Ph.D. Program in Computer Science - XXXIII cycle

Doctoral Thesis

Handling Dynamic Aspects of Argumentation

Carlo Taticchi

Supervisor:
Prof. Stefano Bistarelli
University of Perugia, Italy

Internal advisor:
Prof. Luca Aceto
Reykjavik University, Iceland

April 2021

GSSI Gran Sasso Science Institute
Viale Francesco Crispi, 7 - 67100 L'Aquila - Italy

Abstract

Many applications in the field of artificial intelligence aim to reproduce human behaviour and reasoning in order to allow machines to think and act accordingly. One of the main challenges in this sense is to provide methodologies and tools for expressing a certain kind of knowledge in a formal way so that the machines can use it for reasoning and infer new information. Argumentation pursues the objective of studying how conclusions can be reached, starting from a set of assumptions, through a process of logical reasoning. This process is very similar to the human way of thinking and involves features which can be traced to the conducting of a dialogue between two (or more) people. Indeed, in the most common form of argumentation, a part (which can be, for instance, an interlocutor) in a debate tries to affirm some belief and defends it from the attacks of other parts. Argumentation Theory provides formal models for representing and evaluating arguments that interact with each other and, in particular, Abstract Argumentation Frameworks (AFs) are used to study the acceptability of arguments. Solving an abstract argumentation problem means to identify components of the debate (called extensions) which share certain properties and validate the same proposition, according to a specified semantics (which is a selection criterion).

Besides the static representation of conflicts between different parts, AFs can also handle the evolution of situations in which instances of particular problems undergo changes; variations on the underlying information can be interpreted as modifications in the corresponding framework. Practical implementation of argumentation-based systems should take into account the various changes that are usually introduced in a given knowledge base. The case in which all the information is already known to every party at the beginning of the interaction and thus the conclusion can be drawn without any further step is, indeed, unlikely. Moreover, due to the dynamic nature of certain problems, settling for a solution (in a particular AF) could not be sufficient to guarantee a good outcome in case the problem evolves.

In this thesis, we study the dynamics of AFs from multiple perspectives with the purpose of better understanding how dynamic (and concurrent) processes can be handled in the context of argumentation. In this sense, we provide theoretical results, algorithms and tools which can be useful in many dynamic aspects of argumentation. Before arriving to define our concurrent language, we set the theoretical results we need to work with dynamics aspect of argumentation.

First, we consider aspects of argumentation oriented towards reasoning tasks, as operations that preserve the semantics and ranking functions for the arguments. We start

by investigating some of the problems that can be instantiated in argumentation-based systems and involve a reasoning process to accomplish a given task. As one can expect, introducing changes might lead to obtaining different semantics for the considered AF. We therefore study operations which leave the set of extensions unchanged and we arrive to define a set of operators for which the semantics is an invariant. We also derive a notion of robustness representing the number of syntactical changes an AF can withstand before changing its semantics.

When the number of arguments to take into account is very large, restricting to the set of accepted arguments may still not be sufficient to make a decision concerning a certain problem. Using ranking-based semantics, instead, it is possible to refine the acceptability level of arguments in an AF by sorting them from the best to the worst, according to some evaluation method. We give our contribution to the field by devising a ranking-based semantics that relies on power indexes for estimating the contribution a certain argument brings to each extension. Aware of the fact that abstract frameworks are not sufficient to precisely instantiate problems coming from the real world, for which the structure of the arguments as well as the type of relations between them should be considered, we study the behaviour of ranking-based semantics in a setting where AFs are semi-structured: we use claim-augmented frameworks in which arguments are explicitly associated with the claims they stand for. The work is accompanied by a study of the properties that characterise the various ranking functions.

Then we set the basis for working with the acceptability of arguments both for the classical and the weighted case through four-state labelling-based semantics. Whatever the level of abstraction, the central task in applications that take advantage of argumentation theory is the identification of good arguments: the first step to (be able to) draw conclusions in a controversial situation or when the information is only partial is to separate an acceptable outcome from the rest of non-feasible solutions. Between classical semantics which only distinguish acceptable arguments from rejected ones and ranking-based semantics that just sort the arguments from the best to the worst, labelling-based semantics allow for discriminating up to three statuses of acceptance by assigning labels to the arguments in an AF. In this thesis, we adapt the classical three-state labelling semantics to work with the extra label that we use to mark "unused" arguments in the framework. We also consider the weighted case, in which AFs are extended with values on the attack relations representing the strength of the attacks themselves and we provide labelling functions that generalise the classical approach.

We continue proposing a language able to describe the interactions between debating agents and that uses argumentation as an embedded reasoning engine. Logical frameworks for argumentation have been introduced to fulfil the operational tasks related to

the study of dynamics in AFs, such as the description of AFs, the specification of modifications, and the search for sets of "good" arguments. Since none of these approaches consider the possibility of having concurrent interactions or agents arguing with each other, we introduce a concurrent language for argumentation (ConArg_lang) that aims to be used also for modelling different types of interaction between agents (as negotiations, persuasion, deliberation and dialogues). In particular, our language allows for modelling concurrent processes, inspired by notions such as the *Ask-and-Tell constraint system* and using AFs as a centralised store. The language is thus endowed with primitives for the specification of interaction between agents through the fundamental operations of adding (or removing) and checking arguments and attacks. We also propose a set of AGM-style operations that allow for modifying an AF (which constitutes the shared memory our agents access to communicate) and changing the status of its arguments to allow the implementation of more complex operations, like negotiation and the other forms of dialogues.

Finally, we accompany all our theoretical results with working implementations of tools that are used to both better study the problems we face and prepare the ground for practical applications. The core of the suite consists of a constraint-based solver for AFs, able to compute the set of extensions and test the acceptability of the arguments. The solver can work with classical as well as extended AFs, like weighted and probabilistic ones. Among other functionalities, we provide the possibility to rank the arguments of a given framework using power indexes from cooperative game theory. The suite is also endowed with a web interface in which graphical representations of AFs, labelling semantics and ranking of arguments can be visualised.

To my parents

Acknowledgements

While I was working on this thesis, I had the occasion of meeting many amazing people, and now I'm happy to have the opportunity to thank them properly. First of all, I want to tank Prof. Stefano Bistarelli, who in the last years has not only been my supervisor, but also a mentor and a guide. Tirelessly, he has helped me to grow both academically and as a person, always finding the right words to motivate me and push me to move forward during the most challenging moments. My gratitude also goes to my advisor, Prof. Luca Aceto; His passion for research and his attention to teaching have been a source of inspiration for me. I also want to thank Prof. Francesco Santini for his valuable contribution to my work and for having been able to lighten the mood in difficult moments. A special mention to Dr. Fabio Rossi for his irreplaceable help from the DMI laboratory. Tanks also to Prof. Stefan Woltran and Dr. Wolfgang Dvořák, which made my time in Vienna fun as well as productive.

Aside from the hard work, my life as a Ph.D. student was surrounded by pleasant moments spent with the people I met and the friends who shared this path with me. Life in L'Aquila would have been much less enjoyable and exciting without all the people of GSSI. Among them, I want to thank Sara and Giovanna for their help and, above all, for their friendship. Continuing with fellow Ph.D. students, thanks to Ivan and Francesco (Galt), that, albeit from different places, shared the joys and sorrows of the doctorate with me. At this point, I can't help but thank all the other friends of the "Sezione (L)aurea(ti)" group: Luca, Matteo, Emanuele and Francesco (Fra). You guys have really made a difference.

Concluding with the dearest affections, a heartfelt thanks to my family for their unconditional support. Thanks to Francesca for always being present and sharing every moment with me, good or bad it was. Special thanks also to Francesco and Brunella.

Contents

Chapter 1

Introduction

"*Power is in tearing human minds to pieces and putting them together again in new shapes of your own choosing.*"
– George Orwell

Many applications in the field of artificial intelligence aim to reproduce human behaviour and reasoning in order to allow machines to think and act accordingly. One of the main challenges in this sense is to provide methodologies and tools for expressing a certain kind of knowledge in a formal way so that the machines can use it for reasoning and infer new information. Argumentation pursues the objective of studying how conclusions can be reached, starting from a set of assumptions, through a process of logical reasoning. This process is very similar to the human way of thinking and involves features which can be traced to the conducting of a dialogue between two (or more) people. Indeed, in the most common form of argumentation, a part (which can be, for instance, an interlocutor) in a debate tries to affirm some belief and defends it from the attacks of other parts. Argumentation Theory provides formal models for representing and evaluating arguments that interact with each other. In his seminal work [85], Dung introduces a representation for Argumentation Frameworks in which arguments are abstract, i.e., their internal structure as well as their origin is left unspecified. Abstract Argumentation Frameworks (AFs) have been widely studied from the point of view of the acceptability of arguments. An AF is represented by a pair $\langle \mathcal{A}, \mathcal{R} \rangle$ consisting of a set of arguments and a binary relation of attack defined among them. Solving an abstract argumentation problem means to identify components of the debate (called extensions) which share certain properties and validate the same proposition, according to a specified semantics (which is a selections criterion). For instance, the requirements for a set of arguments could be that nodes inside the set do not attack each other, or that they are capable of

defending themselves against attacks from the outside. This kind of abstraction makes AFs suitable for representing all sort of problems in which the goal is to achieve a conclusion (find a solution, make a decision, reach an agreement, etc.) based on some known facts.

Several authors have investigated the dynamics of AFs, taking into account both theoretical [21, 59, 146] and computational aspects. Besides the static representation of conflicts between different parts, it is important to also provide AFs with the basis for handling the evolution of situations in which instances of particular problems undergo changes; variations on the underlying information can be interpreted as modifications in the corresponding framework. Such modifications are performed through operations of addition or subtraction of arguments and attacks. Argumentation processes model the interaction that takes place in systems where information is controversial, inconsistent and/or incomplete, for example between individuals involved in some kind of dialogue, in recommender/persuasive systems designed for the most varied domains (e.g., legal and medical), in technologies for explainable artificial intelligence, and in any application which may benefit from the representation power of AFs and the automatic reasoning through non-monotone formalisms. In this scenario, any practical implementation of argumentation-based systems should take into account the various changes that are usually introduced in a given knowledge base. The case in which all the information is already known to every party at the beginning of the interaction and thus the conclusion can be drawn without any further step is, indeed, unlikely. Moreover, due to the dynamic nature of certain problems, settling for a solution (in a particular AF) could not be sufficient to guarantee a good outcome in case the problem evolves. For example, think about a politician redacting a public speech. With the information she has at the moment of preparation, she can arrive to conclude that her harangue is flawless. However, it may happen that new arguments, presented at the time of the speech, affect some critical point, changing the meaning (and the outcome) of the speech itself.

This leads to an increasing necessity of tools for handling dynamics of AFs, also considering different perspectives. From the point of view of modifications, two main approaches can be examined. First, one can focus on introducing changes on the knowledge base (most of the time represented by an AF, possibly extended with additional features like weights) dealing only with syntactic constraints [84] and without worrying about the consequences that such modification brings in terms of acceptance. On the other hand, it is also possible to study the impact of such changes with respect to the outcome of the argumentation system (that is the computed set of extensions). For instance, the task of adding an argument to an existing AF [70] may result in the change of the acceptance statuses of all other nodes, and thus the distortion of the meaning of the speech. In this sense, precise operations can be performed to ensure that the new information will

always be accepted [23] or that will not affect the semantics [51]. The implementation of argumentation processes, then, requires systems at the base that enable the communication between intelligent agents [87] and that provide operations working on both syntactic and semantic level.

In this thesis, we study the dynamics of AFs from multiple perspectives:

- first, we consider aspects of argumentation oriented towards robustness, as operations that preserve the semantics;
- we also study of a ranking function that sorts the arguments of an AF (in order to help selection of arguments during a dynamic revision process) through the use of power-indexes;
- then we set the basis for working with the acceptability of arguments both for the classical and the weighted case through four-state labelling-based semantics, adding finer grain selection methodologies in the revision process;
- we continue proposing a language able to describe the interactions between debating agents and that uses argumentation as an embedded reasoning engine;
- finally, we describe the various tools we devised and implemented to aid the research in this filed.

We aim to capture all the features needed for handling dynamic (and concurrent) processes involving argumentation, so as to provide a tool as effective as possible and which can be useful in the many fields resorting to artificial intelligence. Therefore, instead of directly focusing on the definition of our language, we pave the way to best formulate the theoretical tools we need by considering different aspects of argumentation, either explicitly related to dynamics or that serve as foundations for further development.

In detail, we start by investigating some of the problems that can be instantiated in argumentation-based systems and involve a reasoning process to accomplish a given task. For example, we give a notion of robustness (the ability of an AF to resist modifications) to cope with the problem of how to introduce changes in a given AF without modifying its set of extensions for a certain semantics. As one can expect, introducing changes might lead to obtaining different semantics for the considered AF. Modifications on an AF are performed by specifically designed operations, which we can divide in two types: the ones which change the semantics of the system (like those in [69]) and the ones that do not (as examined, for example, in [145]). We study this latter type of operations, which leave the semantics unchanged, reducing to the case of addition (or subtraction) of an attack and

we arrive to define a set of operators for which the semantics is an invariant. In this way, it would also be possible to locate sets of arguments which are essential to preserving the semantics. Every change inside those sets modifies the semantics, but changes outside do not cause any alteration. By removing the non-meaningful part of AFs, it is possible to obtain equivalent frameworks for which the computation of extensions is faster, especially for checking credulous/sceptical acceptance of arguments.

When the number of arguments to take into account is very large, restricting to the set of accepted arguments may still not be sufficient to make a decision concerning a certain problem. Imagine the situation in which a doctor (or an intelligent system acting in her place) has to recommend a cough medicine in a pool of a hundred possible choices. After evaluating the pros and cons of each prescription according to the patient's condition and symptoms, the doctor discards all the medications which present contraindications, remaining with a dozen acceptable choices. At this point, resorting only to the notions of classical argumentation semantics, the doctor would have no option but to choose at random. Using ranking-based semantics [7], instead, it is possible to refine the acceptability level of arguments in an AF by sorting them from the best to the worst, according to some evaluation method. We give our contribution to the field by devising a ranking-based semantics that relies on power indexes [108] for estimating the contribution a certain argument brings to each extension, i.e., how useful it is to accept that argument. Indeed, power indexes (of which the famous Shapley Value [150] is an example) are used in cooperative game theory to quantify the contribution of a player in a coalition.

Aware of the fact that abstract frameworks are not sufficient to precisely instantiate problems coming from the real world, for which the structure of the arguments as well as the type of relations between them should be considered [142], we study the behaviour of ranking-based semantics in a setting where AFs are semi-structured: we use claim-augmented frameworks [90] in which arguments are explicitly associated with the claims they stand for. In particular, we devise a method for lifting an argument-ranking to the level of the claims, avoiding the need to define an *ad hoc* function that specifically works with claims. The work is accompanied by a study of the properties that characterise the various ranking functions. For instance, one of the basic properties that any reasonable ranking-based semantics should satisfy is the "abstraction", meaning the ranking has to be independent of the name of the arguments. We show that using our lifting function, the abstraction property is satisfied with respect to the ranking on the claim if it is satisfied in the first place on the level of the arguments. For other properties we specify under which conditions they are preserved after the lifting; in this way, we allow all the ranking semantics defined in the literature to be used also for ranking claims, taking a first step towards structured argumentation.

Whatever the level of abstraction, the central task in applications that take advantage of argumentation theory is the identification of good arguments: the first step to (be able to) draw conclusions in a controversial situation or when the information is only partial is to separate an acceptable outcome from the rest of non-feasible solutions. Between classical semantics which only distinguish acceptable arguments from rejected ones and ranking-based semantics that just sort the arguments from the best to the worst, labelling-based semantics [65, 105] allow for discriminating up to four statuses of acceptance by assigning labels to the arguments in an AF. While the set of accepted arguments, which are assigned an in label remains the same as in Dung's approach, the rejected arguments can receive different labels. Only those directly attacked by an in argument, and therefore properly defeated, are marked as out; the rest of the arguments, which neither can be accepted nor are rejected, are assigned the undec label. Moreover, it is possible to left arguments which are not of interest in a particular situation with an empty label, allowing one to exclude irrelevant information during a reasoning process, without the need to retract arguments from the AF. In this thesis, we adapt the classical three-state labelling semantics to work with the extra empty label. We also consider the weighted case, in which AFs are extended with values on the attack relations representing the strength of the attacks themselves and we provide labelling functions that generalise the classical approach.

Logical frameworks for argumentation, like the ones presented in [84, 87], have been introduced to fulfil the operational tasks related to the study of dynamics in AFs, such as the description of AFs, the specification of modifications, and the search for sets of "good" arguments. Although some of these languages could be exploited to implement applications based on argumentation, for instance to model debates among political opponents, none of them consider the possibility of having concurrent interactions or agents arguing with each other. The lack of approaches considering concurrency represents a significant gap between the reasoning capacities of AFs and their possible use in real-life tools. As an example, consider the situation in which two debating agents share a knowledge base, represented by an AF, and both of them want to update it with new information, in such a way that the new beliefs are consistent with the previous ones. The agents can act independently and simultaneously. Similarly to what happens in concurrent programming, if no synchronization mechanism is taken into account, the result of update or revision can be unpredictable and can also lead to the introduction of inconsistencies.

Motivated by the above considerations, we introduce a concurrent language for argumentation (ConArg_lang) that aims to be used also for modelling different types of interaction between agents (as negotiations, persuasion, deliberation and dialogues). In particular, our language allows for modelling concurrent processes, inspired by notions such as the *Ask-and-Tell constraint system* [147], and using AFs as a centralised store.

The language is thus endowed with primitives for the specification of interaction between agents through the fundamental operations of adding (or removing) and checking arguments and attacks. Besides specifying a logic for argument interaction, our language can model debating agents (e.g., chatbots) that take part in a conversation and provide arguments. The possibility of defining blocking and atomic operations gives to the language the ability to describe scenarios where the interactions between agents need to be synchronised, e.g., a negotiation in which agents have to wait for the others before responding. Alchourrón, Gärdenfors, and Makinson (AGM) theory [2] gives operations (like expansion, contraction, revision) for updating and revising beliefs on a knowledge base. We also propose a set of AGM-style operations that allow for modifying an AF (which constitutes the shared memory our agents access to communicate) and changing the status of its arguments to allow the implementation of more complex operations, like negotiation and the other forms of dialogues.

Finally, we accompany all our theoretical results with working implementations of tools that are used to both better study the problems we face and prepare the ground for practical applications. All the tools are collected in a suite available online[1]. The core of the suite consists of a constraint-based solver for AFs, able to compute the set of extensions and test the (credulous/sceptical) acceptability of the arguments. The solver can work with classical as well as extended AFs, like weighted and probabilistic ones. Among other functionalities, we provide the possibility to rank the arguments of a given framework using power indexes from cooperative game theory. The suite is also endowed with a web interface in which graphical representations of AFs, labelling semantics and ranking of arguments can be visualised.

1.1 List of Original Publications

In the following we list the original publications which constitute the core of this thesis. We use papers from I to III in Chapter 3, where we discuss the notion of robustness.

I Carlo Taticchi. **A Study of Robustness in Abstract Argumentation Frameworks.** In Viviana Mascardi and Ilaria Torre, editors, *Proceedings of the Doctoral Consortium of AI*IA 2016 Co-Located with the 15th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2016), Genova, Italy, November 29, 2016*, volume 1769 of CEUR Workshop Proceedings, pages 11–16. CEUR-WS.org, 2016.

[1] ConArg website: `http://www.dmi.unipg.it/conarg/`.

II Stefano Bistarelli, Francesco Santini, and Carlo Taticchi. **On Looking for Invariant Operators in Argumentation Semantics.** In Keith Brawner and Vasile Rus, editors, *Proceedings of the Thirty-First International Florida Artificial Intelligence Research Society Conference, FLAIRS 2018, Melbourne, Florida, USA. May 21–23 2018*, pages 537–540. AAAI Press, 2018.

III Stefano Bistarelli, Francesco Santini, and Carlo Taticchi. **Local Expansion Invariant Operators in Argumentation Semantics.** In Beishui Liao, Thomas Ågotnes and Yì N. Wáng, editors, *Dynamics, Uncertainty and Reasoning, The Second Chinese Conference on Logic and Argumentation, CLAR 2018, Hangzhou, China, 16–17 June 2018*, pages 45–62. Springer, 2018.

Papers IV to VIII are the basis for Chapter 4 on ranking-based semantics.

IV Stefano Bistarelli, Paolo Giuliodori, Francesco Santini, and Carlo Taticchi. **A Cooperative-game Approach to Share Acceptability and Rank Arguments.** In Pierpaolo Dondio and Luca Longo, editors, *Proceedings of the 2nd Workshop on Advances In Argumentation In Artificial Intelligence, co-located with XVII International Conference of the Italian Association for Artificial Intelligence, AI^3@AI*IA 2018, 20–23 November 2018, Trento, Italy*, volume 2296 of CEUR Workshop Proceedings, pages 86–90. CEUR-WS.org, 2018.

V Carlo Taticchi. **Power Index-Based Semantics for Ranking Arguments in Abstract Argumentation Frameworks: An Overview.** In Mario Alviano, Gianluigi Greco, Marco Maratea, and Francesco Scarcello, editors, *Discussion and Doctoral Consortium Papers of AI*IA 2019 - 18th International Conference of the Italian Association for Artificial Intelligence, Rende, Italy, November 19–22, 2019*, volume 2495 of CEUR Workshop Proceedings, pages 113–118. CEUR-WS.org, 2019.

VI Stefano Bistarelli, Wolfgang Dvorák, Carlo Taticchi, and Stefan Woltran. **Ranking-Based Semantics from the Perspective of Claims.** In Henry Prakken, Stefano Bistarelli, Francesco Santini, and Carlo Taticchi, editors, *Computational Models of Argument - Proceedings of COMMA 2020, Perugia, Italy, September 4–11*, 2020, volume 326 of Frontiers in Artificial Intelligence and Applications, pages 111–122. IOS Press, 2020.

VII Stefano Bistarelli, Francesco Faloci, and Carlo Taticchi. **Implementing Ranking-Based Semantics in ConArg.** In *31st IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2019, Portland, OR, USA, November 4–6, 2019*, pages 1180–1187. IEEE, 2019.

VIII Stefano Bistarelli and Carlo Taticchi. **Power index-based semantics for ranking arguments in abstract argumentation frameworks.** *Intell. Artif.*, 13(2):137–154, 2019.

In Chapter 5 we use papers IX and X to present characterisations of AFS.

IX Stefano Bistarelli and Carlo Taticchi. **Preliminary Study on Reinstatement Labelling for Weighted Argumentation Frameworks.** In Francesco Santini and Alice Toniolo, editors, *Proceedings of the 3rd Workshop on Advances In Argumentation In Artificial Intelligence Co-Located with the 18th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2019), Rende, Italy, November 19–22, 2019*, volume 2528 of CEUR Workshop Proceedings, pages 45–49. CEUR-WS.org, 2019.

X Stefano Bistarelli and Carlo Taticchi. **A Labelling Semantics for Weighted Argumentation Frameworks.** In Francesco Calimeri, Simona Perri, and Ester Zumpano, editors, *Proceedings of the 35th Italian Conference on Computational Logic - CILC 2020, Rende, Italy, October 13–15, 2020*, volume 2710 of CEUR Workshop Proceedings, pages 263–277. CEUR-WS.org, 2020.

Chapter 6 refers to papers XI and XII which study the ConArg_lang language.

XI Stefano Bistarelli and Carlo Taticchi. **A Concurrent Language for Argumentation: Preliminary Notes.** In *Proceedings of the Workshop on Recent Developments on the Design and Implementation of Programming Languages (DIP2020)*, to appear. OASICS, 2020.

XII Stefano Bistarelli and Carlo Taticchi. **A Concurrent Language for Argumentation**. In Bettina Fazzinga, Filippo Furfaro, and Francesco Parisi, editors, *Proceedings of the Workshop on Advances In Argumentation In Artificial Intelligence 2020 Co-Located with the 19th International Conference of the Italian Association for Artificial Intelligence (AIxIA 2020), Online, November 25–26, 2020*, volume 2777 of CEUR Workshop Proceedings, pages 75–89. CEUR-WS.org, 2020.

Finally, Chapter 7 collects the tools of papers from XIII to XVII.

XIII Stefano Bistarelli, Fabio Rossi, Francesco Santini, and Carlo Taticchi. **Towards visualising security with arguments.** In Davide Ancona, Marco Maratea, and

Viviana Mascardi, editors, *Proceedings of the 30th Italian Conference on Computational Logic, Genova, Italy, July 1–3, 2015*, volume 1459 of CEUR Workshop Proceedings, pages 197–201. CEUR-WS.org, 2015.

XIV Stefano Bistarelli, Theofrastos Mantadelis, Francesco Santini, and Carlo Taticchi. **Using MetaProbLog and ConArg to compute Probabilistic Argumentation Frameworks.** In Pierpaolo Dondio and Luca Longo, editors, *Proceedings of the 2nd Workshop on Advances In Argumentation In Artificial Intelligence, co-located with XVII International Conference of the Italian Association for Artificial Intelligence, AI^3@AI*IA 2018, 20–23 November 2018, Trento, Italy*, volume 2296 of CEUR Workshop Proceedings, pages 6–10. CEUR-WS.org, 2018.

XV Stefano Bistarelli, Theofrastos Mantadelis, Francesco Santini, and Carlo Taticchi. **Probabilistic Argumentation Frameworks with MetaProbLog and ConArg.** In Lefteri H. Tsoukalas, Éric Grégoire, and Miltiadis Alamaniotis, editors, *IEEE 30th International Conference on Tools with Artificial Intelligence, ICTAI*, pages 675–679. IEEE, 2018.

XVI Stefano Bistarelli, Francesco Faloci, Francesco Santini, and Carlo Taticchi. **Studying Dynamics in Argumentation with Rob.** In Sanjay Modgil, Katarzyna Budzynska, and John Lawrence, editors, *Computational Models of Argument - Proceedings of COMMA 2018, Warsaw, Poland, 12–14 September 2018*, volume 305 of Frontiers in Artificial Intelligence and Applications, pages 451–452. IOS Press, 2018.

XVII Stefano Bistarelli, Francesco Faloci, Francesco Santini, and Carlo Taticchi. **A Tool For Ranking Arguments Through Voting-Games Power Indexes.** In Alberto Casagrande and Eugenio G. Omodeo, editors, *Proceedings of the 34th Italian Conference on Computational Logic, Trieste, Italy, June 19–21, 2019*, volume 2396 of CEUR Workshop Proceedings, pages 193–201. CEUR-WS.org, 2019.

1.2 Organization of the Thesis

The rest of this thesis is structured as follows. Chapter 2 contains the background notions of argumentation theory, game theory and multi-agent systems that we are going to use in the rest of the work. In Chapter 3 we introduce invariant operators for dealing with changes in AFs, fully covered in papers [51, 52]. In Chapter 4 we present a methodology for ranking arguments which relies on power indexes, discussed in [34, 46, 55, 154]. We also consider semi-structured AFs, studying a lifting of the ranking from the level of arguments to the level of claims [56]. In Chapter 5 we characterise important aspects of

AFs: first, we map the classical semantics on a four-state labelling, as depicted in [37], and then we focus on weighted AFs, describing the specially designed labelling functions introduced in [35, 38]. Chapter 6 is devoted to the description of the concurrent argumentation language devised in [36, 37], in which we also discuss the link with belief revision. Chapter 7 describes the tools we developed contextually to our studies, covering various aspects of argumentation, including insights from papers on dynamics [47], weighted labelling [38], semantics of probabilistic AFs [48, 49], ranking-based semantics [54], applications in the field of security [41] and the study of robustness [45, 153]. The thesis finds its conclusion in Chapter 8, where we also discuss further directions for our work.

Chapter 2

Preliminaries

In this chapter, we briefly recall the basic concepts of different topics we refer to in our thesis. First of all, we give the fundamental definition for Abstract Argumentation Framework and some of its declinations (where arguments and attacks are enriched with additional information), namely weighted AFs, probabilistic AFs and claim-augmented Frameworks. The latter formalism represents a middle ground between abstract and structured argumentation, since it maintain the information about which claim is supported by which argument. For all the provided representations of AF we show how arguments are evaluated according to different (type of) semantics, also taking into account the role of ranking-based semantics and argumentation-based multi-agent systems. To further develop in this direction, we provide the definitions of some power indexes used in cooperative game theory to quantify the contribution of players in a coalition. Finally, we describe the features that a formal language requires for handling interactions in concurrent systems, using Communicating Sequential Processes (CSP) [101], the Calculus of Communicating Systems (CCS) [127] and Concurrent Constraint Programming (CC) [147] as examples.

2.1 Argumentation Theory

Argumentation is an interdisciplinary field that aims to understand and model the human natural fashion of reasoning. In Artificial Intelligence, argumentation theory allows one to deal with uncertainty in non-monotonic (defeasible) reasoning, and it is used to give a qualitative, logical evaluation to sets of interacting arguments, called extensions. Argumentation deals with both the problems of representing knowledge and deriving information from it, using inference and logic to draw conclusions in a fashion really close to the human way of reasoning. For this, Argumentation can be applied in a wide area of

different disciplines concerning civil debate, dialogue, conversation and persuasion, and can be considered the means by which people protect their beliefs. By neglecting the internal structure of each argument (e.g., *premises* and a *claim*), the framework becomes "abstract", that is we are not interested in the meaning of arguments any more, but we just focus on their relations and we look for general properties. In his seminal paper [85], Dung defines the building blocks of abstract argumentation.

Definition 2.1 (AFs)**.** Let U be the set of all possible arguments[2], which we refer to as the "universe". An Abstract Argumentation Framework is a pair $\langle Arg, R\rangle$ where $Arg \subseteq U$ is a set of instantiated arguments and R is a binary relation on Arg representing instantiated attacks.

AFs can be represented through directed graphs, that we depict using the standard conventions. For two arguments $a, b \in Arg$, the notation $(a, b) \in R$ (or, equivalently, $a \to b$) represents an attack directed from a against b.

Definition 2.2 (Acceptable Argument)**.** Given an AF $F = \langle A, R\rangle$, an argument $a \in A$ is acceptable with respect to $D \subseteq A$ if and only if $\forall b \in A$ such that $(b, a) \in R$, $\exists c \in D$ such that $(c, b) \in R$, and we say that b is **defended** from D.

Given an argument framework F we use A_F to refer to the arguments of F and R_F to refer to the attack relation of F. We define the sets of arguments that attack (and that are attacked by) another argument as follows.

Definition 2.3 (Attacks)**.** Let F $= \langle A, R\rangle$ be an AF, $a \in A$ and $S \subseteq A$. We define the sets $a_F^+ = \{b \in A \mid (a, b) \in R\}$, $a_F^- = \{b \in A \mid (b, a) \in R\}$, $S_F^+ = \bigcup_{a\in S} a_F^+$ and $S_F^- = \bigcup_{a\in S} a_F^-$ (we will omit the subscript F when it is clear from the context).

By using the notion of defence as a criterion for distinguishing acceptable arguments in the framework, one can further refine the set of selected "good" arguments. The goal is to establish which are the acceptable arguments according to a certain semantics, namely a selection criterion. Non-accepted arguments are rejected. Different kinds of semantics have been introduced [15, 85] that reflect qualities which are likely to be desirable for "good" subsets of arguments. We first give the definition for the extension-based semantics (also referred to as Dung semantics), namely admissible, complete, stable, semi-stable, preferred, and grounded semantics (denoted with *adm*, *com*, *stb*, *sst*, *prf* and *gde*, respectively, and generically with σ).

Definition 2.4 (Extension-based semantics)**.** Let $F = \langle Arg, R\rangle$ be an AF. A set $E \subseteq Arg$ is conflict-free in F, denoted $E \in S_{cf}(F)$, if and only if there are no $a, b \in E$ such that $(a, b) \in R$. For $E \in S_{cf}(F)$ we have that:

[2]The set U is not present in the original definition by Dung and we introduce it for our convenience.

- $E \in S_{adm}(F)$ if each $a \in E$ is defended by E;
- $E \in S_{com}(F)$ if $E \in S_{adm}(F)$ and $\forall a \in Arg$ defended by E, $a \in E$;
- $E \in S_{stb}(F)$ if $\forall a \in Arg \setminus E$, $\exists b \in E$ such that $(b, a) \in R$;
- $E \in S_{sst}(F)$ if $E \in S_{com}(F)$ and $E \cup E^+$ is maximal;
- $E \in S_{prf}(F)$ if $E \in S_{adm}(F)$ and E is maximal;
- $E \in S_{gde}(F)$ if $E \in S_{com}(F)$ and E is minimal.

Moreover, if E satisfies one of the above properties for a certain semantics, we say that E is an extension for that semantics (for example, if $E \in S_{adm}(F)$ we say that E is an admissible extension).

A partial order can be defined among the set of extensions for the different semantics. In detail, we know that $S_{stb}(F) \subseteq S_{prf}(F) \subseteq S_{com}(F) \subseteq S_{adm}(F) \subseteq S_{cf}(F)$ and $S_{gde}(F) \subseteq S_{com}(F)$. The grounded semantics, in particular, coincides with the set of arguments sceptically accepted by the complete ones.

Definition 2.5 (Standard semantic equivalence [131])**.** Let F_1 and F_2 be two AFs. We say that F_1 and F_2 are equivalent with respect to a semantics σ if and only if $S_\sigma(F_1) = S_\sigma(F_2)$.

Semantic equivalence between AFs has been studied thoroughly, especially with respect to the dynamics. Besides enumerating the extensions for a certain semantics σ, one of the most common tasks performed on AFs is to decide whether an argument a is accepted in some extension of $S_\sigma(F)$ or in all extensions of $S_\sigma(F)$. In the former case, we say that a is *credulously* accepted with respect to σ; in the latter, a is instead *sceptically* accepted with respect to σ.

Example 2.1. *In Figure 2.1 we provide an example of AF where sets of extensions are given for all the mentioned semantics. We discuss some details: the singleton $\{e\}$ is not conflict-free because e attacks itself. The argument b is not contained in any admissible extension because no other argument (included itself) defends b from the attack of a. The empty set $\{\}$, and the singletons $\{c\}$ and $\{d\}$ are not complete extensions because a, which is not attacked by any other argument, has to be contained in all complete extensions. Only the maximal (with respect to set inclusion) admissible extensions $\{a, c\}$ and $\{a, d\}$ are preferred, while the minimal complete $\{a\}$ is the (unique) grounded extension. Then, the arguments in the subset $\{a, d\}$, that conduct attacks against all the other arguments (namely b, d and e), represent a stable extension. To conclude the example, we want to*

point out that argument a is sceptically accepted with respect to the complete semantics, since it appears in all three subsets of $S_{com}(F)$. On the other hand, arguments c, that is in just one complete extension, is credulously accepted with respect to the complete semantics.

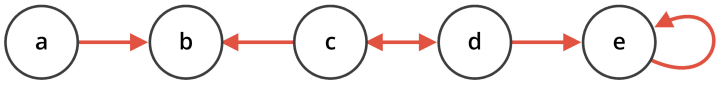


FIGURE 2.1: An argumentation framework F for which we compute the following sets of extensions: $S_{cf}(F) = \{\{\},\{a\},\{b\},\{c\},\{d\},\{a,c\},\{a,d\},\{b,d\}\}$, $S_{adm}(F) = \{\{\}, \{a\},\{c\},\{d\},\{a,c\},\{a,d\}\}$, $S_{com}(F) = \{\{a\},\{a,c\},\{a,d\}\}$, $S_{prf}(F) = \{\{a,c\},\{a,d\}\}$, $S_{stb}(F) = \{\{a,d\}\}$, and $S_{gde}(F) = \{\{a\}\}$.

Many of the above-mentioned semantics (such as the admissible and the complete ones) exploit the notion of defence in order to decide whether an argument is part of an extension or not. The phenomenon for which an argument is accepted in some extension because it is defended by another argument belonging to that extension is known as *reinstatement* [65]. In the same paper, Caminada also give a definition for a reinstatement labelling.

Definition 2.6 (Reinstatement labelling)**.** Let $F = \langle Arg, R\rangle$ be an AF and $\mathbb{L} = \{\mathsf{in}, \mathsf{out}, \mathsf{undec}\}$. A labelling of F is a total function $L : Arg \to \mathbb{L}$. We define $in(L) = \{a \in Arg \mid L(a) = \mathsf{in}\}$, $out(L) = \{a \in Arg \mid L(a) = \mathsf{out}\}$ and $undec(L) = \{a \in Arg \mid L(a) = \mathsf{undec}\}$. We say that L is a reinstatement labelling if and only if it satisfies the following:

- $\forall a, b \in Arg$, if $a \in in(L)$ and $(b, a) \in R$ then $b \in out(L)$;
- $\forall a \in Arg$, if $a \in out(L)$ then $\exists b \in Arg$ such that $b \in in(L)$ and $(b, a) \in R$.

Moreover, we denote with $Arg|_{\mathsf{in}}$ the subset of all and only arguments of Arg whose label is in. Analogously, we use $Arg|_{\mathsf{out}}$ and $Arg|_{\mathsf{undec}}$ to refer to subset of out and undec arguments, respectively.

In other words, an argument is labelled in if all its attackers are labelled out, and it is labelled out if at least an in node attacks it. In all other cases, the argument is labelled undec. A labelling-based semantics [15] associates with an AF a subset of all the possible labellings. In Figure 2.2 we show an example of reinstatement labelling on an AF.

Given a labelling L, it is possible to identify a correspondence with the extension-based semantics [15]. In particular, the set of in arguments coincides with a complete extension,

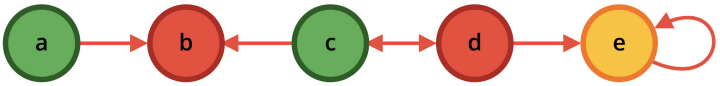


FIGURE 2.2: an example of AF in which reinstatement labelling is showed by using colours. Arguments a and c highlighted in green are in, red ones (b and d) are out, and the the yellow argument e (that attacks itself) is undec.

Labelling restrictions		Semantics
no restrictions		complete
empty	undec	stable
maximal	in	preferred
maximal	out	preferred
maximal	undec	grounded
minimal	in	grounded
minimal	out	grounded
minimal	undec	semi-stable

TABLE 2.1: Reinstatement labelling vs semantics.

while other semantics can be obtained through restrictions on the labelling as shown in Table 2.1.

A labelling for the strongly admissible semantics is given in [66], where the author relies on a numbering on the arguments to assign the correct labels. In every labelling of the various semantics, arguments for which not every attacker is labelled out, and no attacker is labelled in are labelled undec.

A further refinement of labelling for AFs is provided in [105], where the authors distinguish four types of label.

Definition 2.7 (Four-state labelling)**.** A four-state labelling consists of a total mapping $L : Arg \to 2^{\{\mathsf{in},\mathsf{out}\}}$ that satisfies the following conditions:

- $\forall a \in Arg$, if $\mathsf{out} \in L(a)$, then $\exists b \in Arg$ such that $(b, a) \in R$ and $\mathsf{in} \in L(b)$;
- $\forall a \in Arg$, if $\mathsf{in} \in L(a)$, then $\forall b \in Arg$ such that $(b, a) \in R$, $\mathsf{out} \in L(b)$;
- $\forall a \in Arg$, if $\mathsf{in} \in L(a)$, then $\forall c$ such that $(a, c) \in R$, $\mathsf{out} \in L(c)$.

A four-state labelling is said to be total[3] if and only if $\forall a \in Arg$, $L(a) \neq \emptyset$. A labelling which is not total is called partial. Moreover, the four labels form the lattice of Figure 2.3, in which undec (that is the set $\{\mathsf{in}, \mathsf{out}\}$) is the top element and empty is the bottom.

[3]The total labelling is called "complete" in the original definition [105]. We changed it to avoid ambiguity with the complete semantics.

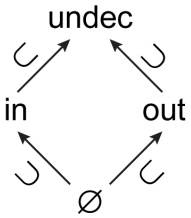


FIGURE 2.3: Lattice of labels in the four-state labelling.

We show an example of labelling in Figure 2.4, where all four labels are used. Note that the arguments labelled in and out in the figure do not satisfy the condition of the reinstatement labelling. Arguments that are assigned the label in are accepted, those that are assigned the label out are rejected, those that are assigned both in and out (which we denote as undec) are neither fully accepted nor fully rejected, and those that are not considered at all are assigned the empty label.

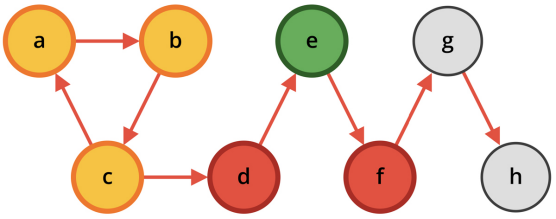


FIGURE 2.4: Labelling of an AF showed through colours. Argument e, highlighted in green, is the only in; red arguments d and f) are out; those in yellow, i.e., a, b and c, are undec; and the grey arguments g and h are left with an empty label empty.

Even though the labelling of Definition 2.7 is more informative than the reinstatement labelling of Definition 2.6 (that does not comprehend an empty label), there not exist a direct connection between labellings and extensions of a certain semantics, as it happens for the reinstatement labelling. We use restatement and four-state labelling in Chapter 5 of this thesis.

Extension- and labelling-based semantics define criteria for deciding if an argument of an AF is to consider accepted or not, without expressing how much an argument is acceptable with respect to the others. In order to distinguish between more acceptability degrees than just in, out and undec, ranking-based semantics [7] can be defined starting from properties that give "precedence" to arguments. For instance, arguments that do not receive attacks are more preferred than arguments that are attacked [62]. In this way, arguments can be ranked from the most to the least acceptable ones.

Definition 2.8 (Ranking-based semantics [7])**.** A ranking-based semantics associates with any $F = \langle A, R \rangle$ a ranking $\succcurlyeq_F$ on A, where $\succcurlyeq_F$ is a pre-order (a reflexive and transitive relation) on A. $a \succcurlyeq_F b$ means that a is at least as acceptable as b ($a \simeq b$ is a shortcut for $a \succcurlyeq_F b$ and $b \succcurlyeq_F a$, and $a \succ_F b$ is a shortcut for $a \succcurlyeq_F b$ and $b \not\succcurlyeq_F b$).

When clear from the context, we will just write $\succcurlyeq$ to denote the ranking. A ranking-based semantics can be characterised by some specific properties that take into account how couples of arguments in an AF are evaluated for establishing their position in the ranking. We provide a list of the properties suggested in [7].

Definition 2.9 (Isomorphism)**.** An isomorphism ι between two AFs $F = \langle A, R\rangle$ and $F' = \langle A', R'\rangle$ is a bijective function $\iota : A \to A'$ such that $\forall a, b \in A$, $(a, b) \in R$ if and only if $(\iota(a), \iota(b)) \in R'$.

We can characterise the role of an argument with respect to another one according to the length of the path between them: an odd path represents an attack, while an even path is considered as a defence.

Definition 2.10 (Attackers and defenders [7])**.** Let $F = \langle A, R\rangle$ be an AF and $a, b \in A$ and denote with $P(b, a)$ a path from b to a. The multi-sets of **defenders** and **attackers** of a are $R_n^+(a) = \{b \mid \exists P(b, a)$ with length $n \in 2\mathbb{N}\}$ and $R_n^-(a) = \{b \mid \exists P(b, a)$ with length $n \in 2\mathbb{N} + 1\}$, respectively. $R_1^-(a) = R^-(a)$ is the set of direct attackers of a.

Besides arguments alone, also sets of arguments can be compared. Two rules apply: the greater the number of arguments, the more preferred the group; in case of two groups with the same size, the more preferred the arguments in a group, the more preferred the group itself.

Definition 2.11 (Group Comparison [7])**.** Let $\succcurlyeq$ be a ranking on the elements of a set S. The associated group comparison $\succcurlyeq^G$ on 2^S is defined as follows. For $S_1, S_2 \subseteq S$: $S_1 \succcurlyeq^G S_2$ if and only if there exists an injective mapping $f : S_2 \to S_1$ such that $\forall a \in S_2$, $f(a) \succcurlyeq a$. Moreover, $S_1 \succ^G S_2$ if and only if $S_1 \succcurlyeq^G S_2$ and either $|S_2| < |S_1|$ or f additionally satisfies $f(a) \succ a$ for some $a \in S_2$. $\succ^G$ is called strict group comparison.

For example, consider the sets of arguments $S_1 = \{a, b, c\}$, $S_2 = \{c, e\}$ and $S_3 = \{d, f\}$, and the ordering $a \succ b \succ c \succ d \succ e \succ f$. By Definition 2.11, we have that $S_1 \succ^G S_2$ since $|S_2| < |S_1|$, and $S_2 \succ^G S_3$ since $c \succ d$ and $e \succ f$. In the following, we list the properties proposed in [7], all defined for a ranking-based semantics σ, for any AF $F = \langle A, R\rangle$ and for all pairs of arguments $a, b \in F$.

Abstraction. The ranking on A is defined only on the basis of the attacks between arguments, that is it is preserved over isomorphisms of the framework.

(Abs) For any isomorphism γ such that $F' = \gamma(F)$, $a \succcurlyeq^\sigma_F b$ if and only if $\gamma(a) \succcurlyeq^\sigma_{F'} \gamma(b)$

Independence. The ranking between two arguments a and b should be independent of any argument that is neither connected to a nor to b.
(Ind) $\forall F' = \langle A', R' \rangle \in cc(F)$, $\forall a, b \in A'$, then $a \succcurlyeq^{\sigma}_{F'} b \Rightarrow a \succcurlyeq^{\sigma}_{F} b$, where $cc(F)$ denotes the set of connected components in F

Void Precedence. A non-attacked argument is ranked strictly higher than any attacked argument.
(VP) $R_1^-(a) = \emptyset$ and $R_1^-(b) \neq \emptyset \Rightarrow a \succ^{\sigma} b$

Self-contradiction. A self-attacking argument is ranked lower than any other non self-attacking argument.
(SC) $(a, a) \notin R$ and $(b, b) \in R \Rightarrow a \succ^{\sigma} b$

Cardinality Precedence. The greater the number of direct attackers for an argument, the weaker the rank of this argument.
(CP) $|R_1^-(a)| < |R_1^-(b)| \Rightarrow a \succ^{\sigma} b$

Quality Precedence. An argument a should be ranked higher than an argument b, if at least one attacker of b is ranked higher than any attacker of a.
(QP) $\exists c \in R_1^-(b)$ such that $\forall d \in R_1^-(a), c \succ^{\sigma} d \Rightarrow a \succ^{\sigma} b$

Counter-Transitivity. If the direct attackers of b are at least as numerous and acceptable as those of a, then a is at least as acceptable as b.
(CT) $R_1^-(b) \succcurlyeq^{G} R_1^-(a) \Rightarrow a \succcurlyeq^{\sigma} b$

Strict Counter-Transitivity. If CT holds and either the direct attackers of b are strictly more numerous or acceptable than those of a, then a is strictly more acceptable than b.
(SCT) $R_1^-(b) \succ^{G} R_1^-(a) \Rightarrow a \succ^{\sigma} b$

Defense Precedence. For two arguments with the same number of direct attackers, a defended argument is ranked higher than a non-defended argument.
(DP) $|R_1^-(a)| = |R_1^-(b)|, R_2^+(a) \neq \emptyset$ and $R_2^+(b) = \emptyset \Rightarrow a \succ^{\sigma} b$

Non-attacked Equivalence. All the non-attacked arguments have the same rank.
(NaE) $R^-(a) = \emptyset$ and $R^-(b) = \emptyset \Rightarrow a \simeq^{\sigma} b$

Totality. All pairs of arguments can be compared. **(ToT)** $a \succcurlyeq^{\sigma} b$ or $b \succcurlyeq^{\sigma} a$.

In [7] and [62] implications between the above properties are studied. In particular, we take into account the following considerations.

Proposition 2.12. *For every ranking-based semantics,*

- *CP and QP are not compatible*
- *SCT implies VP*
- *CT and SCT imply DP*
- *SCT implies CT*
- *CT implies NaE*

Before continuing, we remark that all these properties are not mandatory for obtaining a well-defined ranking, but they are just meant to provide a form of characterization for a ranking-based semantics. In fact, some properties are incompatible, and one might or might not find convenient to have a particular property for a certain application.

2.1.1 Weighted Argumentation Frameworks

In order to compute the set of extensions of a particular AF, attack relations are used to determine the acceptability of the arguments. Since it is not possible to further diversify the relations among arguments, every attack in the AF has the same "strength", that is, the existence or not of an attack is the only thing that matters in determining the semantics. To overcome this limit, Dung's AFs have been extended to Weighted AFs (WAFs) by associating the attacks with a weight that represents the support of the relation [86]. In this kind of frameworks, the acceptability criteria for the arguments also need to consider the weight of incoming and outgoing attacks. Three main approaches have been proposed in the literature: in [122] the attacks are considered individually and the acceptability status of an argument is determined through a one by one comparison on the strengths of the relations; in [74] each attack towards an argument can be defended form a group of arguments, with an overall strength obtained by aggregating the single strengths of the counter-attacks coming from that group; the method used in [50], instead, aggregates the strengths of both the attacks conducted towards and argument and the defences for that argument. In this latter work, the framework is equipped with a c-semiring [29, 39] that provides the operation for composing the weights in order to estimate the effectiveness of a defence. The acceptability of an argument is then determined by comparing the compositions of the attacks with the composition of the defences.

C-semirings are absorptive, commutative semiring, that is commutative semirings with idempotent plus operator (also called tropical semirings) and top element. These structures allow expressing both the values of the weights and the aggregation operators and thus are parametric to the desired notion of defence.

Definition 2.13 (c-semirings [39])**.** A c-semiring is a tuple $\mathbb{S} = \langle S, \oplus, \otimes, \bot, \top \rangle$ such that S is a set, $\top, \bot \in S$, and $\oplus, \otimes : S \times S \to S$ are binary operators making the triples $\langle S, \oplus, \bot \rangle$ and $\langle S, \otimes, \top \rangle$ commutative monoids (semi-groups with identity), satisfying *i)* $\forall s, t, u \in S.\ s \otimes (t \oplus u) = (s \otimes t) \oplus (s \otimes u)$ (distributivity), and *ii)* $\forall s \in S.\ s \otimes \bot = \bot$ (annihilator). Moreover, we have that $\forall s, t \in S.\ s \oplus (s \otimes t) = s$ (absorptiveness). The operator $\oplus$ also defines a preference relation $\leq_{\mathbb{S}}$ over the set S, such that $a \leq_{\mathbb{S}} b \iff a \oplus b = b$, for $a, b \in S$.

We list some of the most common instances of c-semirings.

- $\mathbb{S}_{boolean} = \langle \{false, true\}, \vee, \wedge, false, true \rangle$
- $\mathbb{S}_{fuzzy} = \langle [0, 1], \max, \min, 0, 1 \rangle$
- $\mathbb{S}_{probabilistic} = \langle [0, 1], \max, \times, 0, 1 \rangle$
- $\mathbb{S}_{weighted} = \langle \mathbb{R}^+ \cup \{+\infty\}, min, +, +\infty, 0 \rangle$

Different c-semirings can represent different notions of defence for WAF, by using the operators $\oplus$ and $\otimes$ for obtaining an ordering among the values in S. For simplicity, we refer to these values as weights. Note that the element $\top$ of the c-semiring (*e.g.*, 0 for the weighted and *true* for the boolean) coincides with having no relation between two arguments. We denote with $\text{WAF}_{\mathbb{S}}$ a WAF endowed with a c-semirings $\mathbb{S}$ and we call it a semiring-based WAF.

Definition 2.14 ($\text{WAF}_{\mathbb{S}}$ [40])**.** A semiring-based WAF is a quadruple $\langle \mathcal{A}, \mathcal{R}, W, \mathbb{S} \rangle$, where $\mathbb{S}$ is a c-semiring $\langle S, \oplus, \otimes, \bot, \top \rangle$, $\mathcal{A}$ is a set of arguments, R the attack binary-relation on $\mathcal{A}$, and $W : \mathcal{A} \times \mathcal{A} \longrightarrow S$ is a binary function. Given $a, b \in \mathcal{A}$ and $R(a, b)$, then $W(a, b) = s$ means that a attacks b with a weight $s \in S$. Moreover, we require that $R(a, b)$ if and only if $W(a, b) <_{\mathbb{S}} \top$.

Given a $\text{WAF}_{\mathbb{S}}$ we can evaluate the overall weight of all the attacks from a set of arguments towards another set through the **composition** operator $\otimes$ of the c-semiring $\mathbb{S}$. In particular, we use $\bigotimes$ to indicate the $\otimes$ operator on a set of values (indeed $\otimes$ is a binary operator that composes two weights).

Definition 2.15 (Attacks [40])**.** Let $F = \langle \mathcal{A}, \mathcal{R}, W, \mathbb{S} \rangle$ be a $WAF_{\mathbb{S}}$. A set of arguments $\mathcal{B}$ attacks a set of arguments $\mathcal{D}$ and the weight of such attack is $k \in S$, if

$$W(\mathcal{B}, \mathcal{D}) = \bigotimes_{b \in \mathcal{B}, d \in \mathcal{D}} W(b, d) = k.$$

The previous definition also allows composing the attacks from a set of arguments to another single argument, and from a single argument towards a set of arguments. The frameworks in [74, 122] can be then described as instances of a $\text{WAF}_{\mathbb{S}}$. For example, the attack strength used in [122] corresponds to the strongest weight among all the counter-attacks and can be obtained through a fuzzy semiring. The approach in [74], instead, uses an aggregation function (e.g., + or max) to obtain the overall strength of the attacks coming from the defending arguments; also in this case, a semiring can be selected according to the used aggregation function.

The notion of weighted defence (or w-defence that we use in Chapter 5) can then be expressed in terms of preferences over the weighted attack relations. In particular, if we consider the defence of [122], we obtain the defence $\mathbb{D}_1$; alternatively, we can use the definition in [74] to obtain $\mathbb{D}_2$; the notion introduced in [42], finally, generalises the other two approaches and provides the defence $\mathbb{D}_3$.

Definition 2.16 (w-defence)**.** Let $F = \langle \mathcal{A}, \mathcal{R}, W, \mathbb{S} \rangle$ be a $WAF_{\mathbb{S}}$. Then $\mathcal{B} \subseteq \mathcal{A}$ w-defends $b \in \mathcal{A}$ if and only if $\forall a \in \mathcal{A}$ such that $R(a, b)$,

$\mathbb{D}_1$: $\exists c \in B \mid W(a, b) \geq_{\mathbb{S}} W(c, a)$, or

$\mathbb{D}_2$: $W(a, b) \geq_{\mathbb{S}} W(\mathcal{B}, a)$, or

$\mathbb{D}_3$: $W(a, \mathcal{B} \cup \{b\}) \geq_{\mathbb{S}} W(\mathcal{B}, a)$.

By using one among $\mathbb{D}_1$, $\mathbb{D}_2$ and $\mathbb{D}_3$ for checking the acceptability of the arguments in the weighted framework, it is possible to redefine the classical extension-based semantics.

Definition 2.17 (Extension-based semantics for $\text{WAF}_{\mathbb{S}}$ [33])**.** Let $_{\mathbb{S}}$ $F = \langle \mathcal{A}, \mathcal{R}, W, \mathbb{S} \rangle$ be a WAF. A subset of arguments $\mathcal{B} \subseteq \mathcal{A}$ is w-conflict-free if $W(\mathcal{B}, \mathcal{B}) = \top$. A w-conflict-free subset $\mathcal{B}$ is then

- w-admissible, if $\forall a \in \mathcal{B}^{-}$. $W(a, \mathcal{B}) \geq_{\mathbb{S}} W(\mathcal{B}, a)$ (that is $\mathcal{B}$ w-defend itself from the arguments in $\mathcal{A} \setminus \mathcal{B}$);
- w-complete, if it is w-admissible and each argument $b \in \mathcal{A}$ such that $\mathcal{B} \cup \{b\}$ is w-admissible belongs to $\mathcal{B}$;
- w-stable, if it is w-admissible and $\forall a \notin \mathcal{B}$. $\exists b \in \mathcal{B}$ such that $W(b, a) <_{\mathbb{S}} \top$;
- w-preferred, if it is a maximal (with respect to set inclusion) w-admissible subset of $\mathcal{A}$;
- w-grounded, if it is the maximal (with respect to set inclusion) w-admissible extension included in the intersection of w-complete extensions;

- w-quasi-strongly admissible[4], if $\forall a \in \mathcal{B}^{-}$, $\forall b \in \mathcal{B}$. $\exists \mathcal{C} \subseteq \mathcal{B} \setminus \{b\}$ with $W(a, \mathcal{B}) \succcurlyeq^{G} W(\mathcal{C}, a)$.

A w-grounded extension always exists, according to all three defence approaches. In [33, 74] it is also shown that, using either $\mathbb{D}_2$ or $\mathbb{D}_3$, such extension is unique. Moreover in [33] it corresponds to any maximal w-admissible extension included in the intersection of w-complete extensions. The definition for w-quasi-strongly admissible extensions, first given in [33], states that a subset of arguments $\mathcal{B}$ is w-strongly admissible when for all $b \in \mathcal{B}$, $\mathcal{B}$ is defended by a subset of $\mathcal{B}$ that does not include b. In other words, each argument in $\mathcal{B}$ is defended by the rest of the arguments in $\mathcal{B}$.

2.1.2 Probabilistic Argumentation Frameworks

Many different extensions of AFs from Dung's pioneering work [85] have appeared to better describe different aspects of debates. Sample works include Assumption Based Argumentation [61], extending AFs with support [133], or introducing labels [165]. Others [27, 50] have focused on introducing weights in elements of the AF. Knowledge representation with the use of probabilistic information has been used in many areas of Computer Science, and it is a powerful medium to represent knowledge. For this reason, many researchers have extended AFs by adding probabilistic information. These very prominent extensions of AFs have been categorized in two big groups by Hunter [103]: the **epistemic** and the **constellation** approaches. The epistemic approaches, such as those presented in [157] describe probabilistic AFs (PrAFs) where the uncertainty does not alter the structure of the AFs. This type of AFs use the probability assignments to quantify the existing uncertainty of arguments in AFs and not to introduce new uncertainty. The constellation approaches, such as those presented in [117] introduce probabilistic values that are associated with the elements in an AF; in such a way, the uncertainty related to the structure of the AF can be represented. The constellation approaches generate a set of AFs with a probabilistic distribution and as such they define a probabilistic distribution over the extensions of those AFs. Fazzinga et al. [94] discuss the complexity for computing different semantics in PrAFs. We focus on the constellation approaches.

Definition 2.18 (PrAF [117])**.** A probabilistic AF is a tuple $PrAF = (Args, Atts, P_{Atts})$ where $Args$, $Atts$ define an AF, P_{Atts} is a set of probabilities for each $\rightarrow \in Atts$ with $0 < P_{Atts}(\rightarrow) \leq 1$.

[4]The definition for the w-quasi-strongly admissible semantics is introduced in [33], where the authors refer to it by the term w-strongly admissible. However, differently from the classical case, the defending set $\mathcal{B}' \subseteq \mathcal{B} \setminus \{a\}$ is not required to recursively be w-strongly admissible, and thus we considered it more appropriate to use a different name.

Stating an attack has probability 0 is redundant. A probabilistic attack with 0 probability is not part of any AF that the constellation represents and is omitted. A PrAF defines a probability distribution for all the possible non-probabilistic AFs it contains. Each single possible set of probabilistic attacks of the PrAF is called a **possible world**. The possible worlds of a PrAF are exponential in the number of probabilistic attacks (2^N where N the number of probabilistic attacks).

Definition 2.19 (Probability of Possible World [117])**.** The probability of a possible world equals to the product of the probability of each probabilistic attack that is in the possible world with the product of one minus the probability of each probabilistic attack that is excluded from the possible world.

$$P_{world} = \prod_{e_i \in AF_{world}} P(e_i) \cdot \prod_{e_j \notin AF_{world}} (1 - P(e_j))$$

The usual AF semantics are slightly modified in PrAFs. For example, in PrAFs the inquisitor is not asking if a set Q is admissible in PrAF P; but what is the probability that set Q is admissible in PrAF P, meaning with what probability exists an AF where Q is admissible. Similarly, for different semantics than admissible such as complete, preferable, etc.

2.1.3 Structured Argumentation

In abstract argumentation, arguments have no internal structure and there is no specification of what is an argument or an attack. This abstract perspective provides many advantages for studying the nature of argumentation, but it does not cover all our needs for understanding argumentation or for building tools for supporting or undertaking argumentation. On the other hand, structured argumentation provide a formal language for representing knowledge, and specifying how arguments and counterarguments can be constructed from that knowledge. Below, we discuss different approaches to structured argumentation.

Toulmin [159] developed a method for analysing arguments based on a structure of six elements: claim, grounds, warrant, qualifier, rebuttal and backing. The first three components (claim, grounds and warrant) are the fundamental part of every argument and always need to be specified, while the last three (qualifier, rebuttal and backing) can be added to further develop the argument and provide additional information to work through. An argument written in this manner unfolds to reveal both the strengths and limits of the argument. No argument pretends to be stronger than it is or applies further than it is meant to. The point here is not to win or beat all the counter-arguments, but

to come as close to the truth or as close to a realistic and feasible solution as we possibly can.

$ASPIC^+$ [129] is a framework for specifying argumentation systems that, contrary to AF in the sense of Dung, offers a mechanism for reasoning with the structure of arguments and the nature of attack or defeat. This allow for coping with problems that is not possible to formulate at the abstract level. For example, it is possible to check whether arguments in the same extension have mutually consistent claims. In $ASPIC^+$, arguments constitute a reasoning structure that, starting from a set of premises, provide backing for a claim. The strongest way to remove doubt is to show that the claim deductively follows from grounds. Indeed, arguments are considered as claims supported with one or more premises [140]. There are three different ways in which $ASPIC^+$ arguments can be in conflict. Indeed, arguments can be attacked on a conclusion of a defeasible inference (rebutting attack), on a defeasible inference step itself (undercutting attack), or on an ordinary premise (undermining attack). In general, an argument A can be used as a counter-argument to B, if A successfully attacks, i.e. defeats, B. Whether an attack from A to B (on its sub-argument B') succeeds as a defeat, may depend on the relative strength of A and B', i.e. whether B' is strictly stronger than, or strictly preferred to A. Preferences can be established through an ordering $\succcurlyeq$ on the set of all arguments that can be constructed on the basis of an argumentation theory. We can also instantiate a Dung framework with $ASPIC^+$ arguments and the $ASPIC^+$ defeat relation. Standard semantics [85] can then be used to determine the acceptability of the arguments in the obtained A.

Similarly to what happens for $ASPIC^+$, in Assumption-Based Argumentation (ABA) arguments and attacks are derived from given rules in a deductive system, assumptions, and their contraries [61]. All these elements are abstract and can be instantiated to describe a particular argumentation system, although the abstraction level is not as high as in Dung's frameworks that do not provide information about the internal structure of the arguments. Assumptions are "defeasible" sentences of the language and the only part of the argument that can be attacked. Consequently, argumentation in ABA amounts to identifying "strong" sets of assumptions. Unless an ABA framework contains at least one assumption the argumentation is trivial, since there is nothing to debate about. An Assumption can either be hypothesised or derived using relations to construct arguments, depending on whether it is used in the head or in the body of a rule. Assumptions are attacked on the basis of a contrary function that identifies contrary sentences in the provided language. Each assumption can only have one single contrary while different assumptions can have the same contrary. In ABA, arguments are deductions of claims using rules and supported by sets of assumptions, and attacks are directed at the assumptions in the support of arguments. It follows that attacks between ABA's

arguments correspond to attacks between sets of assumptions. A notion of defence can also be derived for assumptions. This notion is used to determine the justification status of the assumptions themselves.

Defeasible Logic Programming [97] (DeLP) is a formalism in which defeasible argumentation is combined with logic programming. The DeLP language has three components: facts, strict rules and defeasible rules. In a DeLP program, we distinguish the subset of facts and strict rules, and the subset of defeasible rules. The defeasible part is introduced in the language through weak rules that allow to identify guaranteed claims in a knowledge base by using an inference mechanism. Such a derivation is called defeasible because there may exist information in the knowledge base that is in contradiction with some facts, thus preventing its acceptance as a valid conclusion. Conversely, we call strict derivation, a derivation where only strict rules and facts are used. Usually, only the set of defeasible rules of a DeLP program is contradictory, so to maintain a certain internal coherence within set of facts and strict rules. Given a contradictory program, DeLP uses a defeasible argumentation formalism to decide between contradictory goals. The fundamental part of the formalism is the notion of argument, that is a minimal and non-contradictory set of rules used to derive a conclusion. It is wort noticing that strict rules are not even part of the argument structure. Whilee $ASPIC^+$ and ABA can rely on external mechanisms (for instance Dung's semantics) for identifying the acceptable part of controversial information, DeLP uses the notion of *counter-arguments*. An argument can indeed be defeated by some counter-argument if the latter is preferred to the former for some criterion. For an argument structure there can be several counter-arguments either directly or indirectly attacking it. Therefore, all the counter-arguments have to be taken into account in order to verify whether it is defeated or not, on the basis of some preference relation. To establish whether an argument structure is defeated, we need to consider all its defeaters and check if they are defeated in turn by some other argument structures.

Between structured and abstract argumentation, there are intermediate representation forms, called semi-structured, which capture a set of essential features of structured arguments. Example of this approach are LAF-ensembles [19] and Claim-Augmented Argumentation Frameworks [90].

2.1.3.1 Claim-Augmented Argumentation Frameworks

The main idea behind claim-augmented AF is to provide additional information in AFs by associating a claim to each argument: in fact, arguments in an AF should be associated

with claims they support. Hereby, each argument exactly supports one claim, while a claim can be associated with more than one argument.

Definition 2.20 (CAFs [90])**.** A *claim-augmented argumentation framework* (CAF in short) is a triple $(A, R, claim)$ where (A, R) is an AF and $claim\colon A \to X$ assigns a claim (from a given universe of possible claims X) to each argument of A. If $claim(a) = x$ we also say that a supports x. For a CAF $CF = (A, R, claim)$, we use AF_{CF} to refer to AF (A, R) and X_{CF} to denote the set of claims in CF, i.e. $X_{CF} = \{claim(a) \mid a \in A\}$.

Given a CAF $CF = (A, R, claim)$, and claim $x \in X$, we use $A_{CF,x}$ to denote the set of arguments a in CF with $claim(a) = x$. For claims $x, y \in X$, we say that x attacks y in CF if there are arguments $a \in A_{CF,x}, b \in A_{CF,y}$, such that $(a, b) \in R$; we further use $x^+_{CF} = \{y \in X \mid x \text{ attacks } y \text{ in } CF\}$ and $x^-_{CF} = \{y \in X \mid y \text{ attacks } x \text{ in } CF\}$. If clear from the context, we will drop the subscript CF.

Extension-based semantics for a CAF $(A, R, claim)$ are defined by re-interpreting extensions of the standard semantics of the underlying AF (A, R) via the *claim*-function [90].

Definition 2.21. For a semantics σ, we define its claim-based variant σ_c as follows. For any CAF $CF = \langle A, R, claim\rangle$, $\sigma_c(CF) = \{claim(E) \mid E \in \sigma(\langle A, R\rangle)\}$. Given $S_c \in \sigma_c(CF)$, we say that $E \subseteq A$ is a *σ-realization* of S_c in CF if $claim(E) = S_c$ and $E \in \sigma(\langle A, R\rangle)$.

We finally introduce two central subclasses of CAFs [90].

Definition 2.22. Let $CF = (A, R, claim)$ be a CAF with $F = (A, R)$ the underlying AF. CF is called (i) *well-formed* if $a^+ = b^+$ for all $a, b \in A$ with $claim(a) = claim(b)$; (ii) *att-unitary* if $a^- = b^-$ for all $a, b \in A$ with $claim(a) = claim(b)$.

In other words, a CAF is well-formed when arguments with the same claim attack the same arguments, while it is att-unitary when arguments with the same claim are attacked by the same arguments. Examples are given in Figure 2.5 (we use the label a_x to denote an argument a supporting a claim x).

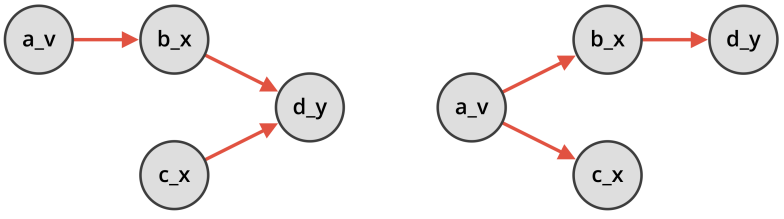


FIGURE 2.5: Examples of a well-formed CAF (left) and an att-unitary CAF (right).

2.1.4 The ConArg Suite

ConArg [31] is a suite of tools developed with the purpose to facilitate research in the field of Argumentation in Artificial Intelligence. Together with its software-library version ConArgLib [44], ConArg is able to verify, check the existence, check the non-emptiness, and enumerate extensions. Moreover, it can decide the credulous and sceptical acceptability of an argument. All the previous problems can be solved for following sets/semantics: conflict-free, admissible, complete, preferred, stable, grounded (all in [85]), semi-stable, stage, ideal, and eager (in successive works [15]). ConArgLib is available for both Linux and Windows OSs, and can be downloaded, as well as for the stand-alone solver, from the website. ConArg has been developed in C++ and is using Gecode, a toolkit for developing constraint-based systems and applications.

The project involves a series of components that address different aspects of argumentation, building on a constraint-based solver for argumentation problems [32, 44]. The tool has also been extended for handling weighted argumentation [43, 50] and, in this thesis, will be endowed with new features (encompassing different aspects of argumentation) and a web interface.

2.2 Power Indexes in Game Theory

In game theory, cooperative games are a class of games where groups of players (or agents) are competing to maximise their goal, through one or more specific rules. Voting games are a particular category of cooperative games in which the profit of coalitions is determined by the contribution of each individual player. In order to identify the "value" brought from a single player to a coalition, power indexes are used to define a preference relation between different agents, computed on all the possible coalitions. In Chapter 3, we use two among the most commonly used power indexes, namely the Shapley Value [150, 163] and the Banzhaf Index [13]. Other power indexes exist, such as the Deegan-Packel Index and the Johnston Index [108], that are relevant in cooperative game theory, and that could be useful for special purpose ranking that we plan to study in the future.

Every power index relies on a characteristic function $v : 2^N \to \mathbb{R}$ that, given the set N of players, associates each coalition $S \subseteq N$ with a real number in such a way that $v(S)$ describes the total gain that agents in S can obtain by cooperating with each other. The expected marginal contribution of a player $i \in N$, given by the difference of gain between S and $S \cup \{i\}$, is $v_{S_i} = v(S \cup \{i\}) - v(S)$.

The Shapley Value $\phi_i(v)$ of the player i, given a characteristic function v, is computed as:

$$\phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!\ (|N| - |S| - 1)!}{|N|!}\ v_{S_i} \tag{2.1}$$

The formula considers a random ordering of the agents, picked uniformly from the set of all $|N|!$ possible orderings. The value $|S|!\ (|N| - |S| - 1)!$ expresses the probability that all the agents in S come before i in a random ordering.

The second fair division scheme we use is the Banzhaf Index $\beta_i(v)$, which evaluates each player i by using the notion of *critical voter*: given a coalition $S \subseteq N \setminus \{i\}$, a critical voter for S is a player i such that $S \cup \{i\}$ is a winning coalition, while S alone is not. In other words, i is a critical voter if it can change the outcome of the coalition it joins.

$$\beta_i(v) = \frac{1}{2^{|N|-1}} \sum_{S \subseteq N \setminus \{i\}} v_{S_i} \tag{2.2}$$

The difference between the perhaps more famous Shapley Value and the Banzhaf index is that the latter does not take into account the order in which the players form the coalitions. Deegan and Packel assume that only minimal winning coalitions are formed, that they do so with equal probability, and that if such a coalition is formed it divides the (fixed) spoils of victory equally among its members. In order to avoid divisions by zero in the formula, we use the interpretation of [6]: let's call $M(v)$ the set of minimal winning coalitions of the game (always assuming $\emptyset \in M(v)$), and $M_i(v)$ the subset of $M(v)$ formed by coalitions $S \subseteq N$ such that $i \in S$. The Deegan-Packel index $\rho_i(v)$ of a player $i \in N$ is computed as follows.

$$\rho_i(v) = \frac{1}{|M(v)|} \sum_{\substack{S \subseteq M_i(v) \setminus \{i\} \\ S \neq \emptyset}} \frac{v_{S_i}}{|S|} \tag{2.3}$$

The last index we implement is the Johnston index [88]. Based on the principle of critical vote, it differs from Banzhaf's for the fact that critical voters in winning coalitions are rewarded with a fractional score instead of one whole unit (that is the score is equally divided among all critical members of the coalition). Let $\varkappa(S)$ denote the number of critical voters in a winning coalition S. The Johnston index $\gamma_i(v)$ of a player $i \in N$ is computed as follows.

$$\gamma_i(v) = \sum_{\substack{S \subseteq N \setminus \{i\} \\ \varkappa(S) \geq 1}} \frac{v_{S_i}}{\varkappa(S)} \tag{2.4}$$

Note that the summation is only done on the coalitions in which there is at least one critical voter.

2.3 Systems of Interacting Agents

Nowadays, the multi-agent system paradigm is widely used in many applications whenever it is required to solve a difficult problem or model complex situations [164]. In such systems, intelligent agents behave in a particular environment, determined by the application domain. For examples, the authors in [126] design agents as chatbots in a travel domain, while the work presented in [78]) consider vehicles that have to coordinate in order to find the route to reach a target destination. Depending on the domain, agents act in order to accomplish certain goals (for instance, travel planning and meeting arrangement [126] or generation of agent schedules [78]). In knowledge-based systems [78, 126], each agent has her own beliefs and aims to exchange information with the others in order to reach conclusions on a given matter. Information can also come from resources external to the system, e.g., humans. In [132], agents are given a knowledge base of commonsense concepts that are used for interacting with humans (and in particular children) for generating bedtime stories.

One of the fundamental tasks for an agent is the communication with other agents in the environment. Note that the view of the knowledge base can be either global or local for every agent. As for human beings, intelligent agents have to share not only a common language (essential to run communication protocols) but also an ontology, that is their understanding of the surrounding world. Many communication languages used in knowledge-based systems are variants of the Knowledge Query and Manipulation Language (KQML) [95]. Languages of this family share a set of primitives specifically designed for spreading and retrieving information.

A well-known paradigm for modelling intelligent agents is the so called **belief-desire-intention** software model (or simply BDI) [143], in which agents are endowed with their own belief about the world (that can also include inference rules), sets of goals (namely desires an agent is willing to pursue), and intentions, that represent what the agent has chosen to do. Although the BDI architecture concerns about individual agents, it also supports interactions between agents in multi-agent systems [100]. Once communication is established, more involved operations can be performed in the systems. For instance, agents can organize themselves for coordinating complex tasks (as crowd simulations in [109]), they can cooperate in finding strategies [160], and negotiate on the desired outcomes [92]. In order to achieve this kind of interactions between agents that act independently in a distributed environment, it is, therefore, necessary a concurrent

approach for the communication protocol. In the following, we summarise the main issues of dealing with concurrent systems, and some of the solutions provided by formal languages.

Agents (or threads) in a distributed system can perform operations that affect the behaviour of the other components. The indeterminacy in the execution order of the processes may lead to inconsistent results for the computation or even cause errors that prevent particular tasks from being completed. We refer to this kind of situation as a *race condition*. If not properly handled, race conditions can cause loss of information, resource starvation and deadlock.

In order to understand the behaviour of agents and devise solutions that guarantee correct executions, many formalisms have been proposed for modelling concurrent systems. Petri nets [137], parallel random-access machines [96] and process calculi such as communicating sequential processes (CSP) [101] and calculus of communicating systems (CCS) [127] are examples of models of concurrency. Process calculi support the description of high-level interactions between processes like communication, in the form of message passing, by using a set of primitives and operators to combine them. Furthermore, processes expressed with a calculus of this kind can be manipulated, checked for properties and be subjected to other kinds of analysis, such as bisimulation and trace equivalence checking. All the process calculi share a set of primitives (events, processes) and basic operations that, nonetheless, are implemented in different ways. Below, we list some of the most significant operations of CSP and CCS, highlighting similarities and differences between the two approaches.

- Action prefixing. Often, processes are required to be executed in an ordered fashion, possibly after a certain event happened or an input is received. The sequential composition allows expressing the behaviour of a process that after performing a particular action a, continue as another process P. This kind of process can be modelled in CCS as $a.P$. Similarly, the CSP construct $a \rightarrow P$ is the process that, after communicating a to the environment, proceeds as P.

- Parallel composition. Opposed to the sequential operator, the parallel composition $P|Q$ characterises in CCS the execution of two processes P and Q in parallel. Intuitively, $P|Q$ is the process that results by interleaving the executions of P and Q; the two processes can also perform a two-way synchronization, resulting in an internal system action, when they can perform complementary input and output actions. A form of multi-way synchronization can be obtained in CSP through the interface parallel operator $P|[\{a\}]|Q$, which requires both P and Q to perform a in order for that event to occur.

- CHOICE. In CCS, the process $P + Q$ can proceed either as P or Q. CSP, instead, distinguishes between external and internal choice. The difference between these two forms of choice is that in the former the environment is allowed to decide which action to perform, while in the latter there is no control over the choice.

- HIDING. This operator allows one to control the interaction between agents and it is crucial for restricting interference and abstracting process behaviours. To hide an events has different meanings in CSP and CCS. In CSP, to hide an event means to make that event unobservable, in such a way that only the consequent process is considered. For instance, $(a \rightarrow P) \setminus \{a\}$ simply reduces to P and every occurrence of a becomes non-observable from the environment. On the other hand, $P \setminus \{a\}$ behaves like P in CCS, but every action (in input or output) on the port a is considered impossible. Therefore, $(a.P) \setminus \{a\}$ results in a deadlock.

Both the formalisms described above use message passing in order to achieve synchronisation between agents. On the other hand, Concurrent Constraint Programming (CC) [147] that we use in Chapter 6 relies on a store of shared variables in which agents can read and write in accordance with constraints posed on the variables. We report the syntax of a CC program in Table 2.2. The basic operations that can be executed by agents in the CC framework are a blocking *Ask* and an atomic *Tell*. These operations realise the interaction with the store and also allow one to deal with partial information. In detail, Ask c succeeds if c is entailed by the store, fails if it is not, and suspends until it can either succeed or fail. Tell c succeeds if and only if c is consistent with the store (and the store is augmented with c). From our point of view, CC is more suitable to be used as a starting point for devising an argumentation-based concurrent language, than CCS and CSP. Indeed, AFs can be seen as the shared store on which agents can perform reasoning tasks.

$$
\begin{aligned}
&P ::= C.A \\
&C ::= p(x) :: A \mid C.C \\
&A ::= success \mid fail \mid tell(c) \rightarrow A \mid E \mid A \| A \mid \exists_x A \mid p(x) \\
&E ::= ask(c) \rightarrow A \mid E + E
\end{aligned}
$$

TABLE 2.2: CC syntax.

Chapter 3

Invariant Operators and Robustness

"*They always say time changes things,*
but you actually have to change them yourself."
– Andy Warhol

Abstract

In this Chapter, we study invariant local addition operators for conflict-free and admissible extensions. Such operators are directly applied on AFs, and are invariant with respect to a chosen semantics (that is with respect to each of the conflict-free/admissible set of arguments). Accordingly, we derive a definition of robustness for AFs in terms of the number of times such operators can be applied without producing any change in the chosen semantics.

AFs provide a basis for handling the evolution of situations in which instances of particular problems undergo changes and variations on the underlying information can be interpreted as modifications in the corresponding graph. Such modifications can be performed through operations of addition or removal of nodes and edges in the AF. As one can expect, introducing these changes might lead to obtain different semantics for the considered AF. We can classify the operations that can be performed on a framework in two types: the ones that change the semantics of the system and the ones that do not. In this chapter, we focus on the latter type of operations (which leave the semantics unchanged), and reducing to the case of addition (or removal) of an attack. Invariant operators can be used by intelligent agents to implement strategies aimed at pursuing their goals, like winning a debate or arriving to a convenient outcome in a negotiation.

In detail, we study a set of local addition [20] operators with respect to which the semantics is not altered. Due to the dynamic nature of certain problems, settling for a solution (in a particular AF) could not be sufficient to guarantee a good outcome in case the problem evolves. Think, for example, to negotiation or persuasion dialogues. With invariant operators at dispose, one could test and possibly "enforce" [23] the strength of its position, that amounts to add attacks such that the current semantics is stronger with respect to additional modification of the framework. Also, invariant operators could be successfully exploited for computing, in an efficient way, the semantics of an evolving AF.

We resort to a notion of "robustness" related to AFs: the main idea is that every argument (and set of arguments) is more or less suitable to undergo changes in a corresponding belief base [82]. Robustness gives a measure of how many changes an AF can withstand before changing its semantics. In particular, we *conflict-free* and *admissible sets* [85], since they are at the centre of any classical semantics and they have never been studied before in these terms. Differently from other works done in this direction (see Section 3.5), we consider how difficult is to modify the whole set of extensions instead of a single one, for instance as in [145]. The motivations behind our study involve several perspectives of implementation of such operators in order to deal with different problem related to argumentation. For instance, the frameworks in which semantics are invariant with respect to the same operations can be grouped in the same class: in this way, further properties of AFs could be studied. The notion of robustness [45] can be exploited to look for stronger clusters of arguments among frameworks of within the same class. At the same time a measure of robustness can be defined starting from the number of invariant operations admitted.

3.1 Robustness

Argumentation pursues the objective of studying how conclusions can be reached through a process of logical reasoning, starting from a set of assumptions. In the most common form of argumentation, a part (which can be, for instance, an interlocutor) in a debate tries to affirm some kind of information and defends it from the attacks of other parts. The purpose of each part (or actor) is to persuade the rest of participants in the dialogue, defeating their assertions through non-monotonic inference. Indeed, this model can be applied to a wide range of situations, for instance in order to solve the issue of finding a suitable outcome (e.g., an agreement in negotiation) for the actors. This raises a number of very interesting questions about the capabilities of argumentation applied to such

problems. In particular, some of the questions we used as a starting point for this work are:

- Is it possible to change the outcome of a debate according to a particular semantics or meaning?
- If so, how easy could it be to perform such a change?
- And which consequences does it bring?

In order to answer these questions and investigate the behaviour of AFs in a dynamic environment, we use the notion of robustness: the robustness of an AF, with respect to a given semantics, is measured by computing the minimum number of changes in the graph needed to change the corresponding extensions set (for instance, addition or removal of nodes or edges). Specifically, we focus on changes in terms of edges (corresponding to attacks relations) and look at the consequences they cause in the chosen semantics.

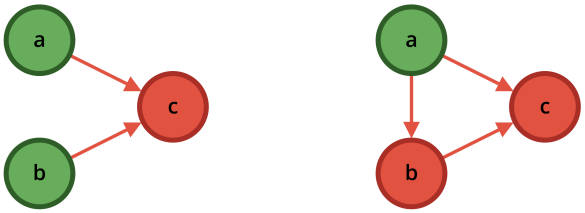
While AFs represent a powerful means to deal with argument-based semantics in static environments, possible evolutions of the supporting knowledge base are usually not taken into account. In this chapter we present different modifying operators able to make changes in the AF by adding an attack between a couple of arguments and for which the semantics is invariant. In this way one can identify the strongest extensions, where strong means to be more resistant to changes, and possibly to strengthen the weakest ones.

After a modification, either a set of arguments is no more acceptable with respect to a given semantics, or a new extension is generated, so the semantics of the AF will change in turn. On the contrary, if we consider the case in which extensions are preserved, further non trivial observations can be made for what concerns the semantics of the AF. For instance, even if the subsets of arguments remain unchanged, an admissible set can become also complete, if the right modifications are applied. Formally, an operator can be defined as follows.

Definition 3.1 (Local addition operator)**.** Let $G = \langle \mathcal{A}, \mathcal{R} \rangle \in \mathcal{F}_{\mathcal{A}}$ be an AF. A local addition operator is a function $m : \mathcal{F}_{\mathcal{A}} \to \mathcal{F}_{\mathcal{A}}$ such that $m(G) = \langle \mathcal{A}, m(\mathcal{R}) \rangle$, where $m(\mathcal{R}) \supseteq \mathcal{R}$.

In other words, a local addition operator is a function m that takes an AF G in input and outputs a modified version of G in which the set of relation $m(\mathcal{R})$ contains all the attacks in $\mathcal{R}$, plus any new attacks. If we consider those operators taking into account also Dung's semantics, we can study changes in the AFs from the point of view of sets of extensions.

Example 3.1. *Consider the AFs in Figure 3.1: in the framework G, the extension $\{a, b\}$ is both admissible and complete while the extension containing argument a is only admissible. After a modification consisting of the addition of the attack $a \to b$, the extension $\{a, b\}$ in the framework $m(G)$ is no longer admissible. On the other hand, after the change on the relations set, the extension $\{a\}$ also becomes complete.*



(a) G (b) $m(G)$

FIGURE 3.1: On the left an AF G, on the right and AF $m(G)$ obtained from G adding an attack from argument a to argument b.

3.2 Semantics Equivalence

Our purpose is to find local addition operators that leave the whole semantics unchanged, so instead of considering changes on the semantics induced by modifications on the graph, we look for the set of allowed changes that leave the semantics unmodified. In this way, we define semantics homomorphisms, namely operators for which the semantics is invariant. In order to preserve the whole semantics, it is necessary to ensure that all the sets will not be modified, hence every set of extensions has to be, in turn, invariant with respect to these operators. We say that if two AFs have the same set of extensions for a certain semantics, then the two frameworks are equivalent for such semantics. For this reason, we need the following definitions.

Definition 3.2 (Extensions set inclusion[5])**.** Let S and S' be two sets of extensions. We say that $S \subseteq S'$ if and only if $\forall E \in S \; \exists E' \in S'$ such that $E \subseteq E'$.

Definition 3.3 (σ-equivalence)**.** Let G and G' be two AFs and σ a semantics. We say that

- $G \equiv_\sigma G'$ if $S_\sigma(G) = S_\sigma(G')$;
- $G \sqsubseteq_\sigma G'$ if $S_\sigma(G) \subseteq S_\sigma(G')$.

[5]The notion of extensions set inclusion is related to that of argument semantics expansion of Definition 6.2 that we use to define operators of the language in Chapter 6.

Adding an attack in an AF can have different consequences. The most intuitive one is that the new attacked argument becomes defeated, and so it is forced to be removed from an extension. If we, instead, consider semantics in which the notion of acceptability is taken into account, defeating an argument could lead to accept another argument. In both the cases in which an argument become acceptable or is removed from an extension, the semantics would change. To distinguish the operators that reduce the set of extensions from those that expand it, we provide Definition 3.4.

Definition 3.4 (Invariant operators)**.** A local addition operator m is said non-decreasing with respect to a semantics σ and an argumentation framework $G = \langle \mathcal{A}, \mathcal{R} \rangle \in \mathcal{F}_{\mathcal{A}}$ if $G \sqsubseteq_\sigma m(G)$, and it is said non-increasing if $m(G) \sqsubseteq_\sigma G$. If m is both non-decreasing and non-increasing, it is an *invariant*: $G \equiv_\sigma m(G)$.

The last case may occur when an attack has no effect on the set of extensions. Our purpose is exactly to find local addition operators that guarantee this last outcome when adding an attack. In the following, an invariant operator will be referred to as h. It is necessary to understand how extensions react to changes in the AF. Since the main issue to deal with is due to the reinstatement, the idea we develop in order to define an invariant operator h is to use the notion of reinstatement labelling. Once the arguments of the AF are labelled (with in, out or undec), there are nine (3^2) different ways an edge can be added among nodes, according to labels of the source and the target of the attack. It is therefore essential to know in advance the possible labels of an argument in the framework and, in particular, if a certain argument is never labelled in, out or undec. Notice that in Chapter 5 we will extend this 3-state labelling to a 4-state one.

3.3 Invariant Operators for Conflict-Free Sets

The conflict-free property is very fragile: introducing a relation between two non conflicting nodes is sufficient to change the conflict-free sets. These sets can only be reduced: no new conflict-free set can be generated after the addition of an attack in the AF. Thus, every operator m able to perform the addition of an edge in a graph G produces another graph $m(G)$ in which the semantics is "smaller" (in the sense that in some extensions of the new AF an argument disappears) or at most equal to the set deriving from G. $\mathcal{R}$ is the set of relations belonging to G, while $m(\mathcal{R})$ is the same set after the addition of an attack introduced by m. We avoid describing the trivial case in which $m(G) = G$, and we only consider the effective transformation of adding an attack (symmetrical conclusions can be drawn in case of removal).

Proposition 3.5. *Every local addition operator m is non-increasing with respect to the conflict-free sets for any argumentation framework $G = \langle \mathcal{A}, \mathcal{R} \rangle \in \mathcal{F}_{\mathcal{A}}$.*

Proof. We have to show that $G \sqsupseteq_{cf} m(G)$ for every local addition operator m, with $G = \langle \mathcal{A}, \mathcal{R} \rangle \in \mathcal{F}_{\mathcal{A}}$. This comes directly from the definition of conflict-free extension, since m is such that $m(\mathcal{R}) \supseteq \mathcal{R}$. □

Corollary 3.6. *Any local addition operator m which is non-decreasing for an argumentation framework $G = \langle \mathcal{A}, \mathcal{R} \rangle \in \mathcal{F}_{\mathcal{A}}$ with respect to conflict-free sets is also invariant: $G \equiv_{cf} m(G)$*

Proof. We know from Proposition 3.5 that every local addition operator m is non-increasing with respect to the conflict-free sets, that is $G \sqsupseteq_{cf} m(G)$. If m is also non-decreasing, we have $G \sqsubseteq_{cf} m(G)$, and thus it is also invariant ($G \equiv_{cf} m(G)$). □

We conclude that an operator m preserves the semantics only if it adds attacks between arguments which already were in conflict. We define an invariant operator h for conflict-free sets with the following theorem.

Theorem 3.7 (Invariant for conflict-free sets)**.** *Let $G = \langle \mathcal{A}, \mathcal{R} \rangle \in \mathcal{F}_{\mathcal{A}}$ be an AF. We have $G \equiv_{cf} h(G)$ if and only if $\forall (a, b) \in h(\mathcal{R})$*

- $(b, a) \in \mathcal{R}$, *or*
- $(a, a) \in \mathcal{R}$, *or*
- $(b, b) \in \mathcal{R}$.

Proof. We show that all conflict-free extensions are preserved if the above conditions hold and vice versa.
"$\Longrightarrow$":
Let's suppose to have an h such that $G \equiv_{cf} h(G)$. If the conditions are not satisfied, then it would exist a relation in $h(\mathcal{R})$ between two arguments belonging to the same extension in $S_{cf}(G)$ and so $G \sqsupseteq_{cf} h(G)$. Indeed, if a and b are never in relation in G and do not attack themselves, then they are also in the same conflict-free extension. We therefore reach a contradiction.
"$\Longleftarrow$":
Suppose that the conditions hold. If $(b, a) \in \mathcal{R}$, then a and b are already in conflict and do not appear together in any conflict-free extension of G. Thus, no extension will be lost in $S_{cf}(h(G))$ when adding an attack between those arguments. On the other hand, if $(a, a) \in \mathcal{R}$ (or $(b, b) \in \mathcal{R}$) then the argument a (b respectively) cannot be in and so it is not possible to change the conflict-free extensions set by adding an attack between a and b. □

If we consider any of the semantics of Definition 2.4 (Chapter 2), we can conclude that adding an attack between two arguments belonging to a certain set always requires (at least) one of those arguments to be removed from that set, changing the semantics in turn. Hence, denying attacks between nodes within the same set (which do not attack each other in G) is a necessary condition in order to leave the semantics unchanged in $h(G)$.

3.4 Invariant Operators for Admissible Sets

Since arguments can be defended and consequently accepted with respect to a certain extension, we need to consider different types of interaction in order to find an operator capable of maintaining the semantics unchanged. Reinstatement labelling provides a powerful means to overcome the issue of comprehending how arguments defend each other inside an extension. Indeed, labellings are a more expressive way than extensions to suggest the acceptance of arguments. We exploit the notion of in, out and undec arguments to define the invariant operator h for the admissible sets. In order to preserve this semantics, we have to guarantee that neither existent extensions will be destroyed, nor new one will be created. To achieve this, an operator h has to ensure that extensions in the set remain conflict-free, in arguments are not defeated from outside and out and undec arguments do not become acceptable. We distinguish between modifications that contract the semantics from modifications that expand it and we give the conditions under which an operator does not allow to perform either kind of change. One of the key points for preserving the in arguments in admissible extensions is to consider the sequences of attacks with even length, which correspond to defence paths.

Theorem 3.8. *Let $G = \langle \mathcal{A}, \mathcal{R} \rangle$ be an AF. A local addition operator h is non-decreasing with respect to the admissible set if and only if $\forall (a, b) \in h(\mathcal{R})$, there does not exist a labelling $\mathcal{L}$ of G such that*

- *$a, b \in in(\mathcal{L})$ or*
- *$a \in out(\mathcal{L})$, $b \in in(\mathcal{L})$, $(b, a) \notin \mathcal{R}$ and either $\nexists c \in out(\mathcal{L})$ such that $(c, b) \in \mathcal{R}$ or there exists a disjoint maximal[6] sequence of attacks towards b in which no argument $c \in in(\mathcal{L})$ is such that $(c, a) \in \mathcal{R}$ or*
- *$a \in undec(\mathcal{L})$, $b \in in(\mathcal{L})$.*

Proof. We have to show that if $G \sqsubseteq_{adm} h(G)$ then the condition holds and vice versa.
"$\Longrightarrow$":

[6] Disjoint with respect to edges of the graph G, maximal with respect to the number of attacks.

An operator h is non-decreasing with respect to admissible sets. Suppose that there exists an attack relation $(a, b) \in h(\mathcal{R})$ such that in some labelling $\mathcal{L}$ of G we have $a, b \in in(\mathcal{L})$ or $a \in undec(\mathcal{L}), b \in in(\mathcal{L})$. In both these cases, the admissible extension corresponding to the labelling $\mathcal{L}$ is lost in $h(G)$ and thus $h(G) \sqsubset_{adm} G$, so we have a contradiction. When $a \in out(\mathcal{L})$, $b \in in(\mathcal{L})$ and $(b, a) \notin \mathcal{R}$ we have two cases: if $\nexists c \in out(\mathcal{L})$ such that $(c, b) \in \mathcal{R}$, then the extension containing the only argument b would be lost. In the other case we have that there exists at least one sequence of attacks with even length that ends in b. If no argument in this sequence attacks a, then the admissible extension composed of b and all the other in arguments in the sequence is no longer admissible in h(G). Both cases lead to a contradiction.
"$\Longleftarrow$":
If the condition holds, it is not possible that an extension in $S_{adm}(G)$ is also in $S_{adm}(h(G))$. Consider any labelling $\mathcal{L}$ of G. If a is in or undec and b is not in, then the addition of an attack $a \rightarrow b$ cannot make an admissible extension of G to become unacceptable in $S_{adm}(h(G))$. If instead a is out, it means that it is already defeated, so every argument b belonging to some admissible extension of G remains acceptable with respect to such extension also in $h(G)$. □

Theorem 3.9. *Let $G = \langle \mathcal{A}, \mathcal{R} \rangle$ be an AF. A local addition operator h is non-increasing with respect to the admissible set if and only if $\forall (a, b) \in h(\mathcal{R})$, there does not exist a labelling $\mathcal{L}$ of G such that*

- *$a, b \in in(\mathcal{L})$ and $\exists c \in out(\mathcal{L})$ such that $(a, c) \notin \mathcal{R}$ and $(b, c) \in \mathcal{R}$ or*
- *$a \in in(\mathcal{L})$, $b \in out(\mathcal{L})$ and $\exists c \in in(\mathcal{L})$ such that $(b, c) \in \mathcal{R}$ or*
- *$a \in in(\mathcal{L})$, $b \in undec(\mathcal{L})$ and $\exists c \in undec(\mathcal{L})$ such that $(c, c) \notin \mathcal{R}$ and $(b, c) \in \mathcal{R}$ or*
- *$a \in out(\mathcal{L})$, $b \in in(\mathcal{L})$, there is an odd length sequence of attacks from b to a and $\nexists c \neq b$ such that there is an odd length sequence of attacks from c to a but not from a to c.*

Proof. We show evidence that no new admissible extensions are generated for G applying the operator h if the conditions of the theorem are satisfied and vice versa.
"$\Longrightarrow$":
Suppose that $h(G) \sqsubseteq_{adm} G$. If there exists a labelling $\mathcal{L}$ for which $a, b \in in(\mathcal{L})$ and $\exists c \in out(\mathcal{L})$ such that $(a, c) \notin \mathcal{R}$ and $(b, c) \in \mathcal{R}$ then arguments a and c would become acceptable together, forming a new admissible extension. The same would happen whenever $a \in in(\mathcal{L}), b \in out(\mathcal{L})$ and $\exists c \in in(\mathcal{L})$ such that $(b, c) \in \mathcal{R}$ or in the case $a \in in(\mathcal{L}), b \in undec(\mathcal{L})$ and $\exists c \in undec(\mathcal{L})$ such that $(c, c) \notin \mathcal{R}$ and $(b, c) \in \mathcal{R}$. If instead the last condition does not hold, then a would be defended from all the incoming

attacks and so it would be accepted in some admissible extension of $h(G)$. In all these cases we reach a contradiction.

"$\Longleftarrow$":

We will see that if the conditions hold, it is not possible that a new admissible extension can be generated. For every labelling $\mathcal{L}$ of G, a non-increasing operator h is allowed to add an attack between arguments a and b only in the following cases:

1. $a, b \in in(\mathcal{L})$ and $\nexists c \in out(\mathcal{L})$ such that $(a, c) \notin \mathcal{R}$ and $(b, c) \in \mathcal{R}$;
2. $a \in in(\mathcal{L}), b \in out(\mathcal{L})$ and $\nexists c \in in(\mathcal{L})$ such that $(b, c) \in \mathcal{R}$;
3. $a \in in(\mathcal{L}), b \in undec(\mathcal{L})$ and $\nexists c \in undec(\mathcal{L})$ such that $(c, c) \notin \mathcal{R}$ and $(b, c) \in \mathcal{R}$;
4. $a \in out(\mathcal{L})$ and there is no odd length sequence of attacks from b to a;
5. $a \in out(\mathcal{L})$ and $\nexists c \neq b$ such that there is an odd length sequence of attacks from c to a but not from a to c.
6. $a \in undec(\mathcal{L})$.

Item 4 means that b is not responsible for a being out, so the attack $a \rightarrow b$ is not sufficient to make a acceptable in a new admissible extension. In case 5, even if a defeats b, it will not become admissible without also defeating c. In all the remaining cases no argument can be defended by a (neither itself), thus no new admissible extensions can be obtained. □

Given Theorem 3.8 and Theorem 3.9, the following holds.

Corollary 3.10. *Let $G = \langle \mathcal{A}, \mathcal{R} \rangle$ be an AF. A local addition operator h is invariant with respect to the admissible set, and we write $G \equiv_{adm} m(G)$, if and only if $\forall (a, b) \in h(\mathcal{R})$, there does not exist a labelling $\mathcal{L}$ of G such that*

- *$a, b \in in(\mathcal{L})$, or*
- *$a \in in(\mathcal{L}), b \in out(\mathcal{L})$ and $\exists c \in in(\mathcal{L})$ such that $(b, c) \in \mathcal{R}$, or*
- *$a \in in(\mathcal{L}), b \in undec(\mathcal{L})$ and $\exists c \in undec(\mathcal{L})$ such that $(c, c) \notin \mathcal{R}$ and $(b, c) \in \mathcal{R}$, or*
- *$a \in out(\mathcal{L})$, $b \in in(\mathcal{L})$, $(b, a) \notin \mathcal{R}$ and either $\nexists c \in out(\mathcal{L})$ such that $(c, b) \in \mathcal{R}$ or there exists a disjoint maximal sequence of attacks towards b in which no argument $c \in in(\mathcal{L})$ is such that $(c, a) \in \mathcal{R}$ or*

- $a \in out(\mathcal{L}), b \in in(\mathcal{L})$, *there is an odd length sequence of attacks from* b *to* a *and* $\nexists c \neq b$ *such that there is an odd length sequence of attacks from* c *to* a *but not from* a *to* c, *or*
- $a \in undec(\mathcal{L}), b \in in(\mathcal{L})$.

Proof. The proof of this corollary is straightforward and comes from the proofs of Theorem 3.8 and Theorem 3.9. In particular, if a labelling $\mathcal{L}$ of G satisfying the properties above does not exist, then the local addition operator h is both non-decreasing (for Theorem 3.8) and non-increasing (for Theorem 3.9) with respect to the admissible semantics. Then h is invariant with respect to the admissible semantics, because the modification on the set of relations does not allow any change in the semantics. Vice-versa, if h is invariant with respect to the admissible semantics, then $G \sqsubseteq_{adm} h(G)$ and $G \sqsupseteq_{adm} h(G)$ must hold. If a labelling exists such that at least one of the given properties is satisfied, then h could be neither non-decreasing, nor non-increasing (or both), according to Theorem 3.8 and Theorem 3.9. Thus, such a labelling can not exist. □

We provide an example of how the conditions given in Corollary 3.10 allow to know in advance if a modification in an AF will change its semantics. Consider the AF in Figure 3.2. The addition of the following attacks do not change the admissible semantics.

- (e, c), indeed c does not attack any other argument, while a (which defends c from the argument b) also attacks e;
- (d, c), same considerations as before;
- (e, d), because d does not attack any other argument and it is never labelled in;
- (c, d), for same reasons of (e, d).

On the other hand, the modifications below change the set of admissible extensions.

- (a, c), because both arguments are in in some extension;
- (e, b), both e and b are in in some extension; moreover e defends the arguments c and d, forming a new extension;
- (c, e), indeed e alone cannot defend itself from c;
- (b, a), because b would defend itself against a, forming in this way a new extension.

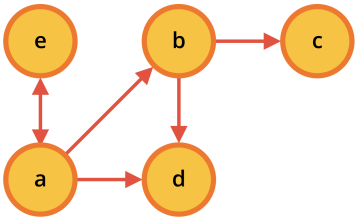
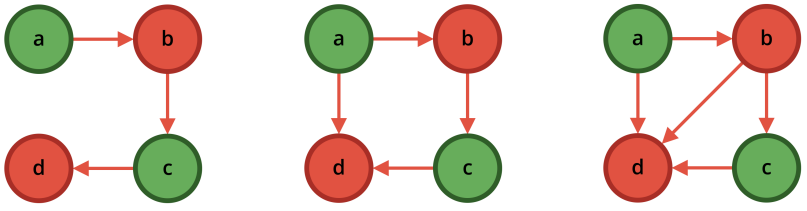


FIGURE 3.2: An example of an AF G for which $S_{adm}(G) = \{\{\}, \{a\}, \{e\}, \{a, c\}, \{b, e\}\}$. The depicted labelling corresponds to the extension $\{\}$.

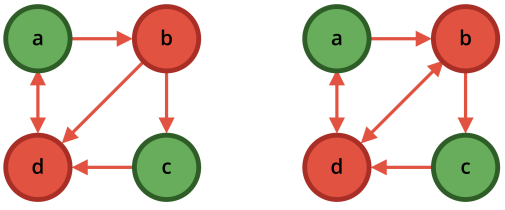
Remark 3.11. In order to determine if an operator m is invariant with respect to the admissible semantics, it is sufficient to consider only labelling in which $in(\mathcal{L})$ is maximal, that is the preferred extensions.

In fact, in non-maximal extensions, some arguments remain labelled undec even if they have different labels in more inclusive extensions (with respect to set inclusion). Thus, looking directly at the most inclusive extension allows for establishing rules able to preserve all the sets.

Invariant operators can be used as a metric to measure the robustness of AFs. The idea is that, starting from G, different invariant operators can be applied in sequence, until no more h exists for the last obtained AF: for example $h_4(h_3(h_2(h_1(G))))$ and no h_5 exists (as in Figure 3.3). Thus, the more operations are allowed for a framework, the more difficult it will be to change the extensions set for such semantics. We define the addition-based robustness of a graph for a generic σ as follows.



(a) G (b) $h(G)$ (c) $h(h(G))$



(d) $h(h(h(G)))$ (e) $h(h(h(h(G))))$

FIGURE 3.3: From figure (a) to (e), the AFs obtained starting from G and each time adding an attack through invariant operator h. The admissible extension $in(\mathcal{L}) = \{a, c\}$ persists.

Definition 3.12 (Local-addition robustness)**.** The local-addition robustness degree of an AF $G = \langle \mathcal{A}, \mathcal{R} \rangle$ with respect to a semantics σ is measured as the maximum number k of invariant operators h_i that can be applied on G such that $G \equiv_\sigma h_k(h_{k-1}(\dots(h_1(G)\dots))$.

3.5 Related Work

In the following of this section we review the most meaningful works related to what presented in this chapter.

Rienstra et al. [145] focus on finding conditions under which the evaluation of an AF remains unchanged when an attack is added or removed. The authors consider grounded, complete, preferred, stable and semi-stable semantics and, for each of them, a set of properties for which extensions are preserved is given. Those properties are in the form: "given a certain labelling, attacks between two arguments with labels X and Y respectively are allowed (or not) for the semantics σ". Invariance is intended with respect to a single extension and not with respect to the whole semantics (as we do). Given conditions works in two directions: existent extensions cannot be cancelled and new extensions cannot be created. In the latter case, invariant properties are defined only for arguments which have the sale labels in all σ labelling.

The problem of finding principles stating whether an extension does not change after adding/removing an attack between two arguments is also addressed by Boella et al. in [60] and [59]. Differently from us, the authors consider only the case in which the semantics of an AF contains exactly one extension, using the grounded semantics as example.

A general theory for handling dynamics in AFs is devised in [118], and extended in [16]. The proposed approach consists in dividing the modified AF in three different parts in which the arguments are: *i*) unaffected by the modifications, *ii*) affected, or *iii*) in relation with the affected arguments, respectively. On the one hand, this kind of division, made on a syntactic level, is different from our work on invariant operations, that is based on the acceptance status of the arguments. On the other hand, it could be interesting to understand how invariant operators behave with respect to the subsets identified through the division-based method of [118].

Cayrol et al [70] studied the impact on the evaluation of an AF when new arguments and attacks are added. They define a number of properties for the change operations according to how the extensions are modified. For instance, a change operation can be "conservative" if the set of extensions is the same after a change. Differently from our

work, the conditions under which the addition of argument does not change the semantics are not studied.

The work in [69] addresses the problem of revising AFs when a new argument is added. In particular, they focus on the impact of new arguments on the set of initial extensions, introducing various kinds of revision operators that have different effects on the semantics. For instance, *Decisive revision* allows for making a decision by providing a revised extensions set with a unique non-empty extension.

In [23] the problem of revising argumentation frameworks according to acquisition of new knowledge is taken into account. While attacks among the old arguments remain unchanged, new arguments and attacks among them can be added. In particular, the authors introduce the notion of *enforcing*, namely the process of modifying an AF (and possibly changing its semantics) in order to obtain a desired set of extensions. This notion departs from our work, in which we instead look for operations that leave the semantics unchanged.

Also Baumann introduces the concepts of update and deletion [22], focusing on modifications that retract arguments and attacks form an AF. New notions of equivalence are characterized through the so called kernels, namely functions that delete redundant attacks from a given framework. We instead concentrate on devising operators that permit both to modify AFs without changing their semantics, and to give a measure of how robust is a given AF, with respect to changes on the attack relations set.

The concept of *desire set* is also studied by Boella et al. in [58] with a work on persuasion in multi-agent systems, addressing the problem of choosing arguments to add into a system in order to maximise their acceptability with respect to the receiving agent. To this purpose, the notion of "more appealing" argument is introduced: in making the choice of a belief to add, an argument is more appealing than another if it does not interact with previous goals and beliefs of the agent. This has a different aim with respect to the work described in this chapter that consists in keeping unaltered the set of extension.

The authors of [77] show that every AF can be augmented in a normal form preserving the semantic properties. In such normal form no argument attacks a conflicting pair of arguments. A σ-*augmentation* is an alteration of an AF that leaves unchanged the semantics σ. The changes in the AF can involve arguments (the only allowed operation is the addition) and attacks.

A different and more restrictive kind of equivalence is introduced in [131]: two AFs G and G' are considered strongly equivalent to each other when they are equivalent after the conjunction with a third AF H (similarly to the notion of bisimulation [135] in state

transition systems). Since our intent is to provide a method for building equivalent AFs through the addition/deletion of attacks on a same framework, the notion of standard equivalence results to be more fitting than the strong equivalence.

The evolution of argumentation semantics is also studied in a series of papers [3–5], where the authors investigate how the sceptical acceptance of arguments, attacks or supports changes when a given ASAF (Attack-Support Argumentation Framework) is updated by adding or removing an element. The papers discuss an incremental algorithm for solving this problem. The evolution of the framework is represented through update functions that introduce changes into ASAFs. The authors identify sets of arguments whose acceptance status may change the update, but, differently from our, their work only concerns the preferred semantics and does not focus on specifically finding operations that are invariant for the set of extensions.

3.6 Conclusion

We defined invariant operators for AFs with respect to the semantics: these operators allow for performing changes on AFs while preserving the semantics. In particular, we have defined two operators, one for the conflict-free and one for the admissible sets, which can be applied to AFs for adding attack relations without resulting in changes to the set of extensions. The operators we have introduced exploit the notion of reinstatement labelling and thus can be applied without even being aware of the extensions admitted for a given semantics. Moreover, we gave a definition for the semantic equivalence between AFs, and we presented a method for computing the addition-based robustness degree of a framework.

Chapter 4

Ranking Arguments in AFs

"To prefer evil to good is not in human nature;
and when a man is compelled to choose one of two evils,
no one will choose the greater when he might have the less."
– Plato

Abstract

In this Chapter, we define a ranking-based semantics that relies on power indexes to refine the acceptability level of arguments in an AF by sorting them from the best to the worst, according to the contribution they bring to each extension. Aware of the fact that abstract frameworks are not sufficient to precisely instantiate problems coming from the real world, we also study the behaviour of ranking-based semantics in a setting where AFs are semi-structured: we use claim-augmented frameworks in which arguments are explicitly associated with the claims they stand for.

From the point of view of dynamics, it is useful to know which parts in a debate/dispute are the most valuable, since they can be involved in revision processes aimed at changing the semantics of the AF. In cooperative game theory [150], power-indexes (like the Shapley Value [150] and the Banzhaf index [13]) are used to evaluate the contribution of each member of a coalition. Following the intuition that arguments in an extension can be seen as players trying to form a winning coalition, we propose a ranking-based semantics that relies on power indexes and notions of coalition formation from cooperative games. Our semantics is parametric to a chosen power index and allows for obtaining a ranking where the arguments are sorted according to their contribution to the acceptability of the other arguments in the various coalitions.

The use of abstract arguments ease the process of instantiation an AF and provide basic reasoning tools for studying acceptability. However, there is common agreement [142] that abstract argumentation should not be treated as an isolated formalism but be embedded in an instantiation procedure which generates the arguments and the relation between them. Arguments in the abstract setting hence should be seen as placeholders for a more complex structure which at least contain a claim the argument stands for. The effect for traditional abstract argumentation semantics in this context has been thoroughly investigated [90, 91], where AFs are endowed with claims. Also complexity-theoretic implications have been pointed out [90]. To complete our study on ranking-based semantics, we consider the context of claim-augmented framework and we conduct a systematic analysis of ranking properties when claims are taken into account.

4.1 Using Game Theory Measures for Ranking Arguments

In this section, we provide a thorough study of a ranking-based semantics we devised using power-indexes as evaluation method to rank arguments in a given AF. First of all, we study our semantics with respect to a set of properties that are used in literature to compare the various existing ranking semantics [62]. This allows us for identifying the advantages of power indexes to rank arguments. To better characterize our semantics, we also discuss the relationship between the acceptability of an argument (in terms of sceptical/credulous acceptability) and the ranking resulting from the evaluation of such argument. Finally, we introduce a property linking the sceptical/credulous acceptability of two arguments with their rank.

4.1.1 Model Description

We give the definition of the ranking-based semantics, which we implement through the joint use of labelling and power indexes. Our approach consists in assigning a value to each argument according to the labels in and out if it satisfies the considered classical semantics. There is no convenience in taking into account also the label undec since it is derived directly from the other two. A further advantage of considering labelling-based semantics is that the characteristic functions only depend on the structure of a given AF, without adding to the picture other parameters, or external/computed values, or ad-hoc functions. Power indexes provides an a priori evaluation of the position of each player in a cooperative game, based on the contribution that each player can make to the different coalitions; in our ranking-based semantics, that we call "PI-based", such coalitions are consist of sets of arguments.

Definition 4.1 (Characteristic function)**.** Consider an AF $F = \langle A, R \rangle$, a Dung semantics σ and the set L_σ of all possible labellings on F satisfying σ. For any $S \subseteq A$, the labelling-based characteristic functions $v_\sigma^I(S)$ and $v_\sigma^O(S)$ are defined as:

$$v_\sigma^I(S) = \begin{cases} 1, & \text{if } S \in in(L_\sigma) \\ 0, & \text{if } otherwise \end{cases}$$

$$v_\sigma^O(S) = \begin{cases} 0, & \text{if } S \in out(L_\sigma) \\ 1, & \text{if } otherwise \end{cases}$$

The function $v_\sigma^I(S)$ takes into account the acceptability of a set of arguments S with respect to a certain semantics σ, assigning to such set a score equal to 1 if there exists a labelling L_σ in which all and only the arguments of S are labelled in. In other words, a set is positively evaluated by $v_\sigma^I(S)$ only if it represents an extension for the semantics σ, and the higher the score of the power index, the better the rank of an argument. A second characteristic function, $v_\sigma^O(S)$, is also introduced to put attention on the negative effect of the attacks received by the arguments. The function $v_\sigma^O(S)$ considers the sets of arguments labelled out by σ, and the evaluation has the usual interpretation: the lower the score according to $v_\sigma^O(S)$, the worse the rank.

Definition 4.2 (PI-based semantics)**.** Let $F = \langle A, R \rangle$ be an AF, σ a Dung semantics, $\pi \in \Pi : \{\phi, \beta\}$ a power index, and v_σ a characteristic function. The PI-based semantics associates to F a ranking $\succcurlyeq_\sigma^\pi$ on A, such that $\forall a, b \in A$,

$$a \succcurlyeq_\sigma^\pi b \iff \pi_a(v_\sigma) \geq \pi_b(v_\sigma)$$

The strict relation is derived in the usual way.

A ranking-based semantics designed in this way has the further advantage of automatically inheriting the properties of the Shaplye Value and the Banzhaf Index, like *efficiency*, *symmetry*, *linearity*, and *zero players* [150]. The lexicographic order on the pairs $(v_\sigma^I(S), v_\sigma^O(S))$ can be used to break possible ties in the final ranking, in the case two arguments of F have the same power index with respect to one of the two characteristic functions. An additional (partial) ordering can also be obtained as the Cartesian product of the two relations.

The PI-based semantics is capable of giving an overview of which are the most valuable arguments in a framework, from the point of view of their contribution to the existence of the various extensions belonging to different semantics. Indeed, it is reasonable to think that an argument which defend many other arguments should be given greater

importance, when looking for sets satisfying the admissible semantics. Giving another example, the highest ranked arguments according to the conflict-free criterion are those that generate less conflict together with all the other possible subsets. The possibility of having a different ranking for each Dung semantics allows for obtaining results that can be more fitting the ultimate purpose (i.e., reasoning about a favourable outcome of a debate) of the ranking. In the following section, we study ranking-based semantics properties with respect to our approach and we show which are satisfied and which are not.

4.1.2 PI-based Semantics Properties

We check which of the properties in [7] are satisfied by the semantics we introduced. In detail, we study which properties among those in Definition 2.10 are satisfied by the PI-semantics obtained through the Shapley Value and the Banzhaf Index, with respect to the various semantics and the two characteristic function of Definition 4.1.

Theorem 4.3. *Consider an AF $F = \langle A, R \rangle$, two arguments $a, b \in A$, a Dung semantics σ and a power index $\pi \in \{\phi, \beta\}$. The PI-based semantics satisfies the following properties:*

- ***Abs***, ***Ind*** *and* ***ToT*** *for any $\sigma \in \{$conflict-free, admissible, complete, preferred, stable$\}$*
- ***SC*** *only for $\sigma =$ conflict-free, with respect to v^I_σ.*
- ***NaE*** *only for $\sigma \in \{$complete, preferred, stable$\}$, with respect to v^I_σ, and for any $\sigma \in \{$conflict-free, admissible, complete, preferred, stable$\}$, with respect to v^O_σ.*

For any $\sigma \in \{$conflict-free, admissible, complete, preferred, stable$\}$, the PI-based semantics does not satisfy ***VP***, ***CP*** *and* ***QP***.

Proof. For each power index, characteristic function and semantics, we state if the properties are satisfied.

- **Abs**: Any extension of every semantics σ is computed starting from the set of attack relations among arguments, thus the ranking is preserved up to isomorphisms of the framework.
- **Ind**: The semantics we propose computes the ranking starting from the sets of extensions of a chosen semantics σ. Since the labelling of each argument a is determined by the other arguments in the same connected component of a, also the ranking between every pair of arguments a and b is independent of any other argument outside the connected component of a and b.

- **VP**: The PI-based semantics never satisfies **VP** for any $\sigma \in \{$conflict-free, admissible, complete, preferred, stable$\}$, with respect to v^I_σ and v^O_σ, and for both ϕ and β. Counterexamples are provided in Figures 4.1, 4.2, and 4.3.

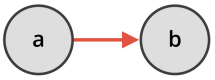


FIGURE 4.1: A counterexample for property **VP**. When σ = conflict-free, we have $a \simeq^\pi_{\sigma,I} b$ for $\pi \in \{\phi, \beta\}$.

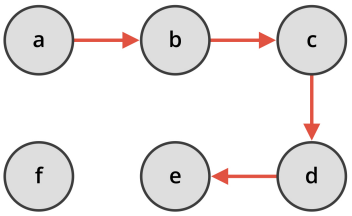


FIGURE 4.2: A counterexample for property **VP**. We have $c \simeq^\pi_{\sigma,I} f$ for $\pi \in \{\phi, \beta\}$ when $\sigma \in \{$complete, preferred, stable$\}$, and for $\pi = \beta$ also when $\sigma =$ admissible. Then, $c \simeq^\pi_{\sigma,O} f$ for $\pi \in \{\phi, \beta\}$ when $\sigma \in \{$admissible, complete, preferred, stable$\}$.

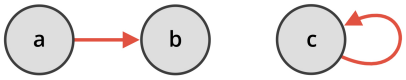


FIGURE 4.3: Counterexample for properties **VP** and **CP**: for $\pi \in \{\phi, \beta\}$ and $\sigma =$ conflict-free, we have $a \simeq^\pi_{\sigma,O} c$. A counterexample for property **SC**: for $\pi \in \{\phi, \beta\}$, we have $c \succ^\pi_{\sigma,I} b$ when $\sigma \in \{$admissible, complete, preferred$\}$, and $c \succ^\pi_{\sigma,O} b$ when $\sigma \in \{$conflict-free, admissible, complete, preferred$\}$. Moreover, $b \simeq^\pi_{\sigma,I} c$ and $b \simeq^\pi_{\sigma,O} c$ when $\sigma =$ stable.

- **SC**: Consider $\sigma =$ conflict-free. If $(a,a) \notin R$ and $(b,b) \in R$, we can state that

$$\exists E \subset A \wedge a \notin E : v^I_\sigma(E \cup \{a\}) - v^I_\sigma(E) > -1 \wedge$$
$$\nexists E \subset A \wedge b \notin E : v^I_\sigma(E \cup \{b\}) - v^I_\sigma(E) > -1$$

  Thus $\pi_a(v^I_\sigma) > \pi_b(v^I_\sigma)$ from which we conclude that $a \succ^\pi_{\sigma,I} b$ when $\sigma =$ conflict-free. In Figure 4.3 we show a counterexample for the other cases.

- **CP**: For both ϕ and β, the **CP** property is not satisfied for any $\sigma \in \{$conflict-free, admissible, complete, preferred, stable$\}$ with respect to v^I_σ and v^O_σ. Counterexamples in Figures 4.3 and 4.4.

- **QP**: For both ϕ and β, **QP** is not satisfied for any $\sigma \in \{$conflict-free, admissible, complete, preferred, stable$\}$ with respect to v^I_σ and v^O_σ. See Figures 4.5, 4.6 and 4.7 for counterexamples.

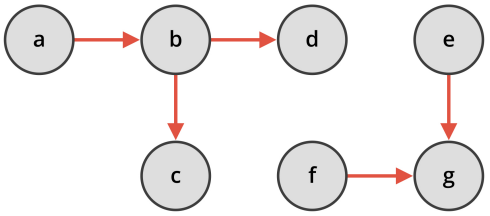


FIGURE 4.4: A counterexample for property **CP** of PI-based semantics. The argument g has more direct attackers than b. However, for $\pi \in \{\phi, \beta\}$, $g \succ^{\pi}_{\sigma,I} b$ when $\sigma =$ conflict-free, while $b \simeq^{\pi}_{\sigma,I} g$ and $b \simeq^{\pi}_{\sigma,O} g$ when $\sigma \in$ {admissible, complete, preferred, stable}.

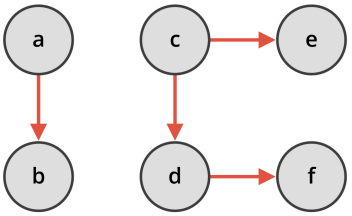


FIGURE 4.5: A counterexample for property **QP** of PI-based semantics: when $\sigma =$ conflict-free, $a \succ^{\pi}_{\sigma,I} c$ and $b \succ^{\pi}_{\sigma,I} d$ for $\pi \in \{\phi, \beta\}$. If $\sigma =$ admissible, we have $c \succ^{\phi}_{\sigma,I} a$ and $d \succ^{\phi}_{\sigma,I} b$, together with $c \succ^{\beta}_{\sigma,I} a$ and $b \succ^{\beta}_{\simeq,I} d$.

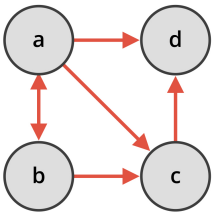


FIGURE 4.6: A counterexample for property **QP** of PI-based semantics. When $\sigma =$ complete, $b \succ^{\phi}_{\sigma,I} c$ and $a \succ^{\phi}_{\sigma,I} d$, while $b \succ^{\beta}_{\sigma,I} c$ and $a \simeq^{\beta}_{\sigma,I} d$. We also have that $b \succ^{\pi}_{\sigma,O} c$ and $a \succ^{\pi}_{\sigma,O} d$ when $\sigma \in$ {conflict-free, admissible, complete} for $\pi = \phi$, and only when $\sigma =$ conflict-free for $\pi = \beta$.

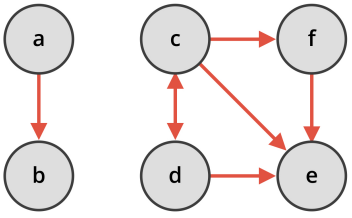


FIGURE 4.7: A counterexample for property **QP**. For $\pi \in \{\phi, \beta\}$, when $\sigma \in$ {preferred, stable}, $a \succ^{\pi}_{\sigma,I} c$ and $b \simeq^{\pi}_{\sigma,I} e$, as well as $a \succ^{\pi}_{\sigma,O} c$ and $b \simeq^{\pi}_{\sigma,O} e$ holds. When $\sigma =$ admissible, we also have that $a \succ^{\beta}_{\sigma,O} c$ and $b \succ^{\beta}_{\sigma,O} e$.

- **NaE**: Non-attacked arguments are labelled in in every complete extension, thus, for any F and $\pi \in \{\phi, \beta\}$, we have that if $a, b \in A$ are non-attacked, then $\pi_a(v^I_\sigma) = \pi_b(v^I_\sigma)$ and $\pi_a(v^O_\sigma) = \pi_b(v^O_\sigma)$. Hence $a \simeq^\pi_{\sigma,I} b$ and $\simeq^\pi_{\sigma,O}$, when $\sigma \in \{\text{complete}\}$. Since all the preferred and stable extensions are also complete, **NaE** holds for $\sigma \in \{\text{complete, preferred, stable}\}$. On the other hand, for $\sigma \in \{\text{conflict-free, admissible}\}$ the property is not satisfied (see the counterexample in Figure 4.8).

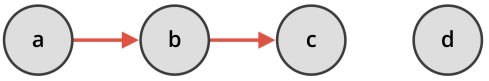


FIGURE 4.8: Counterexample for property **NaE** of PI-based semantics. For $\pi \in \{\phi, \beta\}$, when $\sigma =$ conflict-free, $d \succ^\pi_{\sigma,I} a$, and when $\sigma =$ admissible, $a \succ^\pi_{\sigma,I} d$.

- **ToT**: Any $\pi \in \{\phi, \beta\}$ associate a real number to every arguments of an AF, thus all pairs of arguments can be compared through the order of $\mathbb{R}$.

□

The validity of all the properties we take into account is summarised in Table 4.1: each cell of the table shows if a certain property is satisfied with respect to a power index (between ϕ, β), a Dung semantics (conflict-free, admissible, complete, preferred and stable) and characteristic function (v^I_σ and v^O_σ). The different characteristic functions are represented by alternating in and out rows. We then mark with ✓ the cells representing combinations of power index, semantics and characteristic function for which a property is satisfied, and with ✗ the cell for which it is not. Note that, given a semantics σ and a function v_σ, both the power indexes ϕ and β satisfy the same properties.

To show the correspondence with the classical semantics, in the last column, we also check the properties satisfied by the grounded semantics (that we consider as a degenerate ranking semantics with only two degrees of acceptability). We observe that the PI-based semantics is compatible with the grounded in terms of satisfied properties.

4.1.2.1 Evaluation of the Arguments

Given a ranking, we can correlate the value given to each argument to its credulous/sceptical acceptance. The questions we want to answer are: what does the value of an argument mean? What is the implication one can derive by comparing the values of two arguments? Looking at the acceptability of an argument, we can have in advance some information about the value of its evaluation, without even computing the power index.

		ϕ					β					GRD
		CF	ADM	COM	PRE	STA	CF	ADM	COM	PRE	STA	
Abs	in	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	out	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Ind	in	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	out	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
VP	in	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
	out	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
SC	in	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗
	out	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
CP	in	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
	out	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
QP	in	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
	out	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
NaE	in	✗	✗	✓	✓	✓	✗	✗	✓	✓	✓	✓
	out	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
ToT	in	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	out	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

TABLE 4.1: Properties of the PI-based semantics satisfied by ϕ and β.

Theorem 4.4. *Let $F = \langle A, R \rangle$ be an AF, σ a Dung semantics, $\pi \in \{\phi, \beta\}$ a power index, and v_σ a characteristic function.*

- *if a is sceptically accepted $\implies \pi_a(v_\sigma^I) > 0$;*
- *if a is credulously rejected $\implies \pi_a(v_\sigma^I) < 0$;*

Proof. The proof is straightforward and can be derived from the definition of the power indexes. □

The properties we studied in Theorem 4.3 are mainly related to the structural configuration of the AF (e.g., incoming attacks of the arguments) and do not represent a valid measure of how the contribution of the arguments is considered in the final ranking. In order to make up for this lack, we introduce a coalition formation-related property for ranking semantics.

Sceptically accepted arguments according to any labelling should be ranked higher than credulously accepted and rejected arguments. Analogously, credulously accepted arguments should be ranked higher that rejected arguments.

Definition 4.5. Let $F = \langle A, R \rangle$ be an AF, $a, b \in F$ two arguments, $\pi \in \{\phi, \beta\}$ a power index and σ a Dung semantics.

Sceptical Precedence (σ-SkP). If a is sceptically accepted with respect to σ, while b is not, than $a \succ_\sigma^\pi b$.

Credulous Precedence (σ-CrP). If a is credulously accepted with respect to σ and b is always rejected, than $a \succ_\sigma^\pi b$.

The following proposition holds.

Theorem 4.6. *The PI-based semantics satisfies σ-SkP and σ-CrP for any $\sigma \in \{$conflict-free, admissible, complete, preferred, stable$\}$ and the characteristic function v_σ^I.*

Proof. We give the proof for Theorem 4.6. Given an AF F, a credulously accepted argument i with respect to σ and an evaluation function v_σ^I, there exists at least one subset of arguments S such that $v_\sigma^I(S_{-i} \cup \{i\}) - v_\sigma^I(S_{-i}) > 0$. Thus the value of i will always be higher than that of any rejected argument j, for which $v_\sigma^I(S_{-j} \cup \{j\}) - v_\sigma^I(S_{-j}) < 0$ for any S, and σ-CrP holds. We can make the same consideration for σ-SkP, showing that sceptically accepted arguments have higher value than the others. □

We now discuss the above introduced properties, checking for which existing ranking-based semantics they hold. Besides our semantics, for this study we focus on those surveyed in [62] that return a total ranking, namely Cat, SAF, M&T, Dbs, Bds. Consider the framework in Figure 4.9: when σ = {admissible, complete, preferred, stable}, the argument d is always rejected, while e is credulously accepted. Moreover, e is also sceptically accepted by the complete, preferred and stable semantics. According to the CrP property, then, e should be ranked higher than d. For motivating such evaluation of the arguments, we can imagine that the AF of Figure 4.9 represents a debate we want to win. We know that e is defended by an argument that represent an initiator of the underlying graph (i.e., the argument b), while d is never in according to any Dung semantics. In other words, choosing e over d means to select an argument inside the grounded semantics, that we can consider "winning" or "difficult" to defeat. If we choose d, instead, we are not able to reply to the attack of e and so we are defeated.

Proposition 4.7. *The ranking-based semantics Cat, Dbs and Bds do not satisfy the σ-CrP property when $\sigma \in$ {admissible, complete, preferred, stable}.*

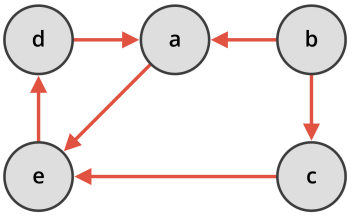


FIGURE 4.9: Example for σ-SkP and σ-CrP. We have $d \succ^{\sigma} e$ for $\sigma \in \{\text{Cat, Dbs, Bds}\}$.

Indeed, considering again the example in Figure 4.9, we have that d is always preferred to e in the rankings obtained by using Cat, Dbs and Bds respectively (see Table 4.2 for details).

Semantics	**Ranking**
Cat	$b \succ^{Cat} d \succ^{Cat} e \succ^{Cat} c \succ^{Cat} a$
Dbs	$b \succ^{Dbs} d \succ^{Dbs} c \succ^{Dbs} e \succ^{Dbs} a$
Bds	$b \succ^{Bds} d \succ^{Bds} c \succ^{Bds} e \succ^{Bds} a$
SAF	$b \succ^{SAF} e \succ^{SAF} d \succ^{SAF} c \succ^{SAF} a$
M&T	$b \succ^{M\&T} e \succ^{M\&T} c \simeq^{M\&T} d \succ^{M\&T} a$

TABLE 4.2: Ranking of the arguments of the AF in Figure 4.9 obtained by using the rankig-based semantics Cat, Dbs, Bds, SAF, M&T.

The categoriser function used by the Cat semantics only takes into account the value of the direct attackers of the arguments in the AF, so it is not possible to establish the importance a of particular argument with respect to the set of extensions of a certain semantics. Even if the CP property is not satisfied, the notion of "strength" that is used by Cat is not related to the acceptability of the arguments. Analogously, both the semantics Dbs and Bds, that satisfy CP and rely on the number of the paths ending to an argument, rank higher arguments that have has less direct attackers, without considering any notion of defence. This is the case of arguments d and e in Figure 4.9: while d is only attacked by a single argument, e is attacked by a and c, so $d \succ^{Dbs} e$ (and $d \succ^{Bds} e$) in the final ranking, although e is defended by the initiator b and d is attacked by e.

Since the authors of [99] assume the sceptical definition for the justification of the arguments, the graded semantics satisfies both σ-SkP and σ-CrP. Also the subgraph-based semantics [83] satisfies σ-SkP and σ-CrP. The ranking is obtained by establishing a lexicographical ordering between the values of a tuple that contains, for each argument a, the label assigned to a by a certain Dung semantics, and the number of times a is labelled l over the total number of subgraphs, for $l =$ in, out and undec, respectively. Although also our approach relies on reinstatement labelling, we omit arguments marked

as undecided in the computation of the ranking: indeed, the set of undec arguments can be obtained starting from the whole set of arguments and subtracting those labelled either in or out.

4.1.3 An Empirical Analysis

In this section, we show the procedure for obtaining a ranking among the arguments of a given AF through the use of a power index. We also compare the results for the Shapley Value and the Banzhaf Index, highlighting the differences in terms of final ordering of the arguments. For our example, we consider the AF in Figure 4.10, that has an initiator (i.e., the argument a, which is not attacked by any other argument), two symmetric attacks (both b, d, and d, e attack each other), and a cycle involving b, d and e.

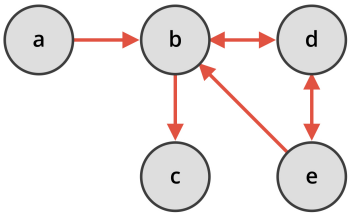


FIGURE 4.10: Example of an AF. The sets of extensions for the conflict-free, admissible, complete, preferred and stable semantics are: $CF = \{\emptyset, \{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \{a, c\}, \{a, d\}, \{a, e\}, \{c, d\}, \{c, e\}, \{a, c, d\}, \{a, c, e\}\}$, $ADM = \{\emptyset, \{a\}, \{d\}, \{e\}, \{a, c\}, \{a, d\}, \{a, e\}, \{c, d\}, \{c, e\}, \{a, c, d\}, \{a, c, e\}\}$, $COM = \{\{a, c\}, \{a, c, d\}, \{a, c, e\}\}$, and $PRE = STB = \{\{a, c, d\}, \{a, c, e\}\}$, respectively.

Below, we report the results for ϕ and β (that correspond to the functions for computing the Shapley Value and the Banzhaf Index, respectively), and the semantics conflict-free, admissible, complete and preferred. We omit the stable one since, in this example, it returns the same set of extensions as the preferred. For each semantics, the values of the power index obtained with respect to the sets of in and out arguments are alternated in each row. Tables 4.3 and 4.4 show the results for the aforementioned indexes.

We now analyse the differences between the obtained rankings, following two levels of detail: we first compare, for each power index, the ranking obtained for all the Dung semantics. Then, for each Dung semantics, we consider the ranking obtained with respect to the different power indexes.

In this example, the Shapley Value (Table 4.3), provides a ranking without indifferences when the conflict-free semantics is considered. While $\phi - com$, $\phi - pre$ and $\phi - stb$ return the same output, where in particular $c \succ d$ and $c \succ e$, the ranking for the admissible semantics gives an opposite interpretation, that is $d \succ c$ and $e \succ c$. This happens because both $\{d\}$ and $\{e\}$ are admissible extensions, while $\{c\}$ is not. Hence, when the admissible semantics is taken into account, d and e are better arguments than c.

TABLE 4.3: Ranking for the AF in Figure 4.10 obtained through the Shapley Value.

	a	**b**	**c**	**d**	**e**	Semantics	Ranking
v^I_{CF}	−0.05000	−0.46667	−0.05000	−0.21667	−0.21667	$\phi - CF$	$a \succ c \succ e \succ d \succ b$
v^O_{CF}	−0.35000	0.06667	−0.26667	−0.18333	−0.26667		
v^I_{ADM}	0.05000	−0.61667	−0.20000	−0.11667	−0.11667	$\phi - ADM$	$a \succ d \simeq e \succ c \succ b$
v^O_{ADM}	−0.31667	0.10000	−0.31667	−0.23333	−0.23333		
v^I_{COM}	0.11667	−0.13333	0.11667	−0.05000	−0.05000	$\phi - COM$	$a \simeq c \succ d \simeq e \succ b$
v^O_{COM}	−0.11667	0.30000	−0.11667	−0.03333	−0.03333		
v^I_{PRE}	0.06667	−0.10000	0.06667	−0.01667	−0.01667	$\phi - PRE$	$a \simeq c \succ d \simeq e \succ b$
v^O_{PRE}	−0.06667	0.10000	−0.06667	0.01667	0.01667		

TABLE 4.4: Ranking for the AF in Figure 4.10 obtained through the Banzhaf Index.

	a	**b**	**c**	**d**	**e**	Semantics	Ranking
v^I_{CF}	−0.06250	−0.68750	−0.06250	−0.31250	−0.31250	$\beta - CF$	$a \succ c \simeq e \succ d \succ b$
v^O_{CF}	−0.31250	0.06250	−0.18750	−0.06250	−0.18750		
v^I_{ADM}	0.06250	−0.68750	−0.06250	−0.18750	−0.18750	$\beta - ADM$	$a \succ c \succ d \simeq e \succ b$
v^O_{ADM}	−0.25000	0.12500	−0.25000	−0.12500	−0.12500		
v^I_{COM}	0.18750	−0.18750	0.18750	−0.06250	−0.06250	$\beta - COM$	$a \simeq c \succ d \simeq e \succ b$
v^O_{COM}	−0.18750	0.18750	−0.18750	−0.06250	−0.06250		
v^I_{PRE}	0.12500	−0.12500	0.12500	0.00000	0.00000	$\beta - PRE$	$a \simeq c \succ d \simeq e \succ b$
v^O_{PRE}	−0.12500	0.12500	−0.12500	0.00000	0.00000		

TABLE 4.5: Ranking for the arguments of the AF in Figure 4.10 obtained through the Deegan-Packel Index.

	a	**b**	**c**	**d**	**e**	Semantics	Ranking
v^I_{COM}	0.50000	0.00000	0.50000	0.00000	0.00000	$\rho - COM$	$a \simeq c \succ b \simeq d \simeq e$
v^O_{COM}	0.00000	0.00000	0.00000	0.00000	0.00000		
v^I_{PRE}	0.33333	0.00000	0.33333	0, 16667	0, 16667	$\rho - PRE$	$a \simeq c \succ d \simeq e \succ b$
v^O_{PRE}	0.00000	0.66667	0.00000	0.33333	0.33333		

When Banzhaf Index is used (Table 4.4), such an inversion of preferences never occurs: there is no semantics for which $d \succ c$ or $e \succ c$. Looking at the formulas of the Shapley Value ϕ (Equation 2.1) and the Banzhaf Index β (Equation 2.2), we can see that the only difference is the factor by which the gain $v(S \cup \{i\}) - v(S)$ is multiplied. Contrary to Shapley, Banzhaf does not consider the order in which the coalitions form; since the acceptability of the arguments does not depend on how the extensions are formed, β produces more consistent results and, therefore, is a more appropriate index to be used for building a ranking-based semantics.

TABLE 4.6: Ranking for the arguments of the AF in Figure 4.10 obtained through the Johnston Index.

	a	**b**	**c**	**d**	**e**	Semantics	Ranking
v^I_{CF}	0.00000	−3.16667	0.00000	−2.50000	−2.50000	$\gamma - CF$	$a \succ c \succ e \succ d \succ b$
v^O_{CF}	−2.50000	1.00000	−2.00000	−0.50000	−1.50000		
v^I_{ADM}	1.00000	−6.16667	0.00000	−1.50000	−1.50000	$\gamma - ADM$	$a \succ c \succ d \simeq e \succ b$
v^O_{ADM}	−2.00000	2.00000	−2.00000	−1.00000	−1.00000		
v^I_{COM}	1.50000	−1.16667	1.50000	−0.50000	−0.50000	$\gamma - COM$	$a \simeq c \succ d \simeq e \succ b$
v^O_{COM}	−2.00000	3.00000	−2.00000	−1.00000	−1.00000		
v^I_{PRE}	0.66667	−0.66667	0.66667	−0.16667	−0.16667	$\gamma - PRE$	$a \simeq c \succ d \simeq e \succ b$
v^O_{PRE}	−1.00000	1.00000	−1.00000	−0.50000	−0.50000		

Using the Deegan-Packel Index for computing the ranking with respect to the conflict-free and the admissible semantics is not meaningful. Indeed, for such semantics, the empty set $\emptyset$ is always an extension, and it also represents the only minimal winning coalitions. Since ρ relies on the set of minimal winning coalitions $M(v)$, when $\emptyset$ is the only element of $M(v)$, all the arguments receive a value of 0, according to Equation 2.3. For this reason, we omit to include $\rho - CF$ and $\rho - ADM$ in Table 4.5.

The ranking obtained through both the power indexes share some common features, that we discuss below. The argument a, that is not attacked by any other, is always in the first position of the rank, for every power index. Consequently, the argument b, that is attacked by a, always results to be the worst argument in the AF, excepted for indifferences. For the complete, preferred and stable semantics, the ranking does not distinguish between a and c, and between d and e. Indeed, the set a, c corresponds to the grounded semantics, that is a and c are equally "important" and should be evaluated the same. Similarly, e and d, that only appear in two distinct maximal admissible sets, receive the same value from both the power indexes. Finally, since the extensions of the preferred and the stable coincide, these two semantics always provide the same final ranking.

4.1.3.1 Comparison of Ranking-Based Semantics

Now we show and discuss the main differences between the PI-based semantics and the ranking-based semantics in [62]. Looking at the various rankings provided for the same AF in Table 4.2 and Table 4.7, we, first of all, notice that $\phi - CF$ and $\beta - CF$, as well as Cat, Dbs and Bds, are not able to capture the reinstatement of the arguments. For this reason, argument e, that is defended by the initiator b, is still considered worse than c or d. Then, we have $d \succ^{Cat} c$ and $d \succ^{Dbs} c$, but also $c \succ^{\phi}_{CF} d$. Indeed, according to our

evaluation, d should not be better that c because it contributes more than c in forming out extensions, that is sets of out labelled arguments.

Semantics	**Ranking through ϕ**	**Ranking through β**
CF	$b \succ c \succ d \succ e \succ a$	$b \succ c \simeq d \succ e \succ a$
ADM	$b \succ e \succ d \succ a \simeq c$	$b \succ e \succ d \succ a \simeq c$
COM	$b \simeq e \succ a \simeq c \simeq d$	$b \simeq e \succ a \simeq c \simeq d$

TABLE 4.7: Ranking of the AF in Figure 4.9 with the PI-based semantics.

For the admissible and complete semantics, the ranking obtained through power indexes takes into account the notion of defence, and so e is evaluated as the best argument after b. Also the SAF and M&T semantics assign to e the second best place in the ranking, after b. On the other hand, M&T does not distinguish between c and d, while $\phi - ADM$, $\beta - ADM$ and SAF produce the ranking $d \succ c$, justified by the fact that c is directly attacked by b and d is not. Notice also that arguments b and e are indifferent for $\phi - COM$ and $\beta - COM$, because both are necessary to obtain a complete extension. As a last observation, all the considered semantics agree on a never being preferred to any other argument. Similarly to ours, the subgraph-based semantics LSI of [83] is parametric to a chosen Dung semantics. Below, we compare it to the PI-based semantics, showing the differences in the obtained rankings. Considering the AF in Figure 4.11(a) and the grounded semantics, we have $b \simeq^{LSI} c \succ^{LSI} a$ and $a \simeq^{\phi}_{GDE} b \simeq^{\phi}_{GDE} c$. The PI-based semantics gives the same position to all the arguments, since none of the them contributes to form the grounded extension (that indeed is empty for the considered AF). In Figure 4.11(b), instead, the rankings are: $a \succ^{LSI} c \succ^{LSI} b$ and $a \simeq^{\phi}_{GDE} c \succ^{\phi}_{GDE} b$. Again, since a and c are both necessary for obtaining the grounded extension, they are given the same value.

(a) The grounded semantics for the depicted AF is GDE $= \emptyset$.

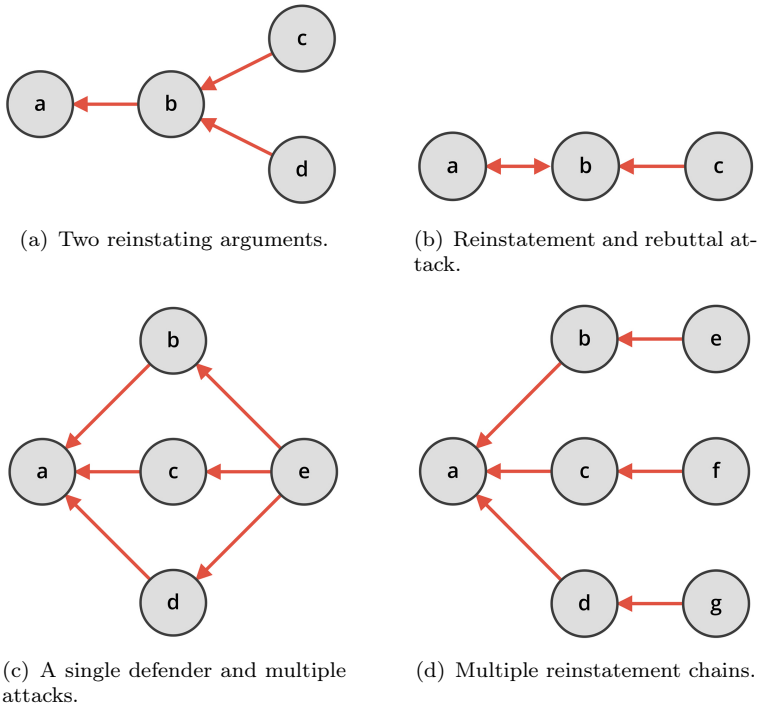
(b) The grounded semantics for the AF in figure is GDE $= \{a, c\}$.

FIGURE 4.11: Example of two AFs for which we provide the grounded extension.

4.1.3.2 A Comparison Over Different Instances

To better understand how the structure of an AF affects the score assigned to an argument, we study the behaviour of the PI-based semantics on some significant instances of AFs given in [83]. These instances represent various possible configurations of attacks

and defences, with the respective rebuttal and reinstatement of arguments. Below, we present some remarks on the 15 AFs configurations we have analysed. The functions ϕ and β return equivalent rankings for the conflict-free, admissible, complete, preferred and stable semantics.



(a) Two reinstating arguments.

(b) Reinstatement and rebuttal attack.

(c) A single defender and multiple attacks.

(d) Multiple reinstatement chains.

FIGURE 4.12: Example of AFs with different rebuttal/restatement configurations.

The ranking function based on the Shapley Value considers, in the computation of the ranking, all the possible permutations of the arguments in a coalition. Since the extensions of any Dung semantics do not depend on a particular ordering of the arguments, this aspect of the ϕ formula is not relevant for establishing the final ranking. Therefore, the Banzhaf Index β represents a more appropriate method for evaluating the contribution of the arguments that form the extensions.

Comparing the ranking resulting from $\beta - CF$ and $\beta - ADM$ for the AF in Figure 4.12(b), we notice that the argument a, that is worse than c according to the conflict-free semantics, is indifferent from c when considering the admissible one. The difference is due to the fact that, in $\beta - ADM$, a is able to defend itself from the attack of b, and so it is reinstated as an admissible extension, while, following the conflict-free semantics, the attack of b is sufficient for making a less preferable than c.

In the AF of Figure 4.12(c), $\beta - CF$ ranks arguments b, c and d in a higher position than the others. Indeed, the Banzhaf Index computed with respect to the set of conflict-free extensions, assigns higher value to arguments that are less involved in attack relations (both as attackers and as attacked). On the other hand, for the admissible semantics,

where the notion of defence plays a fundamental role, arguments b, c and d are worse than e and a. Also in Figure 4.12(d), for $\beta - CF$, a is the worst argument of the framework, because it receives more attacks than any other. According to $\beta - ADM$, instead, the argument a, that is attacked and defended, has a higher position than b, c and d, that are not defended. Hence, we believe that the acceptability conditions of the admissible semantics are more suitable for computing a ranking over the arguments of an AF, that those of the conflict-free.

Considering the admissible semantics also provide a better evaluation for the reinstated arguments than the complete one. For instance, in Figure 4.12(a), $\beta - COM$ assigns the same score to a, c and d, since they appear together in the only complete extension, while the ranking returned by $\beta - ADM$ is $c \simeq d \succ a \succ b$. It is straightforward that the admissible semantics, which correctly captures the notion of defence, is more appropriate for computing the ranking.

Due to the small number of arguments, the complete, preferred and stable semantics share the same set of extensions (and thus produce the same ranking) in all the instances we take into account. Following the previous considerations, $\beta - ADM$ is the more reasonable ranking function among the PI-based semantics.

4.2 Ranking-Based Semantics from the Perspective of Claims

In this section, we provide a study towards an understanding of the functioning of ranking semantics when arguments are not considered to be purely abstract but where each argument stands for a particular claim. In such a setting, standard ranking semantics over arguments implicitly provide an order over claims; however, given the common situation that different arguments can stand for the same claim, it is evident that certain ambiguities arise: consider a framework with three arguments a, b, c, where arguments a and c stand for claim x and argument b stands for claim y (see Figure 4.13), and a ranking of arguments of the form $a \succ b \succ c$ is determined by some ranking semantics (note that such a ranking is not implausible: it might be the case that the support of x in argument a is more plausible than the support of x in argument c). What does the ranking $a \succ b \succ c$ of arguments then tell us when we are interested in a ranking of their claims? Is claim x more acceptable than claim y?

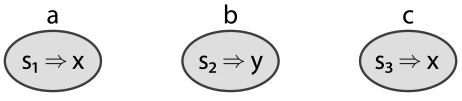


FIGURE 4.13: Arguments a, b and c with claims (x, y) and supports to the claims (s_1, s_2, s_3).

The main objectives are to propose and investigate a "lifting" of an argument-ranking to a claim-ranking. The idea of lifting relies on the intuition that a claim x is more acceptable than a claim y if there is at least one argument for claim x that is more acceptable than all arguments for claim y. In order to investigate the behaviour of such a lifting, we will reformulate several properties (originally proposed to classify argument-based ranking semantics) in a claim-centric perspective. The main insights of this work are to show that statements in the spirit of "for every argument-ranking semantics that satisfies property P, its lifted version satisfies the claim-centric variant of P" hold or are violated. We also study under which conditions such a lifting represents a Galois connection [63]. As a vehicle for these investigations we use claim-augmented argumentation frameworks (CAFs) as introduced in the complexity-study [90]. CAFs provide a natural intermediate layer between structured and purely abstract argumentation, as they carry the necessary information to first compute the extensions and then re-interpret them in terms of the instantiated problem. We extend our analysis also on important subclasses of such frameworks [90]. Notice that CAFs cannot only model the outcome of ASPIC style instantiations but is also applicable to e.g. instantiations from logic programming [90] and assumption-based argumentation (ABA) [61].

Indeed, there would be different ways to come up with a ranking on the claim-level. One would be to avoid the situation where different arguments are related to the same claim (since then, the ranking of claims is immediate from a ranking of arguments). Recently, translations towards such unique-claim frameworks have been investigated [91] and could be coupled with ranking semantics for frameworks with collective attacks as proposed in [168]. Another option would be to define new ranking semantics on claims from scratch, for instance, by taking the logical structure of claims also into account. In this preliminary study, we have opted for the lifting approach outlined above, since (a) it naturally builds on ranking semantics on the argument level which are well understood and (b) it provides first immediate insights on the relationship between rankings on argument- and claim-level which might be useful towards more special-tailored claim-ranking semantics.

4.2.1 CAFs Ranking and Properties

Our goal is to transfer the notion of ranking from classical AFs to CAFs. We call this mapping a *lifting* from arguments to claims. In particular, we are interested in checking whether the properties satisfied by a ranking-based semantics on AFs are preserved (on the level of claims) after the lifting. To conduct this study, we need the concept of ranking-based semantics for CAFs in the first place and then discuss how properties are lifted to the claim level.

Definition 4.8 (Ranking-Based Semantics for CAFs)**.** A *claim-based ranking semantics* associates with any CAF CF a total pre-order $\succcurlyeq_{CF}$ on X_{CF}, called the ranking on CF. $x \succcurlyeq_{CF} y$ means that claim x is at least as acceptable as claim y in CF.

Although many different criteria can be used for sorting the claims of a CAF from the best to the worst, we reshape the properties around some basic concepts we consider to be reasonable for a claims ranking (at least in the context of this study). First of all, we suppose that the ranking of the claims is solely be based on the ranking of their supporters, so claims with strong arguments are ranked higher. Since non-attacked arguments are the strongest ones, a claim supported by a non-attacked argument is better than every claim that only has attacked supporters. Moreover, a claim supported by some strong argument is always better than a claim only supported by weak arguments, no matter their number. Finally, additional arguments supporting a claim always increase (or at least do not harm) its ranking.

In what follows, we define three sets of properties for claim-based ranking semantics to be satisfied. The first group (see Table 4.8) of such properties are concerned with the fundamental properties one expects from a lifting from the ranking of arguments of an AF to a ranking of claims of a CAF. First we require that if the support sets of two claims are comparable (via group comparison), we want that claims with (strictly) stronger support to be (strictly) stronger, see properties **SD** and **SSD**. Second we require that the ranking of a claim is strengthened by additional support. To this end, we define for a CAF $CF = (A, R, \mathit{claim})$ and claim $x \in X_{CF}$, the CAF $CF^{+x} = (A, R, \mathit{claim}')$ where some argument $a \in A$ with $\mathit{claim}(a) \neq x$ gets x as its claim, i.e. $\mathit{claim}'(a) = x$ and $\mathit{claim}'(b) = \mathit{claim}(b)$ for $b \in A \setminus \{a\}$, see property **GSD**. Notice, that (A, R) is unchanged in CF^{+x} and thus the ranking of arguments is not affected.

Support Dependency (**SD**)	$\forall$ CAFs CF, $x, y \in X_{CF}$: $A_x \succcurlyeq^G A_y \implies x \succcurlyeq y$
Strict SD (**SSD**)	$\forall$ CAFs CF, $x, y \in X_{CF}$: $A_x \succ^G A_y \implies x \succ y$
Generalized SD (**GSD**)	$\forall$ CAFs CF, $x \in X_{CF}$: $x \succcurlyeq_{CF} y \implies x \succ_{CF^{+x}} y$

TABLE 4.8: Basic properties for lifted claim-based ranking semantics.

The second group basically rephrases the properties for ranking-based semantics in such a way that the notion of attack on argument-level is replaced by the notion of attack on the claim-level (cf. Definition 2.20)[7]. The resulting properties are called claim-oriented and collected in Table 4.9. We also need adaption of γ-isomorphism and weakly connected components. First, an isomorphism between claims is a bijective function $\gamma_X : X \to X$. Given CAF $CF = (A, R, \mathit{claim})$, we also use $\gamma_X(CF)$ to denote the CAF (A, R, claim')

[7]These properties do not address the different natures of arguments and claims and thus not all of them are expected properties of claim-rankings. However, they are perfectly suited to study which properties are maintained by lifting argument rankings to the claim level.

where $claim'(a) = \gamma_X(claim(a))$ for all $a \in A$, and call CF and CF' to be γ_X-isomorph. Second, the notion of weakly connected components is extended to CAFs in the following way: the claim-connected components $cc_c(CF)$ of a CAF $CF = (A, R, claim)$ are the subset maximal sub-frameworks such that the involved claims are weakly connected via attacks between claims (note that each claim-connected component is thus the union of one or more connected components of (A, R)). Finally, for a CAF CF and a ranking $\preceq$ on X_{CF}, $\preceq^G$ denotes the associated group comparison over subsets of X_{CF}.

(**C-Abs**)	$\forall$ γ_X-isomorph CAFs CF, CF', $x, y \in X_{CF}$: $x \succcurlyeq_{CF} y \iff \gamma_X(x) \succcurlyeq_{CF'} \gamma_X(y)$
(**C-Ind**)	$\forall$ CAFs CF, $CF' \in cc_c(CF)$, $x, y \in X_{CF'}$: $x \succcurlyeq_{CF} y \iff x \succcurlyeq_{CF'} y$
(**C-VP**)	$\forall$ CAF CF, $x, y \in X_{CF}$: $\left(x^- = \emptyset \wedge y^- \neq \emptyset\right) \implies x \succ y$
(**C-SC**)	$\forall$ CAF CF, $x, y \in X_{CF}$: $\left(x \notin x^+ \wedge y \in y^+\right) \implies x \succ y$
(**C-CP**)	$\forall$ CAF CF, $x, y \in X_{CF}$: $\lvert x^-\rvert < \lvert y^-\rvert \implies x \succ y$
(**C-QP**)	$\forall$ CAF CF, $x, y \in X_{CF}$: $\left(\exists z \in y^- : \forall u \in x^- : z \succ u\right) \implies x \succ y$
(**C-CT**)	$\forall$ CAF CF, $x, y \in X_{CF}$: $y^- \succcurlyeq^G x^- \implies x \succcurlyeq y$
(**C-SCT**)	$\forall$ CAF CF, $x, y \in X_{CF}$: $y^- \succ^G x^- \implies x \succ y$
(**C-DP**)	$\forall$ CAF CF, $x, y \in X_{CF}$: $\left(\lvert x^-\rvert = \lvert y^-\rvert, (x^-)^- \neq \emptyset = (y^-)^-\right) \implies x \succ y$

TABLE 4.9: Properties for claim-based ranking semantics.

(**C-Abs**) The ranking of the claims is independent from the naming of the claims.

(**C-Ind**) The ranking between two claims x and y should be independent of any claim that is neither connected to x nor to y.

(**C-VP**) A non-attacked claim is ranked strictly higher than any attacked claim.

(**C-SC**) A self-attacking claim is ranked lower than any non-self-attacking claim.

(**C-CP**) The greater the number of direct attackers of a claim, the weaker the level of acceptability of this claim.

(**C-QP**) The greater the acceptability of one direct attacker for a claim, the weaker the level of acceptability of this claim.

(**C-CT**) If the direct attackers of y are at least as numerous and acceptable as those of x, then x is at least as acceptable as y.

(**C-SCT**) If (**C-CT**) is satisfied and either the direct attackers of y are strictly more numerous or acceptable than those of x, then x is strictly more acceptable than y.

(**C-DP**) For two claims with the same number of direct attackers, a defended claim is ranked higher than a non-defended claim.

The final group of properties provides an alternative to ones in Table 4.9. The intuition is that the simple replacement of attack between arguments by attack between claims might be too general. Take for instance, property **C-VP**: it applies only when each supporter

of claim x in CAF has no attacker ($x^- = \emptyset$) and claim y has at least one supporter being attacked ($y^- \neq \emptyset$). However, it also appears reasonable to apply this property when at least one supporter of x has no attacker, but all supporters of y are attacked. The resulting property is **AC-VP**. Likewise, we occasionally replace x^- (i.e. the set of claims attacking x) with $(A_x)^-$ (i.e. the set of arguments attacking the supporters of x). It is important to note that $(A_x)^-$ is different to the supporters of x^-, i.e. the set $A_{x^-} := \bigcup_{y \in x^-} A_y$; see Figure 4.14.

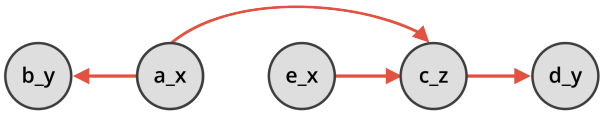


FIGURE 4.14: Example of a CAF where $(A_y)^- = \{a, c\}$ and $A_{y^-} = \{a, c, e\}$.

Table 4.10 presents those refinements where the ranking of claims is obtained from the arguments supporting the claim, and for certain properties (i.e., **AC-QP**, **AC-CT**, and **AC-SCT**) we even take into account that the claim ranking is obtained by lifting an argument ranking. Two things remain to be clarified. For **AC-Abs**, given a CAF $CF = (A, R, \mathit{claim})$, we use a pair of bijective functions $\gamma = (\gamma_A, \gamma_X)$ with $\gamma_A \colon A \to A$ and $\gamma_X : X \to X$. We also use $\gamma(CF)$ to denote the CAF $(A', R', \mathit{claim}')$ where $A' = \{\gamma_A(a) \mid a \in A\}$, $R' = \{(\gamma_A(a), \gamma_A(b)) \mid (a, b) \in R\}$ and $\mathit{claim}'(\gamma_A(a)) = \gamma_X(\mathit{claim}(a))$ for all $a \in A$, and call CF and CF' to be γ-isomorph. Second, for **AC-Ind**, the CAFs in $cc^*(CF)$ are obtained by the weakly connected components of AF $F = AF_{CF}$ together with the *claim*-function from CF restricted to the arguments in that component.

(**AC-Abs**)	$\forall$ γ-isomorph CAFs CF, CF', $x, y \in X_{CF}$: $x \succcurlyeq_{CF} y \iff \gamma_X(x) \succcurlyeq_{CF'} \gamma_X(y)$
(**AC-Ind**)	$\forall$ CAFs CF, $CF' \in cc^*(F)$, $x, y \in X_{CF'}$: $x \succcurlyeq_{CF} y \iff x \succcurlyeq_{CF'} y$
(**AC-VP**)	$\forall$ CAF CF, $x, y \in X_{CF}$: $\big(\exists a \in A_x : a^- = \emptyset \wedge \forall b \in A_y : b^- \neq \emptyset\big) \implies x \succ y$
(**AC-SC**)	$\forall$ CAF CF, $x, y \in X_{CF}$: $\big(\exists a \in A_x : a \notin a^+ \wedge \forall b \in A_y : b \in b^+\big) \implies x \succ y$
(**AC-CP**)	$\forall$ CAF CF, $x, y \in X_{CF}$: $\lvert(A_x)^-\rvert < \lvert(A_y)^-\rvert \implies x \succ y$
(**AC-QP**)	$\forall$ CAF CF, $x, y \in X_{CF}$: $\big(\exists a \in (A_y)^- : \forall b \in (A_x)^- : a \succ b\big) \implies x \succ y$
(**AC-CT**)	$\forall$ CAF CF, $x, y \in X_{CF}$: $(A_y)^- \succcurlyeq^G (A_x)^- \implies x \succcurlyeq y$
(**AC-SCT**)	$\forall$ CAF CF, $x, y \in X_{CF}$: $(A_y)^- \succ^G (A_x)^- \implies x \succ y$
(**AC-DP**)	$\forall$ CAF CF, $x, y \in X_{CF}$: $\big(\lvert(A_x)^-\rvert = \lvert(A_y)^-\rvert, ((A_x)^-)^- \neq \emptyset \wedge ((A_y)^-)^- = \emptyset\big) \implies x \succ y$

TABLE 4.10: Refined properties for claim-based ranking semantics.

4.2.2 Lifting via Lexicographic Order

In this section, we first propose a method for lifting a ranking on arguments to a ranking on claims based on a certain lexicographic order, and then investigate how this claim-based ranking relates to the properties introduced in the previous section. The general

idea is that every ranking criterion for CAFs that does not take into account the acceptability of the arguments in the AFs does not preserve acceptability-related properties after the transformation. Using a lexicographic order is based on the following intuition: if a claim x has one supporter that is better than all the supporters of another claim y, then $x \succ y$. In case the best supporters of x and y are equally acceptable, we look at the second best supporter and so on. Formally, this is captured as follows.

Definition 4.9 (Lexicographic Comparison)**.** Given a set S ordered through a preference relation $\succcurlyeq$, let $\max(S)$ return an element[8] $s \in S$ such that $\nexists t \in S, t \succ s$. We define the *lexicographic order* relation $\succcurlyeq^L$ (based on $\succcurlyeq$) between subsets of S as follows ($A, B \subseteq S$):

- $A \succcurlyeq^L \emptyset$, $\emptyset \not\succcurlyeq^L A$, for $A \neq \emptyset$
- $A \succcurlyeq^L B \iff i)\ \max(A) \succ \max(B)$, or
  $ii)\ \max(A) \succcurlyeq \max(B)$ and $A \setminus \max(A) \succcurlyeq^L B \setminus \max(B)$

As before, we write $A \succ^L B$ in case $A \succcurlyeq^L B$ and $B \not\succcurlyeq^L A$ jointly hold.

As we will show, $\succcurlyeq^L$ is always a refinement of the group-comparison $\succcurlyeq^S$. However, the concepts are clearly different, as illustrated in the example below.

Example 4.1. *Let $S = \{a, b, c\}$ and $a \succ b \succ c$. For the sets $A = \{a\}$ and $B = \{b, c\}$ we have that $A \succ^L B$ as $\max(A) = a \succ \max(B) = b$, but even $A \succcurlyeq^G B$ cannot hold since $|B| > |A|$. Note that also $B \succcurlyeq^G A$ cannot hold since each element in B is worse than a with respect to $\succcurlyeq$. In fact, when $\succcurlyeq$ is a total (pre-)order then also $\succcurlyeq^L$ is a total (pre-)order.*

We now define our central notion of lifting a ranking semantics.

Definition 4.10 (Lexicographic Order)**.** Given a ranking semantics σ that assigns to any AF F a ranking $\succcurlyeq_F$ on A_F, we call a claim-based ranking semantics σ' a *lex-lifting* of σ if for each CAF CF the ranking $\succcurlyeq_{CF}$ assigned by σ' satisfies:

$$\textbf{LO} : \text{for all claims } x, y \in X_{CF} \ : x \succcurlyeq_{CF} y \iff A_x \succcurlyeq^L_{AF_{CF}} A_y$$

Example 4.2. *Consider the ranking $a \simeq b \succ c \succ d \succ e \simeq f$ provided by a semantics σ for the arguments of a CAF $CF = \langle A, R, claim \rangle$. Consider, then, two claims x, y of CF, with $A_x = \{a, c, e\}$ and $A_y = \{b, d, f\}$. If we rank the claims through a semantics σ' satisfying **LO**, then we must have $x \succ y$. Indeed, even if the best supporters of x and y (namely a and b) are indifferent, x has a supporter (i.e., c) that is preferred to any other supporter of y for which does not exist a supporter of x with an equal position in the ranking.*

[8] If there are several such elements $s \in S$ then $\max(S)$ picks an arbitrary of these elements.

As argument rankings only provide the order of arguments according to their strength but no quantitative measure on the difference of their acceptance degrees there are strong limitations on how the strengths of the arguments supporting a claim and the number of arguments supporting a claim can be traded against each other when computing the claim ranking. We consider the strength to be more important, e.g., a claim supported by an unattacked argument should be ranked higher than an argument solely supported by (a large number of) self-attacking arguments. **LO** implements this intuition by first going for the strongest supporting arguments and the number of arguments only becomes relevant when the strongest supporting arguments tie. Alternatives to the lex-lifting would be to order claims by considering the minimum, maximum, or median strength argument supporting the claim. We expect that these approaches (when compared to the lex-lifting approach) will satisfy a smaller number of the analysed properties; a detailed comparison is subject of future work.

We now show which properties are satisfied for lex-liftings, in particular under the assumption that the underlying ranking semantics satisfies the corresponding property on the argument level. As we will see, satisfaction is sometimes conditioned by subclasses of CAFs (cf. Def 2.22). We start with properties from Table 4.8.

Proposition 4.11. *Every lex-lifting of a ranking semantics satisfies **SD**, **SSD**, and **GSD**.*

Proof. For **SD**, it suffices to show that for every ranking $\succcurlyeq$ on some set S, it holds that $\succcurlyeq^G \subseteq \succcurlyeq^L$. Hence suppose $A \succcurlyeq^G B$, for $A, B \subseteq S$. By definition, there is an injective mapping $f : B \to A$ such that $\forall b \in B$, $f(b) \succcurlyeq b$. Without loss of generality we can assume that f is monotone (since, if $f(x) \succ f(y)$ for some $y \succ x$, we can swap the values of $f(x)$ and $f(y)$), and that $f(\max(B)) = \max(A)$ by the same argument. Thus $\max(A) \succcurlyeq \max(B)$. In case $\max(A) \succ \max(B)$ we obtain $A \succ^L B$; otherwise, let $A' = A \setminus \max(A)$, $B' = B \setminus \max(B)$ and consider $f' : B' \to A'$ with $f'(x) = f(x)$ for all $x \in B'$. Since f is injective, f' is injective too, and by definition $\forall b \in B'$, $f(b) \succcurlyeq b$. We thus can continue this argument until the recursion comes to a halt or $B' = \emptyset$. The other two properties can be proven in a similar way.

For **SSD**, it suffices to show that for every ranking $\succcurlyeq$ on some set S, it holds that $\succ^G \subset \succ^L$. Suppose $A \succ^G B$ By definition, there is an injective mapping $f : B \to A$ such that $\forall b \in B$, $f(b) \succcurlyeq b$ and either (a) $|B| < |A|$ or (b) f additionally satisfies $f(b) \succ b$ for some $b \in B$. Again we can assume that f is monotone and that $f(\max(B)) = \max(A)$. By the above we have $A \succcurlyeq^L B$ and it remains to show that $B \not\succcurlyeq^L A$. If $\max(A) \succ \max(B)$, we immediately obtain $B \not\succcurlyeq^L A$. Otherwise, again we consider A', B', F' and continue the argument until (a) $B' = \emptyset$ or (b) eventually $f(b) \succ b$ for some $b \in B$, and obtain that $B \not\succcurlyeq^L A$.

For **GSD**, consider CAF $CF = (A, R, claim)$, AF $F = (A, R)$, $x, y \in X$ and $a \in A$ with $claim(a) \neq x$, and let us assume that $A_x \succcurlyeq^L A_y$. In order to satisfy **GSD** we have to show that $A_x \cup \{a\} \succ^L A_y \setminus \{a\}$. Let i be such that there are exactly i arguments $b \in A_x$ with $b \succcurlyeq a$ and let j be such that there are exactly j arguments $b \in A_y \setminus \{a\}$ with $b \succcurlyeq a$. Now considering the definition of $A_x \cup \{a\} \succ^L A_y \setminus \{a\}$. The first $min(i, j)$ comparisons are exactly the same as in the $A_x \succcurlyeq^L A_y$ test and thus succeed. If the recursion already terminated we have have $A_x \succ^L A_y$ and also $A_x \cup \{a\} \succ^L A_y \setminus \{a\}$. Otherwise consider $k = min(i, j)$ and let $b_k \succcurlyeq c_k$ be the arguments in the k-th comparison of the $A_x \succcurlyeq^L A_y$ test. We now either compare a with c_k and have $a \succ c_k$ (if $i < j$), b_k with c_{k+1} and have $b_k \succ c_{k+1}$ (if $i > j$), or a with c_{k+1} and have $a \succ c_{k+1}$ (if $i = j$). In all cases we obtain $A_x \succ^L A_y$. □

We now continue with the properties from Table 4.9 and their refined versions from Table 4.10. In each proposition, we will oppose a property and its refined version. In case a property does not hold (or only for some subclass of CAFs), corresponding counterexamples are given right after the proposition.

Proposition 4.12. *For every lex-lifting σ' of a ranking semantics σ it holds that:*

1. *σ' satisfies **C-Abs**; and*
2. *σ' satisfies **AC-Abs** whenever σ satisfies **Abs**.*

Proof. 1. holds since no matter how γ_X is chosen, we have $A_x = A_{\gamma_X(x)}$, and thus the property **LO** is not affected. For *2.* Consider a CAF $CF = (A, R, claim)$, let $\succcurlyeq_{CF}$ be the ranking assigned by σ, $\gamma = (\gamma_A, \gamma_X)$ be a pair of bijective functions $\gamma_A \colon A \to A$ and $\gamma_X \colon X_{CF} \to X_{CF}$, and $CF' = (A', R', claim')$ be a CAF γ-isomorphic to CF. Let $x, y \in X_{CF}$ and suppose $x \succcurlyeq_{CF} y$. By **LO**, we know that $A_x \succcurlyeq^L_{(A,R)} A_y$. Now as σ satisfies **Abs**, we also get $\gamma_A(A_x) \succcurlyeq^L_{(A',R')} \gamma_A(A_y)$, where $\gamma_A(S) = \{\gamma_A(a) \mid a \in S\}$. Further as γ is an isomorphism between CF and CF' we have $claim'(\gamma_A(a)) = \gamma_X(claim(a))$ for all $a \in A$. It follows that $\gamma_A(A_x) = A_{CF', \gamma_X(x)}$ and $\gamma_A(A_y) = A_{CF', \gamma_X(y)}$. Thus $\gamma_X(x) \succcurlyeq^L_{AF_{CF'}} \gamma_X(y)$, and by the **LO** property of σ' we arrive at $\gamma_X(x) \succcurlyeq_{CF'} \gamma_X(y)$. The reverse direction can be shown in essentially the same way and we obtain that σ' satisfies **AC-Abs**. □

Obviously, *2.* cannot be satisfied in general. Just consider $\gamma = (\gamma_A, \gamma_X)$ with γ_X the identity function. If σ does not satisfy **Abs**, σ' cannot satisfy **AC-Abs** then, since **Abs** and **AC-Abs** coincide in this setting.

Proposition 4.13. *For every lex-lifting σ' of a ranking semantics σ that satisfies **Ind** it holds that:*

1. *σ' satisfies **C-Ind**; and*

2. *σ' satisfies **AC-Ind** for CAFs being well-formed or att-unitary.*

Proof. 1. basically holds since the supporters of a claim in a CAF CF are guaranteed to occur in exactly one component $CF' \in cc_c(CF)$. From that **C-Ind** for σ' carries over from **Ind** for σ. Considering *2.*, since CF is well-formed or att-unitary, it is guaranteed that all supporters of a claim are contained in the same weakly connected component of AF_{CF} or all supporters of that claim are isolated (no incoming or outgoing attack, no self-attacks). In the former case, the argument of *1.* applies; for the latter, observe that **AC-Ind** does not pose any restriction on ordering those claims. □

If the CAF is neither well-formed nor att-unitary, we have no guarantee that **AC-Ind** is satisfied when lifting a semantics satisfying **Ind**. A counterexample is provided in Figure 4.15. Note that this CAF CF is neither well-formed (since, for instance, e does not attack d) nor att-unitary (since b is not attacked by a). Suppose to have the following ranking over the arguments: $a \simeq b \simeq c \succ d \succ e$, and let CF' the sub-CAF on the left-hand side (with arguments a, d). By **LO**, we have $x \succ_{CF'} y$ (observe that $a \succ d$ carries over to that sub-AF due to **Ind**) but $y \succ_{CF} x$ (since $max(A_x) = a \simeq b = max(A_y)$, following **LO**, it remains to compare d with $\{c, e\}$, where c is preferred over d).

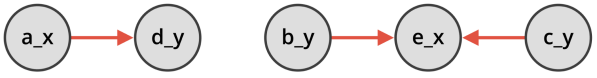


FIGURE 4.15: A CAF with two weakly connected components where lifting violates the property **AC-Ind**.

Proposition 4.14. *For every lex-lifting σ' of a ranking semantics σ that satisfies **VP** it holds that:*

1. *σ' satisfies **C-VP** for att-unitary CAFs; and*

2. *σ' satisfies **AC-VP**.*

Proof. Starting with *2.*, consider a CAF CF and two claims $x, y \in X_{CF}$, such that $\exists a \in A_x : a^- = \emptyset$ and $\forall b \in A_y : b^- \neq \emptyset$. We have to show that $x \succ y$. Since σ satisfies **VP** we know that $a \succ b$ for all $b \in A_y$. Hence, $\max(A_x) \succ \max(A_y)$ and thus $A_x \succ^L A_y$. By **LO**, $x \succ y$. Continuing with *1.*, for an att-unitary CAF CF, we know that for any claim y, with $y^- \neq \emptyset$, $b^- \neq \emptyset$ for all $b \in A_y$. The same reasoning as in *2.* can thus be applied. □

Now consider the well-formed CAF CF with arguments a, b, c, a attacking b and a supports claim x while b, c both support claim y. Assume further a ranking semantics σ that assigns $a \simeq c \succ b$ and thus satisfying **VP**. Via **LO** this is lifted to $y \succ x$ (since $\max(A_x) = a \simeq b = \max(A_y)$ and, by definition, $A_y \setminus \{b\} = \{c\}$ while $A_x \setminus \{a\} = \emptyset$, yields $A_y \succcurlyeq^L A_x$. On the other hand, $x^- = \emptyset$ and $y^- \neq \emptyset$, i.e., **C-VP** is violated.

Proposition 4.15. *For every lex-lifting σ' of a ranking semantics σ that satisfies **SC** it holds that*

1. *σ' satisfies **C-SC** for CAFs that are well-formed and att-unitary; and*
2. *σ' satisfies **AC-SC**.*

Proof. We start with *2.* and consider a CAF CF and two claims $x, y \in X_{CF}$, such that $\exists a \in A_x : a \notin a^+$ and $\forall b \in A_y : b \in b^+$. We have to show that $x \succ y$. Since σ satisfies **SC** we know that $a \succ b$ for all $b \in A_y$. Hence, $\max(A_x) \succ \max(A_y)$ and thus $A_x \succ^L A_y$; by **LO**, $x \succ y$. For *1.* it is sufficient to see that for CAFs that are both well-formed and att-unitary, an argument a with claim x is self-attacking iff all arguments with claim x are self-attacking. The argument from *2.* then applies here as well. □

We show that for CAFs that are either well-formed or att-unitary (but not both), we have no guarantee that **C-SC** is satisfied when lifting a semantics satisfying **SC**. Consider the CAF CF with arguments a, b, c, such that b attacks b. For the case of well-formed CAFs, consider additional attack (c, b); for att-unitary CAFs, consider instead (b, c). Moreover, a supports claim x and b, c both support claim y. Assume further that a ranking semantics σ assigns $a \simeq c \succ b$ to the AF thus satisfying **SC**. Via **LO** this is lifted to $y \succ x$ (since $\max(A_x) = a \simeq b = \max(A_y)$ and, by definition, $A_y \setminus \{b\} = \{c\}$ while $A_x \setminus \{a\} = \emptyset$, yields $A_y \succcurlyeq^L A_x$). On the other hand, $x \notin x^+$ but $y \in y^+$. Hence, **C-SC** is violated.

Proposition 4.16. *For every lex-lifting σ' of a ranking semantics σ that satisfies **CP** it holds that σ' satisfies **AC-CP** for att-unitary CAFs.*

Proof. Consider an att-unitary CAF CF and two claims $x, y \in X_{CF}$, such that $|(A_x)^-| < |(A_y)^-|$. We have to show that $x \succ y$. Note that each $a \in A_x$ has the same attackers, and the same is true for A_y. Hence, for each $a \in A_x$ and each $b \in A_y$, $|a^-| < |b^-|$. Since σ satisfies **CP** we know that $a \succ b$ for all $a \in A_x, b \in A_y$. Hence, $\max(A_x) \succ \max(A_y)$ and thus $A_x \succ^L A_y$, and by **LO**, $x \succ y$. □

Thus, not much can be established for **CP** property. In fact, we next show that **AC-CP** is not guaranteed for well-formed CAFs and **C-CP** is not guaranteed for CAFs that are

both well-formed and att-unitary when lifting a semantics satisfying **CP**. First, consider the CAF CF in Figure 4.16 and a semantics assigning a ranking to the underlying AF that satisfies **CP** and includes the relations $b \simeq c \simeq d \succ a$. Lifting this ranking yields $y \succ x$. On the other hand, we have $|(A_x)^-| = 3$ and $|(A_y)^-| = 6$ and **AC-CP**, that requires $x \succ y$, is thus violated.

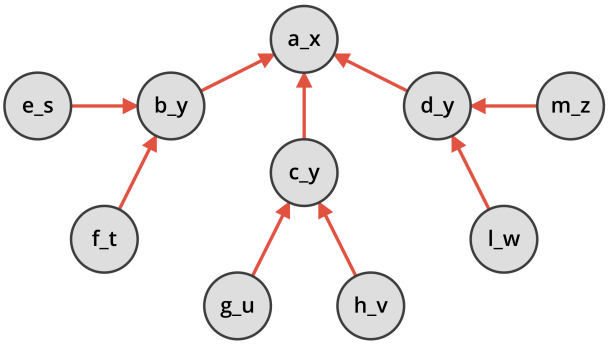


FIGURE 4.16: Example of a well-formed CAF where lifting violates **AC-CP**.

Second, consider the CAF in Figure 4.17 which is both well-formed and att-unitary, and the ranking $e \simeq f \succ b \simeq c \simeq d \succ a$ that satisfies **CP** on the underlying AF. It can be checked that **LO** yields $y \succ x$. However, on the level of claims we have $|x^-| = 1$ (the only claim attacking x is y) and $|y^-| = 2$. Thus, **C-CP** requires $x \succ y$.

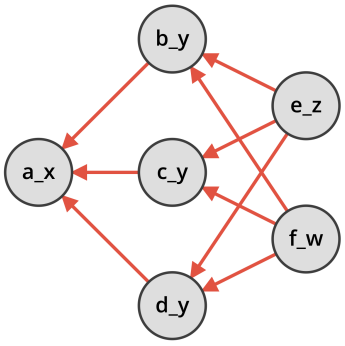


FIGURE 4.17: Example of a well-formed, att-unitary CAF where lifting violates **C-CP**.

Proposition 4.17. *For every lex-lifting σ' of a ranking semantics σ that satisfies **QP** it holds that σ' satisfies **AC-QP** for att-unitary CAFs.*

Proof. Consider a CAF CF and two claims $x, y \in X_{CF}$, such that $\exists c \in (A_y)^- : \forall d \in (A_x)^-, c \succ d$. We have to show that $x \succ y$. Since CF is att-unitary we have that $c \in b^-$ for each $b \in A_y$ and $d \in (A_x)^-$ if and only if $d \in a^-$ for $a \in A_x$. Hence, for each $b \in A_y$ and $a \in A_x$, it holds that $\exists c \in b^- : \forall d \in a^- : c \succ d$. Since σ satisfies **QP**, we have $a \succ b$ for each $b \in A_y$ and $a \in A_x$. It follows that $A_x \succ^L A_y$ and thus, by **LO**, $x \succ y$. □

We show that lifting to **AC-QP** does not work for well-formed CAFs. Consider the CAF CF depicted in Figure 4.18 and suppose to have a ranking on arguments such that $a \simeq e \succ d \succ c \succ b$. Such a ranking satisfies **QP** ($a \succ b \implies d \succ c$), while the depicted CAF is not att-unitary (as we can observe, c is attacked by a and e is not). With $a \in (A_y)^-$ we have an argument such that for all $b \in (A_x)^-$, $a \succ b$. Hence, **AC-QP** would require $x \succ y$. However, the lifting of the ranking via **LO** yields $y \succ x$ (since $e \succ d$).

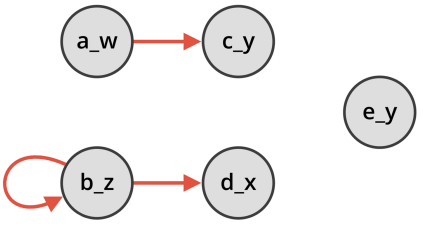


FIGURE 4.18: Example of a well-formed CAF where lifting violates **AC-QP**, **AC-SCT** and **AC-CT**.

Lifting to **C-QP** does not hold, even for CAFs that are both well-formed and att-unitary. Consider the CAF of Figure 4.21 and the ranking $a \simeq b \simeq d \succ f \succ c \simeq e$ that satisfies **QP**. Its lifting yields $z \succ u \succ y \succ x \simeq v$ (note that $|A_z| > |A_u|$, thus $z \succ u$). Now we have $z \in x^-$, $v^- = \{u\}$ and $z \succ u$ but $v \not\succ x$ and thus **C-QP** is violated.

Proposition 4.18. *For every lex-lifting σ' of a ranking semantics σ that satisfies **CT**, **SCT** respectively, it holds that σ' satisfies **AC-CT**, **AC-SCT** respectively, for att-unitary CAFs.*

Proof. Consider a CAF CF and two claims $x, y \in X_{CF}$, such that $(A_y)^- \succcurlyeq^G (A_x)^-$. We show $x \succcurlyeq y$. Since CF is att-unitary $b^- = (A_y)^-$ for all $b \in A_y$, and likewise, $a^- = (A_x)^-$ for all $a \in A_x$. Since σ satisfies **CT**, we obtain $a \succcurlyeq b$ for all $a \in A_x, b \in A_y$. It follows that $A_x \succcurlyeq^L A_y$ and thus, by **LO**, $x \succcurlyeq y$. The proof for **SCT** is analagous to the above. □

To show that lifting to **AC-SCT** does not hold for well-formed CAFs, we reuse the first CAF CF from Figure 4.18 and the ranking $a \simeq e \succ d \succ c \succ b$ satisfying **SCT** ($a \succ b \implies d \succ c$). Since, $(A_y)^- = \{a\}$ and $(A_x)^- = \{b\}$, **AC-SCT** requires $x \succ y$. However, recall that the lifting of the ranking via **LO** yields $y \succ x$. Note that the example applies also to **AC-CT**.

It remains to illustrate the problems with lifting to **C-CT**. To this end, consider the CAF CF depicted in Figure 4.19 which is well-formed and att-unitary, and a ranking of arguments $c \simeq d \simeq e \simeq f \simeq g \succ a \succ b$ (satisfying **SCT** and **CT**). Its lifting yields,

in particular, $x \succ y$. However, $x^- = \{v, w\}$ and $y^- = \{z\}$ and thus $x^- \succ^G y^-$ which requires $y \succ x$ (to satisfy **C-SCT**) or $y \succcurlyeq x$ (to satisfy **C-CT**).

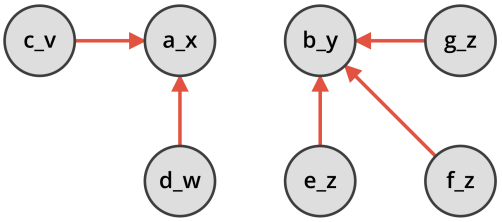


FIGURE 4.19: Example of a well-formed, att-unitary CAF where lifting violates **C-CT** and **C-SCT**

Proposition 4.19. *For every lex-lifting σ' of a ranking semantics σ that satisfies **DP** it holds that σ' satisfies **AC-DP** for att-unitary CAFs.*

Proof. Consider a CAF CF and two claims $x, y \in X_{CF}$, such that $|(A_x)^-| = |(A_y)^-|$ and $((A_x)^-)^- \neq \emptyset = ((A_y)^-)^-$. We show $x \succ y$. Since CF is att-unitary $a^- = (A_x)^-$ for all $a \in A_x$, and likewise, $b^- = (A_y)^-$ for all $b \in A_y$, and this observation carries over to $(a^-)^-$ and $(b^-)^-$. It follows that $|a^-| = |b^-|$ and $(a^-)^- \neq \emptyset = (b^-)^-$. Since σ satisfies **DP**, we obtain $a \succ b$ for all $a \in A_x, b \in A_y$. It follows that $A_x \succ^L A_y$ and thus, by **LO**, $x \succ y$. □

Lifting to **AC-DP** does not hold for well-formed CAFs: let CF be the CAF given in Figure 4.20 with ranking $a \simeq e \simeq f \succ d \succ b \simeq c$. This ranking satisfies **DP** and its lifting implies, in particular, $y \succ x$ since e is preferred over d. On the other hand, we have $|(A_x)^-| = |(A_y)^-| = 1$ and $((A_x)^-)^- = \{f\}$ while $((A_y)^-)^- = \emptyset$. **AC-DP** (which would thus require $x \succ y$) is therefore violated.

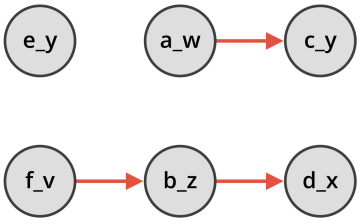


FIGURE 4.20: Example of a well-formed CAF where lifting violates **AC-DP**.

We finally show that lifting to **C-DP** does not hold, even for CAFs that are both well-formed and att-unitary. Consider the CAF of Figure 4.21 and ranking $a \simeq b \simeq c \simeq d \succ f \succ e$ on AF that satisfies **DP**. In particular, we have that $z \simeq x \succ y$. However, $|x^-| = |y^-| = 1$, and moreover, $(y^-)^- \neq \emptyset$ and $(x^-)^- = \emptyset$. By **C-DP**, we would need $y \succ x$; the property is thus violated.

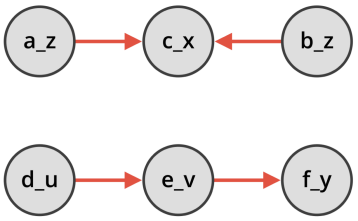


FIGURE 4.21: Example of a well-formed and att-unitary CAF where lifting violates **C-DP** and **C-QP**.

Table 4.11 summarises the results we obtained studying which properties holds for which classes of claims. WF and AU stands for well-formed and att-unitary, respectively.

	Abs	**Ind**	**VP**	**SC**	**CP**	**QP**	**CT**	**SCT**	**DP**
C-	All	All	AU	WF $\wedge$ AU	None	None	None	None	None
AC-	All	WF $\vee$ AU	All	All	AU	AU	AU	AU	AU

TABLE 4.11: Lex-lifting properties.

4.2.3 Galois Connection in CAFs

A Galois connection [63] is a correspondence between two partially ordered set, widely diffused in the field of abstract interpretation [76]. In particular, we are interested in monotone (i.e., order-preserving) Galois connection.

Definition 4.20 (Monotone Galois Connection)**.** Let $(A, \succcurlyeq)$ and $(B, \succcurlyeq)$ two partially ordered sets. A *monotone Galois connection* between these sets consists of two monotone functions $F\colon A \to B$ and $G\colon B \to A$, such that $\forall a \in A, b \in B$, $F(a) \succcurlyeq b \iff a \succcurlyeq G(b)$.

For non-att-unitary CAFs, $(claim, \succcurlyeq)$ and $(G, \succcurlyeq)$ do not represent a Galois connection. Consider as a counter-example the CAF in Figure 4.22, for which we have $claim(e) \succcurlyeq y$ and $e \not\succcurlyeq G(y)$.

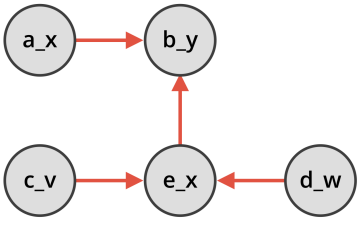


FIGURE 4.22: Well-formed, non-att-unitary CAF where $a \simeq c \simeq d \succ b \succ e$ and $x \simeq v \simeq w \succ y$ following **LO**.

We want to establish the condition under which a monotone Galois connection exists between the set of claims C and the set of arguments A of a CAF. An argument can

be univocally mapped into the claim it supports (by Definition 2.20), but more than one argument can be mapped into the same claim. Therefore, the functions we use for building the Galois connection are $claim\colon A \to C$ and $G\colon X \to A$, where *claim* is the function from Definition 2.20, while $G(x) = max(x_a)$ returns, for any claim $x \in C$, the best supporter of x. In general, the function *claim* is not monotone, as we can see in Figure 4.18 where $d \succ c$ and $claim(d) \not\succ claim(c)$.

Lemma 4.21. *Let $CF = \langle A, R, claim\rangle$ be a CAF with $claim\colon A \to C$. If CF is both well-formed and att-unitary, then claim is monotone.*

Proof. Consider a well-formed, att-unitary CAF CF and any pair of arguments $a, b \in A$ with $claim(a) = x$ and $claim(b) = y$. Since CF is well-formed and att-unitary, any $a \in x_a$ attacks the same set of arguments a^+ and is attacked by the same set of arguments a^-. Therefore, all the supporters of x will be assigned the same rank by any reasonable ranking-based semantics for AFs. The same happens for the claim y and if $a \succ b$, then also $claim(a) \succ claim(b)$ when the claims are ordered through **LO**. □

On the other hand, G is always monotone.

Lemma 4.22. *Let $CF = \langle A, R, claim\rangle$ be a CAF where $claim\colon A \to C$, and $G\colon X \to A$ be a function defined as $G(x) = max(x_a)$. Then G is monotone.*

Proof. Consider two claims $x, y \in C$ for which $x \succ y$ and suppose to have $G(y) \succcurlyeq G(x)$. If this is the case, it means that $G(y) \geq_L G(x)$ and thus either $max(y_a) \succ max(x_a)$, or $max(y_a) \succcurlyeq max(x_a)$ and $max(y_a \setminus max(y_a)) \geq_L max(x_a \setminus max(x_a))$. In both cases, we have, by Definitions 4.9, $y \succ x$, so we reach a contradiction. □

If we consider a well-formed and att-unitary CAF and a ranking $\succcurlyeq$ based on **LO**, we obtain a Galois connection. Indeed, all the arguments supporting a claim have the same rank and cannot be distinguished, so $\forall a \in A, x \in C,\ a \succcurlyeq G(x) \iff claim(a) \succcurlyeq x$.

Theorem 4.23. *Let $CF = \langle A, R, claim\rangle$ be a CAF in which arguments and claims are ordered by $\succcurlyeq_A$ and $\succcurlyeq_C$, respectively. If CF is both well-formed and att-unitary, the functions $claim\colon A \to C$ and $G\colon X \to A$, with $G(x) = max(x_a)$, represent a monotone Galois connection between A and C.*

Proof. We have that $(A, \succcurlyeq_A)$ and $(C, \succcurlyeq_C)$ are two partially ordered sets, and *claim* and G two monotone functions according to Lemmas 4.21 and 4.22. We need to show that, $\forall a \in A, x \in C,\ a \succcurlyeq G(x) \iff claim(a) \succcurlyeq x$.
$\Longrightarrow$) If $a \succcurlyeq G(x)$ the best supporter of x is not preferred to a, so $claim(a) \succcurlyeq x$.

$\Longleftarrow$) Since CF is well-formed and att-unitary, we know that $\forall a, b \in A$ such that $claim(a) = claim(b)$, $a = b$. If $claim(a) \succcurlyeq x$, then $\forall b \in x_a, a \succcurlyeq b$ and in particular a is preferred to the best supporter of x, that is $a \succcurlyeq G(x)$. □

4.3 Related Work

Several works can be found in the literature on ranking-based semantics, with different interpretations of the values associated to the arguments.

The authors in [28] propose a categoriser function that assigns a value to each argument, given the value of its direct attackers. Since only direct attackers are considered in order to compute the ranking, if an argument is attacked by two weak arguments, it is ranked below an argument that is attacked only once by a stronger argument. Differently from our approach, the categoriser does not consider the importance of the arguments, but the structure of the framework.

The semantics described in [68] is based on the principle that an argument is more acceptable if it can be preferred to its attackers. The authors take into account all the ancestor branches of an argument (defending and attacking) and compare their length. However, the produced ranking is only partial: for example, if an argument has strictly more attack branches and more defence branches than another one, then the two arguments are incomparable.

The authors in [116] introduce the formalism for Social Abstract Argumentation Frameworks, an extension of classical AFs, where arguments are associated with a value that represents the social support/vote. In the computation of the ranking, the strength of the attackers is more important than their number, and different methods can be employed for aggregating the votes altogether. This social model-based semantics requires exogenous information, not directly deductible from the relations among the arguments, and thus differs from our intention to provide an approach that can be used with classical settings.

Two different semantics are introduced in [7]: the Discussion-based semantics and the Burden-based semantics. The former, compares arguments by counting the number of paths ending to them; in case of a tie, the length of paths is recursively increased until a difference is found. The latter semantics assigns a *Burden Number* to every argument, which depends on the Burden Number of its direct attackers. In both the presented approaches, the number of attackers is more important than their strength, i.e., the notion of acceptability is not taken into account.

In [123], the strength of a given argument is computed by instantiating a non-cooperative game between a proponent and an opponent of that argument. In this model, defended arguments are stronger than non defended ones, although the final evaluation does not consider the contribution of each arguments in forming acceptable extensions, but is rather determined by the outcome of a fictitious two-person game. In addition, proponent and opponent choose mixed strategies, according to some probability distributions, that have to be fixed in advance.

The work in [10] introduces the concept of *contribution measure*, which evaluates the intensity of each attack in an argumentation framework, in order to establish the loss, in terms of acceptability, undergone by attacked arguments. The attackers of each argument can be then ranked from the most to the least harmful ones, according to their contribution measure. The Shapley Value is shown to be the unique measure that satisfies some crucial axioms, e.g., the independence between the contribution of an argument and its identifier. Also in this case, the notion of defence is not used for the evaluation of the contribution, while our idea of a ranking semantics relies on the acceptability status of arguments in the extensions.

The graded semantics proposed in [99] takes into account extensions of classical semantics in order to determine an ordering between arguments of an AF. The two principles on which the semantics is based are: having fewer attackers is better than having more; having more defenders is better than having fewer. The used order relation is only partial (and thus some of the arguments may be incomparable). Moreover, the ranking being built on the two principles mentioned above does not allow to catch the real contribution of the arguments in forming the extensions, that, instead, is the intention of the power index-based semantics.

Next, we discuss the ranking semantics, based on subgraphs analysis, introduced in [83]. This semantics produces a ranking by counting how many times a certain argument is accepted, rejected or undecided, according to the reinstatement labelling of [65]. The main difference with our approach is that, while we only consider acceptable extensions for obtaining the evaluation of an argument, the semantics in [83] uses all the possible subsets of arguments for computing the ranking, thus not considering the definition of the chosen Dung semantics.

Notions from cooperative game theory have been used in many Artificial Intelligence related works for estimating the contribution of a particular factor to a given phenomenon. For example, in [104] and [156], the Shapley Value is used for obtaining inconsistency measures for knowledge bases. In particular, the measure devised in [104] indicates the contribution of each formula of a belief base to the overall inconsistency of the base, while the author of [156] provide a Shapley Value-based measure that shows how inconsistency

is distributed on a probabilistic knowledge base, so to also identify the causes for the inconsistency. Similarly, in the PI-based semantics, the Shapley Value is used to identify the role that arguments play in forming extensions of a given semantics.

4.4 Conclusion

We introduced a general definition of power index-based semantics, by extending what presented in [46] that only considered the Shapley Value. We use this generalisation to compare how the Shapley value and the Banzhaf index satisfy some of the classical properties of ranking-based semantics in the literature. Differently from other ranking-based semantics defined in the literature, our approach allows for distributing preferences among arguments taking into account classical Dung/Caminada semantics. In this way, we obtain a more accurate ranking with respect to the desired acceptability criterion.

Then, we investigated ranking-based semantics in the context of CAFs, where claims constitute an extension to the abstract structure of the arguments, and we devised a method for lifting an argument-ranking to the level of the claims. To characterise such a lifting, we reformulated some of the classical properties for ranking-based semantics to make them suitable also for CAFs. In detail, we provided two interpretations for each property: one relies solely on claims, while the other takes into account arguments with the same claim. Table 4.11 summarises our results, i.e. which properties hold for which classes of CAFs after a lex-lifting. We suppose for each property of the claim-ranking, the corresponding property on the argument-ranking to hold.

Chapter 5

Extending Labellings

"*Arguments are to be avoided,*
they are always vulgar
and often convincing."
– Oscar Wilde

Abstract

Representation of AFs and their semantics is a crucial step for a sound and complete implementation of the real-world applications we are interested in. In this chapter we propose and discuss some possible characterisations that can be used to handle argumentation processes. In particular, we define a four-state labelling semantics which can be mapped into classical extensions, and we extend labelling-based semantics to also work with WAFs.

Reinstatement (see Definition 2.6) and four-state labelling (Definition 2.7) have both pros and cons. The labelling by Caminada does not allow to leave unlabelled arguments that we do not want to consider in computing acceptability and force all arguments that are neither in nor out to be labelled undec. Indeed, the set of arguments labelled in by the reinstatement labelling showed in Definition 2.6 always correspond to complete extensions (see Table 2.1), while four-state labelling does not necessary correspond to any particular extension. To overcome this problem, we establish a mapping between a modified four-state labelling and the classical semantics of Definition 2.4, which considers not only complete, but also admissible and conflict-free sets. We will use these notions for the semantic domain of the language in Chapter 6. We also we extend the notion of labelling to WAFs and we provide a definition that generalises the reinstatement labelling.

For each weighted semantics, we give the conditions under which a labelling corresponds to a set of extensions.

5.1 A Four-state Labelling Semantics

When considering reinstatement labelling to inspect AFs, the information brought by the undec label can be misleading. The labelling by Caminada, indeed, does not allow to leave unlabelled arguments that we do not want to consider in computing acceptability, and forces all arguments that are neither *in* nor *out* to be labelled *undec*. Consequently, any reinstatement labelling corresponds to a complete extension and cannot identify conflict-free and admissible sets. This inconvenience can be solved using the four-state labelling of Definition 2.7, which produces labellings as the one in Figure 5.1, where the fact of c not being in or undec does not depend on the structure of the framework, but rather on the choice of just ignoring it.

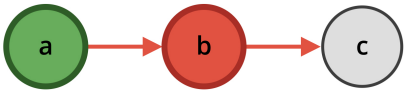


FIGURE 5.1: Example of a four-state labelling where argument c is empty.

Even though the four-state labelling is more informative than the reinstatement labelling (that does not comprehend an empty label), there is no direct connection between labellings and extensions of a certain semantics. To overcome this problem, in the following we establish a mapping between a modified four-state labelling and all the classical extension-based semantics (considering also admissible and conflict-free sets).

Definition 5.1 (Four-state labelling semantics)**.** Let U be a universe of arguments, $F = \langle Arg, R \rangle$ an AF with $Arg \subseteq U$ and $R \subseteq Arg \times Arg$ the arguments and attacks. $L : Arg \to 2^{\{\mathsf{in},\mathsf{out}\}}$ is a four-state labelling on F if and only if:

- $\forall a \in U \setminus Arg . L(a) = \mathsf{empty}$[9];
- $\forall a \in Arg$, if $\mathsf{out} \in L(a)$, then $\exists b \in Arg$ such that $(b, a) \in R$ and $\mathsf{in} \in L(b)$;
- $\forall a \in Arg$, if $\mathsf{in} \in L(a)$, then $\forall b \in Arg$ such that $(b, a) \in R$, $\mathsf{out} \in L(b)$;
- $\forall a \in Arg$, if $\mathsf{in} \in L(a)$, then $\forall c$ such that $(a, c) \in R$, $\mathsf{out} \in L(c)$.

The four labels form a lattice in which *undec* (i.e., the set $\{\mathsf{in}, \mathsf{out}\}$) is the top element and empty is the bottom. Moreover,

[9]Since arguments in $U \setminus Arg$ do not constitute an actual part of the AF, they are always labelled empty.

- L is a conflict-free labelling if and only if:
  - $L(a) = \{\mathsf{in}\} \implies \forall b \in Arg \mid (b,a) \in R. L(b) \neq \{\mathsf{in}\}$ and
  - $L(a) = \{\mathsf{out}\} \implies \exists b \in Arg \mid (b,a) \in R \wedge L(b) = \{\mathsf{in}\}$
- L is an admissible labelling if and only if:
  - $L(a) = \{\mathsf{in}\} \implies \forall b \in Arg \mid (b,a) \in R. L(b) = \{\mathsf{out}\}$ and
  - $L(a) = \{\mathsf{out}\} \implies \exists b \in Arg \mid (b,a) \in R \wedge L(b) = \{\mathsf{in}\}$
- L is a complete labelling if and only if:
  - $L(a) = \{\mathsf{in}\} \iff \forall b \in Arg \mid (b,a) \in R. L(b) = \{\mathsf{out}\}$ and
  - $L(a) = \{\mathsf{out}\} \iff \exists b \in Arg \mid (b,a) \in R \wedge L(b) = \{\mathsf{in}\}$
- L is a stable labelling if and only if:
  - L is a complete labelling and
  - $\nexists a \in Arg \mid L(a) = \{\mathsf{in}, \mathsf{out}\}$
- L is a preferred labelling if and only if:
  - L is an admissible labelling and
  - $\{a \mid L(a) = \{\mathsf{in}\}\}$ is maximal among all the admissible labellings
- L is a grounded labelling if and only if:
  - L is a complete labelling and
  - $\{a \mid L(a) = \{\mathsf{in}\}\}$ is minimal among all the complete labellings

Note that complete, stable, preferred and grounded labelling coincides with those in [65]. We can show there is a correspondence between labellings satisfying the restrictions given in the definition above and the extensions of a certain semantics. We use the notation $L \in S_\sigma(F)$ to identify a labelling L corresponding to an extension of the semantics σ with respect to the AF F.

Theorem 5.2. *A four-state labelling L of an AF $F = \langle Arg, R \rangle$ is a conflict-free (respectively admissible, complete, stable, preferred, grounded) labelling as in Definition 5.1 if and only if the set I of arguments labelled* in *by L is a conflict-free (respectively admissible, complete, stable, preferred, grounded) extension of F.*

Proof. We sketch the proof for the admissible labelling. The conflict-free case is obtained through a similar reasoning and the remaining can be constructed as in [65].

$\Rightarrow$) Consider an admissible labelling L on $F = \langle Arg, R \rangle$. We have to show that there are no $a, b \in I$ such that $(a, b) \in R$ and that each $a \in I$ is defended by I. First of all, arguments labelled in by L can only be attacked by out arguments, so for all $a, b \in I$ we have $(a, b) \notin R$. Then, if a is attacked by an argument b (which we know must be out) that argument is necessarily in turn attacked by at least one in. We conclude that I defends all its elements and therefore it is an admissible extension.

$\Leftarrow$) We have an admissible extension E composed of arguments labelled in by L, and we know that all arguments in E does not attack each other and are defended by E. Hence, in arguments of L cannot be attacked by other arguments with the label in. Finally, arguments that are attacked from E are out. □

The labelling of an AF gives information about the acceptability of the arguments in the framework (according to the various Dung's semantics) and can be used by intelligent agents to represent the state of their beliefs. Each different label can be traced to a particular meaning. For instance, empty stands for "don't care" [105] and identifies arguments that are not considered by the agents. Arguments in $U \setminus Arg$, that are only part of the universe, but not of the shared AF, are labelled with empty since they are outside the interest of the agents. Accepted and rejected arguments (labelled as in and out, respectively), allow agents to discern true beliefs from the false ones. At last, undec arguments possess both in and out labels, meaning that agents cannot decide about the acceptability of a belief ("don't know", indeed). In the next session, where we present our concurrent language for argumentation, the labelling of Definition 5.1 is used to implement both primitives and high level operations that rely on the acceptability state of agent's belief and are able to change the underlying knowledge base accordingly.

5.2 Weighted Labelling

In this section, we extend the four-state labelling to semiring-based argumentation frameworks and we provide a definition that is parametric to a chosen notion of defence (between $\mathbb{D}_1$, $\mathbb{D}_2$ and $\mathbb{D}_3$ of Definition 2.16) and that corresponds to the labelling of Definition 5.1 when a boolean semiring is used. For each weighted semantics, we give the conditions under which a labelling corresponds to a set of extensions. Considering classical AFs, we can use the condition in Definition 5.1 for assigning labels to the arguments in such a way that there is a correspondence between the labelling and the set of extension. We want to obtain the same result also for the weighted case. In order to incorporate the notion of weighted defence in the labelling, we need to take into account the strength of the attack relations.

According to the classical notion of defence, an argument a is defended from the attack of another argument b if there exists a third argument c that attacks b in turn. On the other hand, when a weight is assigned to the attacks, the previous condition cannot ensure alone that the argument a will be defended by c: it can be the case that the attack $c \to b$ is not strong enough to defeat $b \to a$ and thus to justify a. $\mathbb{D}_1$,$\mathbb{D}_2$ and $\mathbb{D}_3$ demand specific considerations for computing the (overall) weight of the attack relations that lead to different outcomes in terms of acceptable arguments. The attacks in [122], where $\mathbb{D}_1$ is used, are ordered by their strength and it is sufficient to compare the weight of two attacks to establish whether an argument is defended (and so labelled in) or not. Following $\mathbb{D}_2$ [74], we need to know the strength resulting from the composition $\bigotimes$ of all the attacks coming form the defending arguments towards the attacker. In particular, an argument b with label out is attacked by the arguments in $b^-|_{\mathsf{in}}$ with a total strength that is expressed by $W(b^-|_{\mathsf{in}}, b)$. According to the definition of collective weighted defence $\mathbb{D}_3$ given in [42], a set of argument is defended from an attacker b only if the $\bigotimes$ of all the defending arguments is stronger than the $\bigotimes$ of the attacks coming from b. This means that the strength of the attacks of the defending arguments is distributed among the defended arguments, so it is not guaranteed for two arguments that are separately w-defended to sill be w-defended when considered together. The intuition behind this representation is that when an argument a attacked by an out b cannot be labelled in because another in argument is "consuming" the attacks of the defending arguments towards b, then a is labelled empty.

In the following, we give a characterisation of the weighted semantics through the notion of labelling of $\text{WAF}_{\mathbb{S}}$.

Definition 5.3 (Labelling-based semantics for $\text{WAF}_{\mathbb{S}}$)**.** Let $F = \langle \mathcal{A}, \mathcal{R}, W, \mathbb{S} \rangle$ be a $\text{WAF}_{\mathbb{S}}$. A labelling L of F is a total function $L : Arg \to 2^{\{\mathsf{in},\mathsf{out}\}}$. For any $A \subseteq \mathcal{A}$, we denote $A|_{\mathsf{in}}$, $A|_{\mathsf{out}}$, $A|_{\mathsf{undec}}$ and $A|_{\mathsf{empty}}$ the set of all and only arguments labelled $\{\mathsf{in}\}$, $\{\mathsf{out}\}$, $\{\mathsf{in},\mathsf{out}\}$ and $\{\}$ by L, respectively. Moreover, let $a \in \mathcal{A}$. L is a

- w-conflict-free labelling for F if and only if:
  - $L(a) = \mathsf{in} \implies a^-|_{\mathsf{in}} = \emptyset$, and
  - $L(a) = \mathsf{out} \implies a^-|_{\mathsf{in}} \neq \emptyset$
- w-admissible labelling for F if and only if:
  - $L(a) = \mathsf{in} \implies a^- = a^-|_{\mathsf{out}} \ \wedge \forall b \in \mathcal{A}$ such that $R(b, a)$,
    - $\mathbb{D}_1$) $\exists c$ with $L(c) = \mathsf{in} \mid W(c, b) \leq_{\mathbb{S}} W(b, a)$, or
    - $\mathbb{D}_2$) $W(b^-|_{\mathsf{in}}, b) \leq_{\mathbb{S}} W(b, a)$, or
    - $\mathbb{D}_3$) $W(b^-|_{\mathsf{in}}, b) \leq_{\mathbb{S}} W(b, b^+|_{\mathsf{in}})$, and

  - $L(a) = \mathsf{out} \implies W(a^-|_{\mathsf{in}}, a) <_{\mathbb{S}} \top$

- w-complete labelling for F if and only if:
  - $L(a) = \mathsf{in} \iff a^- = a^-|_{\mathsf{out}} \wedge \forall b \in \mathcal{A}$ such that $R(b, a)$,
    $\mathbb{D}_1$) $\exists c$ with $L(c) = \mathsf{in} \mid W(c, b) \leq_{\mathbb{S}} W(b, a)$, or
    $\mathbb{D}_2$) $W(b^-|_{\mathsf{in}}, b) \leq_{\mathbb{S}} W(b, a)$, or
    $\mathbb{D}_3$) $W(b^-|_{\mathsf{in}}, b) \leq_{\mathbb{S}} W(b, b^+|_{\mathsf{in}})$, and
  - $L(a) = \mathsf{out} \iff W(a^-|_{\mathsf{in}}, a) <_{\mathbb{S}} \top$

- w-stable labelling for F if and only if
  - L is a w-complete labelling, and
  - $\mathcal{A}|_{\mathsf{undec}} \cup \mathcal{A}|_{\mathsf{empty}} = \emptyset$

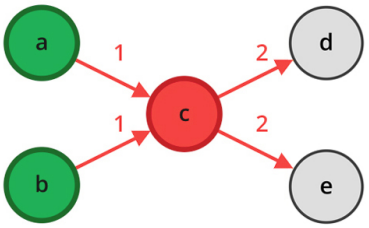
- w-preferred labelling for F if and only if
  - L is a w-admissible labelling, and
  - $\mathcal{A}|_{\mathsf{in}}$ is maximal among all the w-admissible labellings

- w-grounded labelling for F if and only if:
  - $L(a) = \mathsf{in} \iff$ for all w-complete labellings L', $L'(a) = \mathsf{in}$, and
  - $L(a) = \mathsf{out} \iff W(a^-|_{\mathsf{in}}, a) <_{\mathbb{S}} \top$

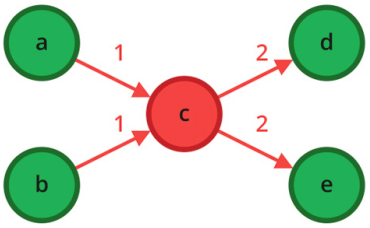
- w-quasi-strongly admissible labelling for F if and only if:
  - $L(a) = \mathsf{in} \implies a^- = a^-|_{\mathsf{out}} \wedge \forall b \in \mathcal{A}$ such that $R(b, a)$,
    $\mathbb{D}_1$) $\exists c$ with $L(c) = \mathsf{in} \mid c \neq a \wedge W(c, b) \leq_{\mathbb{S}} W(b, a)$, or
    $\mathbb{D}_2$) $W(b^-|_{\mathsf{in}} \setminus \{a\}, b) \leq_{\mathbb{S}} W(b, a)$, or
    $\mathbb{D}_3$) $W(b^-|_{\mathsf{in}} \setminus \{a\}, b) \leq_{\mathbb{S}} W(b, b^+|_{\mathsf{in}})$, and
  - $L(a) = \mathsf{out} \implies w_{a^-|_{\mathsf{in}}} <_{\mathbb{S}} \top$

The w-conflict-free labelling coincides with the conflict-free labelling of Definition 5.1: since attacks are not allowed within a conflict-free set of arguments, one does not need to consider the weights. The w-admissible labelling, instead, use the notion of w-defence given in Definition 2.16 with one condition between $\mathbb{D}_1$, $\mathbb{D}_2$ and $\mathbb{D}_3$. If $\mathbb{D}_1$ is used to compute the w-defence, we need to verify that $\exists c$ with $L(c) = \mathsf{in} \mid W(c, b) \leq_{\mathbb{S}} W(b, a)$, that is argument c is strong enough to defend a. Alternatively, the defences can be aggregated with $\mathbb{D}_2$, where $W(b^-|_{\mathsf{in}}, b) \leq_{\mathbb{S}} W(b, a)$ makes sure that the composition of the attacks of the arguments defending a is stronger than the attack of b towards a. Finally, when considering $\mathbb{D}_3$, we have to consider the composition of the attacks of b towards

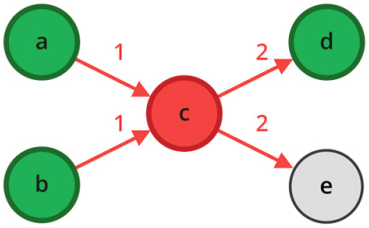
all the defenders. This is obtained through the condition in $W(b^-|_{\mathsf{in}}, b) \leq_{\mathbb{S}} W(b, b^+|_{\mathsf{in}})$. As for the four-state admissible labelling, for an argument a to be out there must exist at least an attack coming from an in argument, so we require $W(a^-|_{\mathsf{in}}, a) <_{\mathbb{S}} \top$ (where $\top$ means that there is no attack between two arguments). Examples of w-admissible labellings are shown in Figure 5.2.



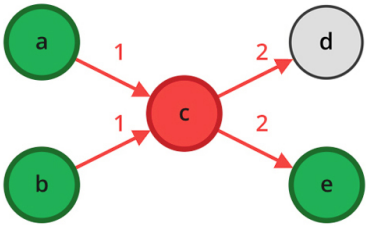
(a) A w-admissible labelling with respect to $\mathbb{D}_1$, $\mathbb{D}_2$ and $\mathbb{D}_3$.



(b) A w-admissible labelling with respect to $\mathbb{D}_2$.



(c) A w-admissible labelling with respect to $\mathbb{D}_2$ and $\mathbb{D}_3$.



(d) A w-admissible labelling with respect to $\mathbb{D}_2$ and $\mathbb{D}_3$.

FIGURE 5.2: Example of different labellings on the same $\text{WAF}_{\mathbb{S}}$.

The definition of the w-complete labelling is similar to the w-admissible one, with the exception that the conditions given for in and out arguments are both necessary and sufficient. An example is shown in Figure 5.3: arguments c and d are both in and out (and therefore are labelled undec), while e is not w-defended and is labelled empty.

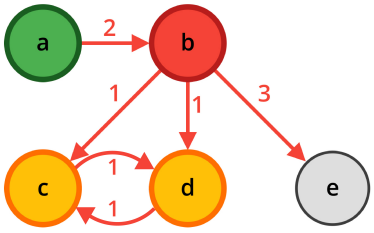


FIGURE 5.3: Example of a w-complete labelling.

A stable semantics partitions the arguments in two disjoint sets: one contains the arguments that are either not attacked or defended by other acceptable arguments, while the other contains the rest of the arguments (i.e., those that are attacked and not defended). In the weighted case, we obtain the same kind of partition. See Figure 5.4 for an example

of w-stable labelling. The example in Figure 5.3, instead, does not represent a w-stable labelling since it has undec and empty arguments (c, d and e, respectively).

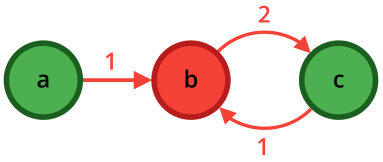


FIGURE 5.4: Example of a $\text{WAF}_{\mathbb{S}}$ with a w-stable labelling.

As for the classical definition, also in the weighted case the w-preferred extensions is the largest admissible sets. The labelling in Figure 5.2(b), that is w-admissible, is also w-preferred, since it has the most in labels possible. For what concerns the grounded semantics, we label in only the sceptically accepted arguments. If we look at the $\text{WAF}_{\mathbb{S}}$ of Figure 5.3, only the argument a is in in all w-complete labellings, and so the depicted labelling is also w-grounded.

Finally, we obtain a w-quasi-strongly admissible (see Definition 2.17) labelling by imposing that every in argument is always defended by other in arguments different from itself. The labelling in Figure 5.4 is not a w-quasi-strongly admissible labelling: in fact, the attack of the in argument a towards the out argument b is not sufficient alone to defend c. On the other hand, all the labellings in Figure 5.2 are w-quasi-strongly admissible.

The sets of arguments labelled in by the above-defined labellings for $\text{WAF}_{\mathbb{S}}$ are equivalent to the extensions of the corresponding semantics.

Theorem 5.4. *A labelling L of a $\text{WAF}_{\mathbb{S}}$ $F = \langle \mathcal{A}, \mathcal{R}, W, \mathbb{S} \rangle$ is a w-admissible (respectively w-complete, w-stable, w-preferred, w-grounded, w-quasi-strongly admissible) labelling if and only if $\mathcal{A}|_{\text{in}}$ is a w-admissible (respectively w-complete, w-stable, w-preferred, w-grounded, w-quasi-strongly admissible) extension of F.*

Proof. We show for each semantics the correspondence between the in arguments and the set of extensions. We refer to Definition 2.17 for the $\text{WAF}_{\mathbb{S}}$ semantics.

- (L is w-admissible $\Rightarrow$ $\mathcal{A}|_{\text{in}}$ is w-admissible.) The out arguments attacking $\mathcal{A}|_{\text{in}}$ are defeated by $\mathcal{A}|_{\text{in}}$. Thus, $\mathcal{A}|_{\text{in}}$ is w-defend from the attacks coming from $\mathcal{A} \setminus \mathcal{A}|_{\text{in}}$ and so it is a w-admissible extension.
- ($\mathcal{A}|_{\text{in}}$ is w-admissible $\Rightarrow$ L is w-admissible.) $\mathcal{A}|_{\text{in}}$ w-defends itself from the attacks of every $b \in \mathcal{A} \setminus \mathcal{A}|_{\text{in}}$, so $W(\mathcal{A}|_{\text{in}}, b) \leq_{\mathbb{S}} W(b, \mathcal{A}|_{\text{in}})$. Moreover, every $a \in \mathcal{A}|_{\text{in}}$, is in and thus L is a w-admissible labelling.

- (L is w-complete $\Rightarrow$ $\mathcal{A}|_{\mathsf{in}}$ is w-complete.) When L is w-complete, then it is also w-admissible and it labels all the arguments w-defended by $\mathcal{A}|_{\mathsf{in}}$ as in. Hence $\mathcal{A}|_{\mathsf{in}}$ is a w-complete extension.

- ($\mathcal{A}|_{\mathsf{in}}$ is w-complete $\Rightarrow$ L is w-complete.) In this case $\mathcal{A}|_{\mathsf{in}}$ is a w-admissible extension where all the w-defended arguments belong to $\mathcal{A}|_{\mathsf{in}}$. Then L is w-complete labelling.

- (L is w-stable $\Rightarrow$ $\mathcal{A}|_{\mathsf{in}}$ is w-stable.) L is a w-complete labelling in which no argument is labelled undec. Thus, the set $\mathcal{A}|_{\mathsf{in}}$ attacks all the other arguments in $\mathcal{A}\setminus\mathcal{A}|_{\mathsf{in}}$, and so $\mathcal{A}|_{\mathsf{in}}$ is a w-stable extension.

- ($\mathcal{A}|_{\mathsf{in}}$ is w-stable $\Rightarrow$ L is w-stable.) We have that the set $\mathcal{A}|_{\mathsf{in}}$ is attacking all the arguments in $\mathcal{A} \setminus \mathcal{A}|_{\mathsf{in}}$, so $\mathcal{A}|_{\mathsf{empty}} = \mathcal{A}|_{\mathsf{undec}} = \emptyset$. Then, since $\mathcal{A}|_{\mathsf{in}}$ is a w-admissible extension containing all the w-defended arguments, $\mathcal{A}|_{\mathsf{in}}$ is a w-complete extension and L a w-stable labelling.

- (L is w-preferred $\Rightarrow$ $\mathcal{A}|_{\mathsf{in}}$ is w-preferred.) The set of arguments labelled in by L coincides with a w-admissible extension which is maximal with respect to the set inclusion. Follows that $\mathcal{A}|_{\mathsf{in}}$ is a w-preferred extension.

- ($\mathcal{A}|_{\mathsf{in}}$ is w-preferred $\Rightarrow$ L is w-preferred.) We have that $\mathcal{A}|_{\mathsf{in}}$ is a maximal w-admissible extension, so L is a w-preferred labelling.

- (L is w-grounded $\Rightarrow$ $\mathcal{A}|_{\mathsf{in}}$ is w-grounded.) If L is w-grounded, all the arguments in $\mathcal{A}|_{\mathsf{in}}$ are also inin any w-complete labelling, thus $\mathcal{A}|_{\mathsf{in}}$ represents the maximal w-admissible extension included in the intersection of w-complete extensions.

- ($\mathcal{A}|_{\mathsf{in}}$ is w-grounded $\Rightarrow$ L is w-grounded.) $\mathcal{A}|_{\mathsf{in}}$ contains all and only arguments that are included in the intersection of w-complete extensions, so L is a w-grounded labelling.

- (L is w-strongly admissible $\Rightarrow$ $\mathcal{A}|_{\mathsf{in}}$ is w-strongly admissible.) The out arguments attacking any argument $a \in \mathcal{A}|_{\mathsf{in}}$ are defeated by $(\mathcal{A} \setminus \{a\})|_{\mathsf{in}}$. Thus, any argument in $\mathcal{A}|_{\mathsf{in}}$ is w-defend by the other arguments in $\mathcal{A}|_{\mathsf{in}}$ from the attacks coming from $\mathcal{A} \setminus \mathcal{A}|_{\mathsf{in}}$ and so $\mathcal{A}|_{\mathsf{in}}$ is a w-strongly admissible extension.

- ($\mathcal{A}|_{\mathsf{in}}$ is w-strongly admissible $\Rightarrow$ L is w-strongly admissible.) Each argument $a \in \mathcal{A}|_{\mathsf{in}}$ is w-defends by $(\mathcal{A} \setminus \{a\})|_{\mathsf{in}}$ from the attacks of every $b \in a^- \cap (\mathcal{A} \setminus \mathcal{A}|_{\mathsf{in}})$, so $W((\mathcal{A} \setminus \{a\}), b) \leq_{\mathbb{S}} W(b, \mathcal{A}|_{\mathsf{in}})$. Hence L is a w-strongly admissible labelling.

□

We summarize in Table 5.1 the conditions specified in Definition 5.3 for obtaining weighted labellings corresponding to the Dung semantics.

	conditions on in arguments	conditions on out arguments	other cond.
w-cf	$L(a) = \mathsf{in} \implies a^-\vert_{\mathsf{in}} = \emptyset$	$L(a) = \mathsf{out} \implies a^-\vert_{\mathsf{in}} \neq \emptyset$	
w-adm	$L(a) = \mathsf{in} \implies a^- = a^-\vert_{\mathsf{out}}$ $\wedge\ \mathcal{A}\vert_{\mathsf{in}}$ w-defends a	$L(a) = \mathsf{out} \iff W(a^-\vert_{\mathsf{in}}, a) <_{\mathbb{S}} \top$	
w-com	$L(a) = \mathsf{in} \iff a^- = a^-\vert_{\mathsf{out}}$ $\wedge\ \mathcal{A}\vert_{\mathsf{in}}$ w-defends a	$L(a) = \mathsf{out} \iff W(a^-\vert_{\mathsf{in}}, a) <_{\mathbb{S}} \top$	
w-stb	$L(a) = \mathsf{in} \iff a^- = a^-\vert_{\mathsf{out}}$ $\wedge\ \mathcal{A}\vert_{\mathsf{in}}$ w-defends a	$L(a) = \mathsf{out} \iff W(a^-\vert_{\mathsf{in}}, a) <_{\mathbb{S}} \top$	$\mathcal{A}\vert_{\mathsf{undec}} = \emptyset$ $\wedge\ \mathcal{A}\vert_{\mathsf{empty}} = \emptyset$
w-pre	$L(a) = \mathsf{in} \implies a^- = a^-\vert_{\mathsf{out}}$ $\wedge\ \mathcal{A}\vert_{\mathsf{in}}$ w-defends a	$L(a) = \mathsf{out} \implies W(a^-\vert_{\mathsf{in}}, a) <_{\mathbb{S}} \top$	$\mathcal{A}\vert_{\mathsf{in}}$ is a max w-adm
w-gde	$L(a) = \mathsf{in} \iff$ $\forall L'$ w-$com.L'(a) = \mathsf{in}$	$L(a) = \mathsf{out} \iff W(a^-\vert_{\mathsf{in}}, a) <_{\mathbb{S}} \top$	
w-qsa	$L(a) = \mathsf{in} \implies a^- = a^-\vert_{\mathsf{out}}$ $\wedge\ \mathcal{A}\vert_{\mathsf{in}} \setminus \{a\}$ w-defends a	$L(a) = \mathsf{out} \implies W(a^-\vert_{\mathsf{in}}, a) <_{\mathbb{S}} \top$	

TABLE 5.1: Summarisation of the introduced labellings for $\text{WAF}_{\mathbb{S}}$.

The conditions we give for the weighted semantics are a generalization of the classical case, and all the labellings for $\text{WAF}_{\mathbb{S}}$ corresponds to the respective classical semantics when the framework is instantiated with a boolean semiring. When the $\text{WAF}_{\mathbb{S}}$ is instantiated with a boolean semiring, all the attacks from an argument to another are associated with the value $false$ and also $w_{a^-\vert_{\mathsf{in}}}$ always corresponds to $false$.

Theorem 5.5. *The labelling of a $\text{WAF}_{\mathbb{S}}$ instantiated with a boolean semiring corresponds to the classical labelling.*

Proof. By Definition 2.13, the weight of an attack between two arguments in a $\text{WAF}_{\mathbb{S}}$ F where $\mathbb{S}$ is boolean always correspond to the value $false$. Since the composition operator is $\wedge$, also the $\otimes$ of every pair of attacks in F is $false$, and thus assigning a labelling boils down to checking the existence of attacks between arguments, as for the crisp case. □

It follows that if L is a w-admissible (respectively w-complete, w-stable, w-preferred, w-grounded) labelling of a $\text{WAF}_{\mathbb{S}}$ F, then L is an admissible (respectively complete, stable, preferred, grounded) labelling of F.

5.3 Related Work

The conditions we specify for obtaining our four-state labelling is similar to the interpretation given in [105], where arguments attacked by an in argument are always out. A different definition of labelling is given in[66], where in arguments can attack both out and undec arguments. Even if the labelling present different sets of out and undec

arguments, nothing changes in terms of corresponding extensions, since the set of in arguments remains the same.

The problem of extending classical AFs with values expressing the strength of arguments and attacks is widely studied, and many different approaches have been presented in the literature. In [9], the authors take into account preference orderings for comparing arguments, while in [27] the success of an attack conducted by an argument toward another one depends on an ordering among the "values" promoted by each argument. A study on bipolar WAFs is conducted in [136], where the authors present an extension for weighted frameworks that takes into account two different types of relations (one for attack and one for support). Another formalism based on a notion of strength is given in [18], were arguments in Quantitative Argumentation Debate Frameworks are evaluated through a score system. The main difference with our work lies in the fact that we take into account the basic definition of WAFs [86], without further refinements on the framework level. Moreover, our study is focused on the interpretation of the labelling in the weighted case.

5.4 Conclusion

In this chapter, we studied some characterisations for AFs and their semantics. First of all, we defined a four-state labelling semantics that marks arguments of an AF with four different labels: in, out, undec and empty. The set of in arguments corresponds to acceptable arguments with respect to the chosen extension-based semantics. Then, we defined a labelling for WAFs, together with a set of labelling conditions corresponding to extensions of weighted semantics, and we showed that our labelling function generalises the classical approach for the non-weighted case. We have considered three different definitions of collective defence from the literature, that can be used to obtain different results in terms of acceptability of the arguments.

Chapter 6

A Concurrent Language for Argumentation

"*Beware of bugs in the above code;*
I have only proved it correct, not tried it."
– Donald Knuth

Abstract

We define a concurrent language for expressing negotiation and debating through the use of argumentation. Using such concurrent language, communication between intelligent agents will be modelled as a dynamic process in which agent beliefs are represented through an Argumentation Framework that changes consequently to the interactions that take pace in the system. High-level primitives will be provided in order to implement such interaction protocols, taking into account belief revision postulates from the theory proposed by Alchourrón, Gärdenfors, and Makinson (AGM). Furthermore, the language will include concepts such as robustness and specific operations for describing common scenarios in the application domain like, for instance, negotiation, automatic debating systems, persuasive chatbots, recommender systems and collaborative robots.

Agents/processes in a distributed/concurrent environment can perform operations that affect the behaviour of other components. The indeterminacy in the execution order of the processes may lead to inconsistent results for the computation or even cause errors that prevent particular tasks from being completed. We refer to this kind of situation as a *race condition*. If not properly handled, race conditions can cause loss of information,

resource starvation and deadlock. In order to understand the behaviour of agents and devise solutions that guarantee correct executions, many formalisms have been proposed for modelling concurrent systems. Concurrent Constraint Programming (CC) [147] (see Table 2.2), in particular, relies on a constraint store of shared variables in which agents can read and write in accordance with some properties posed on the variables. The basic operations that can be executed by agents in the CC framework are a blocking *Ask* and an atomic *Tell*. These operations realise the interaction with the store and also allow one to deal with partial information. Starting from the CC syntax, we enrich the ask and tell operators in order to handle the interaction with an AF used as knowledge base for the agents. We replace the asks with three decisional operations: a syntactic *check* that verifies if a given set of arguments and attacks is contained in the knowledge base, and two semantic *test* operations that we use to retrieve information about the acceptability of arguments in an AF. The tell operation (that we call *add*) augments the store with additional arguments and attack relations.

6.1 Syntax and Semantics

The syntax of our concurrent language for argumentation, ConArg_lang, is presented in Table 6.1, while in Table 6.2 we give the definitions for the transition rules.

$$
\begin{aligned}
&P ::= C.A \\
&C ::= p(a, l, \sigma) :: A \mid C.C \\
&A ::= success \mid add(Arg, R) \rightarrow A \mid rmv(Arg, R) \rightarrow A \mid E \mid A \| A \mid \exists_x A \mid p(a, l, \sigma) \\
&E ::= test_c(a, l, \sigma) \rightarrow A \mid test_s(a, l, \sigma) \rightarrow A \mid check(Arg, R) \rightarrow A \\
&\qquad \mid E + E \mid E +_P E \mid E \|_G E
\end{aligned}
$$

TABLE 6.1: ConArg_lang syntax.

Suppose we have an agent A whose knowledge base is represented by an AF $F = \langle Arg, R \rangle$. An $add(Arg', R')$ action performed by the agent results in the addition of a set of arguments $Arg' \subseteq U$ (where U is the universe) and a set of relations R' to the AF F. When performing an Addition, (possibly) new arguments are taken from $U \setminus Arg$. We want to make clear that the tuple (Arg', R') is not an AF, indeed it is possible to have $Arg' = \emptyset$ and $R' \neq \emptyset$, which allows to perform an addition of only attack relations to the considered AF. It is as well possible to add only arguments to F, or both arguments and attacks. Intuitively, $rmv(Arg, R)$ allows to specify arguments and/or attacks to remove from the knowledge base. Removing an argument from an AF requires to also remove the attack relations involving that argument and trying to remove an argument (or an attack) which

$$\langle add(Arg', R') \to A, \langle Arg, R\rangle\rangle \longrightarrow \langle A, \langle Arg \cup Arg', R \cup R'\rangle\rangle$$ Addition

$$\langle rmv(Arg', R') \to A, \langle Arg, R\rangle\rangle \longrightarrow \langle A, \langle Arg \setminus Arg', R \setminus \{R' \cup R''\}\rangle\rangle \text{ where } R'' = \{(a,b) \in R \mid a \in Arg' \vee b \in Arg'\}$$ Removal

$$\frac{Arg' \subseteq Arg \wedge R' \subseteq R}{\langle check(Arg', R') \to A, \langle Arg, R\rangle\rangle \longrightarrow \langle A, \langle Arg, R\rangle\rangle}$$ Check

$$\frac{\exists L \in S_\sigma(F) \mid l \in L(a)}{\langle test_c(a, l, \sigma) \to A, F\rangle \longrightarrow \langle A, F\rangle}$$ Credulous Test

$$\frac{\forall L \in S_\sigma(F).l \in L(a)}{\langle test_s(a, l, \sigma) \to A, F\rangle \longrightarrow \langle A, F\rangle}$$ Sceptical Test

$$\frac{\langle A_1, F\rangle \longrightarrow \langle A_1', F'\rangle}{\langle A_1 \| A_2, F\rangle \longrightarrow \langle A_1' \| A_2, F'\rangle} \quad \langle A_2 \| A_1, F\rangle \longrightarrow \langle A_2 \| A_1', F'\rangle \qquad \frac{\langle A_1, F\rangle \longrightarrow \langle success, F'\rangle}{\langle A_1 \| A_2, F\rangle \longrightarrow \langle A_2, F'\rangle} \quad \langle A_2 \| A_1, F\rangle \longrightarrow \langle A_2, F'\rangle$$ Parallelism

$$\frac{\langle E_1, F\rangle \longrightarrow \langle A_1, F\rangle, \langle E_2, F\rangle \not\longrightarrow}{\langle E_1 \|_G E_2, F\rangle \longrightarrow \langle A_1, F\rangle} \quad \langle E_2 \|_G E_1, F\rangle \longrightarrow \langle A_1, F\rangle$$ Guarded Parallelism (1)

$$\frac{\langle E_1, F\rangle \longrightarrow \langle A_1, F\rangle, \langle E_2, F\rangle \longrightarrow \langle A_2, F\rangle}{\langle E_1 \|_G E_2, F\rangle \longrightarrow \langle A_1 \| A_2, F\rangle}$$ Guarded Parallelism (2)

$$\frac{\langle E_1, F\rangle \longrightarrow \langle A_1, F\rangle}{\langle E_1 + E_2, F\rangle \longrightarrow \langle A_1, F\rangle} \quad \langle E_2 + E_1, F\rangle \longrightarrow \langle A_1, F\rangle$$ Nondeterminism

$$\frac{\langle E_1, F\rangle \longrightarrow \langle A_1, F\rangle}{\langle E_1 +_P E_2, F\rangle \longrightarrow \langle E_1, F\rangle}$$ If Then Else (1)

$$\frac{\langle E_1, F\rangle \not\longrightarrow, \langle E_2, F\rangle \longrightarrow \langle A_2, F\rangle}{\langle E_1 +_P E_2, F\rangle \longrightarrow \langle E_2, F\rangle}$$ If Then Else (2)

$$\frac{\langle A[y/x], F\rangle \longrightarrow \langle A', F'\rangle}{\langle \exists_x A, F\rangle \longrightarrow \langle A', F'\rangle} \text{ with } y \text{ fresh}$$ Hidden Variables

$$\langle p(b, m, \gamma), F\rangle \longrightarrow \langle A[b/a, m/l, \gamma/\sigma], F\rangle \text{ when } p(a, l, \sigma) :: A$$ Procedure Call

TABLE 6.2: ConArg_lang operational semantics.

does not exist in F will have no consequences. The operation $check(Arg', R')$ is used to verify whether the specified arguments and attack relations are contained in the set of arguments and attacks of the knowledge base, without introducing any further change. If the check is positive, the operation succeeds. Otherwise it suspends. We have two distinct test operations, both requiring the specification of an argument $a \in A$, a label $l \in \{\mathsf{in}, \mathsf{out}, \mathsf{undec}, \mathsf{empty}\}$ and a semantics $\sigma \in \{adm, com, stb, prf, gde\}$. The credulous $test_c(a, l, \sigma)$ succeeds if there exists at least one extension of $S_\sigma(F)$ whose corresponding labelling L is such that $L(a) = l$; otherwise (in the case $L(a) \neq l$ in all labellings) it suspends. Similarly, the sceptical $test_s(a, l, \sigma)$ succeeds if a is labelled l in all possible labellings $L \in S_\sigma(F)$, and suspends in the case $L(a) \neq L$ in some labellings. The guarded parallelism $\|_G$ is designed to allow all the operations for which the guard in the inner expression is satisfied. In more detail, $E_1 \|_G E_2$ is successful when either E_1, E_2 or both are successful and all the operations that can be executed are executed. This behaviour is different both from classical parallelism (for which all the agents have to succeed in order for the procedure to succeed) and from nondeterminism (that only selects one branch). The operator $+_P$ is left-associative and realises an if-then-else construct: if we have $E_1 +_P E_2$ and E_1 is successful, than E_1 will be always chosen over E_2, even if also E_2 is successful, so in order for E_2 to be selected, it has to be the only one that succeeds.

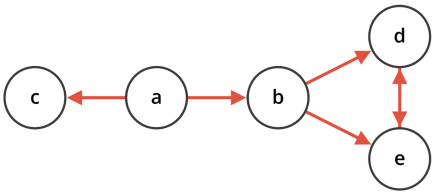
The remaining operators are classical concurrency compositions: an agent in a parallel composition obtained through $\|$ succeeds only if all the agents succeed; any agent composed through $+$ is chosen if its guards succeeds; the existential quantifier $\exists_x A$ behaves like agent A where variables in x are local[10] to A. The parallel composition operator enables the specification of complex concurrent argumentation processes. For example, a debate involving many agents that asynchronously provide arguments can be modelled as a parallel composition of add operations performed on the knowledge base. The procedure call has three parameters that allow the implementation of operators which takes into account an argument, a label and a semantics. Below, we give two examples of ConArg_lang programs.

Example 6.1. *Consider the AF in Figure 6.1(a), where the complete semantics is the set* $\{\{a\}, \{a, e\}, \{a, d\}\}$ *and the preferred coincides with* $\{\{a, d\}, \{a, e\}\}$. *An agent A wants to perform the following operation: if argument d is labelled* out *in all complete extensions, then remove the argument c from the knowledge base. At the same time, an agent B want to add an argument f attacking d only if e is labelled* in *in some preferred extension. If A is the first agent to be executed, the sceptical test on argument d will suspend, since d belongs to the complete extension* $\{a, d\}$. *The credulous test performed by agent B, instead, is successful and so it can proceed to add an argument f that defeats d. Now*

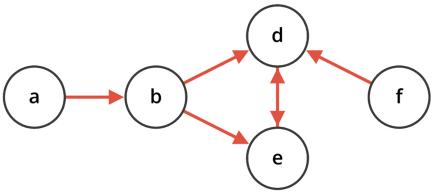
[10] We plan to use existential quantifiers to extend our work by allowing our agents to have local stores.

d is sceptically rejected by the complete semantics and agent A can finally remove the argument c. After the execution of the program below, we obtain the AF of Figure 6.1(b).

$$A : test_s(d, \mathsf{out}, com) \rightarrow rmv(\{c\}, \{(a, c)\}) \rightarrow success$$
$$B : test_c(e, \mathsf{in}, prf) \rightarrow add(\{f\}, \{(f, d)\}) \rightarrow success$$
$$P : A \| B$$



(a) An AF F representing the knowledge base shared between two agents A and B.



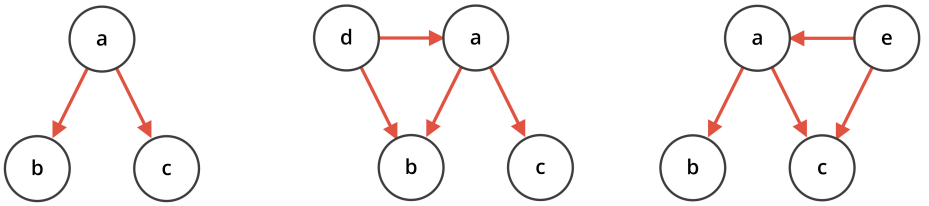
(b) The AF obtained from F after the modifications of agents A and B.

FIGURE 6.1: Example of an AF modified by concurrent agents.

Example 6.2. *In this example we show how the order in which concurrent operations are executed by different agents affects the final appearance of the shared AF. Consider Figure 6.2(a) of an AF F representing the shared memory of two agents A and B. The behaviour of the two agents is described in the following program.*

$$A : test_c(a, \mathsf{in}, com) \rightarrow add(\{d\}, \{(d, a), (d, b)\}) \rightarrow success$$
$$B : test_c(e, \mathsf{in}, prf) \rightarrow add(\{e\}, \{(e, a), (e, c)\}) \rightarrow success$$
$$P : A \| B$$

The agent A tests whether argument a is accepted with respect to the complete semantics (that for F is the singleton {a}) and, if the answer is positive, it adds an argument d and makes d to attack a and b. The agent B executes the same test, but in case of a positive answer it adds an argument e and the attacks from e to a and from e to c. If the execution of A happens before of that of B, we obtain the AF in Figure 6.2(b) and B will not be able to further modify the knowledge base since, at this stage, argument a is no longer acceptable in any complete extension. On the other hand, if the program of B is executed first, the AF is modified as in Figure 6.2(c), where an argument e and the attacks (e, a) and (e, c) are added. In this case the test of agent A can no longer succeed and the AF it will not undergo further changes.



(a) An AF F.

(b) The AF obtained from F if agent A act first.

(c) The AF obtained from F if agent B act first.

FIGURE 6.2: Example of an AF when concurrent agents execute modifications on it.

6.2 Semantics of Failure

The language we presented in the previous section only allows two possible outcomes as result of an operation: it can either succeed, taking the execution to the next step, or suspend (wait). Hence, it may happen that if the right conditions are not satisfied, some processes can get stuck in an endless wait (as it happens for the agent B of Example 6.2 when the program of A is executed first). To solve the issue of termination, we introduce a distinction between the expressions that can be written using ConArg_lang syntax: E^w represents an expression which suspends in the case the condition on its head is not satisfied; E^f can either succeed of fail, but never suspends. We then identify two further categories of operations for checking and testing the knowledge base, one allowing failure and the other which blocks the execution, namely $check^f(Arg', R')$, $test^f_c(Arg', R')$, $test^f_s(Arg', R')$ and $check^w(Arg', R')$, $test^w_c(Arg', R')$, $test^w_s(Arg', R')$, respectively. The revised syntax appears as shown below in Table 6.3.

$$
\begin{aligned}
E &::= E^w \mid E^f \mid E^f +_P E \\
E^w &::= test^w_c(a,l,\sigma) \to A \mid test^w_s(a,l,\sigma) \to A \mid check^w(Arg,R) \to A \mid E^w + E^w \\
E^f &::= test^f_c(a,l,\sigma) \to A \mid test^f_s(a,l,\sigma) \to A \mid check^f(Arg,R) \to A \mid E^f \|_G E^f \mid failure
\end{aligned}
$$

TABLE 6.3: ConArg_lang syntax for expressions with failure and wait.

We change the ConArg_lang operational semantics accordingly (see Table 6.4), establishing the cases in which the expressions produce a failure or a suspension of the program (while the conditions for succeeding remains the same). Allowing expressions to fail, the program can continue the execution even if some of the operation does not succeeds. The $check^w(Arg, R)$ operation suspends when Arg and R are not part of the knowledge base, while $check^f(Arg, R)$ fails. When testing the acceptability of arguments, the $test^w_c(a, l, \sigma)$ and $test^w_s(a, l, \sigma)$ operations suspend in case of a negative response, while $test^f_c(a, l, \sigma)$ and $test^f_s(a, l, \sigma)$ fail. Parallelism and guarded parallelism are also affected by the introduction of failure. The parallel composition of two actions can result in three

possible behaviours: it succeeds when both actions succeed, suspends when at least one action suspends and fail in the remaining case (i.e., when both actions fails). The guarded parallelism executes all branches which satisfy the given condition, and succeeds if at least one expression succeeds. On the other hand, it fails if all the expressions fail. Since only the composition of expressions that can fail are allowed in a guarded parallelism, it cannot suspend under any circumstances.

$$\frac{Arg' \subseteq Arg \wedge R' \subseteq R}{\langle check^*(Arg', R') \rightarrow A, \langle Arg, R\rangle\rangle \longrightarrow \langle A, \langle Arg, R\rangle\rangle} \quad \text{Check (1)}$$

$$\frac{Arg' \not\subseteq Arg \vee R' \not\subseteq R}{\langle check^f(Arg', R') \rightarrow A, \langle Arg, R\rangle\rangle \longrightarrow failure} \quad \text{Check (2)}$$

$$\frac{\exists L \in S_\sigma(F) \mid l \in L(a)}{\langle test^*_c(a, l, \sigma) \rightarrow A, F\rangle \longrightarrow \langle A, F\rangle} \qquad \frac{\forall L \in S_\sigma(F).l \notin L(a)}{\langle test^f_c(a, l, \sigma) \rightarrow A, F\rangle \longrightarrow failure} \quad \text{Credulous Test}$$

$$\frac{\forall L \in S_\sigma(F).l \in L(a)}{\langle test^*_s(a, l, \sigma) \rightarrow A, F\rangle \longrightarrow \langle A, F\rangle} \qquad \frac{\exists L \in S_\sigma(F) \mid l \notin L(a)}{\langle test^f_s(a, l, \sigma) \rightarrow A, F\rangle \longrightarrow failure} \quad \text{Sceptical Test}$$

$$\frac{\langle A_1, F\rangle \longrightarrow \langle A_1', F'\rangle}{\begin{array}{c}\langle A_1 \| A_2, F\rangle \longrightarrow \langle A_1' \| A_2, F'\rangle \\ \langle A_2 \| A_1, F\rangle \longrightarrow \langle A_2 \| A_1', F'\rangle\end{array}} \qquad \frac{\langle A_1, F\rangle \longrightarrow \langle success, F'\rangle}{\begin{array}{c}\langle A_1 \| A_2, F\rangle \longrightarrow \langle A_2, F'\rangle \\ \langle A_2 \| A_1, F\rangle \longrightarrow \langle A_2, F'\rangle\end{array}} \quad \text{Parallelism (1)}$$

$$\frac{\langle A_1, F\rangle \longrightarrow failure}{\begin{array}{c}\langle A_1 \| A_2, F\rangle \longrightarrow failure \\ \langle A_2 \| A_1, F\rangle \longrightarrow failure\end{array}} \quad \text{Parallelism (2)}$$

$$\frac{\langle E_1, F\rangle \longrightarrow \langle A_1, F\rangle, \langle E_2, F\rangle \longrightarrow failure}{\begin{array}{c}\langle E_1 \|_G E_2, F\rangle \longrightarrow \langle A_1, F\rangle \\ \langle E_2 \|_G E_1, F\rangle \longrightarrow \langle A_1, F\rangle\end{array}} \quad \text{Guarded Parallelism (1)}$$

$$\frac{\langle E_1, F\rangle \longrightarrow \langle A_1, F\rangle, \langle E_2, F\rangle \longrightarrow \langle A_2, F\rangle}{\langle E_1 \|_G E_2, F\rangle \longrightarrow \langle A_1 \| A_2, F\rangle} \quad \text{Guarded Parallelism (2)}$$

$$\frac{\langle E_1, F\rangle \longrightarrow \langle A_1, F\rangle}{\langle E_1 +_P E_2, F\rangle \longrightarrow \langle E_1, F\rangle} \quad \text{If Then Else (1)}$$

$$\frac{\langle E_1, F\rangle \longrightarrow failure, \langle E_2, F\rangle \longrightarrow \langle A_2, F\rangle}{\langle E_1 +_P E_2, F\rangle \longrightarrow \langle E_2, F\rangle} \quad \text{If Then Else (2)}$$

TABLE 6.4: ConArg_lang operational semantics for expressions with failure and wait.

6.3 Belief Revision and the AGM Framework

Interaction between agents can be modelled in different ways, according to the purposes of the communication. Negotiating agents need to find a common agreement that is beneficial to all, while, for instance, an agent with the goal of persuading its opponents

has to both defend its position from the attacks of the other agents and defeat all the arguments against its proposal. The operations needed for the implementation of such kinds of interactions must be able to modify the knowledge base shared between the communicating parts so as to model the behaviour of the agents. In particular, usually agents interact modifying part of the shared AF, trying to change the state of acceptance of an argument, often alternating with other agents or concurrently performing syntactic changes to the AF.

The AGM framework [2] provide an approach to the problem of revising knowledge basis by using theories (deductively closed sets of formulae) to represent the beliefs of the agents. A formula α in a given theory can have different statuses for an agent, according to its knowledge base K. If the agent can deduce α from its beliefs, then we say that α is *accepted* ($K \vdash \alpha$). Such a deduction correspond with the entailment of α by the knowledge base. If the agent can deduce the negation of α, then we say that α is *rejected* ($K \vdash \neg\alpha$). Otherwise, the agent cannot deduce anything and α is *undetermined*. The correspondence between accepted/rejected beliefs and in/out arguments in a labelling is straightforward. Since the undetermined status represents the absence of a piece of information (nothing can be deduced in favour of either accepting or rejecting a belief) it can be mapped into the empty label empty. Finally, the undec label is assigned to arguments that are both in and out, boiling down to the notion of inconsistency in AGM. The empty label, in particular, plays a fundamental role in identifying new arguments that agents can bring to the debate to defend (or strengthen) their position. The status of a belief can be changed through some operations (namely expansion $\oplus$, contraction $\oslash$ and revision $\circledast$) on the knowledge base, as depicted in Figure 6.3 (notice the similarity with the lattice in Figure 2.3).

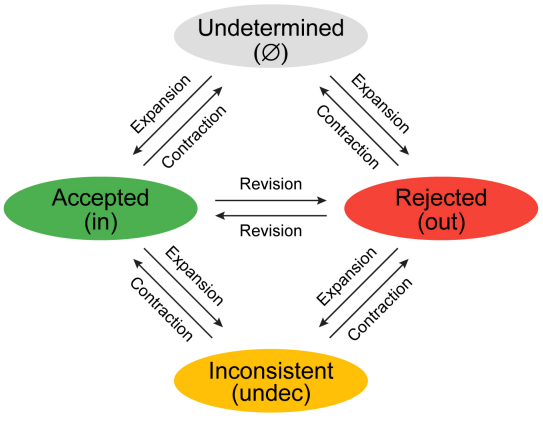


FIGURE 6.3: Transitions between AGM beliefs states.

An expansion basically brings new pieces of information to the base, allowing for undetermined belief to become either accepted or refused. A contraction, on the contrary,

reduces the information an agent can rely on in making its deduction, and an accepted (or refused) belief can become undetermined. A revision introduces conflicting information, making acceptable belief refused and vice-versa. The AGM framework also defines three sets of rationality postulates (one for each operation) that any good operator should satisfy. To give an example, if we want to add a new belief on a knowledge base, then we expect that no other information in the base is removed. AGM operators provide building blocks for realizing complex interaction processes between agents. Below, we provide some examples:

- **Negotiation** is a process that aims to solve conflicts arising from the interaction between two or more parties that have different individual goals (for instance, a request of computational resources in a distributed network), and its outcome is an agreement that translates in common benefits for all participants. Expansion, here, can be used to model the behaviour of an agent presenting claims towards its counterparts, while contraction represents the act of retracting a condition to successfully conclude the negotiation.

- Contrary to negotiation, a **debate** takes place when the goal of the agents in the system is to promote their own point of view and thus "convince" the others about a conclusion or a statement. A debate [98] can be considered as a mechanism through which a decision maker extracts information from two (or more) counterparts, each of them holding different positions with respect to the right choice. In a multi-agent system, a debate is a process carried out as the interaction between more parties, each of them trying to provide arguments strong enough to support their own conclusion. In this case, agents can make their beliefs accepted in different ways, exploiting AGM operators: inconsistent beliefs can be made accepted through a contraction, while expansion can make beliefs which state is undetermined acceptable.

- The notion of **persuasion** in dialogue games [139] aims to solve conflicts of points of view between two counterparts. In order to persuade the opponent, an agent has to defend its position by replying to every attack against its initial claim. If it fails, the opponent wins the game. Agents involved in this kind of persuasive dialogue games have to elaborate strategies [106], for supporting their beliefs and defeating the adversaries, that consist in a sequence of actions to perform in the system. Again, revision operations on the knowledge base are responsible for changing the status of the beliefs of a persuaded agent.

As for knowledge basis in belief revision, AFs can undergo changes that modify the structure of the framework itself, either integrating new information (and so increasing

the arguments and the attacks in the AF) or discarding previously available knowledge. Agents using AFs as the mean for exchanging and inferring information has to rely on operations able to modify such AFs. Besides considering the mere structural changes, also modifications on the semantics level need to be addressed by the operations performed by the agents. In the following, we define three operators for AFs, namely *argument expansion*, *contraction* and *revision*, that comply with classical operators of AGM and that can be built as procedures in our language.

The argumentation frameworks $\langle Arg, R\rangle$ we use as the knowledge base for our concurrent agents are endowed with a universe of arguments U that are used to bring new information. Since arguments in $U \setminus Arg$ do not constitute an actual part of the knowledge base, they are always labelled $\emptyset$, until they are added into the framework and acquire an in and/or an out label. Notice also that changes to the knowledge base we are interested in modelling are restricted to a single argument at a time, miming the typical argument interaction in dynamic AF.

Definition 6.1 (Argument extension expansion, contraction, revision)**.** Let $F = \langle Arg, R\rangle$ be an AF on the universe U, $Arg \subseteq U$, $R \subseteq Arg \times Arg$, σ a semantics, $L \in S_\sigma(F)$ a given labelling, and $a \in U$ an argument.

- An argument extension expansion $\oplus^{\sigma}_{a,L} : AF \to AF$ computes a new AF $F' = \oplus^{\sigma}_{a,L}(F)$ with semantics $S_\sigma(F')$ for which $\exists L' \in S_\sigma(F')$ such that $L'(a) \supseteq L(a)$ (if $L'(a) \supset L(a)$ the expansion is strict).
- An argument extension contraction $\oslash^{\sigma}_{a,L} : AF \to AF$ computes a new AF $F' = \oslash^{\sigma}_{a,L}(F)$ with semantics $S_\sigma(F')$ for which $\exists L' \in S_\sigma(F')$ such that $L(a) \supseteq L'(a)$ (if $L(a) \supset L'(a)$ the contraction is strict).
- An argument extension revision $\circledast^{\sigma}_{a,L} : AF \to AF$ computes a new AF $F' = \circledast^{\sigma}_{a,L}(F)$ with semantics $S_\sigma(F')$ for which $\exists L' \in S_\sigma(F')$ such that if $L(a) = \mathsf{in}/\mathsf{out}$, then $L'(a) = \mathsf{out}/\mathsf{in}$ and $\forall b \in Arg$ with $b \neq a$, $L'(b) = L(b) \vee L'(b) \neq \mathsf{undec}$ (that is no inconsistencies are introduced).

Moreover, we denote with $\oplus^{\sigma,l}_{a,L}(F)$, $\oslash^{\sigma,l}_{a,L}(F)$ and $\circledast^{\sigma,l}_{a,L}(F)$ an argument extension expansion, contraction and revision, respectively, that computes an AF F' with semantics $S_\sigma(F')$ for which $\exists L' \in S_\sigma(F')$ such that $L'(a) = l$.

When performing an argument extension expansion (or contraction, or revision) for a certain argument a of an AF F, the operators of Definition 6.1 take into account a single labelling of the semantics σ and there is no control over the other labellings, for which a can have its label arbitrarily changed. For example, an argument extension

expansion that increases the number of labels of a with respect to a chosen labelling L, may reduce that number in a different labelling. Therefore, we introduce a further definition that considers all the possible labellings $\mathcal{L}^F_\sigma$ of $S_\sigma(F)$. To compare the various labels an argument can have in different labellings, we refer to the order in Figure 2.3 and, calling $\mathcal{L}^F_{\sigma\downarrow a}$ the multi-set of the labels a has in the various $L \in \mathcal{L}^F_\sigma$, we say that $\mathcal{L}^{F'}_{\sigma\downarrow a} \supseteq \mathcal{L}^F_{\sigma\downarrow a}$ if there exists an injective function $f : \mathcal{L}^F_\sigma \rightarrowtail \mathcal{L}^{F'}_\sigma$ such that $\forall l \in \mathcal{L}^F_{\sigma\downarrow a}.l \leq f(l)$. Moreover, we use the notation $\mathcal{L}^F_{\sigma\downarrow a}|_l$ to restrict to l labels in the multi-set $\mathcal{L}^F_{\sigma\downarrow a}$, where $l = \{\mathsf{in}, \mathsf{out}, \mathsf{undec}, \mathsf{empty}\}$.

Definition 6.2 (Argument semantics expansion[11], contraction, revision)**.** Let $F = \langle Arg, R\rangle$ be an AF on the universe U, $Arg \subseteq U$, $R \subseteq Arg \times Arg$, σ a semantics, and $a \in U$ an argument.

- An argument semantics expansion $\oplus^\sigma_a : AF \rightarrow AF$ computes a new AF $F' = \oplus^\sigma_a(F)$ with semantics $S_\sigma(F')$ such that $\mathcal{L}^{F'}_{\sigma\downarrow a} \supseteq \mathcal{L}^F_{\sigma\downarrow a}$.
- An argument semantics contraction $\oslash^\sigma_a : AF \rightarrow AF$ computes a new AF $F' = \oslash^\sigma_a(F)$ with semantics $S_\sigma(F')$ such that $\mathcal{L}^F_{\sigma\downarrow a} \supseteq \mathcal{L}^{F'}_{\sigma\downarrow a}$.
- An argument semantics revision $\circledast^\sigma_a : AF \rightarrow AF$ computes a new AF $F' = \circledast^\sigma_a(F)$ with semantics $S_\sigma(F')$ such that $\forall b \in Arg. \left|\mathcal{L}^F_{\sigma\downarrow b}|_{undec}\right| \geq \left|\mathcal{L}^{F'}_{\sigma\downarrow b}|_{undec}\right|$ (that is no inconsistencies are introduced), and:
  - in-to-out revision: $\left|\mathcal{L}^F_{\sigma\downarrow a}|_{\mathsf{out}}\right| < \left|\mathcal{L}^{F'}_{\sigma\downarrow a}|_{\mathsf{out}}\right| \wedge \left|\mathcal{L}^F_{\sigma\downarrow a}|_{\mathsf{in}}\right| > \left|\mathcal{L}^{F'}_{\sigma\downarrow a}|_{in}\right|$;
  - out-to-in revision: $\left|\mathcal{L}^F_{\sigma\downarrow a}|_{\mathsf{in}}\right| < \left|\mathcal{L}^{F'}_{\sigma\downarrow a}|_{\mathsf{in}}\right| \wedge \left|\mathcal{L}^F_{\sigma\downarrow a}|_{\mathsf{out}}\right| > \left|\mathcal{L}^{F'}_{\sigma\downarrow a}|_{\mathsf{out}}\right|$;

It is important to note that the formalism we present is not monotone: the *add* operation may lead to a contraction, reducing the number of arguments with the labels in and/or out. Similarly, the removal of an argument may lead to an expansion (this is the case of Figure 6.4).

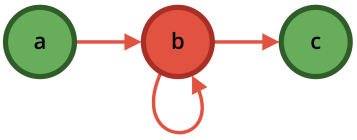


FIGURE 6.4: Example of argument extension expansion. Removing the in argument a makes both b and c undec.

AGM operators have already been studied from the point of view of their implementation in work as [23, 75], especially with regard to enforcement. However, in the previous

[11] The notion of argument semantics expansion is related to that of semantics inclusion of Definition 3.2 that we use in Chapter 3 to define robustness.

literature, realisability of extensions and not of single arguments is considered. The implementation of an argument expansion/contraction/revision operator changes according to the semantics we take into account. In the following, we consider the grounded semantics and show how the operators of Definitions 6.1 can be implemented. Notice that for the grounded semantics, that only has one extension, Definitions 6.1 and 6.2 coincide.

Proposition 6.3. *Let $F = \langle Arg, R\rangle$ be an AF on the universe U, $Arg \subseteq U$, $R \subseteq Arg \times Arg$, $a \in U$ an argument, and L the unique grounded labelling. A possible argument extension expansion $\oplus^{gde,l}_{a,L}(F)$ could act as:*

- *if $L(a) =$ empty and $l =$ in, add a to Arg*
- *if $L(a) =$ empty and $l =$ out,*
  - *if $\exists b \in Arg \mid L(b) =$ in, add $\langle\{a\}, \{(b,a)\}\rangle$ to F*
  - *otherwise, add $\langle\{a,b\}, \{(b,a)\}\rangle$ to F*
- *if $L(a) =$ in and $l =$ undec,*
  - *if $\exists b \in Arg \mid L(b) =$ undec, add (b,a) to R*
  - *otherwise, add (a,a) to R*
- *if $L(a) =$ out and $l =$ undec,*
  - *$\forall b \in Arg \mid L(b) = \{\text{in}\} \wedge (b,a) \in R$, add (a,b) to R*

We show an example of possible implementation in Table 6.5. We make use of some syntactic sugar to simplify the presentation of the results. Let be $|Arg| = n$:

- *$E_1 \wedge E_2 \to A$ represents $E_1 \to E_2 \to A$;*
- *$\sum_{a \in Arg}(E(a))$ represents $E(a_1) + E(a_2) + \cdots + E(a_n)$, $\forall a_i \in Arg$;*
- *$\|_{G\ a \in Arg}(E(a))$ represents $E(a_1) \|_G E(a_2) \|_G \ldots \|_G E(a_n)$, $\forall a_i \in Arg$.*

We also use the letter u to identify fresh arguments taken from $U \setminus Arg$.

Proof. If a has an empty label, it means that $a \in U \setminus Arg$, since the grounded labelling assigns a label different from empty to all arguments in Arg. It is then sufficient to add a to the set of considered arguments Arg to make it in. If the freshly added argument is attacked by another in argument, it becomes out. Continuing, a is labelled undec in the grounded labelling only if it is attacked by an undec argument (included a itself),

$$
\begin{aligned}
&\oplus^{gde,\mathsf{in}}_{a,L}(F) \ : add(\{a\},\{\}) \rightarrow success \\
&\quad{\scriptstyle (L(a)=\mathsf{empty})} \\
&\oplus^{gde,\mathsf{out}}_{a,L}(F) : \sum_{b\in Arg} (test^f_c(b,\mathsf{in},gde) \rightarrow add(\{a\},\{(b,a)\})) \rightarrow success \\
&\quad{\scriptstyle (L(a)=\mathsf{empty})} \\
&\qquad\qquad +_P \\
&\qquad\qquad add(\{a,u\},\{(u,a)\}) \rightarrow success \\
&\oplus^{gde,\mathsf{undec}}_{a,L}(F) : \sum_{b\in Arg} (test^f_c(b,\mathsf{undec},gde) \rightarrow add(\{\},\{(b,a)\})) \rightarrow success \\
&\quad{\scriptstyle (L(a)=\mathsf{in})} \\
&\qquad\qquad +_P \\
&\qquad\qquad add(\{\},\{(a,a)\}) \rightarrow success \\
&\oplus^{gde,\mathsf{undec}}_{a,L}(F) : \Big\|_G \ (test^f_c(b,\mathsf{in},gde) \wedge check^f(\{\},\{(b,a)\}) \\
&\quad{\scriptstyle (L(a)=\mathsf{out})} \qquad\qquad \rightarrow add(\{\},\{(a,b)\})) \rightarrow success
\end{aligned}
$$

TABLE 6.5: Argument extension expansion operator (Proposition 6.3) in CONARG_LANG syntax.

thus, to make an in argument a become undec we can look for an argument b in Arg that is already labelled as undec. If we find such a b then it is sufficient to add the attack relation from b to a to the store. Otherwise, we make a attack itself. Finally, if we want an out argument a to become undec, we make it attack back all its in attackers. Doing so, we obtain three distinct complete labellings: one in which a is accepted and its attackers are not, another one in which the opposite situation occurs, and the third labelling in which neither a nor its attackers are fully accepted or rejected (that is they are undec). Hence, a will be undec in the minimal complete labelling (that, by Definition 5.1, is also grounded). □

Proposition 6.4. *Let $F = \langle Arg, R\rangle$ be an AF on the universe U, $Arg \subseteq U$, $R \subseteq Arg \times Arg$, $a \in U$ an argument, and L the unique grounded labelling. A possible argument extension contraction $\oslash^{gde,l}_{a,L}(F)$ could act as:*

- *if $L(a) =$ undec and $l =$ in, $\forall b \in Arg \mid L(b) =$ undec, remove (b,a) from R*
- *if $L(a) =$ undec and $l =$ out,*
  - *if $\exists b \in Arg \mid L(b) =$ in, add (b,a) to R*
  - *otherwise, add $\langle\{b\},\{(b,a)\}\rangle$ to F*
- *if $L(a) =$ in and $l =$ empty, remove a (and all attacks involving a) from F*
- *if $L(a) =$ out and $l =$ empty, remove a (and all attacks involving a) from F*

An implementation of argument extension contraction can be found in Table 6.6. We use the following syntactic sugar:

- $\sum\limits_{a \in Arg} (E(a))$ *represents* $E(a_1) + E(a_2) + \cdots + E(a_n)$, $\forall a_i \in Arg$;
- $\Big\|_G\limits_{a \in Arg} (E(a))$ *represents* $E(a_1) \|_G E(a_2) \|_G \ldots \|_G E(a_n)$, $\forall a_i \in Arg$.

$$
\begin{aligned}
\oslash^{gde,\mathsf{in}}_{a,L}(F) &: \Big\|_G\limits_{b \in Arg} (test^f_c(b, \mathsf{undec}, gde) \rightarrow rmv(\{\}, \{(b,a)\})) \rightarrow success \\
\oslash^{gde,\mathsf{out}}_{a,L}(F) &: \sum_{b \in Arg} (test^f_c(b, \mathsf{in}, gde) \rightarrow add(\{\}, \{b,a\})) \rightarrow success \\
&\quad +_P \\
&\quad add(\{u\}, \{u,a\}) \rightarrow success \\
\oslash^{gde,\mathsf{empty}}_{a,L}(F) &: rmv(\{a\}, \{\}) \rightarrow success \\
\oslash^{gde,\mathsf{empty}}_{a,L}(F) &: rmv(\{a\}, \{\}) \rightarrow success
\end{aligned}
$$

$(L(a)=\mathsf{undec})$ $(L(a)=\mathsf{undec})$ $(L(a)=\mathsf{in})$ $(L(a)=\mathsf{out})$

TABLE 6.6: Argument extension contraction operator (Proposition 6.4) in CONARG_LANG syntax.

Proof. Consider a grounded labelling. An undec argument a can become in by removing all attacks coming from undec arguments (included a itself). Indeed an argument is undec only if it is attacked by another undec. Note that a cannot be attacked by in arguments, otherwise it would have been out. Therefore, after the changes a is only attacked by out arguments, and thus is in. Alternatively, a can become out when it is attacked by another in argument b (when the store does not contain in arguments, we add one from the universe). If a is either in or out, instead, we can contract its label to undec through the removal of a itself form the store. □

Proposition 6.5. *Let* $F = \langle Arg, R \rangle$ *be an AF on the universe* U, $Arg \subseteq U$, $R \subseteq Arg \times Arg$, $a \in U$ *an argument, and* L *the unique grounded labelling. A possible argument extension revision* $\circledast^{gde,l}_{a,L}(F)$ *could act as:*

- *if* $L(a) = in$,
  - *if* $\exists b \in Arg \mid L(b) = \mathsf{in}$, *add* (b,a) *to* R *and then* $\forall c \in Arg \mid (a,c) \in R$, *add* (b,c) *to* R

  - *otherwise, add* $\langle\{b\}, \{(b,a)\}\rangle$ *to* F *and then* $\forall c \in Arg \mid (a,c) \in R$*, add* (b,c) *to* R

- *if* $L(a) = \mathsf{out}$*,* $\forall b \in Arg \mid L(b) \in \{\mathsf{in}, \mathsf{undec}\}$*, remove* (b,a) *from* R *and then* $\forall c \in Arg \mid (a,c) \in R \wedge L(c) \in \{\mathsf{in}, \mathsf{undec}\}$*, remove* (a,c) *from* R

We give an example of argument extension revision in Table 6.7, where we use the following abbreviations:

- $E_1 \wedge E_2 \to A$ *represents* $E_1 \to E_2 \to A$*;*
- $true$ *represents a dummy* $check^f(\{\}, \{\})$*;*
- $\sum_{a \in Arg} (E(a))$ *represents* $E(a_1) + E(a_2) + \cdots + E(a_n)$*,* $\forall a_i \in Arg$*;*
- $\|_{G}^{a \in Arg} (E(a))$ *represents* $E(a_1) \|_G E(a_2) \|_G \ldots \|_G E(a_n)$*,* $\forall a_i \in Arg$*;*
- $test_c^*(a, S, \sigma) \to A$ *represents* $\sum_{l \in S} (test_c^*(a, l, \sigma))$*.*

$$
\begin{aligned}
\circledast_{a,L}^{gde,\mathsf{out}}(F) : {} & \sum_{b \in Arg} (test_c^f(b, \mathsf{in}, gde) \to add(\{\}, \{(b,a)\}) \\
& \quad \to \|_{G}^{c \in Arg} (check^f(\{c\}, \{a,c\}) \to add(\{\}, \{(b,c)\})) \|_G true \to success \\
& +_P \\
& add(\{b\}, \{(b,a)\}) \\
& \quad \to \|_{G}^{c \in Arg} (check^f(\{c\}, \{a,c\}) \to add(\{\}, \{(b,c)\})) \|_G true) \to success \\
\circledast_{a,L}^{gde,\mathsf{in}}(F) : {} & \|_{G}^{b \in Arg} (test_c^f(b, \{\mathsf{in}, \mathsf{undec}\}, gde) \to rmv(\{\}, \{(b,a)\})) \\
& \quad \to \|_{G}^{c \in Arg} (\\
& \qquad test_c^f(c, \{\mathsf{in}, \mathsf{undec}\}, gde) \wedge check^f(\{c\}, \{a,c\}) \\
& \qquad\quad \to rmv(\{\}, \{(a,c)\}) \|_G true \\
& \qquad) \to success
\end{aligned}
$$

TABLE 6.7: Argument extension revision operator (Proposition 6.5) in CONARG_LANG syntax.

Proof. Given a grounded labelling we want to change the label of a from in to out (or vice versa), while preserving the labels of all other arguments. If a is in, we can look for another argument b labelled in and make b attack a, together with all other arguments attacked by a. If the store does not contain any in argument, we take one from the

universe. If a is out, we remove all the attacks coming from in and undec arguments, so that the only attacks left come from out arguments and a becomes in. To preserve the labels of the other arguments, all attacks from a towards in and undec are removed, since they would have become out after the revision of a. out arguments attacked by a does not need further adjustments. □

Note that the argument extension revision we propose for grounded semantics in Proposition 6.5 is more restrictive than necessary, since ensure all the arguments different from a (that is the argument to be revised) to maintain the exact same labels, while Definition 6.1 only forbids to change the label to undec. For each operator, we also show how to implement it in our language.

Theorem 6.6. *The argument extension expansion, contraction and revision in Propositions 6.4, 6.4 and 6.5, respectively, can be implemented in the* ConArg_lang *language.*

In devising operations of Definitions 6.1 and 6.2, that allow agents for changing the labels of arguments in a shared knowledge base with respect to a given semantics, we reinterpret AGM operators for expansion, contraction and revision. In particular, our operations are restricted to a single argument, rather than considering a set of beliefs as in other approaches like [75] and [23]. Nonetheless, we maintain similarities with the AGM theory, to the point that we can highlight some similarities with the original postulates of [2] that characterise rational operators performing expansion, contraction and revision of beliefs in a knowledge base. For instance, an argument semantics expansion for an argument a and a semantics σ, always produce in output an AF F in which a has its labels incremented in number in at least one labelling of $S_\sigma(F)$.

6.4 Related Work

The research in argumentation deals with both the development of models for representing structured knowledge and the devising of reasoning mechanisms that allow one to draw consistent conclusions (under some semantics). Abstract Argumentation Frameworks are an example of a model that can be used to represent the interaction between debating counterparts and to analyse the acceptability of arguments. Due to the dynamic nature of AFs, much effort has been devoted to study frameworks that can handle changes in terms of structure (arguments and attacks), semantics and extensions. In this section, we review some of the recent works concerning dynamics, description languages for argumentation, and applications in agent-based systems.

Argumentation and Dynamics

A formalism for expressing dynamics in AFs is defined in [146] as a *Dynamic Argumentation Framework* (DAF). The aim of that paper is to provide a method for instantiating Dung-style AFs by considering a universal set of arguments U. A DAF consists of an AF $\langle U, R\rangle$ and a set of evidence, which has the role of restricting $\langle U, R\rangle$ to possible arguments and relations, so to obtain a static instance of the framework. DAFs are built starting from argumental structures, in which a tree of argument supports a claim (corresponding to the root of the tree), and then adding attacks between argumental structures. The dynamic component of a DAF is thus the set of evidence. The introduced approach allows for generalising AFs, adding the possibility of modelling changes, but, contrary to our study, it does not consider how such modifications affect the semantics and does not allow to model the behaviour of concurrent agents.

The impact of modifications on an AF in terms of sets of extensions is studied in [69]. Different kinds of revision are introduced, in which a new argument interacts with an already existing one. The authors describe different kinds of revision differing in the number of extensions that appear in the outcome, with respect to a semantics: a *decisive* revision allows to obtain a unique non-empty extension, a *selective* revision reduces the number of extensions (to a minimum of two), while a *questioning* one increases that number; a destructive *revision* eliminates all extensions, an *expansive* revision maintain the number of extension and increases the number of accepted arguments; a *conservative* revision does not introduce changes on the semantics level (and is strictly connected to the notion of robustness [52]), and an *altering* revision insert and delete arguments in the extensions. All these revisions are obtained through the addition of a single argument, together with a single attack relation either towards or from the original AF, and can be implemented as procedures of our language. The review operator we define in the syntax of our language (as the other two operator for expansion and contraction), instead, does not consider whole extensions, but just an argument at a time, allowing communicating agents to modify their beliefs in a finer grain.

Instead of considering changes in the semantics, the work in [59] focuses on the opposite problem, that is to determine which are the modifications that an AFs can withstand while preserving its set of extensions (similarly to the conservative revision in [69]). The authors focus on grounded and sceptical preferred semantics (which contain exactly one extension) and consider the reinstatement labelling by Caminada [65] in order to characterise changes that preserve the semantics.

Other works, such as [21], define principles for enforcing an argument with the fewest possible number of changes to be made in an AF. In order to formalise the concept of

minimal change, a notion of equivalence between AFs is taken into account. The relations among several types of AF equivalence are investigated in [24], where the authors provide an exhaustive analysis of all the classical semantics. Equivalence represents a useful tool for dealing with dynamics in AFs as it can lead to efficient methods for recomputing semantics when an AF is updated. For example, a special track on dynamics [47] appear in ICCMA 2019[12] [57] (International Competition on Computational Models of Argumentation), where solvers have to compute the set of extensions as changes in the AF are introduced. In our language, we could also consider implementing an operator that takes into account if the AF obtained after the modification is equivalent (with respect to the semantics) to the original one, with the purpose to offer, for instance, an efficient method for checking the admissibility of an argument.

Focusing on syntactic expansion of an AF (the mere addition of arguments and attacks), [23] show under which conditions a set of arguments can be enforced (to become accepted) for a specific semantics. Moreover, since adding new arguments and attacks may lead to a decrease in term of extensions and accepted arguments, the authors also investigate whether an expansion behave in a monotonic fashion, thus preserving the status of all originally accepted arguments. The study is only conducted on the case of weak expansion (that adds further arguments which do not attack previous arguments). The notion of expansion we use in the presented work is very different from that in [23]. First of all, we take into account semantics when defining the expansion, making it more similar to an enforcement itself: we can increment the labels of an argument so to match a desired acceptance status. Then, our expansion results to be more general, being able to change the status of a certain argument not only to accepted, but also rejected, undecided or undetermined. This is useful, for instance, when we want to diminish the beliefs of an opponent agent.

Enforcing is also studied in [75], where the authors consider an expansion of the AF that only allows the addition of new attack relations, while the set of arguments remains the same (differently from [23]). It is shown, indeed, that if no new argument is introduced, it is always possible to guarantee the success of enforcement for any classical semantics. Also in this case, we want to highlight the differences with our work. Starting from the the modifications allowed into the framework, we are not limited to only change the set of relations, since we implement procedures that also add and remove arguments. Moreover, the operators we define are not just enforcement operators, since they allow to modify the acceptability status of a single argument of an AF.

In our model, AFs are equipped with a universe of arguments that agents use to insert new information in the knowledge base. The problem of combining AFs is addressed

[12]ICCMA 2019 website: `http://iccma2019.dmi.unipg.it`.

in [26], that study the computational complexity of verifying if a subset of argument is an extension for a certain semantics in incomplete argumentation frameworks obtained by merging different beliefs. The incompleteness is considered both for arguments and attack relations. Similarly to our approach, arguments (and attacks) can be brought forward by agents and used to build new acceptable extensions. On the other hand, the scope of [26] is focused on a complexity analysis and does not provide implementations for the merging.

Languages for Argumentation

A logical language for handling dynamics in AFs is presented in [84] where arguments and relations are represented by means of propositional variables, and acceptability criteria for arguments are defined through logical formulas. Dynamic evolution is modelled by changing the truth values of variables expressed in the *Dynamic Logic of Propositional Assignments* [12]. Two kinds of modifications can be performed in the framework: adding/removing attack relations and changing the extensions set according to the desired outcome. The authors give a set of postulates expressing some good properties (e.g., the principle of minimal change), and define programs that modify an AF in such a way that a boolean formula becomes true in some extension under a given semantics. However, a straightforward mechanism for dealing with the addition (and deletion) of arguments is not implemented and would require further work. Moreover, this approach considers a global knowledge base and thus it is not suitable for applications in distributed systems, where each agent has its own beliefs.

The revision of agent's beliefs is studied in [93]. A high-level language is provided, that allows one to represent beliefs of agents as *defeasible logic programs*. Ground truth and defeasible arguments are represented with different formalisms, and a reasoning mechanism serves the purpose of changing the belief state of the agents: as new information is acquired, a set of rules revise the logic program of the agent, incorporating (or rejecting) the new knowledge. Also here, no proposal is made of how to handle possible conflicts between agents that could arise in the case of concurrency.

The problem of whether a dialogue action is a legal operation in the system is addressed in [155]. In that paper, there are no assumptions on the order of communications, and the protocols for exchanging arguments guarantee that either an agent cannot cause a violation in the first place or another agent can detect the violation. Such a result is obtained by specifying an operational semantics for *dialogue templates*, namely a formalism that gives conditions under which an agent can send (or receive) a message and how

such message affects the belief base of the sender (and the receiver). This approach is essentially sequential and does not model concurrency between agents.

YALLA is an agent-based language for argumentation proposed in [87]. Similarly to the work in [84], it can be used to represent AFs, characterise extensions and modify the framework. A set of axioms is defined in order to map the logical structure of YALLA into the model of AFs (for instance, an axiom associates the attack relation with a predicate symbol). An AF can be then built using arguments and relations taken from a common universe. In this way, we can consider the belief base of an agent as an AF that is a local, subjective view of such a universe. The notion of defence and the classical Dung semantics are also expressed using logical formulas. The authors also introduce an update operator (based on Katsuno and Mendelzon revision postulates [107]) and a set of properties which are used to describe different possible enforcement operations. As for all the above-mentioned approaches, YALLA does not take into account concurrency between processes (or agents whose belief bases are argumentation frameworks). Consequently, all the operations are done sequentially, as if the agents were taking turns. The authors themselves state that it would be interesting to study the case in which agents can operate simultaneously.

Applications in Agent-Based Systems

Coordination of interactions is a fundamental requirement in multi-agent systems. Concurrent agent-oriented languages, like AgentSpeak [162], provide communication primitives for allowing agents to exchange messages in both synchronous and asynchronous fashion. Also in [162], agents are organised into families, that is classes of agents that offer a set of services to other families. Each family is distinguished by the kind of services it can provide. Three categories of actions are considered: inform, request with wait, and request without wait. A mechanism is also implemented for the prioritisation of concurrent actions. Many works, like [11, 144], use AgentSpeak primitives for implementing more sophisticated reasoning features and the pursuit of goals. However, the simple operations of AgentSpeak-like languages are not sufficient for expressing the more complex interactions in argumentation systems, especially if also humans are involved in the process.

At the moment, chatbots are at the centre of attention for what concern speech interactions and they are studied for their applications in human-computer interaction protocols, like persuasion [64], response selection [166] and also teaching [151]. The authors of a recent survey [1] list some of the most significant techniques for designing intelligent chatbots. What emerges is that the development of the reasoning platform

behind any chatbot is extremely complicated and mostly relies on pattern matching for interpreting the speech in input and providing the output. Moreover, chatbots are usually designed for specific domain applications, viz. health assistance and education, and there is no common approach to address general purpose solutions. The link between chatbots and argumentation as tools for studying and modelling human behaviour is clear. For example, the work in [71] proposes to use chatbots for collecting arguments and counterargument from users in order to generate a knowledge base on a particular matter. On the other hand, our language could provide a reasoning framework for the chatbot itself, by using AFs for modelling interactions with the user, and according to some kind of semantics (like a ranking-based semantics).

Ranking-base semantics allow one to obtain a ranking over the arguments and could be used to determine the preferences of agents involved in negotiation protocols. In [92] a negotiation strategy is presented for agents with ordinal preferences, that is qualitatively expressed in terms of "less than" or "more than". The authors consider the different cases in which agents have global or partial knowledge over the preferences of the others. The presented negotiation protocol requires the agents to take turns for making offers alternately, although other works [114, 152] provide concurrent implementations that we consider more interesting for the work we propose to undertake. We address these and other issues in the next chapter.

Argumentation-Based Systems

A formal model for an argumentation-based system is described in [112], where two agents A and B debate to decide whether B should perform a certain action or not. The reasoning model consists of two components: a model of human motivational sphere which measures how convenient would be for both the agents to let B perform the action, and a reasoning procedure that regulate the final decision. If the two agents have contradictory goals, then they can resort to different strategies for persuading the other by stressing the pleasantness, the usefulness or even the obligation to do the action. If, instead, the agents have a common goal, they can cooperate to find arguments supporting it. An interesting aspect of this model is the possibility of endowing agent B with motivations, for doing the action, which agent A do not know.

The system proposed in [161] allows more than two agents to debate by using a procedure for exchanging arguments and counter-arguments. The agents are divided in two subcategories: the proponents and the opponents, with the usual meaning that the former type of agents proposes arguments supporting a certain issue, and the former tries to defeat those arguments. The system is obtained by using logic programming with weak

and strong negation, together with a simplified version of the argumentation framework from Prakken and Sartor [141] where priorities among rules are not taken into account. The system is centralised, with a communication control allowing agents to take turns to exchange arguments, and if no opponent succeeds in the counterargument, then the proposed argument is justified.

A different approach is implemented by the authors of [134], that develop a mechanism for practical argumentation-based dialogues in multi-agent system. An agent can perform four types of actions: it can *assert*, *accept*, *question* and *justify* a claim. Each agent can assert a claim for which it has an acceptable argument, and can accept a claim for which it has no acceptable arguments against. The action to perform is decided on the basis of an underlying argumentation system that uses unique extension semantics (such as the grounded semantics) to elaborate a strategy for the dialogue game. The dialogue is started by a proponent executing an assertion move and terminates following the acceptance or not of the subject of the dialogue by the opponent. An important achievement of the paper is to make sure that dialogues always terminate, and that they reach the ideal solution under certain conditions. There can also be restrictions on which moves can be performed and by whom. The commitments of each agent are stored in a component which is accessible to all participants in the dialogue. All the agents can read, and only the owner can update the information in its commitment store, making impossible for an agent to hide some of its information. The implementation is obtained using the Jason platform, an extension of the AgentSpeak language.

A general framework for deliberation dialogues in multi-agent systems is introduced in [113]. Deliberation dialogues have a significant cooperative aspect (there is a need for action and the agents need to mutually reach a decision) and differ from persuasion and negotiation, in which conflict plays a major role. Agents in the framework take turns to make moves and propose actions in the dialogue. Preferences on these proposals can also be expressed. A move played from an agent can directly go against a single move of another agent, eventually forming a tree where the dialogical status of each move can be evaluated following a $in \backslash out$ scheme similar to the one of [65]. The outcome of the dialogue is then determined following two orderings of the proposed actions: a preliminary ordering that prefers justifiable options over non-justifiable ones, and an agent option ordering that aggregates preferences of the agents. Additional rules can be introduced, depending on the domain, in order to prevent agents from playing incoherent and inconsistent moves. For example, it may make sense in some applications to prohibit agents from replying to their own moves.

6.5 Conclusion

In this chapter, we introduced a concurrent language for argumentation that can be used by (intelligent) agents to implement different forms of communications. The agents involved in the process share an abstract argumentation framework that serves as a knowledge base and where arguments represent the agreed beliefs. The framework can be changed via a set of primitives that allow the addition and the removal of arguments and attacks. All agents have at their disposal a universe of "unused" arguments to choose from when they need to introduce new information. In order to take into account the justification status of such beliefs (which can be accepted, rejected, undetermined and inconsistent) we considered a four-state labelling semantics. Beside operations at a syntactic level, thus, we also defined semantic operations that verify the acceptability of the arguments in the store. Finally, to allow agents for realising more complex forms of communication (like negotiation and persuasion), we presented three AGM-style operators, namely of expansion, contraction and revision, that change the status of a belief to a desired one; we also showed how to implement them in our language.

Chapter 7

Argumentation Tools

"You cannot mandate productivity,
you must provide the tools to
let people become their best."
– Steve Jobs

Abstract

In this chapter we describe a set of tools we developed to pursue two main objectives. On the one hand we needed to automate certain procedures (e.g., the representation and computation of extensions for AFs) in order to study argumentation dynamics and better understand how to handle the various processing involved in reasoning tasks. On the other hand we wanted to make tools available to facilitate the implementation of practical applications by providing low-level functionalities. The tools we present cover several aspects of argumentation and perform tasks that include computation and representation of labelling semantics for various kind of AFs (classical, weighted, probabilistic), visualisation of invariant operations on AFs, handling of security-related problems where arguments stands for threats and countermeasures, and the implementation for our concurrent language based on argumentation.

7.1 The ConArg Web Interface

Recent works (as the ones presented in [43, 48]) have been carried out with the help of ConArg (see Section 2.1.4). We extended the tool with additional features that allow for handling probabilistic argumentation [48] making use of a probabilistic logic programming language. We also have a module implementing ranking-based semantics for

AFs [55] and a ConArg-based application which deals with cybersecurity issues. All the figures representing AFs in this thesis are made using the ConArg web interface.

7.1.1 Weighted labelling

To facilitate the use of weighted labelling semantics for argumentation-based application, we provide a tool able to represent $\text{WAF}_{\mathbb{S}}$ and visualize the computed labellings for various semantics. For this purpose, we extend ConArg [31] with a series of functionalities for handling weighted argumentation problems. The web interface, which is shown in Figure 7.1, is implemented in JavaScript and relies on a server-side solver written in C.

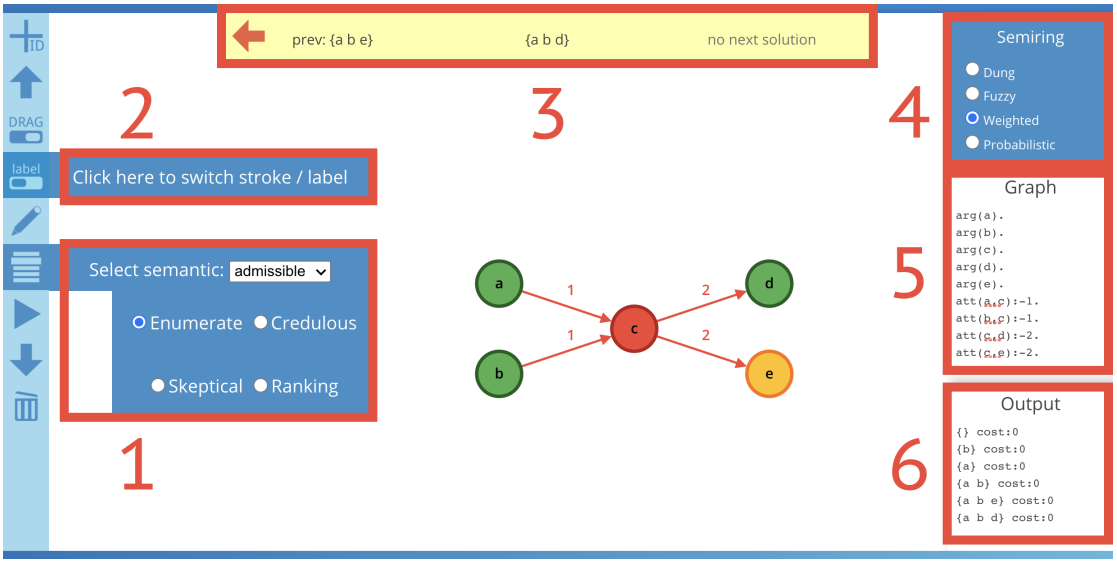


FIGURE 7.1: ConArg web interface displaying a weighted labelling for a $\text{WAF}_{\mathbb{S}}$. The highlighted areas corresponds to: 1) semantics selection, 2) representation of weights by stroke/label, 3) solution selection, 4) semiring selection, 5) input area, 6) output area.

Below, we list the main areas of the interface. **Menu.** Positioned to the left side of the interface, it allows for choosing among different options for both visualising AFs and solving argumentation problems. In particular, it is possible to: import an AF from a local file in aspartix format, change the visualisation of the attacks for weighted AFs, switching the behaviour of the drag action between moving an argument in the canvas and drawing an attack, selecting the kind of problem to solve, run the computation, and save the output as a text file.

Semantics selection panel. Here it is possible to set the parameters for the resolution of several problems. First of all, one is required to select a Dung semantics through the dedicated drop-down menu. For each semantics, four different kinds of problem can be solved. In particular, one can: enumerate the extensions for the chosen semantics, check the credulous/sceptical acceptability of a particular argument, and rank the arguments

by using a PI-based semantics. For the latter problem, it is required to select a power index among the those implemented.

Canvas. This area of the interface has a twofold purpose. On one hand, it is possible to define an AF by drawing nodes and edges. On the other hand, after the calculation of a solution for a certain problem, the canvas allows for visualising the output directly on the displayed AF, through a specific colouration of the arguments. For instance, to display the results of a ranking-based semantics, arguments are assigned a greyscale colour according to their ranking position.

Semiring selection panel. Different paradigms of representation (and solution) for the AFs can be selected, in addition to the classical one. For each semiring, different options on the menu are available.

AF in input. AFs can be entered in this panel. Changes to the canvas also affect this area, that maintains a coherent representation of the AF.

Output panel. The solutions for the various problems solved by ConArg are displayed here. It is also possible to download a text file containing the output.

In the following, we describe an example of use of the tool for weighted argumentation. First of all, we use panel 4 of Figure 7.1 to select a semiring: this determines both the representation of the AF (for instance classical, weighted, probabilistic) and the kind of solution provided by the solver. If weighted is chosen, it is possible to specify a $\text{WAF}_{\mathbb{S}}$ by either using the input area (panel 5) or directly clicking on the canvas to draw arguments and attacks. The next step is to select the semantics (panel 1) for which we want obtain a labelling. Since we selected the weighted semiring, we will obtain a weighted labelling. The solver computes the sets of in arguments, that are then displayed in panel 6. The labellings are directly visible on the $\text{WAF}_{\mathbb{S}}$ through the usual colour scheme: in arguments are green; any arguments attacked by an in is red (that stands for out); all the remaining arguments (i.e., the undec ones) are yellow. In case the solver returns more than one solution for the selected semantics (as happens in Figure 7.1), we can choose which labelling to visualise by using panel 3.

7.1.2 Probabilistic Argumentation

Hunter [103] categorizes probabilistic abstract argumentation frameworks (PrAFs) in two different categories: the *constellation* and the *epistemic* PrAFs. of PrAFs by [117]. A constellation approach to PrAFs defines probabilities over the structure of the AF graph. One can assign probabilities to either the arguments or/and attacks of the AF.

We refer to arguments/attacks with assigned probabilities less than 1 as probabilistic arguments/attacks. We restrict PrAFs to have probabilities attached only in attacks, since in [30] is shown that PrAFs with probabilities only to attacks can represent any general PrAF as defined by [117]. A probabilistic attack $a \to b$ exists in an AF with probability $P(a \to b)$. These probabilistic attacks correspond to random variables, which are assumed to be mutually independent. As such, a PrAF defines a probability distribution over a set of AFs.

We propose a system able to compute the probability of a user given set Q[13] being within the admissible or conflict free semantics under the enumerate inference. One thing we want to point out about semantics in PrAF is that sceptical inference could be seen somewhat differently. When, for example, you are asking whether a set Q is sceptically preferable in a PrAF P the answer would be the probability that exists an AF where Q is sceptically preferable. This is somewhat contraindicative with the notion of sceptical where you want the set to be in all the extensions under the semantics. For that reason in this work we do not consider at all sceptical inference and only focus on credulous inference. We use *MetaProbLog* [120], a framework based on ProbLog [110] probabilistic logic programming language, implemented in Prolog and C. ProbLog extends Prolog programs by annotating facts with probabilities. In that way it defines a probability distribution over all Prolog programs. ProbLog follows the distribution semantics presented by Sato [148]. MetaProbLog extents ProbLog with high order calls. We model the constellation approach through MetaProbLog and we integrate it with the web interface of ConArg [32]. As far as we know, what we present in this work is the first application of this kind which is openly available to the scientific community. Other attempts which are not available online include [102, 117]. MetaProbLog provides several different efficient probabilistic inference methods such as: (i) exact inference based on Reduced Ordered Binary Decision Diagrams (ROBDDs) and dynamic programming [110, 119]; (ii) program (AAF) sampling with memoization [111]; (iii) any-time inference using an iterative deepening algorithm [121].

The web interface exposes two forms of probabilistic inference: exact and AF sampling. The exact inference computes the exact probability; the AF sampling inference is an approximation method. In most cases the exact inference is able to compute the result faster than most approximation methods, such as the AF sampling inference. But exist cases where exact inference is intractable and a user is forced to use an approximation method, for those cases we provide the AF sampling inference. The program sampling inference is based on the use of Monte Carlo methods, that is, to use the ProbLog program to generate large numbers of random subprograms and to use those to estimate

[13] There exist an exponential number of sets Q to the number of arguments that have $0 < P_{sem}(a \in Q) < 1$. For that reason we do not enumerate all.

the probability. We note that each sampled ProbLog program corresponds to sampling an AF. More specifically, such a method proceeds by repeating the following steps:

1. sample a logic (sub)program L from the ProbLog program
2. search for a proof of the initially stated query q in the sample $L \cup BK$
3. estimate the success probability as the fraction P of samples which hold a proof of the query

The implementation of this approach for MetaProbLog, is similar with the one described at [111], and takes advantage of the independence of probabilistic facts to generate samples lazily while proving the query, that is, sampling and searching for proofs which are interleaved. To assess the precision of the current estimate P, at each m samples the width δ of the 95% confidence interval is approximated.

We integrated MetaProbLog with the web interface of ConArg (Figure 7.2) in order to directly take input from the interface and return the probability of queries of PrAFs. The user can select "probabilistic" from the ConArg web interface panel in order to start working with PrAFs, three main tasks can be performed with the aid of the tool: first of all, a framework whose attacks are endowed with a probabilistic value can be given as input and visualized directly in the interface; then the left menu can be used to query for extensions with respect to a given semantics; finally the results of the query are shown both as text and visually on the represented graph. We detail each of these tasks separately.

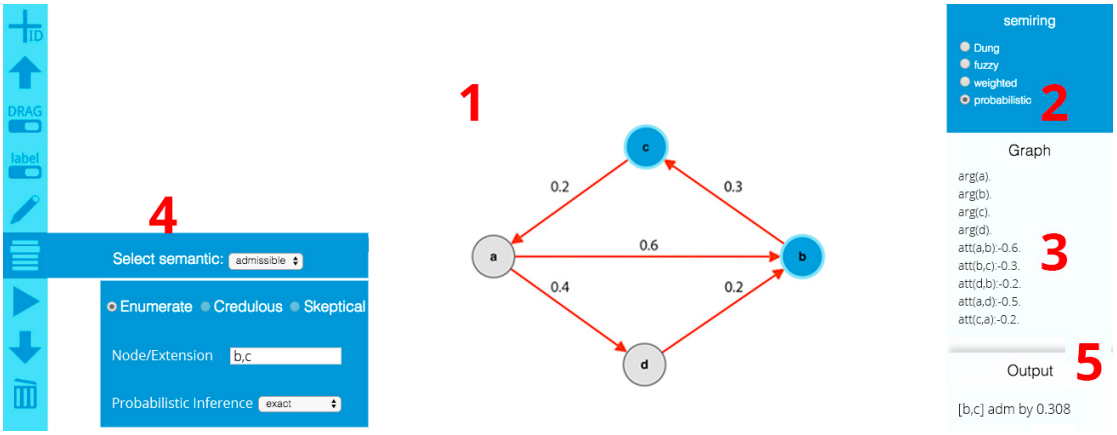


FIGURE 7.2: The ConArg web interface. The marked areas correspond to: 1) example of a PrAF where nodes highlighted in blue represent the subset selected to be checked with respect to a semantics, 2) semiring selection panel, 3) the specification of the PrAF with probabilities attached to attacks, 4) the query panel to select the semantics and specify a node or an extension to test and the type of probabilistic inference, 5) result of the query.

Representation. A visual representation is helpful to study and understand properties of AFs or to look for counterexamples. The input PrAF is entered by drawing on the interface or by typing it in the text box of the right panel (Figure 7.2.3), with the following syntax: arg(a) defines an argument a, and att(a,b):-0.6 denotes that an attack from a to b exists with probability 0.6. The visual and the textual representations are consistent with each other so that a change in one will modify the other accordingly.

Queries. Due to the probabilistic nature of PrAFs, looking for an extension with respect to a semantics means investigating the probability that a set of arguments belongs to that semantics. Once the tool has received the PrAF in input, it is possible to test a subset of arguments in order to obtain the probability with which it is in a certain semantics. All the settings for the query can be configured in a panel of the left menu (Figure 7.2.4). The "Select semantics" options list allows to choose between the conflict free and the admissible semantics, while the arguments can be listed in the "Node/Extension" text field. Moreover, we can specify which type of probabilistic inference has to be used for computing the solution: either "exact" or with "AAF sampling".

Output. The output can be read on the right panel of the web interface (Figure 7.2.5). For instance, the string Set [b,c] is admissible by 0.308 is the output for the query given in the paragraph above as an example, and expresses the fact that the subset $\{b, c\}$ of the PrAF is admissible with a probability of 0.308.

In the constellation approaches, uncertainty in the topology of the graph (probabilities on arguments and attacks) is used to make probabilistic assessments on the acceptance of arguments. As far as we know, this is the first tool based on the constellation approach that is openly offered to the scientific community through a web interface. The goal of this work is to further enrich ConArg and to provide to the scientific community a wider set of problems that can be solved. As future work we plan to also extent the ConArg library [44] to support probabilistic inference. In addition, we would like to investigate and integrate the epistemic approach in ConArg. Contrarily to constellations, in the epistemic approach the topology of the graph is fixed but probabilistic assessments on the acceptance of arguments are evaluated with respect to the relations of the arguments in the graph.

7.1.3 Ranking

In classical argumentation, arguments can be either accepted or rejected according to their justification status, but no further distinction can be done beyond this division

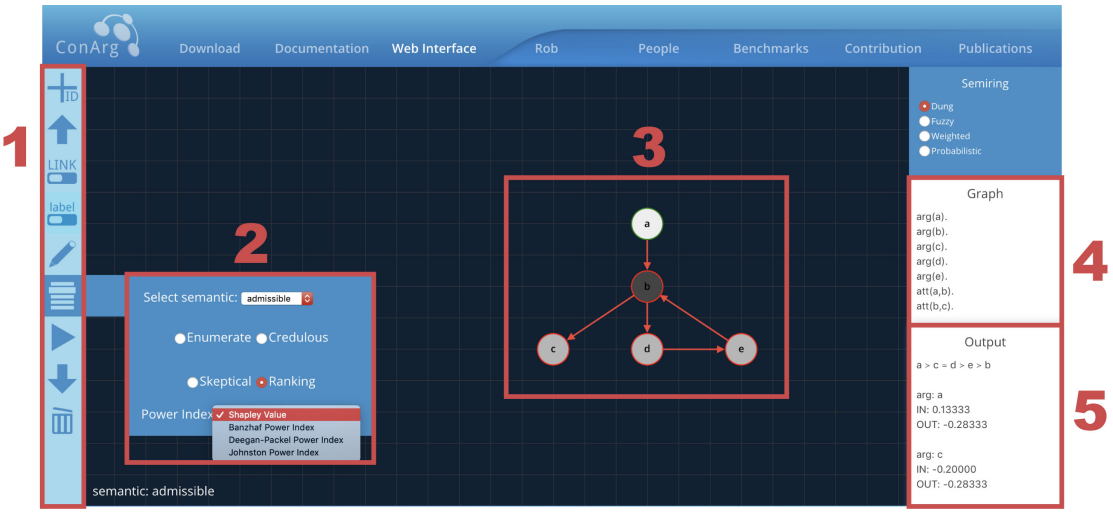


FIGURE 7.3: A screenshot of the ConArg web interface. The highlighted elements are: 1) options menu, 2) semantics selection panel, 3) canvas where the AF is visualised, 4) AF in input, 5) output panel.

into these two categories.[14] On the other hand, ranking semantics allow for assigning an individual score to each argument so that an overall ranking of all arguments can be established by sorting the set of scores. Carrying on the work in [46, 54], we implemented of a ranking function based on the Shapley Value [150], a very well known concept in cooperative game theory, which we use to distribute the scores among the arguments: the more an argument contributes to the acceptability of an extension, the higher its score. In addition, we also take into account a different valuation scheme, the Banzhaf Index [13], and we implement it in order to study the differences with the results obtained through the Shapley Value. Given an argumentation framework, the tool computes the score of every argument over both the ranking schemes introduced above, and its output is a ranking of the arguments with respect to a given semantics. The ConArg Web Interface (see Figure 7.3 for an overview) allows one to easily perform complex argumentation related tasks.

Behind the web interface, ConArg has several modules (like the solver and the ranking script) that allow one to access different functionalities to cope with argumentation problems. In this section, we discuss, in particular, the component of the tool that concerns ranking-based semantics, putting attention on implementation aspects. When we start the computation of the ranking over the arguments of an AF F, the interface calls the ConArg solver that returns the set S of extensions for the chosen Dung semantics σ. These extensions represent the sets of *in* arguments with respect to σ and are formatted as sets of strings (e.g., $S = \{\{a\}, \{a, b\}, \{c, d\}\}$, where a, b, c and d are arguments). Together with the set of extensions, also the framework F and the power index π that we want

[14] More than just two categories have been proposed in the literature, but still from a qualitative point of view.

to use are passed to the ranking script. The script, then, computes the specified power index π for each of the argument in F. The obtained values are approximated to the nearest fifth decimal digit. The two functions that implement the equations of Section 2.2 share a common part, namely v_{S_i}, that represents the evaluation of the contribution of the argument i in forming acceptable extensions. For the sake of efficiency, we compute π only with respect to those sets S such that either S or $S \cup \{i\}$ is an extension for σ. In any other case, the value of $v(S \cup \{i\}) - v(S)$ is zero, so we don't need to do the calculation.

We distinguish between two different characteristic functions: $v^I_\sigma(S)$ and $v^O_\sigma(S)$. As stated in Definition 4.1, the former function takes into account the set of *in* arguments. Given a set of arguments S that does not contain i, if $S \cup \{i\}$ is an extension with respect to σ and S alone is not, then i brings a positive contribution to the coalition, and its own rank will be higher according to $v^I_\sigma(S)$. On the other hand, the latter function ($v^O_\sigma(S)$) only considers arguments that are labelled *out* by σ. In detail, i gets a positive value by π when $S \cup \{i\}$ is a set of *out* arguments and S alone is not. The set $out(L_\sigma)$ is obtained by computing the sets of arguments that are attacked by the extensions of the semantics σ. At this point, each argument of F is associated with the values of the two functions; the resulting structure has the format of an array [arg_name, pi_in, pi_out], where the three components are: the identifier of the argument, the value of the power index π obtained through $v^I_\sigma(S)$, and the value of π obtained through $v^O_\sigma(S)$, respectively.

In order to establish the preference relation between two arguments, the PI-based semantics considers the value pi_in first: the greater the score of an argument with respect to $v^I_\sigma(S)$, the higher its position in the ranking. In case of a tie, i.e., when the value of pi_in is the same for both the arguments that we want to compare, we perform a further control looking at the value of $v^O_\sigma(S)$. Following the principle that accepted arguments are better than rejected ones, the greater the value of an argument with respect to pi_out, the lower its position in the ranking. Consider, for example, two arguments a and b, belonging to F, with the following evaluations obtained through π: [a, 0.2, -0.5] and [b, 0.2, -0.4]. The value pi_in is equal for both a and b, therefore we proceed to confront the values for pi_out. Since $-0.5 < -0.4$, we have that $a \succ^\pi_F b$. The motivation to this kind of ranking is that, while a and b have the same contribution in forming acceptable extensions, a belongs to fewer sets of *out* arguments, that is, a is defeated less times than b. Hence, it is reasonable to prefer a to b.

Finally, when all the arguments are sorted according to the semantics and the power index that we have selected, the results of the computation are displayed in the output panel (frame 5 of Figure 7.3). Along with the overall ranking, we show the pi_in and pi_out values of each argument. For providing a visual hint about which arguments

are the most preferred and which ones the least, we assign a colour to each node of the AF visualised in the canvas. The assigned colours vary in a greyscale, according to the position of the corresponding argument in the obtained ranking: the lighter the colour, the higher the rank (as depicted in the frame 3 of Figure 7.3).

7.1.4 Security Analysis

Starting from ConArg, we develop SecArg [41], a tool to visualise security threats and related countermeasures as arguments, as if security was a continuous dynamic discussion between the administrator and the surveilled system. Existing automated tools to defend a system from such security threats are one potential solution, but a completely automated approach could undervalue the strong analytic capabilities of humans, particularly in problematic situations that require vigilant human oversight. We measure the strength of subsets of arguments and single arguments in accordance with Argumentation Theory. We print such strength degrees in different colours with the purpose to immediately catch the attention of the Security Administrator on what is going on in his system, and help him to take a decision on the set of countermeasures to be considered.

Consider a small research and development company. This company cooperates with other (often large) enterprises for the development of complex goods. Such company possesses high-tech knowledge which has to be protected from competitors. The company needs to efficiently use its resources with the purpose to survive in a highly competitive market. In short, the company has the goal (i.e., asset) of ensuring the productivity of operations (QoS). In this small example, the security-system administrator has identified the following threats and related security controls (in square brackets): hacker penetration (HP) [host IDS (HI), network IDS (NI)] (where IDS stands for Intrusion Detection System), employee abuse (EA) [monitoring functionality (MF), audit procedures (AP)], and compromise of communication channel (CCC) [virtual private network (VPN), encrypted line (EL)]. We would like to emphasise that abstract arguments have no internal structure, and are not "directly linked" to classical logic. For this reason, we can consider multiple sources of information but and belief, such as case law, common sense, and expert opinion. We can consider information coming from multiple network-sensors, in the form of logs, warnings, and errors. Facts and beliefs can be also taken from internal policy documents, and standard documents as well. For instance the Standard of Good Practice for Information Security, is a business-focused, practical and comprehensive guide to identifying and managing information security risks in organizations and their supply chains.

To work on our example we use SecArg[15] (Security with Arguments). The input file passed to SecArg contains the list of arguments partitioned into countermeasures, threats, assets, and attacks between them: for instance, countermeasure(HI), threat(HP), att(HI,HP) (hacker penetration is prevented by a host IDS). SecArg visually represents the different nature of arguments with different colours: green for countermeasures, red for threats, and yellow for assets. A more extended example is represented in Figure 7.4. The

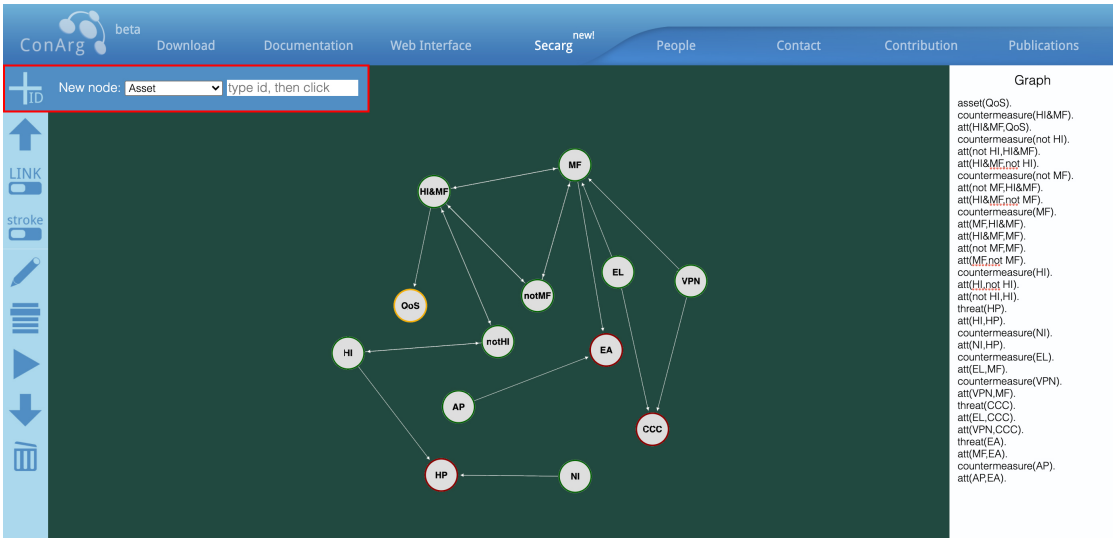


FIGURE 7.4: SecArg interface showing an AF with a QoS asset (represented with a yellow border), controls (green) and threats (red). The panel highlighted in red allows to select the type of node (between asset, threats and countermeasures).

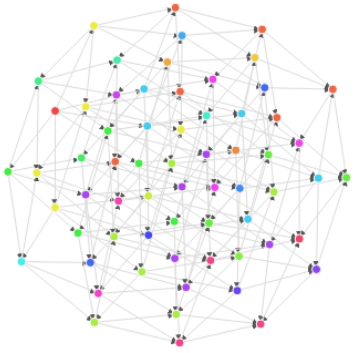
depicted AF represents a situation in which executing a host IDS and a monitoring functionality on the same machine (i.e., HI&MF) impacts on its QoS. Hence, we pose an attack between them, and we also consider not having HI (NotHI) or MF (NotMF). Moreover, we have some countermeasures in conflict, i.e., EL or VPN, and MF. We obtain three stable extensions (we use the stable semantics because it is the most sceptical one): i) {AP, VPN, EL, HI, NI, NotMF, QoS}, ii) {AP, VPN, EL, HI, NI, HI&MF}, and iii) {AP, VPN, EL, NI, NotHI, NotMF, QoS}. In this case, reasoning in terms of stable or preferred semantics is the same, since they both returns the same three extensions. Reasoning on the sceptical acceptance of arguments in such extensions, we obtain that AP, VPN, EL, NI are sceptically accepted. This means that, for the attack/countermeasure scenario we have depicted, having audit procedures, a virtual private network, an encrypted line, and a network IDS is always considered a valid argument. Therefore, they correspond to a strong suggestion for the security administrator. On the other hand, there are some other arguments that are rejected (see Def. 3), that is they never appear in such extensions; for instance EA, HP, MF, and CCC. All three threats are successfully avoided, in the sense that adopted security countermeasures always prevent

[15]SecArg web interface: `http://www.dmi.unipg.it/secarg/`.

all of them. Moreover, also adopting the monitoring functionality countermeasure is not a good idea given this scenario, since it is rejected as well. Finally, the remaining arguments appear sometimes but not always in such three extensions (they are credulously accepted): NotHI (in 1 extension), HI&MF (1), HI (2), NotMF (2), QoS (2). This can be interpreted as a strength-score for these arguments: for instance, having an host IDS beats not having it (2 to 1): hence the administrator is recommended to use it.

7.2 Visualising Robustness with ConArg_rob

In order to better understand problems related to argumentation and robustness (introduced in Chapter 3), we developed ConArg_rob, a graphical tool capable of representing different aspects of an AF. An alpha version of the tool (showed in Figure 7.5) allows to display all the AFs obtainable by any combinations of attacks between a fixed number of arguments.



Semantic Sort | Count Sort

1.	#1	{a b c} {a b} {a c} {b c} {a} {b} {c}
2.	#6	{c}
3.	#1	{a b} {a c} {a} {b} {c}
4.	#1	{a b} {b}
5.	#1	{a b} {b} {c}
6.	#5	{b c} {b} {c}
7.	#1	{a c} {c}
8.	#6	{b}
9.	#1	{a c} {b} {c}
10.	#3	{b} {c}
11.	#5	{a b} {a} {b}
12.	#1	{a b} {a}
13.	#1	{a b} {b c} {a} {b} {c}
14.	#1	{a b} {a} {c}
15.	#1	{a b} {a} {b} {c}
16.	#1	{b c} {b}
17.	#5	{a c} {a} {c}
18.	#1	{b c} {c}
19.	#1	{a c} {a}
20.	#2	
21.	#1	{a c} {b c} {a} {b} {c}
22.	#1	{a c} {a} {b}
23.	#1	{a c} {a} {b} {c}
24.	#6	{a}
25.	#1	{b c} {a} {c}
26.	#3	{a} {c}
27.	#1	{b c} {a} {b}
28.	#1	{b c} {a} {b} {c}
29.	#3	{a} {b}
30.	#1	{a} {b} {c}

toggle all

FIGURE 7.5: Alpha version of the ConArg_rob interface showing the graph with the coloured nodes on the left and the legend on the right. Each node in the graph represents an AF.

Each of these AFs is represented as a node in an oriented graph and each node in this graph is connected by an edge to all other nodes, whose corresponding AF only differs by one attack. All of the nodes in the graph have an assigned color depending on the extensions (computed by ConArg) that satisfy a given semantics: different AFs that return the same set of extensions for that semantics have the same color in the graph.

A legend displays all the sets of extensions divided by color. At last, by mouse hovering a node it is possible to have a preview of the corresponding AF and its description in classical format, together with the computed set of extensions for a given semantics.

Starting from the alpha version, we obtained the ConArg_rob web interface[16], that allows displaying all the AFs obtainable by any combinations of attacks between a fixed number of arguments. Each of these AFs is represented as a node in an oriented graph and each node in this graph is connected by an edge to all other nodes whose corresponding AF only differs by one attack. Moreover, when a certain AF is selected by clicking on it, the tool displays the graph relative to that AF and the sets of extensions obtainable for each semantics. So, the notion of robustness exploited by ConArg_rob is the partial robustness. The tool's menu (Figure 7.6) allows for selecting the number of arguments that the AFs will contain and one can choose whether to take in account or not AFs in which arguments attack themselves (self attacks), defend themselves from every other attack (symmetric AFs) or attack without direct counterattacks. In Figure 7.6, for example, we chose to consider only all the AFs with 3 arguments and no self attacks.

FIGURE 7.6: A screenshot of the tool menu.

Whenever the "draw" button is clicked the lattice of all AFs is drawn. Clicking on a node allows displaying a lattice of extensions for the corresponding AF. These extensions are computed by ConArg. Each set is coloured according to the less inclusive semantics to which it belongs. For instance, a set marked as admissible will imply its belonging to conflict free semantics too. At last, selecting a set of extensions will show all the AFs on the first panel for which there exists a semantics containing those extensions and each AF will be colored according to that semantics. The notion of partial robustness is well represented in Figure 7.7: on the left, we can see the graph in which colored nodes are those representing the AFs allowing the selected extension $\{2\}$ and in the right panel we can see that even if the semantics changes together with the AF, it always contains the extension $\{2\}$.

The features introduced in this version of the tool allowed us to get some significant results. At first, we focused on the set of symmetric AFs, in which every attack is bidirectional, i.e. if a and b are two arguments of an AF and a attacks b, then b attacks a and we found out some interesting properties, also suggested by Coste-Marquis et al. in [73]. For instance, we observed that all the conflict-free extensions are also admissible. Indeed it holds that an extension in a symmetric AF is admissible if and only if it is

[16]Link to ConArg_rob: `http://dmi.unipg.it/conarg/rob.html`.

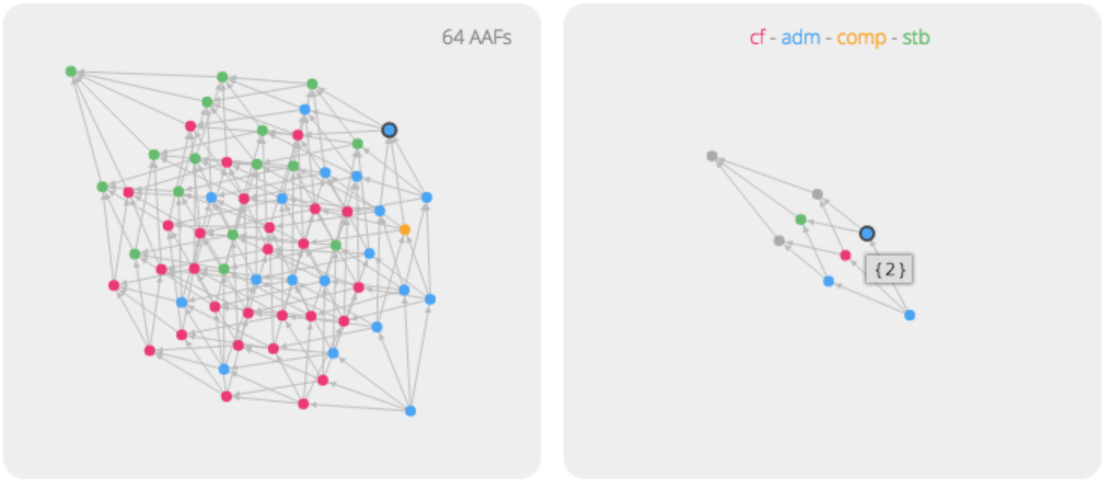


FIGURE 7.7: On the right, the graph of extensions in which the set {2} is selected. On the left, the lattice of AF showing in which semantics the set {2} appears.

conflict-free. Another property we observed concerns the grounded extensions: in a symmetric AF the grounded extension is given by the set of arguments which are not attacked. If every argument is attacked, then the empty set is the grounded extension. In Figure 7.8 we can see that the argument with label 3 (on the left graph) is the only one which is not attacked. Hence the extension set containing the only argument 3 (the yellow node in the right graph) is the grounded extension (and in this example, it is also the only complete extension). Changing the parameters used for the representation (number of arguments and attacks type) it is possible to exploit the functionality of the tool in order to study semantics properties, with respect to the inclusion between AFs, linked to the notion of robustness.

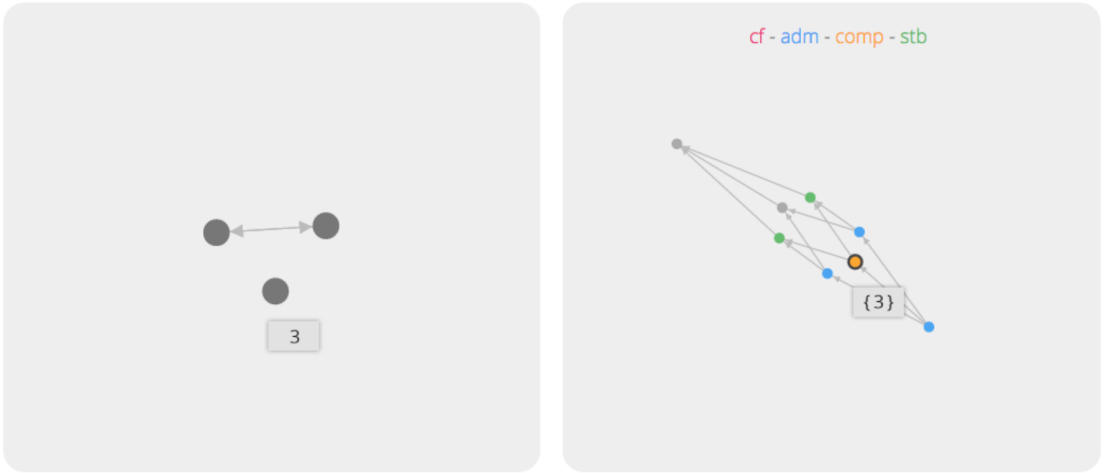


FIGURE 7.8: A representation of 3 arguments AFs lattice with a selected AF drawn on the right side.

The presented work is a first step towards the study of the concept of robustness in AFs. Several works in the literature concern the connection between consistency of beliefs

and the preservation of semantics in knowledge base when changes are introduced, but none of them focuses on studying the implications of the changes in terms of variations of attacks between arguments. By using ConArg_rob as a support for our study, we obtained as a first result the definition of invariant operators with respect to the conflict-free and the admissible semantics [52], able to introduce changes (in terms of addition of an attack) into an AF, while preserving the whole set of extensions. These operators introduce a new possible approach for ordering AFs: instead of attack inclusion that we are currently considering, AFs could be ordered with respect to semantics, using invariant operators for finding adjacent AFs. By our knowledge, this is the first tool that aims to help analyse properties of AFs.

7.3 ConArg_lang Implementation

We develop a working implementation for the language illustrated in Chapter 6. We use python and ANTLR[17] (ANother Tool for Language Recognition) for the server-side interpreter, and the usual web programming technologies (HTML, JavaScript and PHP) for the web interface.

7.3.1 The Interpreter

ANTLR is a parser generator for reading, processing, executing, and translating structured text. Starting from a grammar file, ANTLR generates a parser that can build and walk parse trees. ANTLR provides two ways of traversing the syntax tree: either trough a listener (the default option) or a visitor. The biggest difference between the listener and visitor mechanisms is that listener methods are called independently, whereas visitor methods must walk their children with explicit visit calls. Not invoking visitor methods on the children of a node means those subtrees are not visited. Since we want to implement guards in our language, we need the possibility to decide which part of the tree will be visited, making our choice fall on the visitor approach.

Our projects consists of a grammar file and seven python classes, the most interesting being the *CustomVisitor*, in which we define the behaviour of the parser, and the class *ArgFun* containing all the auxiliary argumentation-related functions used to process the knowledge base of the agents (that is, indeed, an AF). We define our grammar starting from the syntax given in Table 6.1 and we obtain a .g4 file (supported by ANTLR version 4) of which we show the main part in Table 7.1. Capitalized words are placeholder for terminals specifying syntactic elements of the language: for instance, `ARROW` stands for

[17] ANTLR website: `https://www.antlr.org/`.

```
grammar CA;

program
    :   par_action SEMICOLON                                          #prg
    ;

par_action
    :   action (PAR action)*                                          #par
    ;

action
    :   '(' action ')'                                                #pac
    |   'add(' (EMP | ARGS) ',' (EMP | ATKS) ')' ARROW action         #add
    |   'rmv(' (EMP | ARGS) ',' (EMP | ATKS) ')' ARROW action         #rmv
    |   expression                                                    #exp
    |   SUCCESS                                                       #suc
    |   FAILURE                                                       #flr
    ;

expression
    :   '(' expression ')'                                            #pex
    |   expression_w                                                  #exw
    |   expression_f                                                  #exf
    |   'sum(' expression_w (',' expression_w)* ')'                   #ndt
    |   'gpar(' expression_f (',' expression_f)* ')'                  #gpa
    |   expression_f (PPLUS expression)*                              #ite
    ;

expression_w
    :   'checkw(' (EMP | ARGS) ',' (EMP | ATKS) ')' ARROW action     #ckw
    |   'testcw(' (EMP | ARGS) ',' LABEL ',' SEM ')' ARROW action    #tcw
    |   'testsw(' (EMP | ARGS) ',' LABEL ',' SEM ')' ARROW action    #tsw
    ;

expression_f
    :   'checkf(' (EMP | ARGS) ',' (EMP | ATKS) ')' ARROW action     #ckf
    |   'testcf(' (EMP | ARGS) ',' LABEL ',' SEM ')' ARROW action    #tcf
    |   'testsf(' (EMP | ARGS) ',' LABEL ',' SEM ')' ARROW action    #tsf
    ;

ARROW
    :   '->'
    ;

   ...
```

TABLE 7.1: Part of .g4 file specifying the CONARG_LANG grammar.

the symbol `->`, `PAR` corresponds to ||, and `ARGS` is any list of literals enclosed in curly brackets.

Starting from the grammar, ANTLR automatically generates all the components we will use for parsing the language, the most remarkable being the list of used tokens, the interpreter containing names for literals and rules, and symbolic names for the tokens, a lexer which recognises input symbols from a character stream, the parser itself (endowed with all the necessary support code) and the visitor class. Then, we need to manually

override the default methods provided in the visitor to customise the behaviour of the parser. The visit of the parse tree always starts with the execution of the function *visitPrg*, which recursively visits all its children. The parser recognises twenty types of node (the non terminal elements in the grammar), identified through a three-letter code preceded by # (see Table 7.1). These codes are then used as a shortcut to recall nodes for which we want to specify a desired behaviour. Below, we provide details on the implementation of visiting functions.

- *visitPrg*: calls the visit on its children, collects the results and, in case of termination, returns the output of the whole program.
- *visitPar*: starts two separated threads to execute (visit) two actions in parallel, returning true if both succeeds, false if at least one action fails, and suspends if an action is waiting for its guard to become true.
- *visitAdd* and *visitRmv*: modify the AF by either adding or removing part of the AF, respectively. Always succeeds and continues on the children. Note that *visitRmv* succeeds also if the specified arguments and/or attacks are not in the AF. In that case, the AF is left unchanged.
- *visitSuc* and *visitFlr*: correspond to visits to terminal nodes and return true (success) and false (failure), respectively.
- *visitNdt*: implements a concatenation of + operators, inspecting the guards of all its children and randomly selecting a branch to execute among the possible ones. A guard can be a waiting check or either of the waiting tests. If no guards are found with satisfiable conditions, *visitNdt* waits for changes in the AF until some child can be executed.
- *visitGpa*: implements a concatenation of $\|_G$ operators. Execute all its children in separated threads. Contrary to *visitNdt*, *visitGpa* only works with expressions that can fail (and do not suspend), thus allowing for two possible outcomes, that is success if at least one expression succeeds, and failure if all expressions fail.
- *visitIte*: behaves like an if-then-else construct. The first child must be an expression with guaranteed termination (either success or failure). The children are executed in the same order in which they are specified and as soon as a satisfiable guard is found, the corresponding branch is executed. Since some of the children can be waiting expression, *visitIte* is not guaranteed to terminate.
- *visitCkw* and *visitCkf*: check if a given set of arguments and/or attacks is present in the knowledge base. In case of success, both nodes proceed visiting the consequent action. On the other hand, when the knowledge base does not contain

the specified parts of AF, *visitCkw* waits for the condition to become true, while *visitCkf* immediately returns false and and leads to branch failure.

- *visitTcw*, *visitTcf*, *visitTsw* and *visitTsf*: call the ConArg solver to execute credulous and sceptical tests on the acceptability of a given set of arguments. As with the checks, the test functions are also available in two versions, one that always terminates (with either a success or a failure) and the other that possibly suspends and waits for the condition to become true.

In addition to the visiting functions, we have a set of core functions responsible for managing auxiliary tasks, like starting new threads when a parallel composition is detected, making changes to the shared AF and computing the semantics for the test operations. All the components are put together in the *Main* class, which takes in input and runs the user-defined program. First of all, the input stream (a string containing the definition of the program to run) is passed to the lexer, which extracts the tokens and sends them to the parser. Then, the parser uses the tokens to generate a tree ready to be traversed (see Figure 7.9 for an example.). Finally, the visitor walks the tree and executes the program. We conclude this section by illustrating the execution of some ConArg_lang programs.

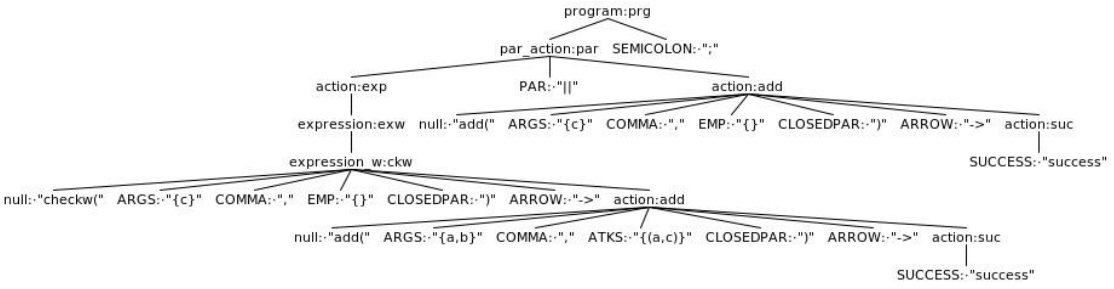


FIGURE 7.9: Parse tree for the ConArg_lang program in Example 7.1.

7.3.2 The Web Interface

To facilitate the use of the tool we develop a web interface[18] exposing the functionalities of ConArg_lang. The interface (see Figure 7.10) consists of a web page divided into three main areas: an input form, one text box for the program output and one for the shared AF. The output of our tool shows, for each step, the executed operation and the remaining part of the program, together with the results of check and test operations.

The user can manually input a program in the designated area (either typing or pasting the code) or can select a sample program from those available in the drop down menu next to the heading. The two buttons below the input area run the program and display

[18] ConArg_lang web interface: `http://www.dmi.unipg.it/conarg/lang`.

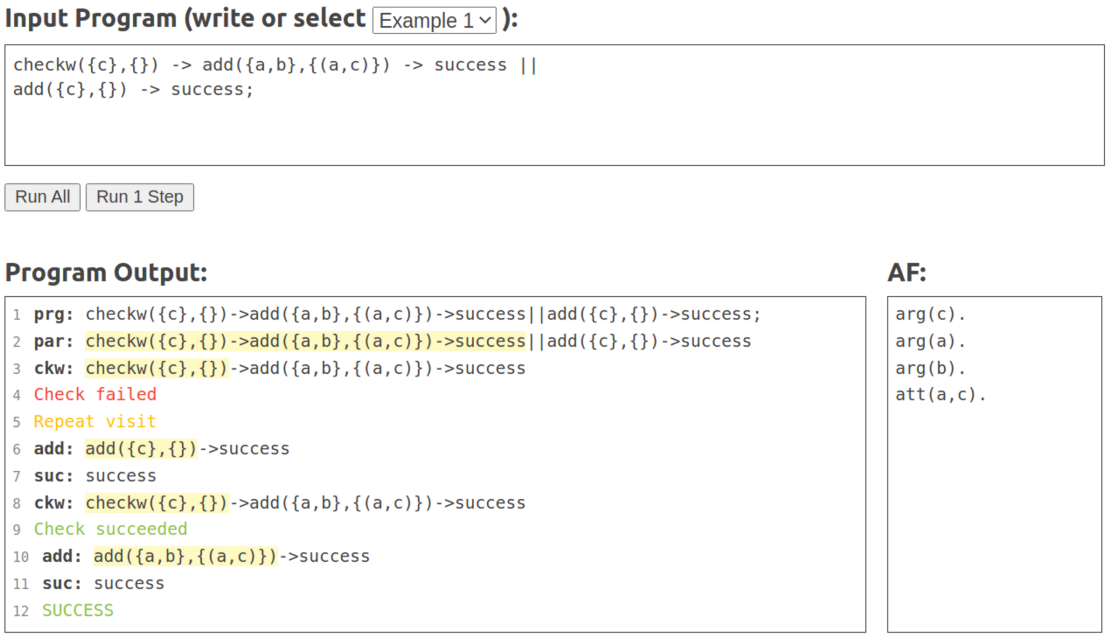


FIGURE 7.10: Execution of a the CONARG_LANG program in Example 7.1.

the result in two different ways. Clicking the button "Run all", the result of the whole program is immediately displayed in the area below and the AF shown on the right represent the final state of the shared store. On the other hand, the button "Run 1 step" shows, as the name suggests, one step at time: each click on the button makes another step of the execution appear in the output area. The AF on the right side is updated after each `add` or `rmv` operation, showing the evolution of the underlying knowledge base. Note that the difference between the two usable modes is only in the visualisation, since both compute the whole result beforehand. This means that if the execution does not terminate, for instance because of the presence of a unsatisfiable waiting operation, the process will hang and nothing will be displayed. Regardless of the chosen method, the executed operation is highlighted in yellow in each line of the output. We conclude this section by illustrating the execution of some CONARG_LANG programs.

Example 7.1 (Parallel actions)**.** *Consider the program below, in which two actions are executed in parallel.*

```
checkw({c},{}) -> add({a,b},{(a,c)}) -> success ||
add({c},{}) -> success;
```

The first action corresponds to a syntactic check followed by ad add, while the second one consists of an add operation with no prerequisites. Running the program produces the results in Figure 7.10. Note that the AF representing the knowledge base is always empty at the beginning.

In line 1 of the output, the parser recognises a valid ConArg_lang *program, i.e., a composition of parallel actions ending with a semicolon. Two threads (one for each action) are started. In this example, the action that occurred first in the program is also executed first, but in general it can happen in any order. In line 3, the program executes a waiting* `checkw`*: if the AF contains an argument c then the visit on that branch can continue (and the add operation is executed). Otherwise, the* `checkw` *is repeated until it (possibly) becomes true. Since the AF is empty by default and no other action has modified it yet, the check on the AF return a negative answer (line 4). In the meanwhile, the* `add` *operation of the second thread is executed in line 6. The AF is modified accordingly, introducing an argument c. AF* $= \langle \{c\}, \{\} \rangle$*. This branch of the execution terminates in line 7 with a success. At this point, the check of the first thread (which had previously given negative results) is repeated, this time giving an affirmative answer (lines 8 and 9). The execution then continues in line 10 with the* `add` *operation which produces further modifications on the AF. At this point,* $AF = \langle \{c, a, b\}, \{(a, c)\} \rangle$*. This branch successfully terminates in line 11 and since both the parallel actions of our program succeed, the whole program terminates with a success (line 12).*

Example 7.2 (Nondeterminism)**.** *We have the following program with a parallel composition and a nondeterministic operation.*

```
add({a,b},{}) -> sum(
checkw({c},{}) -> add({},{(c,a)}) -> success,
testcw({a},in,complete) -> rmv({b},{}) -> success
) ||
add({c},{}) -> success;
```

It is possible to obtain different outcomes according to the order in which the thread handling the parallelism are executed. We show an example in Figure 7.11.

Program Output:

```
1 prg: add({a,b},{})->sum(checkw({c},{})->add({},{(c,a)})-
>success,testcw({a},in,complete)->rmv({b},{})->success)||add({c},{})->success;
2 par: add({a,b},{})->sum(checkw({c},{})->add({},{(c,a)})-
>success,testcw({a},in,complete)->rmv({b},{})->success)||add({c},{})->success
3 add: add({a,b},{})->sum(checkw({c},{})->add({},{(c,a)})-
>success,testcw({a},in,complete)->rmv({b},{})->success)
4 ndt: sum(checkw({c},{})->add({},{(c,a)})->success,testcw({a},in,complete)-
>rmv({b},{})->success)
5 add: add({c},{})->success
6 suc: success
7 tcw: testcw({a},in,complete)->rmv({b},{})->success
8 Test succeeded
9 rmv: rmv({b},{})->success
10 suc: success
11 SUCCESS
```

AF:

```
arg(a).
arg(c).
```

FIGURE 7.11: Execution of a the ConArg_lang program in Example 7.2.

After identifying the program in line 1 and the parallel composition in line 2, the visit of the tree proceeds with the execution of the `add` *operation of the first thread, which introduces in the AF two new arguments, namely a and b (line 3).* $AF = \langle \{a, b\}, \{\} \rangle$. *The node corresponding to a nondeterministic choice on the same thread is visited immediately after in line 4. It is important to note that our implementation of the* `sum` *inspects all the guards on child nodes and selects a verified one (if any) at random. In the program we are analysing, the child of* `sum` *are* `checkw({c},{})` *and* `testcw({a},in,complete)`*, and only the latter is true at the time of the verification, meaning the former will be ignored. Then the program continues executing the other thread, which adds an argument c to the AF and terminates with a success (lines 5 and 6).* $AF = \langle \{a, b, c\}, \{\} \rangle$. *At this point,* `checkw({c},{})` *becomes true, but the choice on which expression will be executed has already been made. The remaining thread resumes its execution performing the* `testcw` *operation (line 7). The waiting test succeeds on the first try in line 8, leading to the removal of argument b (line 9), as specified by the parse tree. Now we have* $AF = \langle \{a, c\}, \{\} \rangle$. *The branch and the whole program also succeed (lines 10 and 11).*

Example 7.3 (If-then-else)**.** *We run the following* ConArg_lang *program, whose result is shown in Figure 7.12.*

```
add({a,b},{(a,b)}) ->
checkf({c},{}) -> add({d},{}) -> success +P
testcf({b},in,complete) -> add({e},{}) -> success;
```

Program Output:

```
1 prg: add({a,b},{(a,b)})->checkf({c},{})->add({d},{})-
>success+Ptestcf({b},in,complete)->add({e},{})->success;
2 par: add({a,b},{(a,b)})->checkf({c},{})->add({d},{})-
>success+Ptestcf({b},in,complete)->add({e},{})->success
3 add: add({a,b},{(a,b)})->checkf({c},{})->add({d},{})-
>success+Ptestcf({b},in,complete)->add({e},{})->success
4 ite: checkf({c},{})->add({d},{})->success+Ptestcf({b},in,complete)->add({e},{})-
>success
5 ckf: checkf({c},{})->add({d},{})->success
6 Check failed
7 tcf: testcf({b},in,complete)->add({e},{})->success
8 Test failed
9 FAILURE
```

AF:

```
arg(a).
arg(b).
att(a,b).
```

FIGURE 7.12: Execution of a the ConArg_lang program in Example 7.3.

After initialising the AF with two arguments and an attack between them in line 3 ($AF = \langle \{a, b\}, \{(a, b)\} \rangle$*.), the program executes an if-then-else construct (line 4). The first condition consists of a* `checkf` *operation, which immediately fails (lines 5 and 6). Thus, the program proceed with the second condition, this time a* `testcf`*, that also fails (lines 7 and 8). Since both conditions fail, also the program terminates with a failure in line 9. We remark that more than two conditions can be declared by the use of* `+P` *and only the last one can be a waiting expression.*

7.4 Related Work

In addition to those presented in this chapter, many other tools can be found in the literature that either use or are used to study argumentation. Below we list some of the most relevant with respect to our research and which, however, depart from the implementations we provided in our work for both functionality and design.

From what concerns semantics computation, it is worth mentioning the abstract argumentation reasoner μ-toksia [130], which ranked first in the main track of ICCMA 2019 [57]. It supports both enumeration (find some/all extensions of an AF) and decision (in the sceptical/credulous sense) tasks for admissible, complete, preferred, stable, semi-stable, stage, grounded and ideal semantics. The system, implemented in C++, is based on a SAT solver that performs iterative computations trough its API. Differently from this work, we didn't focus on the implementation of a reasoner itself, but rather on its integration within a graphical interface.

The *Tweety* library collection to artificial intelligence and knowledge representation [158] offers support for dealing with both abstract and structured argumentation, and present a general and versatile collection of Java classes to deal with various aspects of different approaches. Tweety allows to define and manipulate AFs and compute extension-based semantics among grounded, stable, complete, preferred, ideal, semistable, CF2, and stage. Some ranking-based semantics can be used as well. Concerning structured argumentation, Tweety implements popular approaches as ASPIC+, ABA, DeLP and deductive argumentation, which are then reduced to reasoning on classical AFs for enabling semantics computation.

Other tools have been devised for learning purposes. ArgTeach [149], for instance, is web platform which can be used for teaching labelling-based semantics. The user has to mark the arguments on a proposed AF (selected form a limited set) so as to obtain a complete labelling. Even partial labellings are accepted and suggestions are given on how to label the remaining arguments. The tool then checks the correctness of the solution and provides hints in case of error.

Regarding graphical tools involving argumentation, the authors in [138] present a tool for logic-based argumentation able to deal with priorities over rules and burden of persuasion. The user interface allows to insert in input an argumentation graph and to obtain in output the corresponding argument labelling, together with the step by step resolution process. The tool only works with the grounded semantics and is mainly devoted to legal applications.

7.5 Conclusion

In this chapter, we presented the tools developed following the work done for this thesis. The tools are aimed at dealing with problems arising from the investigation of different aspects of argumentations and helped us in carrying out our research. To begin with, we extended the ConArg suite with many features, ranging from the implementation of the labelling semantics for graphically visualising sets of extensions to the possibility of handling weighted and probabilistic argumentation. We also equipped the web interface with a module for dealing with ranking-based semantics, and we developed an argumentation-based application for treating cybersecurity problems. Then we described ConArg_rob, a tool that allows visualising a lattice of AFs ordered by the number of attack relations and that helped us in the study of robustness. With ConArg_rob, one can examine how semantics changes when attacks are added/removed from an AF, making it easier to identify conditions to obtain invariant operators that preserve the set of extensions. Finally, we discuss the implementation of ConArg_lang, our concurrent language based on argumentation, giving details of both the server-side interpreter and the web interface.

Chapter 8

Conclusions and Future Work

In this thesis, we studied argumentation from the point of view of dynamics and, in particular, we were interested in the ability to manage the evolution of information. We considered different aspects of argumentation and devised theoretical and practical tools useful for developing argumentation-based applications in the context of multi-agent systems, where complex interactions between agents need to be modelled and handled. The main results we obtained are summarised in the following.

Robustness In AFs (Chapter 3)

We introduced the notion of robustness in AFs, which is the property of a framework to withstand syntactic changes while preserving the semantics (in the sense of sets of extensions). The robustness degree of an AF is then computed as the number of changes that can be performed without affecting the semantics: the more the changes, the more robust the framework. We are thus able to sort AFs based on their robustness degree.

Ranking-Based Semantics (Chapter 4)

Besides sorting AFs, we were also interested in sorting the arguments themselves in order to identify the most suitable ones for a revision process. Therefore, we devised a ranking-based semantics which evaluates the arguments of an AF by using power-indexes (a well-established concept from cooperative game theory). Our approach distributes preferences among arguments taking into account classical Dung semantics, allowing for a more accurate ranking with respect to the desired acceptability criterion. Our study on ranking-based semantics also involved semi-structured AFs, in order to get closer to the application on real-world problems. We considered CAFs (AFs extended with claims),

and we proposed a method for lifting a ranking from the level of arguments to the level of claims. We then gave conditions under which the properties of a ranking are preserved after the lifting.

Four-State Labelling (Chapter 5)

We defined a four-state labelling semantics for AFs that marks arguments on a finer grain with the labels in, out, undec and empty, the last one meaning "don't care". We showed that such semantics identifies the same sets of arguments (those labelled in) accepted by the classical Dung semantics while providing additional information on the rejected arguments (that are further partitioned into out, undec and empty). We then extended this kind of labelling to weighted AFs, considering different notions of collective defence from the literature.

Concurrent Language for Argumentation (Chapter 6)

We introduced a concurrent language for argumentation that allow agents to communicate through a shared knowledge base represented by an AF. Agents can add, remove, check the existence and test the acceptability of arguments in the AF through a set of primitives. We used four-state labelling semantics for the test operations and we presented AGM-style operators for expansion, contraction and revision of the knowledge base in order to allow complex interaction between the agents.

Argumentation Tools (Chapter 7)

The studies conducted in this thesis were accompanied by the development of various tools able to deal with problems arising from the investigation of different aspects of argumentations and that helped us in carrying out our research from both theoretical and practical perspectives. For instance, we extended the ConArg suite with many features for handling weighted and probabilistic AFs and for computing ranking-based semantics. We also developed an argumentation-based application for treating cybersecurity problems. Then we created ConArg_rob, another tool that allows visualising a lattice of AFs sorted by the number of attack relations for studying of robustness. Finally, we provided an implementation of ConArg_lang presenting both a server-side interpreter and a web interface.

Future work

To conclude, we discuss some possible directions in which the presented work could be extended.

On Robustness

The study we presented in Chapter 3 concerning the notion of robustness has a wide set of future perspectives. First, we plan to design invariant operators with respect to the complete, stable, semi-stable, preferred and grounded semantics (until now studied only with respect to single extensions [145]). We would like to find the sets of arguments which are essential to preserve the whole semantics. Every change inside those sets modifies the semantics, while changes outside do not cause any alteration. By removing the non-core part of AFs, it is possible to obtain equivalent frameworks for which the computation of extensions is faster, especially for checking credulous/sceptical acceptance of arguments.

Then, different notions of equivalence, e.g. local equivalence [131], could be taken into account, and additional modifications of AFs could be considered, as the deletion of attack or the addition/removal of arguments. We also plan to devise a more general notion of robustness, involving the new modifications proposed above. By relaxing the conditions underlying invariant operators, and thus allowing the semantics to change, other operators could be obtained, that allow reaching "compromises": if two parts of a debate desire two different outcomes in terms of semantics, a compromise can be reached as a third semantics, that is the closest one with respect to those desired by both the counterparts. Definitions of closeness could be devised as well. We also want to study local addition operators for semiring-based weighted AFs [40, 50].

Concerning ConArg_rob, the tool for robustness presented in Chapter 7, many other problems could be approached by inspecting properties of AFs. For instance, we plan to set up a *Constraint Satisfaction Problem* (*CSP*) connecting the sets of extensions to the generating AFs, and mapping belief-revision problems to such graphs. Adding facts that change the consistency of the knowledge base should lead the semantics of the AF to change in turn. In the same way, operations of belief revision should bring the AF to maintain unchanged the set of extensions for a given semantics. Another goal is to obtain a method to find, given a specific semantics (intended as a set of extensions), all the graphs representing an AF with that semantics. We will extract theorems and proofs on the possibility or not that semantics with certain properties exist partitioning the AFs set according to attacks type between arguments. For example, one can focus on the properties of the AFs in which the only attacks are all self attacks or bidirectional attacks,

to find relations between AFs containing the same extensions for a certain semantics. In this way, we could be able to solve some of the open problem related to abstract argumentation, like those proposed by Baumann and Strass in [25]. Some example problems could be: given an AF, can all implicit conflicts be made explicit (by adding one or two attacks between them)? Or even what is the maximal number of extensions for each semantics in an AF with n arguments?

Due to the intrinsic complexity of generating AFs and computing semantics, ConArg_rob can handle AFs with a maximum of 3 arguments. By setting 3 as the number of arguments, 3^2 different attacks can be presented, for a total of 2^9 different AFs. In the future, we intend to bring this threshold to 6, by only considering non-isomorphic semantics to reduce the generated nodes. Currently, we are working on a new feature of the tool that allows for representing the lattice of all extensions sets. In the future, we would like to carry out a more detailed study on subclasses of directed graphs (e.g., symmetric and simple). Depending on the number of arguments, we then would like to study the presence of cycles when this number is even or odd, along with other properties of the obtained graphs. We also would like to study inclusion between extensions of a single AF and between sets of AF in the graph. Finally, plan to extend the concept of robustness to coalitions of arguments, by studying how much a group of arguments derived from partitioning the original set is more robust than another.

On Ranking-Based Semantics

Following the work on ranking-based semantics (Chapter 4), we plan to implement other indexes in the tool, or combinations of them. As a starting point, we could use the work in [115], where various power indexes are grouped according to some criteria that qualify them for certain applications. In particular, we may consider the Public Good index, that is said to detect "special games". We aim to understand which ranking properties (or families of them, i.e., local or global) listed in [62] such indexes can successfully capture. With the comparison of different indexes, we aim to determine if there is a link between ties on rankings and the possible resolution of ambiguities.

So far, we have only captured properties that are local to an argument, i.e., they can be checked by inspecting the immediate neighbourhood of an argument. Global properties derive, instead, from the whole framework structure (e.g., full attacking or defending paths), and could be exploited for further refining the ranking returned by our semantics. We are also interested in extending our work on weighted AFs, where a different notion of defence is used.

Another direction we plan to investigate concerns the characteristic functions we use for evaluating the arguments. Similarly to what is done in [81] for studying coalitions with particular properties, we want to restrict the set of possible extensions by considering only the subsets of arguments that are in a given semantics. In this way, we can exclude all the arguments that are not even credulously accepted. For instance, we could devise a PI-based semantics where the arguments are evaluated with respect to the stable semantics, while the only coalitions to be taken into account are the admissible ones.

The work on claim-augmented AFs, described as well in Chapter 3, has to be seen as a first approach towards ranking semantics on the claim level, hence it can be extended in many directions. First, instead of relying on the lifting of the ranking from arguments to claims, we could also devise a ranking-based semantics directly on claims (e.g. by exploiting the logical structure of arguments).

Second, instead of considering qualitative functions, which only define a preference relation, we could study ranking-based semantics for CAFs when the ranking is induced from scores assigned to arguments (and claims); the quantitative evaluation of the arguments provides additional information that could lead to a more accurate ranking of the claims. Already existing ranking-based semantics (e.g. [34]) could be used for this purpose. This also requires a method for aggregating the values and assigning a score to coalitions of arguments (that is an extension or a set of supporters for a claim). In this context, the notion of robustness [51] is of interest.

Further avenues for future research include a complexity analysis, fuzzy approaches (see, e.g. [72]) and relations to the axioms from [8]. We would also like to analyse the complexity of decision problems for (CAFs) ranking-based semantics. Examples of these problems are: "is claim x the best one?" and "is argument a better than b?". Finally, we want to integrate the notion of CAF into ConArg in order to make the tool compatible with the use of semi-structured argumentation.

On the four-state semantics of WAFs

We plan to extend the work on weighted labellings of Chapter 5 in different directions. The definitions of the labelling-based semantics for WAFs do not include conditions for the undec since they are obtained from in and out arguments. In this sense, we would like to investigate the possible advantages of giving explicit conditions for labelling the undec arguments, similarly to what is done in [128] for classical AFs. An interesting study could then be carried out on the *don't care* and *don't know* labels, that are used in [17] as further differentiation of undec arguments. In our context, the difference between the two labels could be made more continuous by considering the weight on the attack relations.

We also plan to give a definition of w-strongly admissible extension (generalising the one provided in [14] for the crisp case) and introduce the respective labelling.

On the Concurrent Language for Argumentation

We plan to extend the work of Chapter 6 in many directions. First of all, given the known issues of abstract argumentation [142], we want to consider structured AFs and provide an implementation for our expansion, contraction and revision operators, for which different stores (structured and not abstract, indeed) need to be considered. The concurrent primitives are already general enough and do not require substantial changes. To obtain a spendable implementation, we will consider operations that can be done in polynomial time [89], for instance by using the grounded semantics, for which finding and checking extension is an easy task from the point of view of computational complexity.

To further improve the capabilities of our agents and make it more appealing for real-life applications, we want to extend ConArg_lang with the ability to handle processes involving time-critical aspects, similar to how CC is extended with temporal logic in [79, 80]. In this way, we could implement operations that also take into account time constraints.

On the operations level, we are currently only able to modify the acceptance status of the arguments, without further considerations on the obtained semantics. To gain control also over changes on the set of extensions, we want to introduce operators able to obtain a specified semantics (when possible) or to leave it unchanged (this can be done relying on the notion of robustness [52]). Another study we could conduct over such operators concerns their (non-)monotonicity. Since, in the current state of the work, operations like the removal of an argument can lead to an expansion into the considered AF, we would like to investigate the conditions under which, for instance, a contraction can be the only consequence of a removal. To this extent, also other operations on beliefs (like extraction, consolidation and merging) could be taken into account.

Concerning belief revision, we plan to investigate the link between postulates of the AGM theory [2] and our operators for expansion, contraction and revision. We also want to consider postulates from Katsuno and Mendelzon theory for belief update [107]. The link of our work with dialogue games in argumentation [125], then, can be exploited to provide tools for designing strategies that allow the agents to analyse complex scenarios and find the best solution for accomplishing their goals.

As a final consideration, whereas in real-life cases it is always clear which part involved in a debate is stating a particular argument, AFs do not hold any notion of "ownership" for arguments or attacks, that is, any bond with the one making the assertion is lost.

To overcome this problem, we want to implement the possibility of attaching labels on (groups of) arguments and attacks of AFs, in order to preserve the information related to whom added a certain argument or attack, extending and taking into account the work in [124]. Consequently, we can also obtain a notion of locality (or scope) of the belief in the knowledge base: arguments owned by a given agent can be placed into a local store and used in the implementation of specific operators through hidden variables.

Other Lines of Research

A different approach to argumentation problems consists of a matrix representation [167] that also enable for studying extensions and contracting AFs to reduce the number of arguments to consider for computing extensions and that can be used to handle dynamics of AFs. We already extended the above-mentioned results to weighted AFs [53] and, for the future, we plan to integrate the matrix representation into the ConArg suite and conduct a theoretical and empirical analysis on the advantages brought by such alternative approach.

Then, we would like to transpose our work on argumentation dynamics also to other kinds of AFs, as CAFs and bipolar AFs [67], which allow the specification of two different binary relations (one is the classic attack relation, while the other denotes support between arguments). In this way, we could access the capabilities of such extended AFs and integrate them into our tools.

On the application level, we want to explore the possibility of realising a chatbot using the technologies we developed in our research. Giving some insights, we could use ConArg_lang as a reasoning platform to implement the interaction between the user and the chatbot itself. Given a topic, the chatbot should be able to bring forward arguments to either support or reject the acceptability of certain arguments, we could model this behaviour through the join use of some kind of semantics (e.g., labelling-based semantics) and revision operators provided by our language.

Finally, to improve the usability and ease of maintenance of the tools belonging to the ConArg suite, we plan to distribute our functionalities through Docker[19] containers, namely packages containing software running on a virtualised operative system.

[19]Docker website: `https://www.docker.com`.

Bibliography

[1] Sameera A. Abdul-Kader and Jhon Woods. Survey on Chatbot Design Techniques in Speech Conversation Systems. *Int. J. Adv. Comput. Sci. Appl.*, 6(7), 2015. DOI 10.14569/IJACSA.2015.060712.

[2] Carlos E. Alchourrón, Peter Gärdenfors, and David Makinson. On the logic of theory change: Partial meet contraction and revision functions. *J. Symb. Log.*, 50 (02):510–530, June 1985. ISSN 0022-4812, 1943-5886. DOI 10.2307/2274239.

[3] Gianvincenzo Alfano, Sergio Greco, and Francesco Parisi. An Efficient Algorithm for Skeptical Preferred Acceptance in Dynamic Argumentation Frameworks. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 18–24. ijcai.org, 2019. DOI 10.24963/ijcai.2019/3.

[4] Gianvincenzo Alfano, Andrea Cohen, Sebastian Gottifredi, Sergio Greco, Francesco Parisi, and Guillermo Ricardo Simari. Dynamics in Abstract Argumentation Frameworks with Recursive Attack and Support Relations. In Giuseppe De Giacomo, Alejandro Catalá, Bistra Dilkina, Michela Milano, Senén Barro, Alberto Bugarín, and Jérôme Lang, editors, *ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020)*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pages 577–584. IOS Press, 2020. DOI 10.3233/FAIA200141.

[5] Gianvincenzo Alfano, Sergio Greco, and Francesco Parisi. Computing Skeptical Preferred Acceptance in Dynamic Argumentation Frameworks with Recursive Attack and Support Relations. In Henry Prakken, Stefano Bistarelli, Francesco Santini, and Carlo Taticchi, editors, *Computational Models of Argument - Proceedings of COMMA 2020, Perugia, Italy, September 4-11, 2020*, volume 326 of *Frontiers in Artificial Intelligence and Applications*, pages 67–78. IOS Press, 2020. DOI 10.3233/FAIA200493.

[6] José María Alonso-Meijide, Mikel Álvarez-Mozos, and María Gloria Fiestras-Janeiro. Power Indices and Minimal Winning Coalitions in Simple Games with Externalities. UB Economics Working Papers 2015/328, Universitat de Barcelona, Facultat d'Economia i Empresa, UB Economics, 2015. URL `https://ideas.repec.org/p/ewp/wpaper/328web.html`.

[7] Leila Amgoud and Jonathan Ben-Naim. Ranking-Based Semantics for Argumentation Frameworks. In Weiru Liu, V. S. Subrahmanian, and Jef Wijsen, editors, *Scalable Uncertainty Management - 7th International Conference, SUM 2013*, volume 8078 of *LNCS*, pages 134–147. Springer, 2013. ISBN 978-3-642-40380-4.

[8] Leila Amgoud and Jonathan Ben-Naim. Argumentation-based Ranking Logics. In Gerhard Weiss, Pinar Yolum, Rafael H. Bordini, and Edith Elkind, editors, *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, Istanbul, Turkey, May 4-8, 2015*, pages 1511–1519. ACM, 2015. URL `http://dl.acm.org/citation.cfm?id=2773344`.

[9] Leila Amgoud and Claudette Cayrol. On the Acceptability of Arguments in Preference-based Argumentation. In Gregory F. Cooper and Serafín Moral, editors, *UAI '98: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, University of Wisconsin Business School, Madison, Wisconsin, USA, July 24-26, 1998*, pages 1–7. Morgan Kaufmann, 1998. ISBN 978-1-55860-555-8. URL `https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=226&proceeding_id=14`.

[10] Leila Amgoud, Jonathan Ben-Naim, and Srdjan Vesic. Measuring the Intensity of Attacks in Argumentation Graphs with Shapley Value. In Carles Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 63–69. ijcai.org, 2017. DOI 10.24963/ijcai.2017/10.

[11] Malte Aschermann, Sophie Dennisen, Philipp Kraus, and Jörg P. Müller. LightJason, a Highly Scalable and Concurrent Agent Framework: Overview and Application. In Elisabeth André, Sven Koenig, Mehdi Dastani, and Gita Sukthankar, editors, *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15, 2018*, pages 1794–1796. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM, 2018. URL `http://dl.acm.org/citation.cfm?id=3237980`.

[12] Philippe Balbiani, Andreas Herzig, and Nicolas Troquard. Dynamic Logic of Propositional Assignments: A Well-Behaved Variant of PDL. In *28th Annual ACM/IEEE*

Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013, pages 143–152. IEEE Computer Society, 2013. ISBN 978-1-4799-0413-6. DOI 10.1109/LICS.2013.20.

[13] John F. Banzhaf. Weighted Voting Doesn't Work: A Mathematical Analysis. *Rutgers Law Review*, 19(2):317–343, 1965.

[14] 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.

[15] Pietro Baroni, Martin Caminada, and Massimiliano Giacomin. An introduction to argumentation semantics. *Knowl. Eng Rev.*, 26(4):365–410, 2011. DOI 10.1017/S0269888911000166.

[16] 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.

[17] Pietro Baroni, Massimiliano Giacomin, and Bei Shui Liao. I don't care, I don't know ... I know too much! On Incompleteness and Undecidedness in Abstract Argumentation. In Thomas Eiter, Hannes Strass, Miroslaw Truszczynski, and Stefan Woltran, editors, *Advances in Knowledge Representation, Logic Programming, and Abstract Argumentation - Essays Dedicated to Gerhard Brewka on the Occasion of His 60th Birthday*, volume 9060 of *Lecture Notes in Computer Science*, pages 265–280. Springer, 2015. ISBN 978-3-319-14725-3. DOI 10.1007/978-3-319-14726-0_18.

[18] Pietro Baroni, Marco Romano, Francesca Toni, Marco Aurisicchio, and Giorgio Bertanza. Automatic evaluation of design alternatives with quantitative argumentation. *Argum. Comput*, 6(1):24–49, 2015. DOI 10.1080/19462166.2014.1001791.

[19] Pietro Baroni, Massimiliano Giacomin, and Beishui Liao. A general semi-structured formalism for computational argumentation: Definition, properties, and examples of application. *Artificial Intelligence*, 257:158–207, April 2018. ISSN 00043702. DOI 10.1016/j.artint.2018.01.003.

[20] Ringo Baumann. Normal and strong expansion equivalence for argumentation frameworks. *Artif Intell*, 193:18–44, 2012. DOI 10.1016/j.artint.2012.08.004.

[21] Ringo Baumann. What Does it Take to Enforce an Argument? Minimal Change in abstract Argumentation. *Front. Artif. Intell. Appl.*, pages 127–132, 2012. ISSN 0922-6389. DOI 10.3233/978-1-61499-098-7-127.

[22] Ringo Baumann. Context-free and Context-sensitive Kernels: Update and Deletion Equivalence in abstract Argumentation. In Torsten Schaub, Gerhard Friedrich, and Barry O'Sullivan, editors, *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pages 63–68. IOS Press, 2014. DOI 10.3233/978-1-61499-419-0-63.

[23] Ringo Baumann and Gerhard Brewka. Expanding Argumentation Frameworks: Enforcing and Monotonicity Results. In Pietro Baroni, Federico Cerutti, Massimiliano Giacomin, and Guillermo Ricardo Simari, editors, *Computational Models of Argument: Proceedings of COMMA 2010, Desenzano Del Garda, Italy, September 8-10, 2010*, volume 216 of *Frontiers in Artificial Intelligence and Applications*, pages 75–86. IOS Press, 2010. DOI 10.3233/978-1-60750-619-5-75.

[24] Ringo Baumann and Gerhard Brewka. The equivalence zoo for Dung-style semantics. *J Log Comput*, 28(3):477–498, 2018. DOI 10.1093/logcom/exv001.

[25] Ringo Baumann and Hannes Strass. Open Problems in Abstract Argumentation. In Thomas Eiter, Hannes Strass, Miroslaw Truszczynski, and Stefan Woltran, editors, *Advances in Knowledge Representation, Logic Programming, and Abstract Argumentation - Essays Dedicated to Gerhard Brewka on the Occasion of His 60th Birthday*, volume 9060 of *Lecture Notes in Computer Science*, pages 325–339. Springer, 2015. DOI 10.1007/978-3-319-14726-0_22.

[26] Dorothea Baumeister, Daniel Neugebauer, Jörg Rothe, and Hilmar Schadrack. Verification in incomplete argumentation frameworks. *Artif Intell*, 264:1–26, 2018. DOI 10.1016/j.artint.2018.08.001.

[27] Trevor J. M. Bench-Capon. Persuasion in Practical Argument Using Value-based Argumentation Frameworks. *J Log Comput*, 13(3):429–448, 2003. DOI 10.1093/logcom/13.3.429.

[28] Philippe Besnard and Anthony Hunter. A logic-based theory of deductive arguments. *Artif Intell*, 128(1-2):203–235, 2001. DOI 10.1016/S0004-3702(01)00071-6.

[29] Stefano Bistarelli and Fabio Gadducci. Enhancing constraints manipulation in semiring-based formalisms. In Gerhard Brewka, Silvia Coradeschi, Anna Perini, and Paolo Traverso, editors, *ECAI 2006, 17th European Conference on Artificial Intelligence, August 29 - September 1, 2006, Riva Del Garda, Italy, Including Prestigious Applications of Intelligent Systems (PAIS 2006), Proceedings*, volume 141 of *Frontiers in Artificial Intelligence and Applications*, pages 63–67.

IOS Press, 2006. URL http://www.booksonline.iospress.nl/Content/View.aspx?piid=1647.

[30] Stefano Bistarelli and Theofrastos Mantadelis. A Possible World View and a Normal Form for the Constellation Semantics. In Francesco Calimeri, Nicola Leone, and Marco Manna, editors, *Logics in Artificial Intelligence - 16th European Conference, JELIA 2019, Rende, Italy, May 7-11, 2019, Proceedings*, volume 11468 of *Lecture Notes in Computer Science*, pages 58–68. Springer, 2019. DOI 10.1007/978-3-030-19570-0_4.

[31] Stefano Bistarelli and Francesco Santini. ConArg: A Constraint-Based Computational Framework for Argumentation Systems. In *IEEE 23rd International Conference on Tools with Artificial Intelligence, ICTAI*, pages 605–612. IEEE Computer Society, 2011. ISBN 978-1-4577-2068-0. DOI 10.1109/ICTAI.2011.96.

[32] Stefano Bistarelli and Francesco Santini. Modeling and Solving AFs with a Constraint-Based Tool: ConArg. In Sanjay Modgil, Nir Oren, and Francesca Toni, editors, *Theorie and Applications of Formal Argumentation - First International Workshop, TAFA 2011. Barcelona, Spain, July 16-17, 2011, Revised Selected Papers*, volume 7132 of *Lecture Notes in Computer Science*, pages 99–116. Springer, 2011. DOI 10.1007/978-3-642-29184-5_7.

[33] Stefano Bistarelli and Francesco Santini. A Hasse Diagram for Weighted Sceptical Semantics with a Unique-Status Grounded Semantics. In *Proceedings of the 14th International Conference on Logic Programming and Nonmonotonic Reasoning*. Lecture Notes in Computer Science, July 2017. DOI 10.1007/978-3-319-61660-5_6.

[34] Stefano Bistarelli and Carlo Taticchi. Power index-based semantics for ranking arguments in abstract argumentation frameworks. *Intell. Artif.*, 13(2):137–154, 2019. DOI 10.3233/IA-190031.

[35] Stefano Bistarelli and Carlo Taticchi. Preliminary Study on Reinstatement Labelling for Weighted Argumentation Frameworks. In Francesco Santini and Alice Toniolo, editors, *Proceedings of the 3rd Workshop on Advances In Argumentation In Artificial Intelligence Co-Located with the 18th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2019), Rende, Italy, November 19-22, 2019*, volume 2528 of *CEUR Workshop Proceedings*, pages 45–49. CEUR-WS.org, 2019. URL http://ceur-ws.org/Vol-2528/4_Bistarelli_et_al_AI3_2019.pdf.

[36] Stefano Bistarelli and Carlo Taticchi. A Concurrent Language for Argumentation: Preliminary Notes. In *Workshop on Recent Developments on the Design and Implementation of Programming Languages (DIP2020), to Appear*, OASICS. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[37] Stefano Bistarelli and Carlo Taticchi. A Concurrent Language for Argumentation. In Bettina Fazzinga, Filippo Furfaro, and Francesco Parisi, editors, *Proceedings of the Workshop on Advances In Argumentation In Artificial Intelligence 2020 Co-Located with the 19th International Conference of the Italian Association for Artificial Intelligence (AIxIA 2020), Online, November 25-26, 2020*, volume 2777 of *CEUR Workshop Proceedings*, pages 75–89. CEUR-WS.org, 2020. URL `http://ceur-ws.org/Vol-2777/paper67.pdf`.

[38] Stefano Bistarelli and Carlo Taticchi. A Labelling Semantics for Weighted Argumentation Frameworks. In Francesco Calimeri, Simona Perri, and Ester Zumpano, editors, *Proceedings of the 35th Italian Conference on Computational Logic - CILC 2020, Rende, Italy, October 13-15, 2020*, volume 2710 of *CEUR Workshop Proceedings*, pages 263–277. CEUR-WS.org, 2020. URL `http://ceur-ws.org/Vol-2710/paper17.pdf`.

[39] Stefano Bistarelli, Ugo Montanari, and Francesca Rossi. Semiring-based constraint satisfaction and optimization. *J. ACM*, 44(2):201–236, March 1997. ISSN 0004-5411, 1557-735X. DOI 10.1145/256303.256306.

[40] Stefano Bistarelli, Daniele Pirolandi, and Francesco Santini. Solving Weighted Argumentation Frameworks with Soft Constraints. In Wolfgang Faber and Nicola Leone, editors, *Proceedings of the 25th Italian Conference on Computational Logic, Rende, Italy, July 7-9, 2010*, volume 598 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2010. URL `http://ceur-ws.org/Vol-598/paper18.pdf`.

[41] Stefano Bistarelli, Fabio Rossi, Francesco Santini, and Carlo Taticchi. Towards visualising security with arguments. In Davide Ancona, Marco Maratea, and Viviana Mascardi, editors, *Proceedings of the 30th Italian Conference on Computational Logic, Genova, Italy, July 1-3, 2015*, volume 1459 of *CEUR Workshop Proceedings*, pages 197–201. CEUR-WS.org, 2015. URL `http://ceur-ws.org/Vol-1459/paper28.pdf`.

[42] Stefano Bistarelli, Fabio Rossi, and Francesco Santini. A Collective Defence Against Grouped Attacks for Weighted Abstract Argumentation Frameworks. In Zdravko Markov and Ingrid Russell, editors, *Proceedings of the Twenty-Ninth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2016*, pages

638–643. AAAI Press, 2016. ISBN 978-1-57735-756-8. URL https://www.aaai.org/ocs/index.php/FLAIRS/FLAIRS16/paper/view/12778/12645.

[43] Stefano Bistarelli, Fabio Rossi, and Francesco Santini. ConArg: A Tool for Classical and Weighted Argumentation. In Pietro Baroni, Thomas F. Gordon, Tatjana Scheffler, and Manfred Stede, editors, *Computational Models of Argument - Proceedings of COMMA*, volume 287 of *Frontiers in Artificial Intelligence and Applications*, pages 463–464. IOS Press, 2016. ISBN 978-1-61499-685-9. DOI 10.3233/978-1-61499-686-6-463.

[44] Stefano Bistarelli, Fabio Rossi, and Francesco Santini. A ConArg-Based Library for Abstract Argumentation. In *29th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2017, Boston, MA, USA, November 6-8, 2017*, pages 374–381. IEEE Computer Society, 2017. DOI 10.1109/ICTAI.2017.00065.

[45] Stefano Bistarelli, Francesco Faloci, Francesco Santini, and Carlo Taticchi. Studying Dynamics in Argumentation with Rob. In Sanjay Modgil, Katarzyna Budzynska, and John Lawrence, editors, *Computational Models of Argument - Proceedings of COMMA 2018, Warsaw, Poland, 12-14 September 2018*, volume 305 of *Frontiers in Artificial Intelligence and Applications*, pages 451–452. IOS Press, 2018. DOI 10.3233/978-1-61499-906-5-451.

[46] Stefano Bistarelli, Paolo Giuliodori, Francesco Santini, and Carlo Taticchi. A Cooperative-game Approach to Share Acceptability and Rank Arguments. In Pierpaolo Dondio and Luca Longo, editors, *Proceedings of the 2nd Workshop on Advances In Argumentation In Artificial Intelligence, Co-Located with XVII International Conference of the Italian Association for Artificial Intelligence, AI(^{3})@AI*IA 2018, 20-23 November 2018, Trento, Italy*, volume 2296 of *CEUR Workshop Proceedings*, pages 86–90. CEUR-WS.org, 2018. URL http://ceur-ws.org/Vol-2296/AI3-2018_paper_8.pdf.

[47] Stefano Bistarelli, Lars Kotthoff, Francesco Santini, and Carlo Taticchi. Containerisation and Dynamic Frameworks in ICCMA'19. In Matthias Thimm, Federico Cerutti, and Mauro Vallati, editors, *Proceedings of the Second International Workshop on Systems and Algorithms for Formal Argumentation (SAFA 2018) Co-Located with the 7th International Conference on Computational Models of Argument (COMMA 2018), Warsaw, Poland, September 11, 2018*, volume 2171 of *CEUR Workshop Proceedings*, pages 4–9. CEUR-WS.org, 2018. URL http://ceur-ws.org/Vol-2171/paper_1.pdf.

[48] Stefano Bistarelli, Theofrastos Mantadelis, Francesco Santini, and Carlo Taticchi. Probabilistic Argumentation Frameworks with MetaProbLog and ConArg.

In Lefteri H. Tsoukalas, Éric Grégoire, and Miltiadis Alamaniotis, editors, *IEEE 30th International Conference on Tools with Artificial Intelligence, ICTAI 2018, 5-7 November 2018, Volos, Greece*, pages 675–679. IEEE, 2018. DOI 10.1109/ICTAI.2018.00107.

[49] Stefano Bistarelli, Theofrastos Mantadelis, Francesco Santini, and Carlo Taticchi. Using MetaProbLog and ConArg to compute Probabilistic Argumentation Frameworks. In Pierpaolo Dondio and Luca Longo, editors, *Proceedings of the 2nd Workshop on Advances In Argumentation In Artificial Intelligence, Co-Located with XVII International Conference of the Italian Association for Artificial Intelligence, AI(^{3})@AI*IA 2018, 20-23 November 2018, Trento, Italy*, volume 2296 of *CEUR Workshop Proceedings*, pages 6–10. CEUR-WS.org, 2018. URL `http://ceur-ws.org/Vol-2296/AI3-2018_paper_2.pdf`.

[50] Stefano Bistarelli, Fabio Rossi, and Francesco Santini. A novel weighted defence and its relaxation in abstract argumentation. *Int J Approx Reason.*, 92:66–86, 2018. DOI 10.1016/j.ijar.2017.10.006.

[51] Stefano Bistarelli, Francesco Santini, and Carlo Taticchi. Local Expansion Invariant Operators in Argumentation Semantics. In Beishui Liao, Thomas Ågotnes, and Yì N. Wáng, editors, *Dynamics, Uncertainty and Reasoning, The Second Chinese Conference on Logic and Argumentation, CLAR 2018, Hangzhou, China, 16-17 June 2018*, pages 45–62. Springer, 2018. DOI 10.1007/978-981-13-7791-4_3.

[52] Stefano Bistarelli, Francesco Santini, and Carlo Taticchi. On Looking for Invariant Operators in Argumentation Semantics. In Keith Brawner and Vasile Rus, editors, *Proceedings of the Thirty-First International Florida Artificial Intelligence Research Society Conference, FLAIRS 2018, Melbourne, Florida, USA. May 21-23 2018*, pages 537–540. AAAI Press, 2018. URL `https://aaai.org/ocs/index.php/FLAIRS/FLAIRS18/paper/view/17671`.

[53] Stefano Bistarelli, Alessandra Tappini, and Carlo Taticchi. A Matrix Approach for Weighted Argumentation Frameworks. In Keith Brawner and Vasile Rus, editors, *Proceedings of the Thirty-First International Florida Artificial Intelligence Research Society Conference, FLAIRS 2018, Melbourne, Florida, USA. May 21-23 2018*, pages 507–512. AAAI Press, 2018. URL `https://aaai.org/ocs/index.php/FLAIRS/FLAIRS18/paper/view/17658`.

[54] Stefano Bistarelli, Francesco Faloci, Francesco Santini, and Carlo Taticchi. A Tool For Ranking Arguments Through Voting-Games Power Indexes. In Alberto Casagrande and Eugenio G. Omodeo, editors, *Proceedings of the 34th Italian Conference on Computational Logic, Trieste, Italy, June 19-21, 2019*, volume

2396 of *CEUR Workshop Proceedings*, pages 193–201. CEUR-WS.org, 2019. URL http://ceur-ws.org/Vol-2396/paper29.pdf.

[55] Stefano Bistarelli, Francesco Faloci, and Carlo Taticchi. Implementing Ranking-Based Semantics in ConArg. In *31st IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2019, Portland, OR, USA, November 4-6, 2019*, pages 1180–1187. IEEE, 2019. DOI 10.1109/ICTAI.2019.00163.

[56] Stefano Bistarelli, Wolfgang Dvorák, Carlo Taticchi, and Stefan Woltran. Ranking-Based Semantics from the Perspective of Claims. In Henry Prakken, Stefano Bistarelli, Francesco Santini, and Carlo Taticchi, editors, *Computational Models of Argument - Proceedings of COMMA 2020, Perugia, Italy, September 4-11, 2020*, volume 326 of *Frontiers in Artificial Intelligence and Applications*, pages 111–122. IOS Press, 2020. DOI 10.3233/FAIA200497.

[57] Stefano Bistarelli, Lars Kotthoff, Francesco Santini, and Carlo Taticchi. A First Overview of ICCMA'19. In Bettina Fazzinga, Filippo Furfaro, and Francesco Parisi, editors, *Proceedings of the Workshop on Advances In Argumentation In Artificial Intelligence 2020 Co-Located with the 19th International Conference of the Italian Association for Artificial Intelligence (AIxIA 2020), Online, November 25-26, 2020*, volume 2777 of *CEUR Workshop Proceedings*, pages 90–102. CEUR-WS.org, 2020. URL http://ceur-ws.org/Vol-2777/paper76.pdf.

[58] Guido Boella, Célia da Costa Pereira, Andrea Tettamanzi, and Leendert W. N. van der Torre. Making Others Believe What They Want. In Max Bramer, editor, *Artificial Intelligence in Theory and Practice II, IFIP 20th World Computer Congress, TC 12: IFIP AI 2008 Stream, September 7-10, 2008, Milano, Italy*, volume 276 of *IFIP*, pages 215–224. Springer, 2008. DOI 10.1007/978-0-387-09695-7_21.

[59] Guido Boella, Souhila Kaci, and Leendert W. N. van der Torre. Dynamics in Argumentation with Single Extensions: Attack Refinement and the Grounded Extension (Extended Version). In Peter McBurney, Iyad Rahwan, Simon Parsons, and Nicolas Maudet, editors, *Argumentation in Multi-Agent Systems, 6th International Workshop, ArgMAS 2009. Revised Selected and Invited Papers*, volume 6057 of *Lecture Notes in Computer Science*, pages 150–159. Springer, 2009. ISBN 978-3-642-12804-2. DOI 10.1007/978-3-642-12805-9_9.

[60] Guido Boella, Souhila Kaci, and Leendert W. N. van der Torre. Dynamics in Argumentation with Single Extensions: Abstraction Principles and the Grounded

Extension. In Claudio Sossai and Gaetano Chemello, editors, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty, 10th European Conference, ECSQARU 2009, Verona, Italy, July 1-3, 2009. Proceedings*, volume 5590 of *Lecture Notes in Computer Science*, pages 107–118. Springer, 2009. DOI 10.1007/978-3-642-02906-6_11.

[61] Andrei Bondarenko, Francesca Toni, and Robert A. Kowalski. An Assumption-Based Framework for Non-Monotonic Reasoning. In Luís Moniz Pereira and Anil Nerode, editors, *Logic Programming and Non-Monotonic Reasoning, Proceedings of the Second International Workshop, Lisbon, Portugal, June 1993*, pages 171–189. MIT Press, 1993.

[62] Elise Bonzon, Jérôme Delobelle, Sébastien Konieczny, and Nicolas Maudet. A Comparative Study of Ranking-Based Semantics for Abstract Argumentation. In Dale Schuurmans and Michael P. Wellman, editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 914–920. AAAI Press, 2016. ISBN 978-1-57735-760-5. URL `https://arxiv.org/pdf/1602.01059.pdf`.

[63] Ferdinand Börner. Basics of Galois Connections. In Nadia Creignou, Phokion G. Kolaitis, and Heribert Vollmer, editors, *Complexity of Constraints - An Overview of Current Research Themes [Result of a Dagstuhl Seminar]*, volume 5250 of *Lecture Notes in Computer Science*, pages 38–67. Springer, 2008. DOI 10.1007/978-3-540-92800-3_3.

[64] Petter Bae Brandtzæg and Asbjørn Følstad. Chatbots: Changing user needs and motivations. *Interactions*, 25(5):38–43, 2018. DOI 10.1145/3236669.

[65] Martin Caminada. On the Issue of Reinstatement in Argumentation. In Michael Fisher, Wiebe van der Hoek, Boris Konev, and Alexei Lisitsa, editors, *Logics in Artificial Intelligence, 10th European Conference, JELIA*, volume 4160 of *LNCS*, pages 111–123. Springer, 2006. ISBN 978-3-540-39625-3. DOI 10.1007/11853886_11.

[66] Martin Caminada. Strong Admissibility Revisited. In Simon Parsons, Nir Oren, Chris Reed, and Federico Cerutti, editors, *Computational Models of Argument - Proceedings of COMMA 2014, Atholl Palace Hotel, Scottish Highlands, UK, September 9-12, 2014*, volume 266 of *Frontiers in Artificial Intelligence and Applications*, pages 197–208. IOS Press, 2014. ISBN 978-1-61499-435-0. DOI 10.3233/978-1-61499-436-7-197.

[67] Claudette Cayrol and Marie-Christine Lagasquie-Schiex. On the Acceptability of Arguments in Bipolar Argumentation Frameworks. In Lluis Godo, editor, *Symbolic*

and Quantitative Approaches to Reasoning with Uncertainty, 8th European Conference, ECSQARU 2005, Barcelona, Spain, July 6-8, 2005, Proceedings, volume 3571 of *Lecture Notes in Computer Science*, pages 378–389. Springer, 2005. ISBN 978-3-540-27326-4. DOI 10.1007/11518655_33.

[68] Claudette Cayrol and Marie-Christine Lagasquie-Schiex. Graduality in Argumentation. *J Artif Intell Res*, 23:245–297, 2005. DOI 10.1613/jair.1411.

[69] Claudette Cayrol, Florence Dupin de Saint-Cyr, and Marie-Christine Lagasquie-Schiex. Revision of an Argumentation System. In Gerhard Brewka and Jérôme Lang, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Eleventh International Conference, KR 2008, Sydney, Australia, September 16-19, 2008*, pages 124–134. AAAI Press, 2008. ISBN 978-1-57735-384-3. URL http://www.aaai.org/Papers/KR/2008/KR08-013.pdf.

[70] Claudette Cayrol, Florence Dupin de Saint-Cyr, and Marie-Christine Lagasquie-Schiex. Change in Abstract Argumentation Frameworks: Adding an Argument. *J Artif Intell Res*, 38:49–84, 2010. DOI 10.1613/jair.2965.

[71] Lisa A. Chalaguine, Fiona L. Hamilton, Anthony Hunter, and Henry W. W. Potts. Argument Harvesting Using Chatbots. In Sanjay Modgil, Katarzyna Budzynska, and John Lawrence, editors, *Computational Models of Argument - Proceedings of COMMA 2018, Warsaw, Poland, 12-14 September 2018*, volume 305 of *Frontiers in Artificial Intelligence and Applications*, pages 149–160. IOS Press, 2018. ISBN 978-1-61499-905-8. DOI 10.3233/978-1-61499-906-5-149.

[72] Esther Anna Corsi and Christian G. Fermüller. Connecting fuzzy logic and argumentation frames via logical attack principles. *Soft Comput*, 23(7):2255–2270, 2019. DOI 10.1007/s00500-018-3513-2.

[73] Sylvie Coste-Marquis, Caroline Devred, and Pierre Marquis. Symmetric Argumentation Frameworks. In Lluís Godo, editor, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty, 8th European Conference, ECSQARU 2005, Barcelona, Spain, July 6-8, 2005, Proceedings*, volume 3571 of *Lecture Notes in Computer Science*, pages 317–328. Springer, 2005. DOI 10.1007/11518655_28.

[74] Sylvie Coste-Marquis, Sébastien Konieczny, Pierre Marquis, and Mohand Akli Ouali. Weighted Attacks in Argumentation Frameworks. In Gerhard Brewka, Thomas Eiter, and Sheila A. McIlraith, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR 2012, Rome, Italy, June 10-14, 2012*. AAAI Press, 2012. ISBN 978-1-57735-560-1. URL https://www.aaai.org/ocs/index.php/KR/KR12/paper/view/4543/4931.

[75] Sylvie Coste-Marquis, Sébastien Konieczny, Jean-Guy Mailly, and Pierre Marquis. Extension Enforcement in Abstract Argumentation as an Optimization Problem. In Qiang Yang and Michael J. Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 2876–2882. AAAI Press, 2015. URL `http://ijcai.org/Abstract/15/407`.

[76] Patrick Cousot and Radhia Cousot. A Galois connection calculus for abstract interpretation. In Suresh Jagannathan and Peter Sewell, editors, *The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14, San Diego, CA, USA, January 20-21, 2014*, pages 3–4. ACM, 2014. DOI 10.1145/2535838.2537850.

[77] Cosmina Croitoru and Timo Kötzing. A Normal Form for Argumentation Frameworks. In Elizabeth Black, Sanjay Modgil, and Nir Oren, editors, *Theory and Applications of Formal Argumentation - Second International Workshop, TAFA 2013, Beijing, China, August 3-5, 2013, Revised Selected Papers*, volume 8306 of *Lecture Notes in Computer Science*, pages 32–45. Springer, 2013. DOI 10.1007/978-3-642-54373-9_3.

[78] Stephen G. Daugherty. *Multi-Agent Routing in Shared Guidepath Networks*. PhD Thesis, Georgia Institute of Technology, Atlanta, GA, USA, 2018. URL `http://hdl.handle.net/1853/59196`.

[79] Frank S. de Boer, Maurizio Gabbrielli, and Maria Chiara Meo. Semantics and Expressive Power of a Timed Concurrent Constraint Language. In Gert Smolka, editor, *Principles and Practice of Constraint Programming - CP97, Third International Conference, Linz, Austria, October 29 - November 1, 1997, Proceedings*, volume 1330 of *Lecture Notes in Computer Science*, pages 47–61. Springer, 1997. DOI 10.1007/BFb0017429.

[80] Frank S. de Boer, Maurizio Gabbrielli, and Maria Chiara Meo. A Timed Concurrent Constraint Language. *Inf Comput*, 161(1):45–83, 2000. DOI 10.1006/inco.1999.2879.

[81] Jean Derks and Hans Peters. A Shapley Value for Games with Restricted Coalitions. *Int. J. Game Theory*, 21(4):351–60, December 1993. URL `https://link.springer.com/article/10.1007/BF01240150`.

[82] Jürgen Dix, Sven Ove Hansson, Gabriele Kern-Isberner, and Guillermo Ricardo Simari. Belief change and argumentation in multi-agent scenarios. *Ann Math Artif Intell*, 78(3-4):177–179, 2016. DOI 10.1007/s10472-016-9530-x.

[83] Pierpaolo Dondio. Ranking Semantics Based on Subgraphs Analysis. In Elisabeth André, Sven Koenig, Mehdi Dastani, and Gita Sukthankar, editors, *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS*, pages 1132–1140. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM, 2018. URL `http://dl.acm.org/citation.cfm?id=3237864`.

[84] Sylvie Doutre, Andreas Herzig, and Laurent Perrussel. A Dynamic Logic Framework for Abstract Argumentation. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR 2014, Vienna, Austria, July 20-24, 2014*, 2014. URL `http://www.aaai.org/ocs/index.php/KR/KR14/paper/view/7993`.

[85] 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.

[86] 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.

[87] Florence Dupin de Saint-Cyr, Pierre Bisquert, Claudette Cayrol, and Marie-Christine Lagasquie-Schiex. Argumentation update in YALLA (Yet Another Logic Language for Argumentation). *Int. J. Approx. Reason.*, 75:57–92, August 2016. ISSN 0888613X. DOI 10.1016/j.ijar.2016.04.003.

[88] Encarnación A. Durán, Jesús M. Bilbao, Julio R. F. García, and J. J. López. Computing power indices in weighted multiple majority games. *Math. Soc. Sci.*, 46(1):63–80, 2003. DOI 10.1016/S0165-4896(02)00086-0.

[89] Wolfgang Dvorák and Paul E. Dunne. Computational Problems in Formal Argumentation and their Complexity. *FLAP*, 4(8), 2017. URL `http://www.collegepublications.co.uk/downloads/ifcolog00017.pdf`.

[90] Wolfgang Dvorák and Stefan Woltran. Complexity of Abstract Argumentation under a Claim-Centric View. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 2801–2808. AAAI Press, 2019. DOI 10.1609/aaai.v33i01.33012801.

[91] Wolfgang Dvorák, Anna Rapberger, and Stefan Woltran. On the relation between claim-augmented argumentation frameworks and collective attacks. In *ECAI2020 (to Appear)*, 2020.

[92] Sefi Erlich, Noam Hazon, and Sarit Kraus. Negotiation Strategies for Agents with Ordinal Preferences. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 210–218. ijcai.org, 2018. ISBN 978-0-9992411-2-7. DOI 10.24963/ijcai.2018/29.

[93] Marcelo A. Falappa, Alejandro Javier García, and Guillermo Ricardo Simari. Belief dynamics and defeasible argumentation in rational agents. In James P. Delgrande and Torsten Schaub, editors, *10th International Workshop on Non-Monotonic Reasoning (NMR 2004), Whistler, Canada, June 6-8, 2004, Proceedings*, pages 164–170, 2004. ISBN 978-92-990021-0-0. URL `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.549.4021&rep=rep1&type=pdf`.

[94] Bettina Fazzinga, Sergio Flesca, and Francesco Parisi. On the Complexity of Probabilistic Abstract Argumentation. In Francesca Rossi, editor, *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, pages 898–904. IJCAI/AAAI, 2013. URL `http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6563`.

[95] Timothy W. Finin, Richard Fritzson, Donald P. McKay, and Robin McEntire. KQML As An Agent Communication Language. In *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM'94), Gaithersburg, Maryland, USA, November 29 - December 2, 1994*, pages 456–463. ACM, 1994. DOI 10.1145/191246.191322.

[96] Steven Fortune and James Wyllie. Parallelism in Random Access Machines. In Richard J. Lipton, Walter A. Burkhard, Walter J. Savitch, Emily P. Friedman, and Alfred V. Aho, editors, *Proceedings of the 10th Annual ACM Symposium on Theory of Computing, May 1-3, 1978, San Diego, California, USA*, pages 114–118. ACM, 1978. DOI 10.1145/800133.804339.

[97] Alejandro Javier García and Guillermo Ricardo Simari. Defeasible Logic Programming: An Argumentative Approach. *Theory Pr. Log Program*, 4(1-2):95–138, 2004. DOI 10.1017/S1471068403001674.

[98] Jacob Glazer and Ariel Rubinstein. Debates and Decisions: On a Rationale of Argumentation Rules. *Games Econ. Behav.*, 36(2):158–173, 2001. DOI 10.1006/game.2000.0824.

[99] Davide Grossi and Sanjay Modgil. On the Graded Acceptability of Arguments. In Qiang Yang and Michael J. Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI*, pages 868–874. AAAI Press, 2015. URL `http://ijcai.org/Abstract/15/127`.

[100] Alejandro Guerra-Hernández, Amal El Fallah-Seghrouchni, and Henry Soldano. Learning in BDI Multi-agent Systems. In Jürgen Dix and João Alexandre Leite, editors, *Computational Logic in Multi-Agent Systems, 4th International Workshop, CLIMA IV, Fort Lauderdale, FL, USA, January 6-7, 2004, Revised Selected and Invited Papers*, volume 3259 of *Lecture Notes in Computer Science*, pages 218–233. Springer, 2004. ISBN 978-3-540-24010-5. DOI 10.1007/978-3-540-30200-1_12.

[101] C. A. R. Hoare. Communicating Sequential Processes. *Commun ACM*, 21(8): 666–677, 1978. DOI 10.1145/359576.359585.

[102] Nguyen Duy Hung. Probabilistic Argumentation for Decision Making: A Toolbox and Applications, 2016.

[103] Anthony Hunter. Some Foundations for Probabilistic Abstract Argumentation. In Bart Verheij, Stefan Szeider, and Stefan Woltran, editors, *Computational Models of Argument - Proceedings of COMMA 2012, Vienna, Austria, September 10-12, 2012*, volume 245 of *Frontiers in Artificial Intelligence and Applications*, pages 117–128. IOS Press, 2012. DOI 10.3233/978-1-61499-111-3-117.

[104] Anthony Hunter and Sébastien Konieczny. On the measure of conflicts: Shapley Inconsistency Values. *Artif Intell*, 174(14):1007–1026, 2010. DOI 10.1016/j.artint.2010.06.001.

[105] Hadassa Jakobovits and Dirk Vermeir. Robust Semantics for Argumentation Frameworks. *J Log Comput*, 9(2):215–261, 1999. DOI 10.1093/logcom/9.2.215.

[106] Magdalena Kacprzak, Katarzyna Budzynska, and Olena Yaskorska. A logic for strategies in persuasion dialogue games. In Manuel Graña, Carlos Toro, Jorge Posada, Robert J. Howlett, and Lakhmi C. Jain, editors, *Advances in Knowledge-Based and Intelligent Information and Engineering Systems - 16th Annual KES Conference, San Sebastian, Spain, 10-12 September 2012*, volume 243 of *Frontiers in Artificial Intelligence and Applications*, pages 98–107. IOS Press, 2012. ISBN 978-1-61499-104-5. DOI 10.3233/978-1-61499-105-2-98.

[107] Hirofumi Katsuno and Alberto O. Mendelzon. On the Difference between Updating a Knowledge Base and Revising It. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91). Cambridge, MA, USA, April 22-25, 1991.*, pages 387–394, 1991.

[108] Dowding Keith. *Encyclopedia of Power*. SAGE Publications, Inc., 2011. ISBN 978-1-4129-2748-2.

[109] Peter M. Kielar and André Borrmann. Spice: A cognitive agent framework for computational crowd simulations in complex environments. *Auton. Agents Multi-Agent Syst.*, 32(3):387–416, 2018. DOI 10.1007/s10458-018-9383-2.

[110] Angelika Kimmig, Bart Demoen, Luc De Raedt, Vítor Santos Costa, and Ricardo Rocha. On the implementation of the probabilistic logic programming language ProbLog. *Theory Pr. Log Program*, 11(2-3):235–262, 2011. DOI 10.1017/S1471068410000566.

[111] Angelika Kimmig, Bernd Gutmann, and Vitor Santos Costa. Trading Memory for Answers: Towards Tabling ProbLog. In *International Workshop on Statistical Relational Learning, (SRL)*, 2019.

[112] Mare Koit. Developing a Formal Model of Argumentation-based Dialogue. In H. Jaap van den Herik and Joaquim Filipe, editors, *Proceedings of the 8th International Conference on Agents and Artificial Intelligence (ICAART 2016), Volume 2, Rome, Italy, February 24-26, 2016*, pages 258–263. SciTePress, 2016. DOI 10.5220/0005665502580263.

[113] Eric M. Kok, John-Jules Ch Meyer, Henry Prakken, and Gerard Vreeswijk. A Formal Argumentation Framework for Deliberation Dialogues. In Peter McBurney, Iyad Rahwan, and Simon Parsons, editors, *Argumentation in Multi-Agent Systems - 7th International Workshop, ArgMAS 2010, Toronto, ON, Canada, May 10, 2010 Revised, Selected and Invited Papers*, volume 6614 of *Lecture Notes in Computer Science*, pages 31–48. Springer, 2010. DOI 10.1007/978-3-642-21940-5_3.

[114] Kostas Kolomvatsos, Kyriaki Panagidi, Ioannis Neokosmidis, Dimitris Varoutas, and Stathes Hadjiefthymiades. Automated concurrent negotiations: An artificial bee colony approach. *Electron. Commer. Res. Appl.*, 19:56–69, 2016. DOI 10.1016/j.elerap.2016.09.002.

[115] Sascha Kurz. Which criteria qualify power indices for applications? - A comment to "The story of the poor Public Good index". *CoRR*, abs/1812.05808, 2018. URL `http://arxiv.org/abs/1812.05808`.

[116] João Leite and João Martins. Social Abstract Argumentation. In Toby Walsh, editor, *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 2287–2292. IJCAI/AAAI, 2011. URL `http://ijcai.org/Proceedings/11/Papers/381.pdf`.

[117] Hengfei Li, Nir Oren, and Timothy J. Norman. Probabilistic Argumentation Frameworks. In Sanjay Modgil, Nir Oren, and Francesca Toni, editors, *Theorie and Applications of Formal Argumentation - First International Workshop, TAFA 2011. Barcelona, Spain, July 16-17, 2011, Revised Selected Papers*, volume 7132 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2011. DOI 10.1007/978-3-642-29184-5_1.

[118] 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.

[119] Theofrastos Mantadelis and Gerda Janssens. Dedicated Tabling for a Probabilistic Setting. In Manuel V. Hermenegildo and Torsten Schaub, editors, *Technical Communications of the 26th International Conference on Logic Programming, ICLP 2010, July 16-19, 2010, Edinburgh, Scotland, UK*, volume 7 of *LIPIcs*, pages 124–133. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010. DOI 10.4230/LIPIcs.ICLP.2010.124.

[120] Theofrastos Mantadelis and Gerda Janssens. Nesting Probabilistic Inference. *CoRR*, abs/1112.3785, 2011. URL `http://arxiv.org/abs/1112.3785`.

[121] Theofrastos Mantadelis and Ricardo Rocha. Using Iterative Deepening for Probabilistic Logic Inference. In Yuliya Lierler and Walid Taha, editors, *Practical Aspects of Declarative Languages - 19th International Symposium, PADL 2017, Paris, France, January 16-17, 2017, Proceedings*, volume 10137 of *Lecture Notes in Computer Science*, pages 198–213. Springer, 2017. DOI 10.1007/978-3-319-51676-9_14.

[122] Diego C. Martínez, Alejandro Javier García, and Guillermo Ricardo Simari. An Abstract Argumentation Framework with Varied-Strength Attacks. In Gerhard Brewka and Jérôme Lang, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Eleventh International Conference, KR 2008, Sydney, Australia, September 16-19, 2008*, pages 135–144. AAAI Press, 2008. ISBN 978-1-57735-384-3. URL `https://pdfs.semanticscholar.org/5f08/4fa5a2ca6949d6a639bace68205fce272ea0.pdf`.

[123] Paul-Amaury Matt and Francesca Toni. A Game-Theoretic Measure of Argument Strength for Abstract Argumentation. In Steffen Hölldobler, Carsten Lutz, and Heinrich Wansing, editors, *Logics in Artificial Intelligence, 11th European Conference, JELIA*, volume 5293 of *LNCS*, pages 285–297. Springer, 2008. ISBN 978-3-540-87802-5.

[124] Nicolas Maudet, Simon Parsons, and Iyad Rahwan. Argumentation in Multi-Agent Systems: Context and Recent Developments. In *Argumentation in Multi-Agent*

Systems, Third International Workshop, ArgMAS 2006, Hakodate, Japan, May 8, 2006, Revised Selected and Invited Papers, pages 1–16, 2006. DOI 10.1007/978-3-540-75526-5_1.

[125] Peter McBurney and Simon Parsons. Dialogue Games for Agent Argumentation. In *Argumentation in Artificial Intelligence*, pages 261–280. Springer, 2009. DOI 10.1007/978-0-387-98197-0_13.

[126] Arunas Miliauskas and Dale Dzemydiene. An Approach to Designing Belief-Desire-Intention Based Virtual Agents for Travel Assistance. In Audrone Lupeikiene, Raimundas Matulevicius, and Olegas Vasilecas, editors, *Joint Proceedings of Baltic DB&IS 2018 Conference Forum and Doctoral Consortium Co-Located with the 13th International Baltic Conference on Databases and Information Systems (Baltic DB&IS 2018), Trakai, Lithuania, July 1-4, 2018*, volume 2158 of *CEUR Workshop Proceedings*, pages 94–103. CEUR-WS.org, 2018. URL `http://ceur-ws.org/Vol-2158/paper10.pdf`.

[127] Robin Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer, 1980. ISBN 978-3-540-10235-9. DOI 10.1007/3-540-10235-3.

[128] Sanjay Modgil and Martin Caminada. Proof Theories and Algorithms for Abstract Argumentation Frameworks. In Guillermo Ricardo Simari and Iyad Rahwan, editors, *Argumentation in Artificial Intelligence*, pages 105–129. Springer, 2009. ISBN 978-0-387-98196-3. DOI 10.1007/978-0-387-98197-0_6.

[129] Sanjay Modgil and Henry Prakken. The ASPIC$^+$ framework for structured argumentation: A tutorial. *Argum. Comput.*, 5(1):31–62, 2014. DOI 10.1080/19462166.2013.869766.

[130] Andreas Niskanen and Matti Järvisalo. (μ)-toksia: An Efficient Abstract Argumentation Reasoner. In Diego Calvanese, Esra Erdem, and Michael Thielscher, editors, *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning, KR 2020, Rhodes, Greece, September 12-18, 2020*, pages 800–804, 2020. DOI 10.24963/kr.2020/82.

[131] Emilia Oikarinen and Stefan Woltran. Characterizing strong equivalence for argumentation frameworks. *Artif Intell*, 175(14-15):1985–2009, 2011. DOI 10.1016/j.artint.2011.06.003.

[132] Dionne Tiffany Ong, Christine Rachel De Jesus, Luisa Katherine Gilig, Junlyn Bryan Alburo, and Ethel Ong. Building a Commonsense Knowledge Base

for a Collaborative Storytelling Agent. In Kenichi Yoshida and Maria R. Lee, editors, *Knowledge Management and Acquisition for Intelligent Systems - 15th Pacific Rim Knowledge Acquisition Workshop, PKAW 2018, Nanjing, China, August 28-29, 2018, Proceedings*, volume 11016 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2018. ISBN 978-3-319-97288-6. DOI 10.1007/978-3-319-97289-3_1.

[133] Nir Oren and Timothy J. Norman. Semantics for Evidence-Based Argumentation. In Philippe Besnard, Sylvie Doutre, and Anthony Hunter, editors, *Computational Models of Argument: Proceedings of COMMA 2008, Toulouse, France, May 28-30, 2008*, volume 172 of *Frontiers in Artificial Intelligence and Applications*, pages 276–284. IOS Press, 2008. URL `http://www.booksonline.iospress.nl/Content/View.aspx?piid=9287`.

[134] Alison R. Panisson, Felipe Meneguzzi, Renata Vieira, and Rafael H. Bordini. Towards Practical Argumentation-Based Dialogues in Multi-agent Systems. In *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, WI-IAT 2015, Singapore, December 6-9, 2015 - Volume II*, pages 151–158. IEEE Computer Society, 2015. DOI 10.1109/WI-IAT.2015.208.

[135] David Park. Concurrency and automata on infinite sequences. In Peter Deussen, editor, *Theoretical Computer Science*, volume 104, pages 167–183. Springer-Verlag, Berlin/Heidelberg, 1981. ISBN 978-3-540-10576-3. DOI 10.1007/BFb0017309.

[136] Andrea Pazienza, Stefano Ferilli, and Floriana Esposito. Constructing and Evaluating Bipolar Weighted Argumentation Frameworks for Online Debating Systems. In Stefano Bistarelli, Massimiliano Giacomin, and Andrea Pazienza, editors, *Proceedings of the 1st Workshop on Advances In Argumentation In Artificial Intelligence Co-Located with XVI International Conference of the Italian Association for Artificial Intelligence, AI(^{3})@AI*IA 2017, Bari, Italy, November 16-17, 2017*, volume 2012 of *CEUR Workshop Proceedings*, pages 111–125. CEUR-WS.org, 2017. URL `http://ceur-ws.org/Vol-2012/AI3-2017_paper_12.pdf`.

[137] Carl Adam Petri and Wolfgang Reisig. Petri net. *Scholarpedia*, 3(4):6477, 2008. DOI 10.4249/scholarpedia.6477.

[138] Giuseppe Pisano, Roberta Calegari, Andrea Omicini, and Giovanni Sartor. Arg-tuProlog: A tuProlog-based Argumentation Framework. In Francesco Calimeri, Simona Perri, and Ester Zumpano, editors, *Proceedings of the 35th Italian Conference on Computational Logic - CILC 2020, Rende, Italy, October 13-15, 2020*, volume 2710 of *CEUR Workshop Proceedings*, pages 51–66. CEUR-WS.org, 2020. URL `http://ceur-ws.org/Vol-2710/paper4.pdf`.

[139] Henry Prakken. Models of Persuasion Dialogue. In Guillermo Ricardo Simari and Iyad Rahwan, editors, *Argumentation in Artificial Intelligence*, pages 281–300. Springer, 2009. ISBN 978-0-387-98196-3. DOI 10.1007/978-0-387-98197-0_14.

[140] Henry Prakken. An abstract framework for argumentation with structured arguments. *Argum. Comput.*, 1(2):93–124, 2010. DOI 10.1080/19462160903564592.

[141] Henry Prakken and Giovanni Sartor. Argument-Based Extended Logic Programming with Defeasible Priorities. *J Appl Non Cl. Log.*, 7(1):25–75, 1997. DOI 10.1080/11663081.1997.10510900.

[142] Henry Prakken and Michiel De Winter. Abstraction in Argumentation: Necessary but Dangerous. In Sanjay Modgil, Katarzyna Budzynska, and John Lawrence, editors, *Computational Models of Argument - Proceedings of COMMA 2018, Warsaw, Poland, 12-14 September 2018*, volume 305 of *Frontiers in Artificial Intelligence and Applications*, pages 85–96. IOS Press, 2018. DOI 10.3233/978-1-61499-906-5-85.

[143] Anand S. Rao and Michael P. Georgeff. Modeling Rational Agents within a BDI-Architecture. In James F. Allen, Richard Fikes, and Erik Sandewall, editors, *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91). Cambridge, MA, USA, April 22-25, 1991*, pages 473–484. Morgan Kaufmann, 1991. ISBN 978-1-55860-165-9.

[144] Alessandro Ricci, Rafael Heitor Bordini, Jomi Fred Hübner, and Rem Collier. AgentSpeak(ER): An Extension of AgentSpeak(L) improving Encapsulation and Reasoning about Goals. In Elisabeth André, Sven Koenig, Mehdi Dastani, and Gita Sukthankar, editors, *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15, 2018*, pages 2054–2056. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM, 2018. URL `http://dl.acm.org/citation.cfm?id=3238069`.

[145] Tjitze Rienstra, Chiaki Sakama, and Leendert W. N. van der Torre. Persistence and Monotony Properties of Argumentation Semantics. In Elizabeth Black, Sanjay Modgil, and Nir Oren, editors, *Theory and Applications of Formal Argumentation - Third International Workshop, TAFA 2015, Buenos Aires, Argentina, July 25-26, 2015, Revised Selected Papers*, volume 9524 of *Lecture Notes in Computer Science*, pages 211–225. Springer, 2015. DOI 10.1007/978-3-319-28460-6_13.

[146] Nicolas D. Rotstein, Martın O. Moguillansky, Alejandro J. Garcia, and Guillermo R. Simari. An abstract argumentation framework for handling dynamics. In *Proceedings of the Argument, Dialogue and Decision Workshop in NMR*

2008, Sydney, Australia, pages 131–139, 2008. URL https://cs.uns.edu.ar/~mom/publications/nmr2008-DAF.pdf.

[147] Vijay A. Saraswat and Martin Rinard. Concurrent constraint programming. In *Proceedings of the 17th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages - POPL '90*, pages 232–245, San Francisco, California, United States, 1990. ACM Press. ISBN 978-0-89791-343-0. DOI 10.1145/96709.96733.

[148] Taisuke Sato. A Statistical Learning Method for Logic Programs with Distribution Semantics. In Leon Sterling, editor, *Logic Programming, Proceedings of the Twelfth International Conference on Logic Programming, Tokyo, Japan, June 13-16, 1995*, pages 715–729. MIT Press, 1995.

[149] Claudia Schulz and Dragos Dumitrache. The ArgTeach Web-Platform. In Pietro Baroni, Thomas F. Gordon, Tatjana Scheffler, and Manfred Stede, editors, *Computational Models of Argument - Proceedings of COMMA 2016, Potsdam, Germany, 12-16 September, 2016*, volume 287 of *Frontiers in Artificial Intelligence and Applications*, pages 475–476. IOS Press, 2016. DOI 10.3233/978-1-61499-686-6-475.

[150] Lloyd S. Shapley. *Contributions to the Theory of Games*, volume II of *AM-28*. Princeton University Press, 1953.

[151] Bayan Abu Shawar. Integrating CALL Systems with Chatbots as Conversational Partners. *Comput. Sist.*, 21(4), 2017. URL http://www.cys.cic.ipn.mx/ojs/index.php/CyS/article/view/2868.

[152] Gopal Kirshna Shyam, Sunilkumar S. Manvi, and Bhanu Prasad. Concurrent and cooperative negotiation of resources in cloud computing: A game theory based approach. *Multiagent Grid Syst.*, 14(2):177–202, 2018. DOI 10.3233/MGS-180287.

[153] Carlo Taticchi. A Study of Robustness in Abstract Argumentation Frameworks. In Viviana Mascardi and Ilaria Torre, editors, *Proceedings of the Doctoral Consortium of AI*IA 2016 Co-Located with the 15th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2016), Genova, Italy, November 29, 2016*, volume 1769 of *CEUR Workshop Proceedings*, pages 11–16. CEUR-WS.org, 2016. URL http://ceur-ws.org/Vol-1769/paper02.pdf.

[154] Carlo Taticchi. Power Index-Based Semantics for Ranking Arguments in Abstract Argumentation Frameworks: An Overview. In Mario Alviano, Gianluigi Greco, Marco Maratea, and Francesco Scarcello, editors, *Discussion and Doctoral Consortium Papers of AI*IA 2019 - 18th International Conference of the Italian Association for Artificial Intelligence, Rende, Italy, November 19-22, 2019*, volume

2495 of *CEUR Workshop Proceedings*, pages 113–118. CEUR-WS.org, 2019. URL http://ceur-ws.org/Vol-2495/paper14.pdf.

[155] Bas Testerink and Floris J. Bex. Specifications for Peer-to-Peer Argumentation Dialogues. In Bo An, Ana L. C. Bazzan, João Leite, Serena Villata, and Leendert W. N. van der Torre, editors, *PRIMA 2017: Principles and Practice of Multi-Agent Systems - 20th International Conference, Nice, France, October 30 - November 3, 2017, Proceedings*, volume 10621 of *Lecture Notes in Computer Science*, pages 227–244. Springer, 2017. ISBN 978-3-319-69130-5. DOI 10.1007/978-3-319-69131-2_14.

[156] Matthias Thimm. Measuring Inconsistency in Probabilistic Knowledge Bases. In Jeff A. Bilmes and Andrew Y. Ng, editors, *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, June 18-21, 2009*, pages 530–537. AUAI Press, 2009.

[157] Matthias Thimm. A Probabilistic Semantics for abstract Argumentation. In Luc De Raedt, Christian Bessiere, Didier Dubois, Patrick Doherty, Paolo Frasconi, Fredrik Heintz, and Peter J. F. Lucas, editors, *ECAI 2012 - 20th European Conference on Artificial Intelligence. Including Prestigious Applications of Artificial Intelligence (PAIS-2012) System Demonstrations Track, Montpellier, France, August 27-31 , 2012*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, pages 750–755. IOS Press, 2012. DOI 10.3233/978-1-61499-098-7-750.

[158] Matthias Thimm. The Formal Argumentation Libraries of Tweety. In Elizabeth Black, Sanjay Modgil, and Nir Oren, editors, *Theory and Applications of Formal Argumentation - 4th International Workshop, TAFA 2017, Melbourne, VIC, Australia, August 19-20, 2017, Revised Selected Papers*, volume 10757 of *Lecture Notes in Computer Science*, pages 137–142. Springer, 2017. DOI 10.1007/978-3-319-75553-3_9.

[159] Stephen E. Toulmin. *The Uses of Argument, Updated Edition*. Cambridge University Press, 2008. ISBN 978-0-521-53483-3.

[160] Joanna Turner, Qinggang Meng, Gerald Schaefer, and Andrea Soltoggio. Distributed Strategy Adaptation with a Prediction Function in Multi-Agent Task Allocation. In Elisabeth André, Sven Koenig, Mehdi Dastani, and Gita Sukthankar, editors, *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15, 2018*, pages 739–747. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM, 2018. URL http://dl.acm.org/citation.cfm?id=3237492.

[161] Yuichi Umeda and Hajime Sawamura. Towards an argument-based agent system. In Lakhmi C. Jain, editor, *Third International Conference on Knowledge-Based Intelligent Information Engineering Systems, KES 1999, Adelaide, South Australia, 31 August - 1 September 1999, Proceedings*, pages 30–33. IEEE, 1999. DOI 10.1109/KES.1999.820112.

[162] Devindra Weerasooriya, Anand S. Rao, and Kotagiri Ramamohanarao. Design of a Concurrent Agent-Oriented Language. In Michael Wooldridge and Nicholas R. Jennings, editors, *Intelligent Agents, ECAI-94 Workshop on Agent Theories, Architectures, and Languages, Amsterdam, The Netherlands, August 8-9, 1994, Proceedings*, volume 890 of *Lecture Notes in Computer Science*, pages 386–401. Springer, 1994. ISBN 978-3-540-58855-9. DOI 10.1007/3-540-58855-8_25.

[163] Eyal Winter. Chapter 53 The shapley value. In *Handbook of Game Theory with Economic Applications*, volume 3, pages 2025–2054. Elsevier, January 2002. DOI 10.1016/S1574-0005(02)03016-3.

[164] Michael J. Wooldridge. *An Introduction to MultiAgent Systems (2. Ed.)*. Wiley, 2009. ISBN 978-0-470-51946-2.

[165] Yining Wu and Martin Caminada. A Labelling-Based Justification Status of Arguments. *Stud. Log.*, 3(4):12–29, 2010.

[166] Yu Wu, Zhoujun Li, Wei Wu, and Ming Zhou. Response selection with topic clues for retrieval-based chatbots. *Neurocomputing*, 316:251–261, 2018. DOI 10.1016/j.neucom.2018.07.073.

[167] Yuming Xu and Claudette Cayrol. The Matrix Approach for Abstract Argumentation Frameworks. In Elizabeth Black, Sanjay Modgil, and Nir Oren, editors, *Theory and Applications of Formal Argumentation - Third International Workshop, TAFA 2015, Buenos Aires, Argentina, July 25-26, 2015, Revised Selected Papers*, volume 9524 of *Lecture Notes in Computer Science*, pages 243–259. Springer, 2015. ISBN 978-3-319-28459-0. DOI 10.1007/978-3-319-28460-6_15.

[168] Bruno Yun, Srdjan Vesic, and Madalina Croitoru. Ranking-Based Semantics for Sets of Attacking Arguments. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 3033–3040. AAAI Press, 2020. URL `https://aaai.org/ojs/index.php/AAAI/article/view/5697`.