cases C is a co-coloring of n.

We continue with the *insert attack* nodes. Again we show that there is a co-coloring C^* in node n' s.t. for the added attack (T,h) we have $C = C^* + (T,h)$ if $C'((T,h)) \in \{out_{\mathfrak{r}}, pout_{\mathfrak{r}}\}$, $C = C^* \dot{+} (T,h)$ if $C'((T,h)) = in_{\mathfrak{r}}$, and $C = C^* \ddot{+} (T,h)$ if $C'((T,h)) \in \{und_{\mathfrak{r}}, pund_{\mathfrak{r}}\}$.

- Let $C'((T,h)) \in \{out_{\mathfrak{r}}, pout_{\mathfrak{r}}\}$, then C^* is a co-coloring of n' where C^* coincides with C on each $b, (S,t) \in X_{n'}$. To show that C^* is a co-coloring of n' we need to show that the conditions (1)-(8) of complete colorings are satisfied. (1) and (2) carry over from C, as $X_{>n} = X_{>n'}$. (3)-(8) immediately carry over from C as C and C^* coincide on all relevant colors on $X_{n'}$. Hence, C^* is complete for n' and therefore a co-coloring, and $C = C^* + (T,h)$ is a co-coloring for n (as with insert

argument nodes, the correct color $out_{\mathfrak{r}}, pout_{\mathfrak{r}}$ follows from C being complete and the properties of tree decompositions).

- Now on the other side let $C'((T,h)) = in_{\mathfrak{r}}$, then C^* is a co-coloring of n' where C^* coincides with C on each $b, (T,h) \in X_{n'}$ with the exception the argument h (given that $h \in X^A_{n'}$) if $\nexists (S,h) \in X^R_{\geq n'} : C'((S,h)) = in_{\mathfrak{r}}$, in this case we set $C^*(h) = pout_{\mathfrak{a}}$. Again we show that the conditions (1)-(8) are satisfied. (1) and (2) carry over from C, as $X_{>n} = X_{>n'}$. (3), (5), (6), (7), and (8) immediately carry from C, as C and C^* coincide on all relevant colors on $X_{n'}$. Regarding (4), note that this clearly carries over from C for all arguments $a \in X^A_{n'} \setminus \{h\}$, and for the argument h it holds by definition of C^*. Then since C^* is complete from condition (3) it follows $(h \notin X^A_n \vee C(h) \neq in_{\mathfrak{a}})$ and from condition (5) it follows $\forall t \in T \cap X^A_n : C(t) = in_{\mathfrak{a}}$, hence, $C = C^* \dot{+} (T,h)$ is a co-coloring for n.

- Finally, let $C'((T,h)) \in \{und_{\mathfrak{r}}, pund_{\mathfrak{r}}\}$, then C^* is a co-coloring of n' where C^* coincides with C on each $b, (S,t) \in X_{n'}$ with the exception the argument h (given that $h \in X^A_{n'}$) if $\nexists (S,h) \in X^R_{\geq n'} : C'((S,h)) \in \{und_{\mathfrak{r}}, pund_{\mathfrak{r}}, out_{\mathfrak{r}}, pout_{\mathfrak{r}}\}$, in this case we set $C^*(h) = pund_{\mathfrak{a}}$. To show that C^* is a co-coloring of n' we need to show that the conditions (1)-(8) of complete colorings are satisfied. (1) and (2) carry over from C, as $X_{>n} = X_{>n'}$. (3)-(8) immediately carry over from C as C and C^* coincide on all relevant colors on $X_{n'}$, or by construction of C^*. Hence, C^* is complete for n' and therefore a co-coloring, and $C = C^* + (T,h)$ is a co-coloring for n (the correct color $und_{\mathfrak{r}}, pund_{\mathfrak{r}}$ follows from C being complete and the properties of tree decompositions).

We showed that in all three cases C is a co-coloring of n. □

5.5.4 Join Nodes

We now consider the join nodes where we combine matching colorings. Intuitively, we perform the same steps as in admissible semantics (see Definition 5.45). The only difference is that we also have to consider "upgrades" for the undecided colors, which in join nodes behave exactly like the *out*/*pout* idea.

Definition 5.63 (co-coloring: Join)**.** *Let n be a join node with children n', n'', let C be a co-coloring for n', and let D be a co-coloring for n''. If C and D are compatible, i.e.,*

1. $\{a \in X^A_{n'} \mid C(a) = in_{\mathfrak{a}}\} = \{a \in X^A_{n''} \mid D(a) = in_{\mathfrak{a}}\}$,
2. $\{r \in X^R_{n'} \mid C(r) = in_{\mathfrak{r}}\} = \{r \in X^R_{n''} \mid D(r) = in_{\mathfrak{r}}\}$,
3. $\{a \in X^A_{n'} \mid C(a) \in \{und_{\mathfrak{a}}, pund_{\mathfrak{a}}\}\} = \{a \in X^A_{n''} \mid D(a) \in \{und_{\mathfrak{a}}, pund_{\mathfrak{a}}\}\}$, *and*
4. $\{r \in X^R_{n'} \mid C(r) \in \{und_{\mathfrak{r}}, pund_{\mathfrak{r}}\}\} = \{r \in X^R_{n''} \mid D(r) \in \{und_{\mathfrak{r}}, pund_{\mathfrak{r}}\}\}$

then $C \bowtie D$ is an ad-coloring for n, where

$$(C \bowtie D)(b) = \begin{cases} in_{\mathfrak{a}} & \text{if } C(b) = D(b) = in_{\mathfrak{a}} \\ und_{\mathfrak{a}} & \text{if } C(b) = und_{\mathfrak{a}} \vee D(b) = und_{\mathfrak{a}} \\ pund_{\mathfrak{a}} & \text{if } C(b) = D(b) = pund_{\mathfrak{a}} \\ out_{\mathfrak{a}} & \text{if } C(b) = out_{\mathfrak{a}} \vee D(b) = out_{\mathfrak{a}} \\ pout_{\mathfrak{a}} & \text{if } C(b) = D(b) = pout_{\mathfrak{a}} \\ in_{\mathfrak{r}} & \text{if } C(b) = D(b) = in_{\mathfrak{r}} \\ und_{\mathfrak{r}} & \text{if } C(b) = und_{\mathfrak{r}} \vee D(b) = und_{\mathfrak{r}} \\ pund_{\mathfrak{r}} & \text{if } C(b) = D(b) = pund_{\mathfrak{r}} \\ out_{\mathfrak{r}} & \text{if } C(b) = out_{\mathfrak{r}} \vee D(b) = out_{\mathfrak{r}} \\ pout_{\mathfrak{r}} & \text{if } C(b) = D(b) = pout_{\mathfrak{r}} \end{cases}$$

Lemma 5.64. *If for the child nodes n', n'' of a join node n each co-coloring is complete, then each co-coloring C of n is complete.*

Proof. We need to show that each of the conditions (1)-(8) of Definition 5.50 is met. (1) and (2) immediately carry over from n' and n'' since $X^A_n = X^A_{n'} = X^A_{n''}$ and $X^R_n = X^R_{n'} = X^R_{n''}$. (3) holds because it has to hold both for all (S, a) in $X^R_{n'}$ and in $X^R_{n''}$ by assumption. (4) is satisfied, as we set an argument to $out_{\mathfrak{a}}$ only if it is colored $out_{\mathfrak{a}}$ in at least one of the two colorings of n' and n''. Similarly, (5) and (6) carry over from n' and n''. (7) is satisfied, similarly to (4), because we set an attack to $und_{\mathfrak{r}}$ only if it is colored $und_{\mathfrak{r}}$ in at least one of the two colorings of n' and n''. Likewise, (8) is true because we set an argument to $und_{\mathfrak{a}}$ only if it is colored $und_{\mathfrak{a}}$ in at least one of the two colorings of n' and n''. □

The join node 4 of our Figure 5.12 combines the co-colorings of nodes 5 and 11. Note that 4 contains fewer co-colorings than both of its child nodes. While in general we have to consider all possible combinations of colorings, we expect that in most SETAFs many co-colorings will be "filtered out" by join nodes due to sufficiently "different" co-colorings in the two branches, as it is the case in our example. One possible optimization strategy for implementations could be to actively encourage this situation to happen by cleverly choosing the tree decomposition.

Lemma 5.65. *Let C be a complete coloring for a join node n. If in the child nodes n', n'' of n complete colorings and co-colorings coincide then C is a co-coloring for n.*

Proof. Let C' be the extended coloring of C on $X_{\geq n}$ s.t. the conditions of Definition 5.50 are satisfied.

We will show that there are co-colorings C^* for n' and D^* for n'' such that $C = C^* \bowtie D^*$. We define C^* such that it coincides with C on each $a, (T, h) \in X_n$ with the following three exceptions, each accounting for one of the provisional colors $pout_{\mathfrak{a}}$, $pout_{\mathfrak{r}}$, $pund_{\mathfrak{a}}$, and $pund_{\mathfrak{r}}$:

1. arguments b s.t. $C(b) = out_{\mathfrak{a}}$ and $\nexists (S,b) \in X^R_{\geq n'} : C'((S,b)) = in_{\mathfrak{r}}$, for these arguments we set $C^*(b) = pout_{\mathfrak{a}}$,
2. attacks (S,b) s.t. $C((S,b)) = out_{\mathfrak{r}}$ and $\nexists s \in S \cap X^A_{\geq n'} : C'(s) \in \{out_{\mathfrak{a}}, pout_{\mathfrak{a}}\}$, for these attacks we set $C^*((S,b)) = pout_{\mathfrak{r}}$,
3. arguments b s.t. $C(b) = und_{\mathfrak{a}}$ and $\nexists (S,b) \in X^R_{\geq n'} : C'((S,b)) \in \{und_{\mathfrak{r}}, pund_{\mathfrak{r}}\}$, for these arguments we set $C^*(b) = pund_{\mathfrak{a}}$,
4. attacks (S,b) s.t. $C((S,b)) = und_{\mathfrak{r}}$ and $\nexists s \in S \cap X^A_{\geq n'} : C'(s) = und_{\mathfrak{a}}$, for these attacks we set $C^*((S,b)) = pund_{\mathfrak{r}}$.

We show that properties (1)-(8) of complete colorings are satisfied for C^*. (1) and (2) carry over from C, as $X_{>n} = X_{>n'} = X_{>n''}$. (3) is satisfied as for C and C^* it holds

$$\{(T,h) \in X^R_n \mid C((T,h)) \in \{out_{\mathfrak{r}}, pout_{\mathfrak{r}}\})\} = \{(T,h) \in X^R_{n'} \mid C^*((T,h)) \in \{out_{\mathfrak{r}}, pout_{\mathfrak{r}}\})\}.$$

(4), (6), (7), and (8) are satisfied by definition of C^* for the "updated" colors, for the others the property carries over from C. (5) immediately carries over from C. Hence, C^* is complete (and D^* as well due to symmetry). Finally, note that by construction of C^* and D^* it holds

1. $\{a \in X^A_{n'} \mid C^*(a) = in_{\mathfrak{a}}\} = \{a \in X^A_{n''} \mid D^*(a) = in_{\mathfrak{a}}\}$,
2. $\{r \in X^R_{n'} \mid C^*(r) = in_{\mathfrak{r}}\} = \{r \in X^R_{n''} \mid D^*(r) = in_{\mathfrak{r}}\}$,
3. $\{a \in X^A_{n'} \mid C^*(a) \in \{und_{\mathfrak{a}}, pund_{\mathfrak{a}}\}\} = \{a \in X^A_{n''} \mid D^*(a) \in \{und_{\mathfrak{a}}, pund_{\mathfrak{a}}\}\}$, and
4. $\{r \in X^R_{n'} \mid C^*(r) \in \{und_{\mathfrak{r}}, pund_{\mathfrak{r}}\}\} = \{r \in X^R_{n''} \mid D^*(r) \in \{und_{\mathfrak{r}}, pund_{\mathfrak{r}}\}\}$

Hence, $C = C^* \bowtie D^*$ is a co-coloring for n. □

5.5.3 Final Steps for Complete Semantics

We can again sum up our results for each node type to obtain the soundness and completeness for our algorithm to characterize the complete extensions.

Proposition 5.66. *Let $\mathcal{T} = (V_{\mathcal{T}}, E_{\mathcal{T}})$ be a nice tree-decomposition of a SETAF SF. Then in each node $n \in V_{\mathcal{T}}$ co-colorings and complete colorings coincide.*

Proof. ($\subseteq$) Follows by structural induction over the tree-decomposition structure with the leaves as a base (Lemma 5.53) and the forget, insert, and join nodes as steps (Lemma 5.57, Lemma 5.61, Lemma 5.64).

($\supseteq$) First note that by ($\subseteq$) all co-colorings in each node are complete. Then the statement follows by structural induction over the tree-decomposition structure with the leaves as a base (Lemma 5.54) and the forget, insert, and join nodes as steps (Lemma 5.58, Lemma 5.62, Lemma 5.65). □

We then obtain the main result for complete semantics. We want to emphasize that the runtime is bounded by the same function as with admissible sets. This is due to the fact that in join nodes we have sort in three "categories": in, out, undecided. The addition of the provisional undecided color for arguments does not add any additional complexity w.r.t. our complexity analysis.

Theorem 5.67. *The problems $Cred_{adm} = Cred_{com} = Cred_{pref}$ as well as counting the number of complete sets can be done in time $O(9^k \cdot k \cdot (|A| + |R|))$. Moreover, we can enumerate all complete extensions with linear delay.*

Proof. The correctness of the algorithm (i.e., we account *only* for complete colorings in the root node) and the completeness of the algorithm (i.e., we account for *all* the complete colorings in the root node) is shown in Proposition 5.66. Then, from the fact that in the root node n we have $X_n^A = X_n^R = \emptyset$, we trivially get $X_{>n}^A = A$ and $X_{>n}^R = R$. Hence, by property (1) and (2) of complete colorings we know that the colorings in the root node can be extended to SF such that no provisional colors are used. By Proposition 5.51 we then get that the colorings in the root node exactly characterize $com(SF)$.

We can assume the number of nodes to be bounded by $O(|A| + |R|)$ and that we can find and access rows in linear time w.r.t. k. For each node, the number of valid colorings (i.e., rows in our tables of colorings) is bounded by 5^k.

In leaf nodes, we can check the conditions for leaves in time $O(k^2)$ for each of the $O(5^k)$ possible colorings, resulting in $O(5^k \cdot k^2)$. In forget nodes, we can check whether the condition is satisfied and compute eventually resulting colorings in time $O(k)$ for each of the $O(5^k)$ colorings of the child node, resulting in $O(5^k \cdot k)$. In insert nodes, we can check whether the condition is satisfied and compute eventually resulting colorings in time $O(k^2)$ for each of the $O(5^k)$ colorings of the child node, resulting in $O(5^k \cdot k^2)$. Finally, for join nodes we have to consider $5^k \cdot 5^k = 25^k$ pairs. However, we only need to consider 9^k pairs if we assume the data structure to be properly sorted, e.g. lexicographically by treating the colors $in_\mathfrak{a}/in_\mathfrak{r}$ as 0, $und_\mathfrak{a}/pund_\mathfrak{a}/und_\mathfrak{r}/pund_\mathfrak{r}$ as 1, and $pout_\mathfrak{a}/out_\mathfrak{a}/pout_\mathfrak{r}/out_\mathfrak{r}$ as 2. As each table has $O(5^k)$ rows, sorting is in $O(5^k \cdot k)$. Let C be a coloring such that $m \leq k$ arguments/attacks are colored as $in_\mathfrak{a}/in_\mathfrak{r}$. There exist at most 2^{k-m} "distinct" colorings C' (distinct within their group when we group colorings as follows: $\{in_\mathfrak{a}, in_\mathfrak{r}\}, \{und_\mathfrak{a}, pund_\mathfrak{a}, und_\mathfrak{r}, pund_\mathfrak{r}\}, \{out_\mathfrak{a}, pout_\mathfrak{a}, out_\mathfrak{r}, pout_\mathfrak{r}\}$) with $\forall x : (C(x) \in \{in_\mathfrak{a}, in_\mathfrak{r}\} \Leftrightarrow C'(x) \in \{in_\mathfrak{a}, in_\mathfrak{r}\})$. There are $\binom{k}{m}$ possibilities resulting from the choice of m. Finally, we have to distinguish 2^{k-m} combinations of sets $\{x \mid C(x) \in \{und_\mathfrak{a}, pund_\mathfrak{a}, und_\mathfrak{r}, pund_\mathfrak{r}\}\}$ and $\{x \mid C(x) \in \{out_\mathfrak{a}, pout_\mathfrak{a}, out_\mathfrak{r}, pout_\mathfrak{r}\}\}$, resulting in $\sum_{m=0}^{k} \binom{k}{m} \cdot 2^{k-m} \cdot 2^{k-m} \cdot 2^{k-m} = 9^k$ join pairs. We can then compute $C \bowtie D$ in $O(k)$, resulting in $O(9^k \cdot k)$ for join nodes, dominating the runtime of the other node types. The resulting runtime for the algorithm is $O(9^k \cdot k \cdot (|A| + |R|))$.

We can decide credulous acceptance by flagging those colorings that contain the argument in question. In each node we update this flag accordingly, and the flag in the root node indicates whether the argument is credulously accepted. We can keep the count of

complete extensions corresponding to each coloring (cf. [DPW12]). Finally, to enumerate the complete extensions once the dynamic programming algorithm is done we can traverse the tree top-down and output the complete extensions with linear delay (cf. [DPW12]). □

5.6 Discussion

In this chapter, we investigated the treewidth parameter for reasoning tasks in SETAFs. We showed that reasoning with constant primal-treewidth remains hard (contrasting the results for the special case of AFs), while constant incidence-treewidth allows us to reason and count in polynomial time. The parameterized problems are in FPT. Finally, we improved these generically obtained results by providing a dynamic programming algorithm tailored for SETAFs, highlighting interesting differences to the AF-case that arise from the generalization step. While we presented DP algorithms for stable, admissible, and complete semantics, the thereby introduced ideas form the basis also for other semantics such as preferred and semi-stable extensions. For this, the ideas for AFs are also applicable in the context of SETAFs [DPW12, BHW16]. In particular, Dvořák et al. [DPW12] characterize preferred extensions for AFs in a tree decomposition based dynamic programming approach by storing for each admissible coloring a set Γ of admissible colorings that characterize sets that are proper supersets of the characterized extensions of the coloring. These "certificates" intuitively capture the admissible sets that prevent an admissible set from being preferred (due to violated subset-maximality). This approach can be adapted to characterize preferred extensions in SETAFs.

The underlying structure of SETAFs is a directed hypergraph. While there are measures available for general hypergraphs, the directed case is not as well explored. Moreover, while there are several systems available to compute the treewidth of undirected simple graphs efficiently—be it exactly or heuristically—the situation for implementations of hyper-treewidth is less advanced (see e.g. [GLS01, GGS14] for an overview). Finally, reasoning in frameworks with fixed *directed* graph parameters (e.g., cycle rank, directed path-width, etc.) already turned out to be intractable for AFs [DPW12]; which carries over to SETAFs. Hence, we decided to focus on the treewidth-based measures, so that we can implement the presented algorithms in the future.

Considering SETAFs in recent additions to the treewidth literature in the context of argumentation constitutes interesting topics for future research, see e.g. [FHMM21].

CHAPTER 6

Conclusion

In this thesis, we thoroughly investigated argumentation frameworks with collective attacks (SETAFs). SETAFs oftentimes proved useful due to their rich syntax (compared to AFs), and many key semantic properties have been shown to generalize to SETAFs: for instance, already when first discussing collective attacks, Nielsen and Parsons showed that the fundamental lemma of Dung AFs generalizes to SETAFs. However, SETAFs have not been investigated in great detail w.r.t. computational properties by means of formal complexity-theoretic methods, barring algorithmic ideas on the computation of preferred extensions [NP06a]. In this thesis, we fill this gap. In 2019, it was shown that it is possible to generalize most common semantics to SETAFs both for extensions as well as labelings [FB19], while still preserving many desirable properties. In the same year it was shown that SETAFs are strictly more expressive than AFs in most semantics [DFW19]. In the light of these promising results one would expect computational drawbacks, such as raised upper bounds or the inapplicability of advanced algorithmic ideas. While it was shown that in general the same upper bounds in terms of our considered decision problems hold for AFs and SETAFs [DGW18], the latter question remained open.

We cannot reasonably assume to come up with efficient algorithms to reason on every SETAF due to the well-known hardness-results and the widely believed relationship $P \neq NP$. However, on restricted classes (like with bounded size of strongly-connected-components (SCCs), low treewidth, or small backdoor size) we can show significant improvements via advanced algorithmic techniques. We showed that indeed these techniques can be applied to SETAFs, either on their primal-graph or incidence-graph structure. In the following, we summarize our findings in this regard. In particular, Section 6.1.1 summarizes our findings regarding principles and SCC-recursiveness of Chapter 3, in Section 6.1.2 we recall the findings of Chapter 4 where we establish the backdoor notion for SETAFs and provide backdoor-based reasoning algorithms, and in Section 6.1.3 we reiterate our results regarding the parameter treewidth. Section 6.1.4

contains the insights that we gain when comparing the different approaches regarding pairwise compatibility. Finally, in Section 6.2 we discuss future work.

6.1 Summary & Insights

In the following, we provide a concise summary of the results and insights of Chapters 3–5.

6.1.1 Principles and Incremental Computation

In Chapter 3 we established that for SETAFs most principles—when carefully generalized with their intuitions in mind—still apply in the same manner as for AFs. A notable exception here is the *tightness* principle: its violation for the SETAF semantics is due to their increased expressiveness, and can therefore be seen as either an advantage or a disadvantage, depending on the application. Intuitively, tightness is satisfied if for an extension E and an argument $a \notin E$ the reason why E is not acceptable w.r.t. E is at least one argument $b \in E$ which is never jointly accepted alongside a. While many AF semantics satisfy this principle, in SETAFs due to their set attacks tightness is violated, as the reason for an unacceptable argument can be a *set* of arguments rather than a single one. Along the way we discovered intricate details that hide interesting phenomena in AFs behind their simple syntax, such as the difference between *unattacked* sets and *uninfluenced* sets. We argued that indeed the influence of a path in the primal graph exceeds the mere meaning of attacks, but also influences defense and undecidedness in the targeted part of a framework. While on AFs both notions coincide, the richer structure of SETAFs allows us to formally investigate this and other important distinctions. Finally, we introduced novel principles for semantics of SETAFs such as Allowing Partial Conflicts I–III, Tail Strengthening, and Attack Weakening that in the future will help guide the development of new semantics. These principles are genuinely applicable in the context of collective attacks, as they trivialize or exceed the syntactic possibilities of AFs.

Moreover, we showed that SETAFs can be split along the structure of the SCCs of the primal-graph, and the verification of preferred extensions is in FPT w.r.t. the size of the largest SCC. Note that this result captures a general characterization of argumentation semantics in the sense that they can be evaluated in parts (along the SCC-structure). This is also applicable for other admissibility-based semantics studied in this thesis. We then generalized this result by utilizing the tractable fragments of SETAFs, exploiting SCCs that are acyclic, even- or odd-cycle-free, or primal-bipartite. In a nutshell, if an SCC either contains less than a constant k many arguments or belongs to one of these fragments, we can efficiently compute the corresponding sub-problem. If all SCCs have this property, we can verify preferred extensions in FPT time for the whole SETAF. However, this idea is not applicable to SCCs that are fully-symmetric under projection, as we have also shown. Moreover, the results give rise to incremental algorithms to enumerate extensions along the SCC-structure.

6.1.2 The Backdoor-Based Approach

In Chapter 4 we showed that the backdoor approach is applicable to SETAFs as well, again via the route of tractable fragments in the primal graph. This is possible for deletion backdoors to acyclicity and even-cycle-freeness—for the other tractable fragments under our consideration it was shown already for AFs that this is impossible (under standard complexity-theoretic assumptions) [DOS12]. We have shown that we can characterize the extensions in time $O(2^p \cdot poly(|SF|))$ instead of $O(3^p \cdot poly(|SF|))$ (with p being the size of the backdoor), which was the state-of-the-art in AFs. We achieve this improvement by instead of making a 3-valued guess (in, out, undecided), making only a 2-valued guess (in, "not in")—the second option is later resolved to correspond either to "out" or "undecided", depending on the context. Assuming the Strong-Exponential-Time-Hypothesis holds, we established that no further improvement is possible in this regard, as then our algorithm is optimal (w.r.t. the exponential part). We cannot hope to characterize complete extensions with a 2^p approach, as we have shown there can be $O(3^p)$ many w.r.t. a backdoor of size p. Hence, in this case we generalized the 3^p technique of AFs to to SETAFs. In summary, we showed that the backdoor approach is indeed applicable to SETAFs, and along the way even improved the state-of-the-art in the special case of AFs.

6.1.3 The Treewidth-Based Approach

In Chapter 5 we investigated the parameter treewidth. We first established that even with constant primal-treewidth it is impossible to obtain polynomial time reasoning (as always under standard complexity theoretic assumptions). This is in contrast to our positive results regarding SCC-recursiveness and backdoors; intuitively, the reason for this discrepancy is that we can encode the "hard parts" of a SETAF in the collective attacks, while the primal structure remains simple—cf. Figure 5.1. In this example the clauses of a propositional formula in CNF are encoded as collective attacks of a SETAF, and the "choice" between the positive and negative instance of a literal is simply modeled as a symmetric attack. As the focus of the backdoor approach lies on the arguments rather than the attacks, this route is indifferent to the "overlapping attacks". Hence, we obtain a backdoor size to even-cycle-freeness (and acyclicity) of k for the reduction in Figure 5.1 for k variables in the original propositional formula, while we obtain constant treewidth.

Given this negative result, the focus of our investigations shifted to the incidence graph, which more accurately captures the structure of a SETAF (note that while many SETAFs can map to the same primal graph, the incidence graph uniquely identifies a SETAF). To this end we first established that FPT time reasoning is possible for all semantics under our consideration w.r.t. incidence-treewidth via Courcelle's theorem and a characterization in monadic second order logic. We then refined these results by providing tailored algorithms for selected semantics (each illustrating a key idea). Similar to our backdoor approach, we were able to obtain an improvement in the asymptotic runtime over the state-of-the-art in AFs. This is obtained by a technical change of the use of colors: while in the AF approach of [DPW12] it is possible to "upgrade" an "undecided" color to an out "color",

we disallow this change in our approach. Instead, we fully rely on provisional colors, which allows us to obtain an improved asymptotic runtime for join nodes (which are the computational bottleneck), as far fewer candidate solutions have to be compared. In summary, we showed that the treewidth approach is applicable for SETAFs for the incidence graph (but not for the primal graph), and the close look to the structure again allowed us to obtain runtime improvements over the state-of-the-art even in the special case of AFs.

6.1.4 Comparison and Compatibility

We have seen tree different general approaches to achieve computational advantages. In the following we want to briefly highlight why it makes sense to talk about these techniques specifically, as they are, in a sense, pairwise orthogonal. It follows from the respective AF result (see [DHK+22]), together with the fact that in the special case of AFs the primal-treewidth equals the incidence-treewidth, that in a SETAF the parameters treewidth and backdoors are unrelated. That is, there is a family of AFs/SETAFs where one parameter can be arbitrarily high while the other remains constantly low. Hence, it makes sense to initially consider both variants, and depending on the problem at hand choose whichever is more suitable. In this context is is noteworthy that also a hybrid parameter backdoor-treewidth has been investigated for AFs [DHK+22], where the new parameter value dominates both the backdoor and treewidth parameter value. While it is reasonable to assume that this technique generalizes to SETAFs, the drawback of this approach is that there are as of yet no efficient methods to *find* the parameter value (or its witness, in this case a backdoor with small treewidth in the so-called torso graph, which is needed to perform the actual calculations). On the other hand, a backdoor to low treewidth (a "treewidth-backdoor") has not yet been investigated. While this parameter might work in principle, it combines the worst of the both worlds, as finding such a backdoor poses a major computational issue, and in addition the final computation along the tree decomposition is everything but straight-forward.

Finally, regarding SCC-recursiveness it is easy to see that both the number as well as the size of the SCCs is independent of both treewidth and backdoor size (other than that each non-trivial SCC induces at least one cycle). However, our work has shown that any technique that allows for the consideration of mitigated attacks and is closed under argument- and attack-deletion can be used for the intermediate step of evaluating one SCC in an incremental computation procedure. Both the backdoor- and treewidth approach trivially allow for argument- and attack-deletion (in that the parameter value does not increase), and the consideration of mitigated attacks should not pose a major issue [10].

In summary, our approaches are orthogonal, but can be combined in the future.

[10] In this regard it is noteworthy that mitigated attacks can also be "simulated" by adding a novel self-attacking argument to the tail of a mitigated attack, as is discussed for SETAFs in the context of splitting [BDKW24] or is done for SCC-recursiveness on ADFs [GRS21]. This effectively means that we do not need to consider mitigated attacks in a solver if we perform this pre-processing step.

6.2 Future Work & Outlook

With the introduction of the SETAF-reduct and the modularization property, the family of semantics based on weak admissibility can be defined for SETAFs. Preliminary work in this regard in connection with structured argumentation was recently published [BKU24]. While it is reasonable to assume that problems w.r.t. weak-admissibility for SETAFs will be PSPACE-complete as for AFs [DUW22], thorough investigations regarding computational aspects have not yet been conducted. The notion of SCC-recursiveness for SETAFs gives rise to the possibility to define new semantics such as *cf2* [BGG05] and *stage2* [DG12]. Moreover, our investigations serve as a starting point for a structural analysis of HYPAFs—frameworks with sets of arguments attacking sets of arguments [DDK+23]—which generalize SETAFs. HYPAFs have recently been shown to capture the semantics of assumption-based argumentation with preferences (ABA$^+$) [DDK+24]. For these frameworks, future work includes a principle-based analysis, as well as a thorough investigation of computational aspects. Finally, while our considerations are mainly theoretical analyses, in the future these approaches can be implemented and measured in comparison to existing SETAF solvers, such as via Answer-Set-Programming (ASP) [DGW18]. While the backdoor-approach has recently been implemented to solve problems in structured argumentation [AU24], this is still open for the other approaches.

We showed that SETAFs indeed yield the same advantages w.r.t. principles (barring tightness), and that the advanced algorithmic ideas under our consideration are applicable in the context of SETAFs. SETAFs are a minimally-invasive generalization of AFs which allow for additional modeling power while retaining the advantages of simplicity and the computational upper bounds of AFs. Now that the computational landscape for SETAFs is more clear, in the future we might see more and more applications that make use of their extended syntax. This is in line with recent trends in the computational argumentation community that call for a more in depth analysis of algorithmic ideas for abstract argumentation [Dun22].

Overview of Generative AI Tools Used

For this work no generative AI tools were used. In particular, this work contains no generated text, neither in full nor in changed form. Moreover, all figures, tables, etc. were created by the author without the help of AI tools. For spellchecking the Latex IDE TeXstudio was used, version 4.8.1 (`https://texstudio.org/`).

List of Figures

2.1 The standard reduction 19

4.1 SETAF primal graph, running example for Chapter 4 73
4.2 A NOEVEN-backdoor and the remaining SETAF after removal 74
4.3 Example for the function $\mathsf{propagate}_{\mathsf{IO}}$ 78
4.4 Example for the function $\mathsf{propagate}_{\mathsf{U}}$ 80
4.5 Illustration of high number of complete extensions w.r.t. backdoor size . . 85
4.6 Example for the backdoor-algorithm for complete extensions 87

5.1 Hardness-proof for primal-treewidth 97
5.2 Hardness-proof for primal-treewidth (preferred semantics) 98
5.3 Running example for Chapter 5 101
5.4 Sub-frameworks per node of the running example 103
5.5 All st-colorings for our running example 107
5.6 St-colorings for leaf nodes and forget argument nodes 108
5.7 St-colorings forget attack nodes 109
5.8 St-colorings for insert attack nodes 112
5.9 St-colorings for insert argument nodes 113
5.10 St-colorings for join nodes 117
5.11 Selected ad-colorings for our running example 124
5.12 Selected co-colorings for our running example 140

List of Tables

2.1 Complexity of reasoning in SETAFs 18

3.1 Overview of SETAF principles 23

Bibliography

[AMW17] Michael Abseher, Nysret Musliu, and Stefan Woltran. htd - A free, open-source framework for (customized) tree decompositions and beyond. In *Proceedings of the 14th International Conference on Integration of AI and OR Techniques in Constraint Programming, CPAIOR 2017*, pages 376–386. Springer, 2017. doi: 10.1007/978-3-319-59776-8_30.

[AS17] Ryuta Arisaka and Ken Satoh. Coalition formability semantics with conflict-eliminable sets of arguments. In *roceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017*, pages 1469–1471. ACM, 2017. doi: 10.5555/3091125.3091332.

[AU24] Kiet Ngyuen Anh and Markus Ulbricht. Preferred reasoning in ABA by cycle-breaking. In *Proceedings of the 33rd International Joint Conference on Artificial Intelligence, IJCAI 2024*, pages 3523–3531, 2024. doi: 10.24963/ijcai.2024/390.

[AW14] Amir Abboud and Virginia Vassilevska Williams. Popular Conjectures Imply Strong Lower Bounds for Dynamic Problems. In *Proceedings of the 55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014*, pages 434–443, 2014. doi: 10.1109/FOCS.2014.53.

[Bau11] Ringo Baumann. Splitting an argumentation framework. In *Proceedings of the 11th International Conference on Logic Programming and Nonmonotonic Reasoning, LPNMR 2011*, volume 6645 of *LNCS*, pages 40–53. Springer, 2011. doi: 10.1007/978-3-642-20895-9_6.

[BB20] Meghyn Bienvenu and Camille Bourgaux. Querying and repairing inconsistent prioritized knowledge bases: Complexity analysis and links with abstract argumentation. In *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning, KR 2020*, pages 141–151, 2020. doi: 10.24963/KR.2020/15.

[BBU20a] Ringo Baumann, Gerhard Brewka, and Markus Ulbricht. Comparing weak admissibility semantics to their dung-style counterparts - reduct, modularization, and strong equivalence in abstract argumentation. In

Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning, KR 2020, pages 79–88, 2020. doi: 10.24963/kr.2020/9.

[BBU20b] Ringo Baumann, Gerhard Brewka, and Markus Ulbricht. Revisiting the foundations of abstract argumentation - semantics based on weak admissibility and weak defense. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence, AAAI 2020*, pages 2742–2749. AAAI Press, 2020. doi: 10.1609/aaai.v34i03.5661.

[BCD+21] Antonis Bikakis, Andrea Cohen, Wolfgang Dvořák, Giorgos Flouris, and Simon Parsons. Joint attacks and accrual in argumentation frameworks. *FLAP*, 8(6):1437–1501, 2021.

[BCG11] Pietro Baroni, Martin Caminada, and Massimiliano Giacomin. An introduction to argumentation semantics. *Knowledge Eng. Review*, 26(4):365–410, 2011. doi: 10.1017/S0269888911000166.

[BDKM17] Elise Bonzon, Jérôme Delobelle, Sébastien Konieczny, and Nicolas Maudet. A parametrized ranking-based semantics for persuasion. In *Proceedings of the 11th International Conference on Scalable Uncertainty Management, SUM 2017*, volume 10564 of *LNCS*, pages 237–251. Springer, 2017. doi: 10.1007/978-3-319-67582-4_17.

[BDKU24] Giovanni Buraglio, Wolfgang Dvořák, Matthias König, and Markus Ulbricht. Justifying argument acceptance with collective attacks: Discussions and disputes. In *Proceedings of the 33rd International Joint Conference on Artificial Intelligence, IJCAI 2024*, pages 3281–3288, 2024. doi: 10.24963/ijcai.2024/363.

[BDKW24] Giovanni Buraglio, Wolfgang Dvořák, Matthias König, and Stefan Woltran. Splitting argumentation frameworks with collective attacks. In *Proceedings of the 5th International Workshop on Systems and Algorithms for Formal Argumentation, SAFA 2024*, CEUR Workshop Proceedings, pages 41–55, 2024.

[BDST23] Stefano Bistarelli, Victor David, Francesco Santini, and Carlo Taticchi. Temporal probabilistic argumentation frameworks. In *Proceedings of the 38th Italian Conference on Computational Logic, CILC 2023*, volume 3428 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2023.

[BG07] Pietro Baroni and Massimiliano Giacomin. On principle-based evaluation of extension-based argumentation semantics. *Artif. Intell.*, 171(10-15):675–700, 2007. doi: 10.1016/j.artint.2007.04.004.

[BGG05] Pietro Baroni, Massimiliano Giacomin, and Giovanni Guida. SCC-recursiveness: a general schema for argumentation semantics. *Artif. Intell.*, 168(1-2):162–210, 2005. doi: 10.1016/j.artint.2005.05.006.

[BGGvdT18] Pietro Baroni, Dov Gabbay, Massimiliano Giacomin, and Leendert van der Torre, editors. *Handbook of Formal Argumentation*. College Publications, 2018.

[BGL14] Pietro Baroni, Massimiliano Giacomin, and Beishui Liao. On topology-related properties of abstract argumentation semantics. A correction and extension to dynamics of argumentation systems: A division-based method. *Artif. Intell.*, 212:104–115, 2014. doi: 10.1016/j.artint.2014.03.003.

[BHW16] Bernhard Bliem, Markus Hecher, and Stefan Woltran. On efficiently enumerating semi-stable extensions via dynamic programming on tree decompositions. In *Proceedings of the 6th International Conference on Computational Models of Argument, COMMA 2016*, volume 287 of *FAIA*, pages 107–118. IOS Press, 2016. doi: 10.3233/978-1-61499-686-6-107.

[BJN+21] Dorothea Baumeister, Matti Järvisalo, Daniel Neugebauer, Andreas Niskanen, and Jörg Rothe. Acceptance in incomplete argumentation frameworks. *Artif. Intell.*, 295:103470, 2021. doi: 10.1016/J.ARTINT.2021.103470.

[BK23] Michael Bernreiter and Matthias König. From qualitative choice logic to abstract argumentation. In *Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning, KR 2023*, pages 737–741, 2023. doi: 10.24963/KR.2023/73.

[BKU24] Lydia Blümel, Matthias König, and Markus Ulbricht. Weak admissibility for ABA via abstract set attacks. In *Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning, KR 2024*, pages 178–188, 2024. doi: 10.24963/kr.2024/17.

[Boc03] Alexander Bochman. Collective argumentation and disjunctive logic programming. *J. Log. Comput.*, 13(3):405–428, 2003. doi: 10.1093/logcom/13.3.405.

[Bod96] Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996. doi: 10.1137/S0097539793251219.

[BRT19] Pietro Baroni, Antonio Rago, and Francesca Toni. From fine-grained properties to broad principles for gradual argumentation: A principled spectrum. *Int. J. Approx. Reason.*, 105:252–286, 2019. doi: 10.1016/j.ijar.2018.11.019.

[BW10] Gerhard Brewka and Stefan Woltran. Abstract dialectical frameworks. In *Proceedings of the 12th International Conference on Principles of Knowledge Representation and Reasoning, KR 2010*, pages 780–785. AAAI Press, 2010.

[Cam14] Martin Caminada. Strong admissibility revisited. In *Proceedings of the 5th International Conference on Computational Models of Argument, COMMA 2014*, volume 266 of *FAIA*, pages 197–208. IOS Press, 2014. doi: 10.3233/978-1-61499-436-7-197.

[CCD12] Martin Caminada, Walter A. Carnielli, and Paul E. Dunne. Semi-stable semantics. *J. Log. Comput.*, 22:1207–1254, 2012.

[CGGS15] Andrea Cohen, Sebastian Gottifredi, Alejandro Javier García, and Guillermo Ricardo Simari. An approach to abstract argumentation with recursive attack and support. *J. Appl. Log.*, 13(4):509–533, 2015. doi: 10.1016/J.JAL.2014.12.001.

[CGVZ14] Federico Cerutti, Massimiliano Giacomin, Mauro Vallati, and Marina Zanella. An SCC recursive meta-algorithm for computing preferred labellings in abstract argumentation. In *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning, KR 2014*. AAAI Press, 2014.

[Cha12] Günther Charwat. Tree-decomposition based algorithms for abstract argumentation frameworks. Master's thesis, TU Wien, 2012.

[CKRU24] Martin Caminada, Matthias König, Anna Rapberger, and Markus Ulbricht. Attack semantics and collective attacks revisited. *Argument and Computation*, 2024. Pre-press. doi: 10.3233/AAC-230011.

[CLL+08] 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. doi: 10.1145/1411509.1411511.

[CMDM05] Sylvie Coste-Marquis, Caroline Devred, and Pierre Marquis. Symmetric argumentation frameworks. In *Proceedings of the 8th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, ECSQARU 2005*, volume 3571 of *LNCS*, pages 317–328. Springer, 2005. doi: 10.1007/11518655_28.

[Cou87] Bruno Courcelle. Recognizability and second-order definability for sets of finite graphs. Technical Report I-8634, Université de Bordeaux, 1987.

[Cou90] Bruno Courcelle. Graph rewriting: an algebraic and logic approach. In *Handbook of theoretical computer science, Vol. B*, pages 193–242. Elsevier, Amsterdam, 1990. doi: 10.1016/B978-0-444-88074-1.50010-X.

[DBC01] 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.

[DBC02] Paul E. Dunne and Trevor J. M. Bench-Capon. Coherence in finite argument systems. *Artif. Intell.*, 141(1/2):187–203, 2002. doi: 10.1016/S0004-3702(02)00261-8.

[DD18] Wolfgang Dvořák and Paul E. Dunne. Computational problems in formal argumentation and their complexity. In *Handbook of Formal Argumentation*, chapter 14, pages 631–687. College Publications, 2018. Also appears in IfCoLog Journal of Logics and their Applications 4(8):2557–2622.

[DDK+23] Yannis Dimopoulos, Wolfgang Dvořák, Matthias König, Anna Rapberger, Markus Ulbricht, and Stefan Woltran. Sets attacking sets in abstract argumentation. In *Proceedings of the 21st International Workshop on Nonmonotonic Reasoning, NMR 2023*, volume 3464 of *CEUR Workshop Proceedings*, pages 22–31. CEUR-WS.org, 2023.

[DDK+24] Yannis Dimopoulos, Wolfgang Dvořák, Matthias König, Anna Rapberger, Markus Ulbricht, and Stefan Woltran. Redefining ABA+ semantics via abstract set-to-set attacks. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence, AAAI 2024*, pages 10493–10500. AAAI Press, 2024. doi: 10.1609/AAAI.V38I9.28918.

[DDLW15] Paul E. Dunne, Wolfgang Dvořák, Thomas Linsbichler, and Stefan Woltran. Characteristics of multiple viewpoints in abstract argumentation. *Artif. Intell.*, 228:153–178, 2015. doi: 10.1016/j.artint.2015.07.006.

[DFW19] Wolfgang Dvořák, Jorge Fandinno, and Stefan Woltran. On the expressive power of collective attacks. *Argument Comput.*, 10(2):191–230, 2019. doi: 10.3233/AAC-190457.

[DG12] Wolfgang Dvořák and Sarah Alice Gaggl. Incorporating stage semantics in the scc-recursive schema for argumentation semantics. In *Proceedings of the 14th International Workshop on Non-Monotonic Reasoning, NMR 2012*, 2012.

[DG16] Wolfgang Dvořák and Sarah Alice Gaggl. Stage semantics and the SCC-recursive schema for argumentation semantics. *J. Log. Comput.*, 26(4):1149–1202, 2016.

[DGW18] Wolfgang Dvořák, Alexander Greßler, and Stefan Woltran. Evaluating SETAFs via answer-set programming. In *Proceedings of the 2nd International Workshop on Systems and Algorithms for Formal Argumentation, SAFA 2018*, volume 2171 of *CEUR Workshop Proceedings*, pages 10–21. CEUR-WS.org, 2018.

[DHK+22] Wolfgang Dvořák, Markus Hecher, Matthias König, André Schidler, Stefan Szeider, and Stefan Woltran. Tractable abstract argumentation via

backdoor-treewidth. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence, AAAI 2022*, pages 5608–5615. AAAI Press, 2022. doi: 10.1609/aaai.v36i5.20501 .

[DHM+11] 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. doi: 10.1016/j.artint.2010.09.005.

[DJWW14] Wolfgang Dvořák, Matti Järvisalo, Johannes Peter Wallner, and Stefan Woltran. Complexity-sensitive decision procedures for abstract argumentation. *Artif. Intell.*, 206(0):53 – 78, 2014. doi: 10.1016/j.artint.2013.10.001.

[DKUW21] Wolfgang Dvořák, Matthias König, Markus Ulbricht, and Stefan Woltran. A reduct-driven study of argumentation frameworks with collective attacks. In *Proceedings of the 19th International Workshop on Non-Monotonic Reasoning, NMR 2021*, pages 285–294, 2021.

[DKUW22] Wolfgang Dvořák, Matthias König, Markus Ulbricht, and Stefan Woltran. Rediscovering argumentation principles utilizing collective attacks. In *Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning, KR 2022*, pages 122–131, 2022. doi: 10.24963/kr.2022/13.

[DKUW24] Wolfgang Dvořák, Matthias König, Markus Ulbricht, and Stefan Woltran. Principles and their computational consequences for argumentation frameworks with collective attacks. *J. Artif. Intell. Res.*, 79:69–136, 2024. doi: 10.1613/JAIR.1.14879.

[DKW21a] Wolfgang Dvořák, Matthias König, and Stefan Woltran. Graph-classes of argumentation frameworks with collective attacks. In *Proceedings of the 17th European Conference on Logics in Artificial Intelligence, JELIA 2021*, volume 12678 of *LNCS*, pages 3–17. Springer, 2021.

[DKW21b] Wolfgang Dvořák, Matthias König, and Stefan Woltran. On the complexity of preferred semantics in argumentation frameworks with bounded cycle length. In *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021*, pages 671–675, 2021. doi: 10.24963/kr.2021/67.

[DKW22a] Wolfgang Dvořák, Matthias König, and Stefan Woltran. Deletion-backdoors for argumentation frameworks with collective attacks. In *Proceedings of the 4th International Workshop on Systems and Algorithms for Formal Argumentation, SAFA 2022*, volume 3236 of *CEUR Workshop Proceedings*, pages 98–110. CEUR-WS.org, 2022.

[DKW22b] Wolfgang Dvořák, Matthias König, and Stefan Woltran. Treewidth for argumentation frameworks with collective attacks. In *Proceedings of the 9th International Conference on Computational Models of Argument, COMMA 2022*, pages 140–151. IOS Press, 2022.

[DKW23] Wolfgang Dvořák, Atefeh Keshavarzi Zafarghandi, and Stefan Woltran. Expressiveness of SETAFs and support-free ADFs under 3-valued semantics. *J. Appl. Non Class. Logics*, 33(3-4):298–327, 2023. doi: 10.1080/11663081.2023.2244361.

[DKW24] Wolfgang Dvořák, Matthias König, and Stefan Woltran. Parameterized complexity of abstract argumentation with collective attacks. *Argument & Computation*, 2024. Under review.

[DKZLW20] Martin Diller, Atefeh Keshavarzi Zafarghandi, Thomas Linsbichler, and Stefan Woltran. Investigating subclasses of abstract dialectical frameworks. *Argument & Computation*, 11:191–219, 2020. doi: 10.3233/AAC-190481.

[DOS12] Wolfgang Dvořák, Sebastian Ordyniak, and Stefan Szeider. Augmenting tractable fragments of abstract argumentation. *Artificial Intelligence*, 186(0):157–173, 2012. doi: 10.1016/j.artint.2012.03.002.

[DPW12] Wolfgang Dvořák, Reinhard Pichler, and Stefan Woltran. Towards fixed-parameter tractable algorithms for abstract argumentation. *Artif. Intell.*, 186:1 – 37, 2012. doi: 10.1016/j.artint.2012.03.005.

[DRW20] Wolfgang Dvořák, Anna Rapberger, and Stefan Woltran. Argumentation semantics under a claim-centric view: Properties, expressiveness and relation to SETAFs. In *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning, KR 2020*, pages 341–350, 2020. doi: 10.24963/kr.2020/35.

[DSW12] Wolfgang Dvořák, Stefan Szeider, and Stefan Woltran. Abstract argumentation via monadic second order logic. In *Proceeedings of the 6th International Conference on Scalable Uncertainty Management, SUM 2012*, volume 7520 of *LNCS*, pages 85–98. Springer, 2012. doi: 10.1007/978-3-642-33362-0_7.

[DT96] Yannis Dimopoulos and Alberto Torres. Graph theoretical structures in logic programs and default theories. *Theor. Comput. Sci.*, 170(1-2):209–244, 1996. doi: 10.1016/S0304-3975(96)80707-9.

[Dun95] 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. doi: 10.1016/0004-3702(94)00041-X.

[Dun07] Paul E. Dunne. Computational properties of argument systems satisfying graph-theoretic constraints. *Artif. Intell.*, 171(10-15):701–729, 2007. doi: 10.1016/j.artint.2007.03.006.

[Dun09] Paul E. Dunne. The computational complexity of ideal semantics. *Artif. Intell.*, 173(18):1559–1591, 2009. doi: 10.1016/j.artint.2009.09.001.

[Dun22] Paul E. Dunne. Well, to be honest, I wouldn't start from here at all. In *Proceedings of the 9th International Conference on Computational Models of Argument, COMMA 2022*, volume 353 of *FAIA*, pages 3–14. IOS Press, 2022. doi: 10.3233/FAIA220134.

[Dun24] Paul E. Dunne. Decidability in argumentation semantics. *Argument Comput.*, 15(2):191–204, 2024. doi: 10.3233/AAC-220020.

[DUW22] Wolfgang Dvořák, Markus Ulbricht, and Stefan Woltran. Recursion in abstract argumentation is hard - on the complexity of semantics based on weak admissibility. *J. Artif. Intell. Res.*, 74:1403–1447, 2022. doi: 10.1613/JAIR.1.13603.

[Dvo12] Wolfgang Dvořák. *Computational Aspects of Abstract Argumentation*. PhD thesis, Vienna University of Technology, Institute of Information Systems, 2012.

[DW10] Wolfgang Dvořák and Stefan Woltran. Complexity of semi-stable and stage semantics in argumentation frameworks. *Inf. Process. Lett.*, 110(11):425–430, 2010. doi: 10.1016/j.ipl.2010.04.005.

[DW11] Wolfgang Dvořák and Stefan Woltran. On the intertranslatability of argumentation semantics. *J. Artif. Intell. Res. (JAIR)*, 41:445–475, 2011.

[FB19] Giorgos Flouris and Antonis Bikakis. A comprehensive study of argumentation frameworks with sets of attacking arguments. *Int. J. Approx. Reason.*, 109:55–86, 2019. doi: 10.1016/j.ijar.2019.03.006.

[FHM19] Johannes K. Fichte, Markus Hecher, and Arne Meier. Counting complexity for reasoning in abstract argumentation. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence, AAAI 2019*, pages 2827–2834. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.33012827.

[FHMM21] Johannes Klaus Fichte, Markus Hecher, Yasir Mahmood, and Arne Meier. Decomposition-guided reductions for argumentation and treewidth. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence, IJCAI 2021*, pages 1880–1886, 2021. doi: 10.24963/ijcai.2021/259.

[FS15] Johannes Klaus Fichte and Stefan Szeider. Backdoors to tractable answer set programming. *Artif. Intell.*, 220:64–103, 2015. doi: 10.1016/j.artint.2014.12.001.

[FU21] Tom Friese and Markus Ulbricht. On the relationship of modularity notions in abstract argumentation. In *Proceedings of the 19th International Workshop on Non-Monotonic Reasoning, NMR 2021*, pages 51–60, 2021.

[Gab09] Dov M. Gabbay. Semantics for higher level attacks in extended argumentation frames part 1: Overview. *Stud Logica*, 93(2-3):357–381, 2009. doi: 10.1007/S11225-009-9211-4.

[GGS14] Georg Gottlob, Gianluigi Greco, and Francesco Scarcello. Treewidth and hypertree width. In *Tractability: Practical Approaches to Hard Problems*, pages 3–38. Cambridge University Press, 2014. doi: 10.1017/CBO9781139177801.002.

[GGST21] Dov Gabbay, Massimiliano Giacomin, Guillermo R. Simari, and Matthias Thimm, editors. *Handbook of Formal Argumentation*. College Publications, 2021.

[GLS01] Georg Gottlob, Nicola Leone, and Francesco Scarcello. Hypertree decompositions: A survey. In *Proceedings of the 26th International Symposium on Mathematical Foundations of Computer Science, MFCS 2001*, volume 2136 of *LNCS*, pages 37–57. Springer, 2001. doi: 10.1007/3-540-44683-4_5.

[GMAB04] Guido Governatori, Michael J. Maher, Grigoris Antoniou, and David Billington. Argumentation semantics for defeasible logic. *J. Log. Comput.*, 14(5):675–702, 2004. doi: 10.1093/logcom/14.5.675.

[GMO+14] Serge Gaspers, Neeldhara Misra, Sebastian Ordyniak, Stefan Szeider, and Stanislav Zivný. Backdoors into heterogeneous classes of SAT and CSP. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence, AAAI 2014*, pages 2652–2658. AAAI Press, 2014.

[GOS17] Serge Gaspers, Sebastian Ordyniak, and Stefan Szeider. Backdoor sets for CSP. In *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*, pages 137–157. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

[GRS21] Sarah Alice Gaggl, Sebastian Rudolph, and Hannes Straß. On the decomposition of abstract dialectical frameworks and the complexity of naive-based semantics. *J. Artif. Intell. Res.*, 70:1–64, 2021. doi: 10.1613/jair.1.11348.

[IP99] Russell Impagliazzo and Ramamohan Paturi. Complexity of k-SAT. In *Proceedings of the 14th Annual IEEE Conference on Computational Complexity, CCC 1999*, pages 237–240, 1999. doi: 10.1109/CCC.1999.766282.

[IPZ98] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which Problems Have Strongly Exponential Complexity? In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science, FOCS 1998*, pages 653–662, 1998.

[JPW09] Michael Jakl, Reinhard Pichler, and Stefan Woltran. Answer-set programming with bounded treewidth. In *Proceedings of the 21st International*

Joint Conference on Artificial Intelligence, IJCAI 2009, pages 816–822, 2009.

[Kar72] Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972.

[Klo94] Ton Kloks. *Treewidth, Computations and Approximations*, volume 842 of *LNCS*. Springer, 1994. doi: 10.1007/BFb0045375.

[Kön20] Matthias König. Graph-classes of argumentation frameworks with collective attacks. Master's thesis, TU Wien, 2020.

[KRU22] Matthias König, Anna Rapberger, and Markus Ulbricht. Just a matter of perspective. In *Proceedings of the 9th International Conference on Computational Models of Argument, COMMA 2022*, volume 353 of *FAIA*, pages 212–223. IOS Press, 2022. doi: 10.3233/FAIA220154.

[KvdTV18] Souhila Kaci, Leendert W. N. van der Torre, and Serena Villata. Preference in abstract argumentation. In *Proceedings of the 7th International Conference on Computational Models of Argument, COMMA 2018*, volume 305 of *FAIA*, pages 405–412. IOS Press, 2018. doi: 10.3233/978-1-61499-906-5-405.

[KVV22] Atefeh Keshavarzi Zafarghandi, Rineke Verbrugge, and Bart Verheij. Strong admissibility for abstract dialectical frameworks. *Argument Comput.*, 13(3):249–289, 2022. doi: 10.3233/AAC-210002.

[LAD+24] Francesco Leofante, Hamed Ayoobi, Adam Dejl, Gabriel Freedman, Deniz Gorur, Junqi Jiang, Guilherme Paulino-Passos, Antonio Rago, Anna Rapberger, Fabrizio Russo, Xiang Yin, Dekai Zhang, and Francesca Toni. Contestable AI needs computational argumentation. *CoRR*, abs/2405.10729, 2024. doi: 10.48550/ARXIV.2405.10729.

[Lin14] Thomas Linsbichler. Splitting abstract dialectical frameworks. In *Proceedings of the 5th International Conference on Computational Models of Argument, COMMA 2014*, volume 266 of *FAIA*, pages 357–368. IOS Press, 2014. doi: 10.3233/978-1-61499-436-7-357.

[LJK11] Bei Shui Liao, Li Jin, and Robert C. Koons. Dynamics of argumentation systems: A division-based method. *Artif. Intell.*, 175(11):1790–1814, 2011. doi: 10.1016/j.artint.2011.03.006.

[LPS16] Thomas Linsbichler, Jörg Pührer, and Hannes Strass. A uniform account of realizability in abstract argumentation. In *Proceedings of the 22nd European Conference on Artificial Intelligence ECAI 2016*, volume 285 of *FAIA*, pages 252–260. IOS Press, 2016. doi: 10.3233/978-1-61499-672-9-252.

[Mod09] Sanjay Modgil. Reasoning about preferences in argumentation frameworks. *Artif. Intell.*, 173(9-10):901–934, 2009. doi: 10.1016/J.ARTINT.2009.02.001.

[NP06a] Søren Holbech Nielsen and Simon Parsons. Computing preferred extensions for argumentation systems with sets of attacking arguments. In *Proceedings of the 1st International Conference on Computational Models of Argument, COMMA 2006*, volume 144 of *FAIA*, pages 97–108. IOS Press, 2006.

[NP06b] Søren Holbech Nielsen and Simon Parsons. A generalization of Dung's abstract framework for argumentation: Arguing with sets of attacking arguments. In *Proceedings of the 3rd International Workshop on Argumentation in Multi-Agent Systems, ArgMAS 2006*, volume 4766 of *LNCS*, pages 54–73. Springer, 2006. doi: 10.1007/978-3-540-75526-5_4.

[ON08] Nir Oren and Timothy J. Norman. Semantics for evidence-based argumentation. In *Proceedings of 2nd International Conference on Computational Models of Argument, COMMA 2008*, volume 172 of *FAIA*, pages 276–284. IOS Press, 2008.

[OSS21] Sebastian Ordyniak, André Schidler, and Stefan Szeider. Backdoor DNFs. In *Proceedings of Proceedings of the 30th International Joint Conference on Artificial Intelligence, IJCAI 2021*, pages 1403–1409. ijcai.org, 2021. doi: 10.24963/ijcai.2021/194.

[Pol16] Sylwia Polberg. Understanding the abstract dialectical framework. In *Proceedings of the 15th European Conference On Logics In Artificial Intelligence, JELIA 2016*, volume 10021 of *LNCS*, pages 430–446, 2016. doi: 10.1007/978-3-319-48758-8_28.

[Pol17] Sylwia Polberg. *Developing the Abstract Dialectical Framework*. PhD thesis, Vienna University of Technology, Institute of Information Systems, 2017.

[PW23] Andrei Popescu and Johannes Peter Wallner. Reasoning in assumption-based argumentation using tree-decompositions. In *Proceedings of the 18th European Conference on Logics in Artificial Intelligence, JELIA 2023*, volume 14281 of *LNCS*, pages 192–208. Springer, 2023. doi: 10.1007/978-3-031-43619-2_14.

[RS86] Neil Robertson and P. D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *J. Algorithms*, 7(3):309–322, 1986. doi: 10.1016/0196-6774(86)90023-4.

[RS09] Iyad Rahwan and Guillermo R. Simari. *Argumentation in Artificial Intelligence*. Springer, 2009.

[RST99] Neil Robertson, P. D. Seymour, and Robin Thomas. Permanents, pfaffian orientations, and even directed circuits. *Annals of Mathematics*, 150(3):929–975, 1999.

[VBvdT11] Serena Villata, Guido Boella, and Leendert W. N. van der Torre. Attack semantics for abstract argumentation. In *Proceedings of Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI 2011*, pages 406–413. IJCAI/AAAI, 2011. doi: 10.5591/978-1-57735-516-8/IJCAI11-076.

[vdTV17] Leon van der Torre and Srdjan Vesic. The principle-based approach to abstract argumentation semantics. *FLAP*, 4(8):2735–2778, 2017.

[Ver96] Bart Verheij. *Rules, reasons, arguments: formal studies of argumentation and defeat*. PhD thesis, Maastricht University, Netherlands, January 1996. doi: 10.26481/dis.19961205hv.

[WCG09] Yining Wu, Martin Caminada, and Dov M. Gabbay. Complete extensions in argumentation coincide with 3-valued stable models in logic programming. *Studia Logica*, 93(2-3):383–403, 2009. doi: 10.1007/s11225-009-9210-5.

[YCQ+21] Liuwen Yu, Dongheng Chen, Lisha Qiao, Yiqi Shen, and Leendert van der Torre. A principle-based analysis of abstract agent argumentation semantics. In *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021*, pages 629–640, 2021. doi: 10.24963/kr.2021/60.

[YVC20] Bruno Yun, Srdjan Vesic, and Madalina Croitoru. Sets of attacking arguments for inconsistent datalog knowledge bases. In *Proceedings of the 8th International Conference on Computational Models of Argument, COMMA 2020*, volume 326 of *FAIA*, pages 419–430. IOS Press, 2020. doi: 10.3233/FAIA200526.