Between Argument and Conclusion

Argument-based Approaches to Discussion, Inference and Uncertainty

UNIVERSITÉ DU LUXEMBOURG

PhD-FSTC-2012-12
The Faculty of Sciences, Technology and Communication

DISSERTATION

Defense held on 18/05/2012 in Luxembourg
to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG

EN INFORMATIQUE

by

Yining Wu
Born on 22/04/1978 in Anshan (Liaoning, P.R. China)

Between Argument and Conclusion
Argument-based Approaches to Discussion, Inference and Uncertainty

Acknowledgement

First of all , I would like to express my gratitude to my supervisor Martin Caminada and Leon van der Torre, for giving me the chance to do my PhD in the ICR group. Thanks for the advice, the encouragement, enduring patience and constant support. Thanks for leading me to the real scientific research.

Secondly, I would thank Henry Prakken, for his suggestions, comments and guidance which were invaluable to the completion of this work.

Thirdly, I would thank my coauthors: Martin Caminada, Patrizio Barbini, Mikolaj Podlaszewski and Dov Gabbay because scientific research is a communal endeavor.

Additionally, I would thank Mikolaj Podlaszewski, Leon van der Torre, Martin Caminada, Henry Prakken and Emil Weydert who helped me with Chapter 6.

Finally, I must thank my parents. They always stand behind me with continuing and loving support. I could not have finished this work without their support.

Abstract

Formal argumentation has become a popular approach for nonmonotonic reasoning and multi-agent communication in Artificial Intelligence and Computer Science. The approach of Dung on the semantics of abstract argumentation frameworks, which specifies different criteria for selecting arguments given an argumentation framework, received a lot of follow-up.

This work aims to enhance today's generation of argumentation formalisms and make it become suitable for a wider variety of real-life applications.

A number of dialectical proof procedures are associated with some argumentation semantics, e.g., grounded and preferred semantics. However, there were still some argumentation semantics, e.g., stable semantics that did not have associated dialectical proof procedures. In this thesis, we fill the gap by providing a dialectical discussion game for stable semantics.

Since there could be more than one set of arguments that are selected with respect to complete semantics, we provide a more refined notion of the overall status of arguments based on the notion of a complete labelling and give the procedure to determine and defend the justification status. The approach is implemented by two dialectical proof procedures, namely the grounded discussion game and the preferred discussion game.

Furthermore, argumentation can be viewed as a special form of logic programming with negation of failure, which leads to the question whether there exist correspondences between semantics in argumentation and semantics in logic programming. In this work, we show that the complete extensions in abstract argumentation coincide with the 3-valued stable models in logic programming. Since many argumentation semantics can be described based on complete semantics and many logic programming semantics can be described based on 3-valued stable semantics, the correspondences between them lead to new possible correspondences, which enabled one to apply already existing algorithms and computational results of one area to the other area.

One particular property of Dung's abstract argumentation frameworks is that they do not consider the internal structure of arguments and the nature of the defeat relation. Various approaches have been proposed to instantiate Dung's abstract argumentation framework. However, an important question is what postulates a proper instantiation of Dung's theory should satisfy. There are several postulates that have been introduced, e.g., closure, direct consistency, indirect consistency, non-interference and crash resistance. In this thesis, we consider the ASPIC Lite system. We define the postulates of closure, direct consistency, indirect consistency, non-interference and crash resistance in the specific case of the ASPIC Lite formalism. Then we identity a set of conditions under which the ASPIC Lite system satisfies all the mentioned rationality postulates under complete semantics.

Contents

Chapter 1

Introduction

Argumentation is everywhere. Debates between authors and reviewers are common scenarios in the academic world. Argumentation theory aims to naturally capture the way people argue to justify their solutions to many social problems. Over the past decade, argumentation has become increasingly central in Artificial Intelligence (AI). There has been a large growth of interest in the subject from formal and technical perspectives in Computer Science and Artificial Intelligence, and wide use of argumentation technologies in practical applications [28]. In recent years, philosophers and lawyers have become interested in the idea of "argumentation" and argumentation has been applied to problems of defeasible reasoning in law.

In computer science and artificial intelligence, argumentation is supposed to be mechanical procedure for events interpretation, documents organisation and presentation and decision making. On the one hand, formal argumentation is contributing significantly to Artificial Intelligence, e.g., defining semantics of logic programs, implementing persuasive medical diagnostic systems, and studying negotiation dialogues in multi-agent systems. On the other hand, Artificial Intelligence influences argumentation theory and practices by providing formal methods for argument analysis, evaluation and visualisation.

The contribution of argumentation to computer science can be viewed from two perspectives. First, from a theoretical point of view, argumentation constructs a new framework which helps to enrich classical forms of reasoning (including logical deduction, induction, abduction and plausible reasoning, supporting discussion and negotiation in computer-supported cooperative work and learning). Second, from a human-computer interaction perspective, argumentation provides a novel technique that facilitates natural system behavior and is more easily understood by human users and operators. Generally speaking, argumentation has an effect on providing information and advice for human users or agents.

1.1 The nature of argumentation and its study in AI

The computational theory of argumentation has benefited from many diverse disciplines, e.g., philosophy, logic and law. Thus, argumentation is a good example of interdisciplinary interchange and mutual benefits. Figure 1.1 shows the relation between argumentation and three other fields, particularly dialogue game, logic programming and nonmonotonic reasoning. Roughly, many major approaches to nonmonotonic reasoning in AI are special forms of Dung's theory of argumentation. The correspondences between abstract argumentation semantics and logic programming semantics reflect the overlap between abstract argumentation and logic programming. Dialogical games can be used as a proof procedure that tests the statuses of arguments with respect to abstract argumentation semantics.

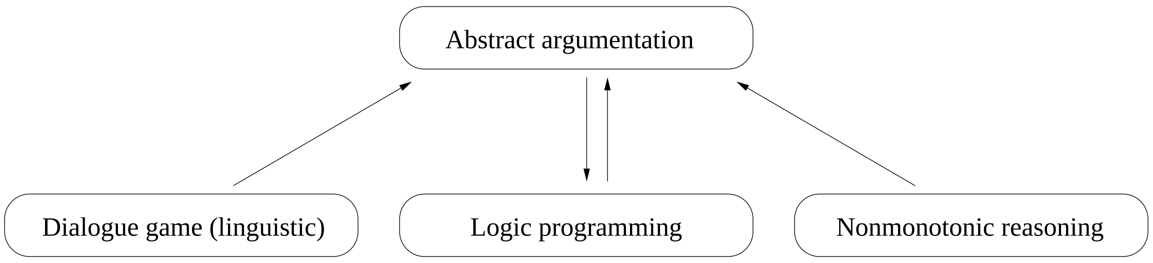


Figure 1.1: Foundations of abstract argumentation

Argumentation is a major mechanism for interaction among agents. “Argumentation is more than ‘reasoning’ in the recesses of single minds, since it crucially involves interaction.”(Johan van Benthem, foreword in [85]). The aims of argumentation are not only deciding the status of arguments by a single party but also exchanging ideas and positions involving several parties. Typically, argumentation concerning an issue can be seen as a dialogical process. Analysing argument justifiability via dialogue games has been fruitfully adapted to argumentation in AI. The demand for collaboration and communication between several autonomous agents keeps increasing in complex technical systems and services, since their goals and tasks are inherently interdependent. Complex technical systems and services depend more and more upon conversations concerned with negotiation or persuasion where agents have different capabilities and viewpoints. Such dialogues exchange proposals, claims or offers. The essential difference between argumentation-based discussions and other approaches is that proposals can be supported by the reasons that justify or oppose them. This is more flexible than it is in other decision-making and communication schemes because information or knowledge that is not being considered can be used to persuade agents to change their view of a claim.

Since the early 1960s, Lorenz and Lorenzen [58, 61, 60] have proposed the concept of dialogue:

> “Fully spelled out it means that for an entity to be a proposition there must exist a dialogue game associated with this entity, i.e., the proposition A, such that an individual play of the game where A occupies the initial position, i.e., a dialogue $D(A)$ about A, reaches a final position with either win or loss after a finite number of moves according to definite rules: the dialogue game is defined as a finitary open two-person zero-sum game.” [59].

To briefly sketch the central concept of dialogue in the field of argumentation, dialogues are games where what is at stake is an argument, when applied to abstract argumentation theory. The set of rules is designed in such a way that a dialogue can be won by the proponent of an argument if and only that argument is justified in a semantic sense. The definition shows an important aspect of constructivity in dialogical logic: infinity plays no role, i.e., dialogical logic does not rely on a quantification over the set of models. The dialogical proof theories are based on the notion of game and dialogue games are finitary. Therefore, in this thesis we consider only finite sets of arguments.

Dialogues provide a way to specify the semantics of a given logic [55]. A tradition in dialogical logic consists of the study of concrete dialogues, for example, dialogues that occur in natural languages. The aim of this tradition was to study the underlying logical regularities of these concrete dialogues. This includes Toulmin's argumentation theory [90] and Woods' work in argumentation theory [98], etc. Another tradition is that the dialogical approach yielded a semantics for intuitionistic and classical logic. In the context of the logical analysis of legal and nonmonotonic reasoning, efforts have been made to bridge the two traditions of dialogical logic. For more on these efforts, see the work of Prakken [76]. There are dialogue games that have been defined for some of the argumentation semantics introduced by Dung [40]. For instance, the discussion game for grounded [80, 75, 20] and the discussion game for preferred semantics [97]. However, there are still remaining argumentation semantics that have not been associated with any dialogue games. In this thesis, we present a dialogue game for stable semantics.

Let us use a well-known example of Prakken [76] to illustrate how agents interact with each other and what is argumentation.

Example 1.1. *The conversation below is between Paul and Olga about whether a car is safe.*

Paul:	*My car is very safe.*
Olga:	*Why?*
Paul:	*Since it has an airbag.*
Olga:	*It is true that your car has an airbag, but I do not think that this makes your car safe, because airbags are unreliable: the newspapers had several reports on cases where airbags did not work.*
Paul:	*I also read that report but a recent scientific study showed that cars with airbags are safer than cars without airbags, and scientific studies are more important than newspaper reports.*

The following arguments are from the conversation:

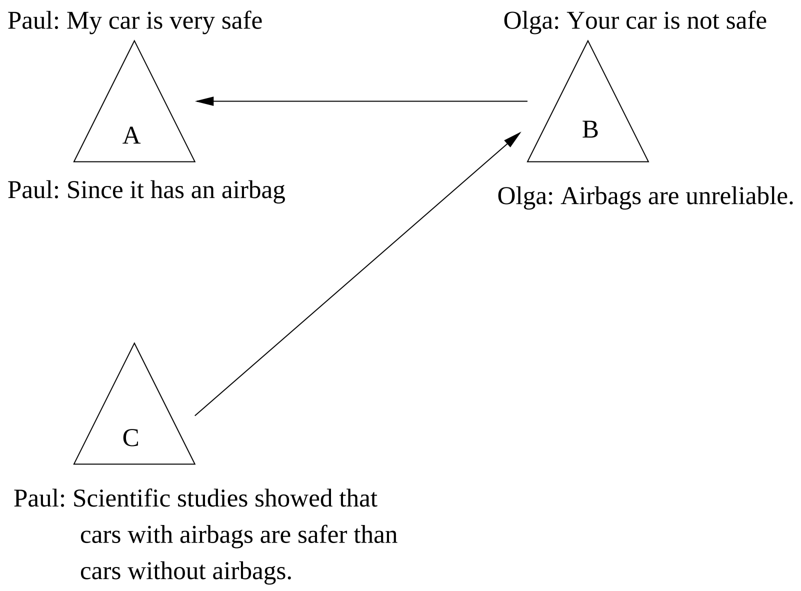

Paul: My car is very safe
A
Paul: Since it has an airbag
Olga: Your car is not safe
B
Olga: Airbags are unreliable.
C
Paul: Scientific studies showed that cars with airbags are safer than cars without airbags.

Figure 1.2: An example (Prakken): airbags make cars safe.

In this conversation, argument A is defeated by argument B and argument B is defeated by argument C. Then argument A is accepted.

The structure of arguments and the acceptability of arguments are fundamental for a computer system to be able to deal with the exchanges of arguments. Much work has been done to build such computer systems. Argument systems which can be applied in political dialogues have been built by Alvarado [1] and Birnbaum *et al.* [17, 18, 64]. Dung's work [40] answered the question of how to understand the acceptability of arguments. The central notion of Dung's abstract argumentation framework is the acceptability of arguments.

Roughly, the idea of argumentation reasoning is that a statement is acceptable if it can be defended against attacking arguments. In other words, whether or not a rational agent accepts a statement depends on whether or not the argument supporting this statement can be successfully defended against the counterarguments. The abstract argumentation frameworks introduced by Dung [40] come together with semantics which specify subsets of arguments, so called extensions, consisting of the arguments which are accepted. Several kinds of argumentation labelings have been proposed either for algorithmic or logical purposes [32, 92, 93] for a more fine-grained distinction of arguments. In this thesis we consider three-valued labelings as proposed by Caminada and Gabbay [32]. The idea is that such labelings partition the arguments of a framework into three sets. One set is the set of arguments that are accepted and labelled by `in`. The other two sets of arguments are not accepted. One of them is the set of arguments that are rejected and labelled by `out`. One of them is the set of arguments that are neither accepted or rejected and labelled `undec`.

There are two acceptance statuses of arguments decided by the traditional extension-based approaches: credulous acceptance and skeptical acceptance. Credulous acceptance distinguishes arguments contained in at least one extension and arguments which are in no extension at all. Skeptical acceptance means that arguments are contained in each extension. However such a characterisation completely ignores the additional information provided by labelings. To take this information into account, in this thesis we propose a labeling-based justification status of an argument, which allows to distinguish different levels of acceptance (and rejection) for arguments based on the labelings of an argumentation framework.

Many different approaches to nonmonotonic reasoning have been proposed in AI [70, 66, 63, 68, 39, 45, 46, 82, 89]. At first sight, they are very different but it turns out that some of the major approaches to nonmonotonic reasoning in AI and logic programming are special forms of Dung's theory of argumentation. Dung's work [40] has presented argumentation theory with the necessary mathematical abstractions for it to reach scientific maturity, and become an independent research area within the field of non-monotonic reasoning. Furthermore, Dung [40] has shown that argumentation can be viewed as a special form of logic programming with negation as failure. This result shows that argumentation can be implemented in logic programming. Dung [40] has presented the correspondence between well-founded models in logic programming and the grounded extension in abstract argumentation and the correspondence between stable models in logic programming and stable extensions in abstract argumentation. Due to the fact that there are other semantics in logic programming, e.g., 3-valued stable, regular and L-stable semantics and the fact that argumentation also has more different semantics, e.g., complete, preferred and semi-stable semantics, it is interesting for practical purposes to find the correspondence relations between those semantics in two fields. In this thesis, we demonstrate that 3-valued stable models in logic programming coincide with complete extensions in abstract argumentation.

Abstract argumentation leaves the internal structure of the arguments and the nature of the defeat relation completely abstract. A number of approaches have been proposed with respect to the structure of arguments, including lists [80], trees [4] or sets of assumptions [16]. For the defeat relation, there are assumption attacks, e.g., the one implemented by Besnard and Hunter [16], Pollock-style undercutting of rules [69, 71], conclusion rebutting [69, 71] and Vreeswijk's notion of rule rebutting. Several advanced forms of arguments have been shown to fit in Dung's standard notion of an argumentation framework. For instance, Caminada's Socratic-style arguments [20] and Prakken's notion of accrued arguments [77]. Various forms of dealing with priorities, including weakest link, last link or the notion of defeasible priorities [80] are taken into account for the defeat relation. However, a proper instantiation of Dung's theory is not a trivial task. Various argumentation formalisms have been criticized [21] for yielding unexpected and undesired outcomes as result of instantiations. Caminada and Amgoud [31] have defined three rationality postulates that can be used to judge the quality of an argumentation system, namely direct consistency, indirect consistency and closure. Caminada, Carnielli and Dunne [34] continued this line of work by defining two more rationality postulates: non-interference and crash resistance. The ASPIC framework [4] is a framework for instantiating Dung frameworks which was developed by Leila Amgoud, Martin Caminada, Claudette Cayrol, Marie-Christine Lagasquie-Schieux, Henry Prakken and Gerard Vreeswijk. In Prakken's work [78] conditions have been identified under which instantiations of the ASPIC framework satisfy the postulates of consistency and closure, extending the results of Caminada and Amgoud [31]. In this thesis, we consider a similar argumentation system which is similar to ASPIC. We call it ASPIC Lite. We show that the ASPIC Lite system does not satisfy the postulates of non-interference and crash resistance. Then we identity a set of conditions under which the ASPIC Lite system satisfies all the mentioned rationality postulates under complete semantics.

1.2 Research questions

The field of formal argumentation can be traced back to the work of Pollock [68, 69], Vreeswijk [95], Lin and Shoham [57], Loui [62], Konolige [56] and Simari and Loui [89]. The idea is that (nonmonotonic) reasoning can be performed by constructing and evaluating arguments, which are composed of a number of reasons that together support a particular claim. Arguments distinguish themselves from proofs by the fact that they are defeasible, that is, the validity of their conclusions can be disputed by other arguments. Whether a claim can be accepted therefore depends not only on the existence of an argument that supports this claim, but also on the existence of possible counter arguments, that can then themselves be defeated by counter arguments, etc.

A great impulse has been the work of Dung [40], in which the concept of an *argumentation framework* was first proposed. An argumentation framework, in essence, is a directed graph in which the nodes represent arguments and the arrows the defeat relations among these arguments. The innovation of Dung [40] was that it shows that various formalisms for nonmonotonic entailment can be described in alternative way using the concept of an argumentation framework. The process of doing so consists of 3 steps. Starting from an underlying knowledge base (e.g. a default theory or logic program) one constructs a graph (step 1) in which the nodes are essentially defeasible proofs that defeat each other. Based on such a graph, one then applies a criterion for selecting zero or more sets of nodes (also called extensions) (step 2). Several of such criteria (also called argumentation semantics) have been stated in the literature under names like grounded, complete, preferred and stable semantics [40], semi-stable semantics [25, 92], ideal semantics [41], eager semantics [27] and CF2 semantics [12, 11, 10].

Although in Dung's original paper, abstract argumentation (step 2) served simply as an intermediate reasoning step in an overall entailment process, one can to some extent also examine abstract argumentation on its own. One particular question one may ask is what it is that the different selection criteria (argumentation semantics) actually constitute. Several argumentation semantics have associated dialectical proof procedures, where a proponent and an opponent argue about the status of a main argument by exchanging counter arguments against each other. This leads to the following research question:

Question 1. *What is a discussion game for stable semantics?*

The discussion game associated with credulous preferred semantics has a Socratic nature [29]. The proponent takes a particular position (argument A is in at least one preferred extension) and then the opponent (assuming the role of Socrates) tries to lead the proponent towards a contradiction by pointing out the consequences of proponent's own position and asking the proponent for additional explanation. The discussion game associated with grounded semantics, on the other hand, has a merely persuasive nature. Here, the proponent tries to convince the opponent that he should accept a particular argument A since its acceptance cannot be avoided. In the current thesis, we examine the case of stable semantics and provide a formal discussion game for it, not only from the technical perspective, but also from a conceptual perspective. It should be emphasized that discussion games are more than just proof procedures of the associated abstract argumentation semantics. Unlike classical logic, argumentation theory is not so much about what is true in a model theoretical sense, but more about what can be defended in rational discussion. Several types of rational discussion exist. As we have seen before, some of these can be associated with preferred and grounded semantics. In this thesis, we examine the type of rational discussion that is associated with stable semantics.

One particular issue not only of abstract argumentation but also of non-abstract argumentation is the particular status of an argument (respectively of a conclusion). As was mentioned before, an abstract argumentation semantics yields zero or more sets of arguments (extensions). The question then becomes what this means for the overall status of an individual argument.

Question 2. *How to define justification statuses of arguments, what are its properties, and how can we compute them?*

Traditionally, the justification status of an argument has been determined based on the associated extensions only. The justification status of arguments can be used as a measurement of uncertainty since the justification status of arguments indicates the level of the acceptance uncertainty of arguments. An argument is called *sceptically justified* if and only if it is in all extensions. An argument is called *credulously justified* if and only if it is in some (but not necessarily all) extensions. In the current thesis, we aim to refine these notions using the concept of argument labellings. Argument labelings [23, 32, 71, 53, 92] provide an alternative and to some extent more informative way to express abstract argumentation semantics. In essence, an argumentation semantics provides zero or more reasonable positions that one can take given the conflicting information represented by the argumentation framework. In the extensions approach, each such position is represented by a set of arguments that is accepted. In the labellings approach, on the other hand, each such position is represented by an opinion on which argument to accept, which argument to reject and which argument to leave undecided. In the current thesis, we show how the concept of argument labellings can be used to obtain a more refined notion of the justification status of arguments than it is provided by the traditional extensions approach and how this notion fits with other more traditional notions of argumentation semantics and their associated justification statuses.

For purposes of nonmonotonic entailment, what matters are not so much the arguments themselves, but mostly the *conclusions* they support. Therefore, after obtaining zero or more sets of arguments (the extensions) by applying an argumentation semantics, we have to identify the corresponding conclusions of each of these sets (step 3). Since an argument is essentially a defeasible proof, this means that each argument from a set of arguments yields its conclusion to the corresponding set of conclusions. Various existing formalisms for nonmonotonic reasoning have been reformulated in terms of the above described 3 step process. For default logic, for instance, it has been proved that the default extensions of a particular default theory are precisely the same as the sets of conclusions one obtains at the end of step 3 when applying stable semantics at step 2 (details can be found in [40]). This leads to the following question:

Question 3. *Which argumentation semantics corresponds to 3-valued stable semantics in logic programming?*

Dung's theory shows that the 3-step process can be used to model default logic, logic programming under the stable model semantics and logic programming under the well-founded model semantics [40]. Logic programming is characterized by programming with relations and inference. The operational semantics of logic programs correspond to logical inference. Governatori *et al.* showed that Nute's defeasible logic can also be modelled using the 3-step process [51]. In the current thesis, we contribute to answering question 3 by showing that also logic programming under the 3-valued stable model semantics can be modelled using the 3-step process. The interesting thing is that, compared to logic programming under the stable model semantics and logic programming under the well founded model semantics, the only thing that needs to be changed is step 2. That is, step 1 and step 3 remain the same, regardless of whether one is modelling logic programming under the stable model semantics, well founded model semantics or 3 valued stable model semantics. Moreover, where Dung showed that stable model semantics corresponds with implementing stable semantics in step 2, and well founded semantics corresponds with implementing grounded semantics in step 2, we show that 3-valued stable semantics corresponds with implementing complete semantics in step 2.

Question 4. *How to make the ASPIC Lite system satisfy crash resistance and non-interference?*

The 3-step argumentation process, elegant as it may seem, does have an important limitation. In step 2, selecting the sets of arguments that can collectively be accepted (by applying the argumentation semantics) is done without actually examining the logical contents of the arguments, purely based on the structure of the graph. However, if one cannot see the contents of these arguments, then how can one be sure that the selected set will have reasonable properties from logical perspective (for instance, that the union of its conclusions is consistent)? Consistency is a proof-theoretic notion meaning that no contradiction can be derived in a calculus. Even a set of arguments that is conflict-free does not guarantee the consistency of the associated set of conclusions.

Finding combinations of how to perform step 1, 2 and 3 in order to obtain a reasonable outcome is not a trivial exercise. Caminada and Amgoud [31] for instance, have found that if one would like to apply the *unrestricted rebut* (Definition 9 in [31]) when constructing an argumentation framework (step 1), one has to apply grounded semantics on the abstract level (step 2) in order for the overall outcome to make sense (to satisfy the postulates of closure and indirect consistency). Similarly, if one wants to apply for instance preferred semantics on the abstract level (step 2) then one needs to apply the slightly less intuitive principle of *restricted rebut* (Definition 15 in [31]) when constructing the argumentation framework (step 1). In the current thesis, we make a new contribution to the overall question of how to make the three step argumentation process (with its "blind" selection criterion in step 2) work. In particular, we describe the postulates of *crash-resistance* and *non-interference* and show how these can be violated by argumentation formalisms that aim to combine classical logic with defeasible reasoning steps (e.g. by some instantiations of ASPIC framework [78]). Moreover, we identify conditions under which the ASPIC Lite system not only satisfies the newly identified postulates of *crash-resistance* and *non-interference*, but also keeps on satisfying the previously identified postulates [31] of direct consistency, indirect consistency and closure. We consider all propositional inferences when we construct argumentation frameworks.

This thesis is not based on a single idea, nor does it provide a new theory from scratch. The idea is that argumentation theory, at lease at the current stage, can best be regarded as an unfinished building that still needs many bricks. In the current PhD thesis, we aim to provide several of such bricks by contributing on questions like how general the 3-step argumentation paradigm actually is, when it comes to modelling concrete instances of non-monotonic formalisms (question 3), what are the notions of rational discussion underlying the various argumentation semantics (question 1), how can one reasonably determine the overall status of a particular argument in the presence of several extensions or labellings (question 2) and which properties does one want the overall argumentation process to satisfy, and how to achieve this (question 4). Our aim is not only to increase theoretical understanding of argumentation theory, but also to enrich it in order to increase its suitability for real world applications.

1.3 Relevance

In the past few years, the field of argumentation in Artificial Intelligence has grown significantly. Some approaches for formal argumentation already existed in the early 1990s. Dung's work [40] defined a number of semantics of abstract argumentation frameworks. There is a lot of following of the approach of Dung, including those who applied the abstract semantics as one of the components in their argumentation formalism, e.g., Prakken and Sartor [80] and Caminada and Amgoud [30]), those who proposed alternative argument-based semantics given a Dung-style argumentation framework, e.g., Baroni *et al.* [12], Caminada [25]) and those who have proposed to extend Dung's notion of an argumentation framework with new functionality, e.g., Bench-Capon [14], Amgoud *et al.* [7]. One can distinguish the following three levels of research regarding the functionality of formal argumentation:

1. "Dung-base". Abstract argument semantics, using argumentation frameworks as defined by Dung [40]. The central research question is the acceptability of arguments given an argumentation framework. Apart from the well-known approaches of complete, stable, preferred and grounded semantics, various alternatives have been stated, such as CF2 semantics [12], prudent semantics [37] and semi-stable semantics [25], stage semantics [92, 15], ideal semantics [41] and eager semantics [27].

2. "Dung-instantiated". In order to consider the internal structure of the arguments and the nature of the defeat relation, various approaches have been proposed. Lists [80], trees [4] or sets of assumptions [16] were proposed for the structure of arguments. For the defeat relation, there are assumption attacks [16], Pollock-style undercutting of rules [69, 71], conclusion rebutting [69, 71] and Vreeswijk's notion of rule rebutting. Caminada's Socratic-style arguments [20] and Prakken's notion of accrued arguments [77] have been shown to fit in Dung's standard notion of an argumentation framework. The ASPIC framework [78] combines ideas of Pollock, Vreeswijk and others on the structure of arguments and the nature of defeat relation. Preferences are considered in the ASPIC framework.

3. "Dung-extended". It has been argued that particular forms of natural argument and informal argumentation concepts require a formalization that is more than Dung's standard notion of an argumentation framework. Bench-Capon [14] proposed a value based argument framework for the representation an evaluation of arguments in practical reasoning. Amgoud *et al.* [7] have presented an approach of bipolar argumentation frameworks and shown that bipolarity may be used at different levels of an argumentation process. Amgoud and Cayrol [6] introduced an approach to deal with argument-strenth. Preference-based argumentation frameworks [5] and collective argumentation [19] also take place at the level of Dung-extended.

The range of our current work is over the level of "Dung-base" (item 1) and the level of "Dung-instantiated" (item 2).

1.4 Outline

The remaining parts of this thesis are organized as follows.

In Chapter 2, we restate some preliminaries on argumentation, including semantics of the traditional extension based approach and the labelling based approach, which is an alternative way to describe the standard admissibility based argumentation semantics. We show the three steps of constructing argumentation frameworks and then illustrate the three-steps of argumentation procedure by introducing the argumentation system introduced by Caminada and Amgoud [31]. We also introduce three desirable properties of argumentation systems, namely direct consistency, indirect consistency and closure. We provide a reformulated version of Vreeswijk and Prakken's preferred discussion game [97] and the grounded discussion game [80, 75, 20], which can be used to determine whether an argument is in at least one preferred extension or in the grounded extension respectively.

In Chapter 3, we present a discussion game for argumentation under stable semantics which is inspired by Vreeswijk and Prakken's preferred discussion game [97]. Besides, we show a discussion game for sceptical stable semantics by using the credulous discussion game. Chapter 3 is based on the joint work of Caminada and Wu [33].

In Chapter 4, we define a labelling-based justification status of the arguments in an argumentation framework. We show that the justification status of an argument can be determined by applying the grounded discussion game and the preferred discussion game and we also describe a software implementation that can compute the justification statuses. We compare the existing argumentation semantics and our approach and define a possible approach for justification statuses of conclusions. Chapter 4 is based on the joint work of Wu and Caminada [99].

In Chapter 5, we prove that the complete extensions in abstract argumentation coincide with 3-valued stable models in logic programming. First, we state some preliminaries on logic programming. Then we describe the transformation from argumentation frameworks to logic programs and the transformation from logic programs to argumentation frameworks. Then we show the correspondence between complete extensions in abstract argumentation and 3-valued stable models in logic programming. Chapter 5 is based on the joint work of Wu, Caminada and Gabbay [100].

In Chapter 6, we introduce an argumentation system called ASPIC Lite which is similar to the one treated by Caminada and Amgoud [31]. Two more desirable properties of argumentation systems are described. It is shown that the ASPIC Lite system does not satisfy these two properties under complete semantics. Then we identify conditions under which the ASPIC Lite system satisfies all five postulates under complete semantics.

We conclude the thesis in Chapter 7.

Chapter 2

Preliminaries

This chapter restates some preliminaries regarding argumentation frameworks, argument semantics, argument-labellings, argumentation games and argumentation schemes. Since we present a discussion game for stable semantics of abstract argumentation in Chapter 3 and dialogue games are finitary (as mentioned in the introductory chapter), also for simplicity, in this thesis we consider only finite argumentation frameworks.

2.1 Argumentation Framework

During discussions and debates, we automatically build argumentation frameworks, which can be used to determine which arguments can be ultimately accepted. Let us consider the following example (taken from [56]):

> "Suppose Ralph normally goes fishing on Sundays, but on the Sunday which is Mother's day, he typically visits his parents. Furthermore, in the spring of each leap year his parents take a vacation, so that they cannot be visited." [56]

Suppose it is Sunday, Mother's day and a leap year. Then one can formulate three arguments related to whether Ralph goes fishing or not:

Argument A: Ralph goes fishing because it is Sunday.
Argument B: Ralph does not go fishing because it is Mother's day, so he visits his parents.
Argument C: Ralph cannot visit his parents, because it is a leap year, so they are on vacation.

We say that an argument B defeats an argument A iff B is a reason against A.

In the above example, Ralph goes fishing because it is Sunday. It is also Mother's day, so Ralph should visit his parents instead of going fishing. But it is a leap year, his parents are supposed to be on vacation. Therefore, Ralph cannot visit his parents. Thus Ralph goes fishing. Argument A is finally accepted.

An argumentation framework consists of a set of arguments and the defeat relations among those arguments.

Definition 2.1. *An* argumentation framework *is a pair* (Ar, def) *where* Ar *is a finite set of arguments and* $def \subseteq Ar \times Ar$.

We say that argument A *defeats* argument B iff $(A, B) \in def$.

An argumentation framework is essentially a directed graph in which the nodes represent arguments and the arrows represent a defeat relation. For example, the argumentation framework of the "Ralph goes fishing" example is shown in Figure 2.1. In this example, argument A is defeated by argument B. So it seems that A should not be accepted which means Ralph does not go fishing that day. When we see argument C which is a defeater of B and is acceptable because C does not have any defeater, B becomes unacceptable. It makes A acceptable again. Hence, in this framework, A and B are acceptable which means that Ralph goes fishing that day.

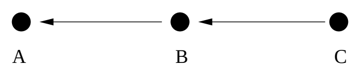


Figure 2.1: Arguments, attacks and reinstatement

2.2 Three-Step Argumentation Process

Nowadays, much research on the topic of argumentation is based on the abstract argumentation theory of Dung [40]. The central concept in this work is that of an *argumentation framework*, which, as just explained, is a directed graph in which the arguments are represented as nodes and the defeat relation is represented by the arrows. Given such a graph, one can then examine the question which set(s) of arguments can be accepted: answering this question corresponds to defining an *argumentation semantics*. Various proposals have been formulated in this respect, and in the current section we describe some of the mainstream approaches. It is, however, important to keep in mind that the issue of argumentation semantics is only one specific aspect (although an important one) in the overall theory of formal argumentation. For instance, if one wants to use argumentation theory for the purpose of (nonmonotonic) entailment, one can distinguish three steps (see Figure 2.2). First of all, one would use an underlying knowledge base to generate a set of arguments and determine in which ways these arguments defeat each other (step 1). The result is an argumentation framework, to be represented as a directed graph in which the internal structure of the arguments, as well as the nature of the defeat relation has been abstracted away. Based on this argumentation framework, the next step is to determine the sets of arguments that can be accepted, using a pre-defined criterion called an argumentation semantics (step 2). After the set(s) of accepted arguments have been identified, one then has to identify the set(s) of accepted conclusions (step 3), for which there exist various approaches.

2.2.1 Step 1: constructing the argumentation framework

Given a knowledge base, the question becomes how to construct the associated argumentation framework.

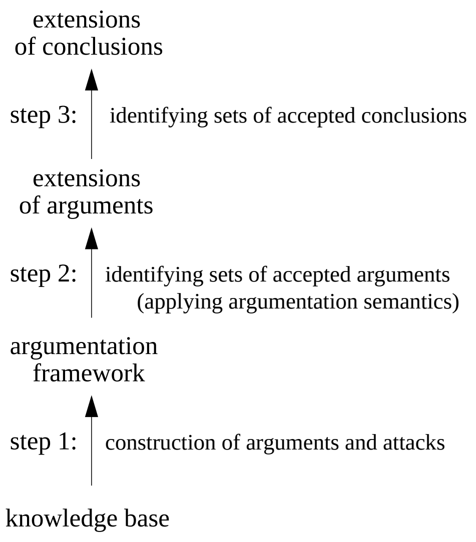

extensions of conclusions
step 3:
identifying sets of accepted conclusions
extensions of arguments
step 2:
identifying sets of accepted arguments (applying argumentation semantics)
argumentation framework
step 1:
construction of arguments and attacks
knowledge base

Figure 2.2: Argumentation for inference

Constructed Arguments

In order to illustrate how to construct Dung-style abstract argumentation frameworks from knowledge bases, we treat an argumentation formalism introduced by Caminada and Amgoud [31].

In the following, let $\mathcal{L}$ be a set of literals.

Arguments consist of strict or defeasible rules [57, 68, 95].

Definition 2.2 (Strict and defeasible rules ([31])). *Let $\varphi_1, \ldots, \varphi_n, \varphi \in \mathcal{L}$ $(n \geq 0)$.*

- *A* strict rule *is of the form $\varphi_1, \ldots, \varphi_n \rightarrow \varphi$.*
- *A* defeasible rule *is of the form $\varphi_1, \ldots, \varphi_n \Rightarrow \varphi$.*

$\varphi_1, \ldots, \varphi_n$ are called the *antecedents* of the rule and φ its *consequent*. A strict rule of the form $\varphi_1, \ldots, \varphi_n \rightarrow \varphi$ indicates that if $\varphi_1, \ldots, \varphi_n$ hold, then without exception it holds that φ. A defeasible rule of the form $\varphi_1, \ldots, \varphi_n \Rightarrow \varphi$ indicates that if $\varphi_1, \ldots, \varphi_n$ hold, then *usually* it holds that φ.

Definition 2.3 (Closure of a set of formulas (Definition 5 in [31])). *Let $\mathcal{P} \subseteq \mathcal{L}$. The closure of $\mathcal{P}$ under the set $\mathcal{S}$ of strict rules, denoted $\mathcal{C}l_{\mathcal{S}}(\mathcal{P})$, is the smallest set such that:*

- *$\mathcal{P} \subseteq \mathcal{C}l_{\mathcal{S}}(\mathcal{P})$.*
- *if $\phi_1, \ldots, \phi_n \rightarrow \psi \in \mathcal{S}$ and $\phi_1, \ldots, \phi_n \in \mathcal{C}l_{\mathcal{S}}(\mathcal{P})$ then $\psi \in \mathcal{C}l_{\mathcal{S}}(\mathcal{P})$.*

The consistency of a set in $\mathcal{L}$ is defined according to classical negation.

Definition 2.4 (Consistent set (Definition 6 in [31])). *Let $\mathcal{P} \subseteq \mathcal{L}$. $\mathcal{P}$ is consistent iff $\nexists\ \psi, \varphi \in \mathcal{P}$ such that $\psi = \neg\varphi$.*

Definition 2.5. *A defeasible theory $\mathcal{T}$ is a pair $(\mathcal{S}, \mathcal{D})$ where $\mathcal{S}$ is a set of strict rules and $\mathcal{D}$ is a set of defeasible rules.*

Arguments can be built from a defeasible theory. The conclusion of an argument is returned by a function `Conc` which is the consequence of the root rule of the argument. `Sub` returns all sub-arguments of the argument and functions `StrictRules` and `DefRules` return all the strict rules and the defeasible rules respectively.

Definition 2.6 (Argument (Definition 7 in [31])). *Let $\mathcal{T} = (\mathcal{S}, \mathcal{D})$ be a defeasible theory. An argument A is:*

- $A_1, \ldots, A_n \rightarrow \psi$ $(n \geq 0)$ *if* $A_1, \ldots, A_n$ *are arguments such that there exists a strict rule* $r \in \mathcal{S}$ *and* $r = \texttt{Conc}(A_1), \ldots, \texttt{Conc}(A_n) \rightarrow \psi$,

  $\texttt{Conc}(A) = \psi$,

  $\texttt{Sub}(A) = \texttt{Sub}(A_1) \cup \ldots \cup \texttt{Sub}(A_n) \cup \{A\}$,

  $\texttt{TopRule}(A) = r$,

  $\texttt{DefRules}(A) = \texttt{DefRules}(A_1) \cup \ldots \cup \texttt{DefRules}(A_n)$.

- $A_1, \ldots, A_n \Rightarrow \psi$ $(n \geq 0)$ *if* $A_1, \ldots, A_n$ *are arguments such that there exists a defeasible rule* $r \in \mathcal{D}$ *and* $r = \texttt{Conc}(A_1), \ldots, \texttt{Conc}(A_n) \Rightarrow \psi$,

  $\texttt{Conc}(A) = \psi$,

  $\texttt{Sub}(A) = \texttt{Sub}(A_1) \cup \ldots \cup \texttt{Sub}(A_n) \cup \{A\}$,

  $\texttt{TopRule}(A) = r$,

  $\texttt{DefRules}(A) = \texttt{DefRules}(A_1) \cup \ldots \cup \texttt{DefRules}(A_n) \cup \{\texttt{Conc}(A_1), \ldots, \texttt{Conc}(A_n) \Rightarrow \psi\}$.

Let $\mathcal{A}rgs$ be the set of all arguments that can be built from $\mathcal{T}$ and let $A, A' \in \mathcal{A}rgs$.

- *A' is a* subargument *of A iff* $A' \in \texttt{Sub}(A)$.
- *A' is a* direct subargument *of A iff* $A' \in \texttt{Sub}(A)$, $A' \neq A$, $\nexists A'' \in \mathcal{A}rgs$ *such that* $A'' \in \texttt{Sub}(A)$ *and* $A' \in \texttt{Sub}(A'')$, $A \neq A''$ *and* $A' \neq A''$.
- *A is an* atomic argument *iff* $\nexists A' \in \mathcal{A}rgs$, $A' \neq A$ *and* $A' \in \texttt{Sub}(A)$.
- *The* depth *of A (*$\mathrm{depth}(A)$*) is* 1 *if A is an atomic argument, or else* $1 + \mathrm{depth}(A')$ *where A' is a direct subargument of A such that* $\mathrm{depth}(A')$ *is maximal.*

We extend `Sub` to a set of arguments, i.e., $\texttt{Sub}(\mathcal{A}rgs) = \texttt{Sub}(A_1) \cup \ldots \cup \texttt{Sub}(A_n)$ where $A_1, \ldots, A_n \in \mathcal{A}rgs$.

An argument is strict if it is constructed only by strict rules, otherwise it is defeasible.

Definition 2.7 (Strict argument and defeasible argument (Definition 8 in [31])). *An argument A is*

- *strict if* $\texttt{DefRules}(A) = \emptyset$;
- *defeasible if* $\texttt{DefRules}(A) \neq \emptyset$;

An argument can defeat arguments in two different ways: undercutting and rebutting.

The definition of undercutting, taken from [31], applies the objectification operator $(\ulcorner \ldots \urcorner)$ introduced by Pollock. The idea is to translate a meta-level expression (in our case: a rule) to an object-level expression (in our case: an element of $\mathcal{L}$) [69, 71]. Undercutting an argument means that there is a defeasible rule in the argument that is claimed not to be applicable.

Definition 2.8 (Undercutting (Definition 10 in [31])). *Argument A* undercuts *argument B (on B') iff* $\texttt{Conc}(A) = \neg\lceil B''_1, \ldots, B''_n \Rightarrow \psi\rceil$ *for some* $B' \in \texttt{Sub}(B)$ *of the form* $B''_1, \ldots, B''_n \Rightarrow \psi$.

Rebutting an argument means that there is a contrary conclusion of the conclusion of a defeasible rule in the argument so that the conclusion of the defeasible rule is argued against.

Definition 2.9 (Rebutting (Definition 15 in [31])). *Argument A* rebuts *argument B (on B') iff* $\texttt{Conc}(A) = \neg\varphi$ *for some* $B' \in \texttt{Sub}(B)$ *of the form* $B''_1, \ldots, B''_n \Rightarrow \varphi$.

Using the notions of rebut and undercut, one can then subsequently define the notion of defeat.

Definition 2.10 (Defeat (Definition 16 in [31])). *Argument A* defeats *argument B iff A rebuts B or A undercuts B.*

Overall, given a defeasible theory $(\mathcal{S}, \mathcal{D})$, one can construct the associated argumentation framework by applying Definition 2.6 and Definition 2.10.

Definition 2.11 (Argumentation framework). *An* abstract argumentation framework *AF built from a defeasible theory $\mathcal{T}$ is a pair (Ar, def) such that:*

- *Ar is the set of arguments on the basis of $\mathcal{T}$ as defined by Definition 2.6,*
- *def is the relation on Ar given by Definition 2.10.*

Definition 2.11 completes the first step in the overall argumentation process: constructing an argumentation framework given a knowledge base.

2.2.2 Step 2: applying abstract argumentation semantics

Given the argumentation framework as provided at the end of step 1 (Definition 2.11) the next question then becomes how to determine the associated sets of arguments that can collectively be accepted. Follow Dung's approach, determining these sets (extensions) is done without looking at the logical content of the arguments.

Definition 2.12 (defense / conflict-free).
Let $AF = (Ar, def)$ be an argumentation framework, $A \in Ar$ and $\mathcal{A}rgs \subseteq Ar$.
We define A^+ as $\{B \in Ar \mid A \ def\ B\}$ and $\mathcal{A}rgs^+$ as $\{B \in Ar \mid A \ def\ B$ for some $A \in \mathcal{A}rgs\}$.
We define A^- as $\{B \in Ar \mid B \ def\ A\}$ and $\mathcal{A}rgs^-$ as $\{B \in Ar \mid B \ def\ A$ for some $A \in \mathcal{A}rgs\}$.
$\mathcal{A}rgs$ is conflict-free *iff $\mathcal{A}rgs \cap \mathcal{A}rgs^+ = \emptyset$.*
$\mathcal{A}rgs$ defends *an argument A iff $A^- \subseteq \mathcal{A}rgs^+$.*
We define $F_{AF} : 2^{Ar} \to 2^{Ar}$ as the function such that: $F_{AF}(\mathcal{A}rgs) = \{A \mid A$ is defended by $\mathcal{A}rgs\}$.

When only one argumentation framework is concerned, F is used as the shortening of F_{AF}.

Definition 2.13 (acceptability semantics). *Let (Ar, def) be an argumentation framework. A conflict-free set $\mathcal{A}rgs \subseteq Ar$ is called*

- *an* admissible set *iff $\mathcal{A}rgs \subseteq F(\mathcal{A}rgs)$.*

- *a* complete extension *iff* $\mathcal{A}rgs = F(\mathcal{A}rgs)$.

- *a* grounded extension *iff* $\mathcal{A}rgs$ *is a minimal complete extension.*

- *a* preferred extension *iff* $\mathcal{A}rgs$ *is a maximal complete extension.*

- *a* stable extension *iff* $\mathcal{A}rgs$ *is a complete extension that defeats every argument in* $Ar \backslash \mathcal{A}rgs$.

- *a* semi-stable extension *iff* $\mathcal{A}rgs$ *is a complete extension such that* $\mathcal{A}rgs \cup \mathcal{A}rgs^{+}$ *is maximal.*

2.2.3 Step 3: determining the sets of justified conclusions

Depending on the particular abstract argumentation semantics, step 2 provides zero or more extensions of arguments. However, what one is often interested in for practical purposes are not so much the arguments themselves, but the *conclusions* supported by these arguments. That is, for each set (extension) of arguments, one needs to identify the associated set (extension) of conclusions.

Definition 2.14. *Let* $\mathcal{A}rgs$ *be a set of arguments whose structure complies with Definition 2.6. We define* $\texttt{Concs}(\mathcal{A}rgs)$ *as* $\{\texttt{Conc}(A) \mid A \in \mathcal{A}rgs\}$.

Definition 2.14 makes it possible to refer to the extensions of conclusions under various argumentation semantics. For instance, the extensions of conclusions under preferred semantics are simply the associated conclusions (Definition 2.14) of each preferred extension of arguments.

The following proposition states that if an argument is in a given extension, then all its sub-arguments are also in that extension.

Proposition 2.1 (Proposition 6.1 in [78]). *Let* (Ar, def) *be an argumentation framework and* E *any of its extensions under a given semantics subsumed by complete semantics. Then for all* $A \in E$, *if* $A' \in \texttt{Sub}(A)$ *then* $A' \in E$.

The justified conclusions [31] are conclusions that are supported by at least one argument in each extension.

Definition 2.15 (Justified conclusions (Definition 12 in [31])). *Let* (Ar, def) *be an argumentation framework, and* $\{E_1, \ldots, E_n\} (n \geq 1)$ *be its set of extensions under a given semantics subsumed by complete semantics.*

- $\texttt{Concs}(E_i) = \{\texttt{Conc}(A) \mid A \in E_i\}$ $(1 \leq i \leq n)$.

- Output $= \bigcap_{i=1,\ldots,n} \texttt{Concs}(E_i)$. Output *is the set of justified conclusions under the given semantics.*

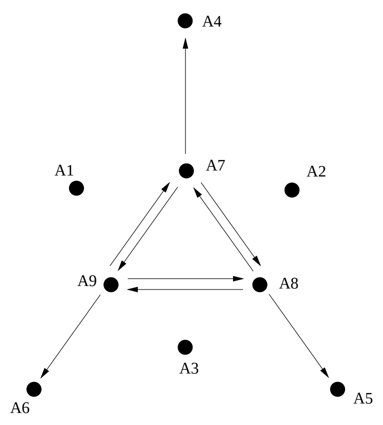


Figure 2.3: The argumentation framework generated by defeasible theory $\mathcal{T}$.

2.2.4 An example of the three step procedure

To illustrate the three step procedure of applying argumentation theory for the purpose of non-monotonic entailment, consider the example of a defeasible theory $\mathcal{T} = (\mathcal{S}, \mathcal{D})$:

$\mathcal{S} = \{\rightarrow jw; \quad \rightarrow mw; \quad \rightarrow sw; \quad mt, st \rightarrow \neg jt; \quad jt, st \rightarrow \neg mt; \quad jt, mt \rightarrow \neg st\}$,

$\mathcal{D} = \{jw \Rightarrow jt; \quad mw \Rightarrow mt; \quad sw \Rightarrow st\}$.

One can interpret this example as follows. John, Mary and Suzy want to go cycling in the countryside ($\rightarrow jw; \quad \rightarrow mw; \quad \rightarrow sw$). They have a tandem bicycle, on which they all want to grab a seat ($jw \Rightarrow jt, \quad mw \Rightarrow mt; \quad sw \Rightarrow st$). However, since the tandem only has two seats, they cannot all three be on it ($mt, st \rightarrow \neg jt; \quad jt, st \rightarrow \neg mt; \quad jt, mt \rightarrow \neg st$). Using Definition 2.6, one can construct the following arguments:

$A_1 :\rightarrow jw$
$A_2 :\rightarrow mw$
$A_3 :\rightarrow sw$
$A_4 : A_1 \Rightarrow jt$
$A_5 : A_2 \Rightarrow mt$
$A_6 : A_3 \Rightarrow st$
$A_7 : A_5, A_6 \rightarrow \neg jt$
$A_8 : A_4, A_6 \rightarrow \neg mt$
$A_9 : A_4, A_5 \rightarrow \neg st$

Using the notion of defeat specified in Definition 2.10, one obtains the argumentation framework of Figure 2.3.

Here, the grounded extension is $\{A_1, A_2, A_3\}$, yielding the associated set of conclusions $\{jw, mw, sw\}$. There are three preferred extensions (that are also stable and semi-stable): $\{A_1, A_2, A_3, A_5, A_6, A_7\}$, $\{A_1, A_2, A_3, A_4, A_6, A_8\}$ and $\{A_1, A_2, A_3, A_4, A_5, A_9\}$, yielding three associated sets of conclusions: $\{jw, mw, sw, \neg jt, mt, st\}$, $\{jw, mw, sw, jt, \neg mt, st\}$, and $\{jw, mw, sw, jt, mt, \neg st\}$.

2.3 Argumentation Semantics

Given an argumentation framework, one can examine which arguments can be accepted according to various semantics. The traditional approach is the extensions approach, which identifies the set of arguments that are accepted. A different way of defining argumentation semantics than the traditional extensions approach is the labellings approach [23, 32, 71, 53, 92]. The labellings approach identifies the set of arguments that are accepted, rejected and the set of arguments that are left undecided.

2.3.1 Extension Based Semantics

In this section, we treat five definitions according to which extensions of arguments can be determined.

Given an argumentation framework, different sets of arguments can be accepted according to various argument based semantics such as complete, grounded, preferred and stable semantics [40] and semi-stable semantics [25].

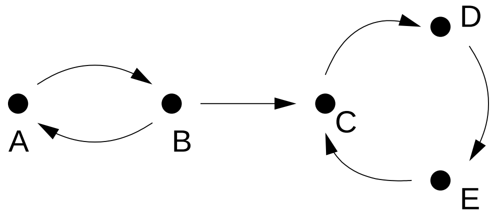


Figure 2.4: An argumentation framework

Complete Semantics

First, we discuss complete semantics. A complete extension is a set of arguments that is conflict-free and defends exactly itself.

Definition 2.16 (Complete extension (Definition 8 in [40])). *Let (Ar, def) be an argumentation framework. A conflict-free set $\mathcal{A}rgs \subseteq Ar$ is called a* complete *extension iff $\mathcal{A}rgs = F(\mathcal{A}rgs)$.*

Example 2.1. *In Figure 2.4, $\{A\}$, $\{B, D\}$ and $\emptyset$ are complete extensions since they are conflict-free and defend exactly themselves.*

Proposition 2.2. *Let $AF = (Ar, def)$ be an argumentation framework and $A \in Ar$. A is in at least one complete extension iff it is in at least one admissible set.*

The validity of Proposition 2.2 can be seen as follows. Since every complete extension is also an admissible set, it follows that if A is in a complete extension, it is also in an admissible set. Furthermore, if A is in an admissible set, then from [40] it follows that A is also in a preferred extension, and every preferred extension is also a complete extension.

Grounded Semantics

Complete semantics has a property that there may exist more than one complete extension. Then whether an argument is in every complete extension becomes relevant. Grounded semantics is defined according to this principle. If an argument is in every complete extension then it is in the grounded extension.

Definition 2.17 (Grounded extension (Definition 7 and Theorem 2 in [40]))**.** *Let (Ar, def) be an argumentation framework and $\mathcal{A}rgs \subseteq Ar$. $\mathcal{A}rgs$ is called a* grounded *extension iff $\mathcal{A}rgs$ is a minimal (with respect to set-including) complete extension.*

Example 2.2. *In Figure 2.4, $\emptyset$ is the grounded extension, since $\emptyset$ is the minimal complete extension.*

Proposition 2.3. *Let $AF = (Ar, \mathit{def})$ be an argumentation framework and $A \in Ar$. A is in all complete extensions iff A is in the grounded extension.*

The validity of Proposition 2.3 can be seen as follows. Since the grounded extension is a complete extension, it follows that if an argument is in every complete extension, it is also in the grounded extension. Furthermore, since the grounded extension is the smallest complete extension, it follows that if an argument is in the grounded extension, it is also in every complete extension.

Preferred Semantics

Grounded semantics is a very sceptical approach which is a disadvantage in situations that we do not need very sceptical results. Preferred semantics is an approach which can be used when people need a more credulous approach. Preferred extensions are maximal complete extensions.

Definition 2.18 (Preferred extension (Definition 4 in [40]))**.** *Let (Ar, def) be an argumentation framework and $\mathcal{A}rgs \subseteq Ar$. $\mathcal{A}rgs$ is called a* preferred *extension iff $\mathcal{A}rgs$ is a maximal (with respect to set-including) complete extension.*

Example 2.3. *In Figure 2.4, there exist two preferred extensions $\{A\}$ and $\{B, D\}$ since they are complete extensions and they are not subsets of any other complete extension.*

Stable Semantics

Stable semantics is one of the oldest semantics for argumentation and non-monotonic reasoning. A set of arguments is a stable extension if and only if it defeats each argument that is not an element of it.

Definition 2.19 (Stable extension (Definition 5 in [40]))**.** *Let (Ar, def) be an argumentation framework and $\mathcal{A}rgs \subseteq Ar$. $\mathcal{A}rgs$ is called a* stable *extension iff $\mathcal{A}rgs^+ = Ar \backslash \mathcal{A}rgs$.*

From the above definition, it follows that a stable extension is conflict-free because otherwise it would hold that $\mathcal{A}rgs \cap \mathcal{A}rgs^+ \neq \emptyset$ and therefore $\mathcal{A}rgs^+ \neq Ar \backslash \mathcal{A}rgs$. Furthermore a stable extension is a complete extension because it defends exactly itself. A stable extension is an admissible set since a stable extension is a complete extension and a complete extension is also an admissible set. A stable extension is also a preferred extension, because any strict superset of a stable extension is not conflict-free.

There are different ways to characterize a stable extension.

Proposition 2.4 (Proposition 1 in [33])**.** *Let (Ar, def) be an argumentation framework. The following statements, describing the concept of stable semantics, are equivalent:*

1. *$\mathcal{A}rgs$ defeats exactly the arguments in $Ar \backslash \mathcal{A}rgs$.*

2. *$\mathcal{A}rgs$ is a conflict-free set that defeats each argument in $Ar \backslash \mathcal{A}rgs$.*

3. *$\mathcal{A}rgs$ is an admissible set that defeats each argument in $Ar \backslash \mathcal{A}rgs$.*

4. *$\mathcal{A}rgs$ is an complete extension that defeats each argument in $Ar \backslash \mathcal{A}rgs$.*

5. *$\mathcal{A}rgs$ is a preferred extension that defeats each argument in $Ar \backslash \mathcal{A}rgs$.*

Example 2.4. *In Figure 2.4, there is one stable extension $\{B, D\}$ since B defeats A and C, D defeats E.*

It is possible that there exists no stable extension. For instance, there is no stable extension in the argumentation framework in Figure 2.5.

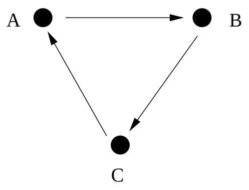


Figure 2.5: An argumentation framework without stable extension

For complete, grounded and preferred semantics, irrelevant arguments cannot influence whether an argument is in the extension or not. However, in stable semantics, irrelevant argument can influence whether an argument is in a stable extension or not.

Definition 2.20 (Relevant (Definition 5 in [25])). *Let (Ar, def) be an argumentation framework. An argument $A \in Ar$ is relevant with respect to an argument $B \in Ar$ iff there exists an undirected path between A and B.*

Example 2.5. *Consider the argumentation framework in Figure 2.6. Argument A is not relevant with respect to B, C and D. However, argument A is the reason why there is no stable extension containing B and D.*

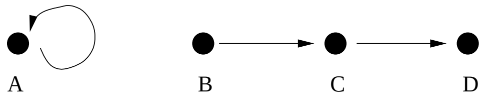


Figure 2.6: Stable semantics does not satisfy relevance.

Semi-stable Semantics [25]

For each finite argumentation framework there always exists a grounded extension and at least one preferred extension. But it is possible that there exists no stable extension in a finite argumentation framework. The idea of semi-stable semantics is that instead of requiring empty `undec`, semi-stable semantics requires minimal `undec`. Then for each finite argumentation framework, there always exists at least one semi-stable extension.

Definition 2.21 (Semi-stable extension (Definition 4 in [25]))**.** *Let (Ar, def) be an argumentation framework and $\mathcal{A}rgs \subseteq Ar$. $\mathcal{A}rgs$ is called a* semi-stable *extension iff $\mathcal{A}rgs$ is a complete extension where $\mathcal{A}rgs \cup \mathcal{A}rgs^+$ is maximal.*

Example 2.6. *In Figure 2.4, there is one semi-stable extension $\{B, D\}$ because it is a complete extension and $\{B, D\} \cup \{A, C, E\}$ is maximal.*

From Example 2.6, we see that if there exists one stable extension then all semi-stable extensions are stable extensions because $\mathcal{A}rgs \cup \mathcal{A}rgs^+$ is equal to Ar which is maximal.

These above semantics can be seen as restricted cases of complete semantics; an overview is provided in the Figure 2.7. The facts that every stable extension is also a semi-stable extension and that every semi-stable extension is also a preferred extension has been proved in [25]. The facts that every preferred extension is also a complete extension and that the grounded extension is also a complete extension have been stated in [40].

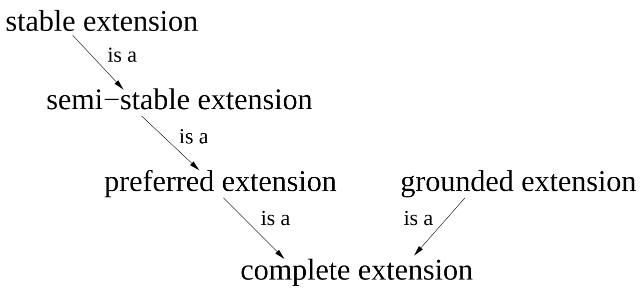


Figure 2.7: An overview of the different semantics

2.3.2 Argument Labelling

The concepts of admissibility, as well as those of complete, grounded, preferred, stable and semi-stable semantics were originally stated in terms of sets of arguments. It is equally well possible, however, to express this concept in terms of *argument labellings*. The approach of (argument) labellings has been used by Pollock [71], by Jakobovits and Vermeir [53] and by Verheij [92] , and has recently been extended by Caminada [23], Vreeswijk [96] and Verheij [93]. The idea of a labelling is to associate with each argument exactly one label, which can either be `in`, `out` or `undec`. The label `in` indicates that the argument is explicitly accepted, the label `out` indicates that the argument is explicitly rejected, and the label `undec` indicates that the status of the argument is undecided, meaning that one abstains from an explicit judgment whether the argument is `in` or `out`.

Definition 2.22 (Labelling (Definition 4 in [23]))**.** *A* labelling *is a function $\mathcal{L} : Ar \longrightarrow \{\texttt{in}, \texttt{out}, \texttt{undec}\}$.*

We write $\texttt{in}(\mathcal{L})$ for $\{A \mid \mathcal{L}(A) = \texttt{in}\}$, $\texttt{out}(\mathcal{L})$ for $\{A \mid \mathcal{L}(A) = \texttt{out}\}$ and $\texttt{undec}(\mathcal{L})$ for $\{A \mid \mathcal{L}(A) = \texttt{undec}\}$. We say that an argument A is *legally* `in` iff $\mathcal{L}(A) = \texttt{in}$ and all the defeaters of A are labelled `out`. We say that an argument A is *legally* `out` iff $\mathcal{L}(A) = \texttt{out}$ and there exists a defeater of A which is labelled `in`. We say that an argument A is *legally* `undec` iff $\mathcal{L}(A) = \texttt{undec}$ and there is no defeater of A that is labelled `in` and not all the defeaters of A are labelled `out`.

Definition 2.23 (Definition 6 in [32]). *Let $\mathcal{L}$ be a labelling of argumentation framework (Ar, def) and $A \in Ar$. We say that:*

1. *A is* legally $\texttt{in}$ *iff* $\mathcal{L}(A) = \texttt{in}$ *and* $\forall B \in Ar : (B \ def \ A \supset \mathcal{L}(B) = \texttt{out})$
2. *A is* legally $\texttt{out}$ *iff* $\mathcal{L}(A) = \texttt{out}$ *and* $\exists B \in Ar : (B \ def \ A \wedge \mathcal{L}(B) = \texttt{in})$.
3. *A is* legally $\texttt{undec}$ *iff* $\mathcal{L}(A) = \texttt{undec}$
   and $\neg\forall B \in Ar : (B \ def \ A \supset \mathcal{L}(B) = \texttt{out})$
   and $\neg\exists B \in Ar : (B \ def \ A \wedge \mathcal{L}(B) = \texttt{in})$.

We say that an argument A is *illegally* $\texttt{in}$ iff $\mathcal{L}(A) = \texttt{in}$ but A is not legally $\texttt{in}$. We say that an argument A is *illegally* $\texttt{out}$ iff $\mathcal{L}(A) = \texttt{out}$ but A is not legally $\texttt{out}$. We say that an argument A is *illegally* $\texttt{undec}$ iff $\mathcal{L}(A) = \texttt{out}$ but A is not legally $\texttt{undec}$.

Definition 2.24 (Admissible labelling (Definition 12 in [32])). *Let (Ar, def) be an argumentation framework and $\mathcal{L} : Ar \longrightarrow \{\texttt{in}, \texttt{out}\}$ be a partial function. We say that $\mathcal{L}$ is an* admissible labelling *iff it satisfies the following:*

- *each argument that is labelled* $\texttt{in}$ *is legally* $\texttt{in}$.
- *each argument that is labelled* $\texttt{out}$ *is legally* $\texttt{out}$.

The idea of a *complete labelling* [23, 32] is that for a labelling to be reasonable, one should be able to give reasons for each argument one accepts (all defeaters are rejected), for each argument one rejects (it has at least one defeater that is accepted) and for each argument one abstains from expressing an explicit opinion about (there are insufficient grounds to accept it and insufficient grounds to reject it). This is made formal in the following definition.

Definition 2.25 ([32]). *Let $\mathcal{L}ab$ be a labelling of argumentation framework (Ar, def). We say that $\mathcal{L}ab$ is a* complete labelling *iff for each $A \in Ar$ it holds that:*

1. *If* $\mathcal{L}ab(A) = \texttt{in}$ *then* $\forall B \in Ar : (B \ def \ A \supset \mathcal{L}ab(B) = \texttt{out})$
2. *If* $\mathcal{L}ab(A) = \texttt{out}$ *then* $\exists B \in Ar : (B \ def \ A \wedge \mathcal{L}ab(B) = \texttt{in})$.
3. *If* $\mathcal{L}ab(A) = \texttt{undec}$ *then*
   $\neg\forall B \in Ar : (B \ def \ A \supset \mathcal{L}ab(B) = \texttt{out})$ *and*
   $\neg\exists B \in Ar : (B \ def \ A \wedge \mathcal{L}ab(B) = \texttt{in})$.

The grounded labelling *is a complete labelling $\mathcal{L}$ where* $\texttt{in}(\mathcal{L})$ *is minimal (w.r.t. set inclusion).*
A preferred labelling *is a complete labelling $\mathcal{L}$ where* $\texttt{in}(\mathcal{L})$ *is maximal (w.r.t. set inclusion).*
A stable labelling *is a complete labelling $\mathcal{L}$ where* $\texttt{undec}(\mathcal{L}) = \emptyset$.
A semi-stable labelling *is a complete labelling $\mathcal{L}$ where* $\texttt{in}(\mathcal{L}) \cup \texttt{out}(\mathcal{L})$ *is maximal (w.r.t. set inclusion).*

Similar to stable extensions, stable labellings might not exist for some argumentation frameworks.

From Definition 2.25 we can see that the complete labellings are used as bases for describing other argument labellings in abstract argumentation. Figure 2.8 shows the relation among those argument labellings.

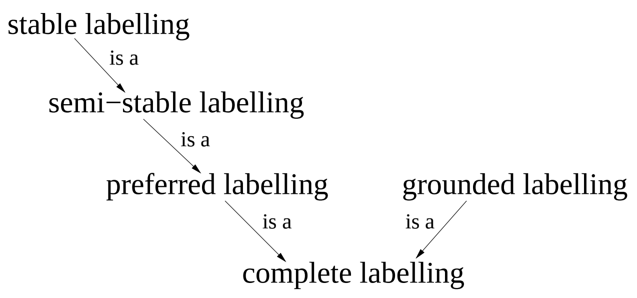


Figure 2.8: An overview of the different argument labelings

Example 2.7. *Consider the argumentation framework in Figure 2.4. Here* $\emptyset, \{(A, \texttt{in}), (B, \texttt{out})\}, \{(A, \texttt{out}), (B, \texttt{in})\}$ *and* $\{(A, \texttt{out}), (B, \texttt{in}), (C, \texttt{out}), (D, \texttt{in}), (E, \texttt{out})\}$ *are examples of admissible labellings. Only* $\emptyset, \{(A, \texttt{in}), (B, \texttt{out}), (C, \texttt{undec}), (D, \texttt{undec}), (E, \texttt{undec})\}$ *and* $\{(A, \texttt{out}), (B, \texttt{in}), (C, \texttt{out}), (D, \texttt{in}), (E, \texttt{out})\}$ *are examples of complete labellings.* $\emptyset$ *is the grounded labelling. Only* $\{(A, \texttt{in}), (B, \texttt{out}), (C, \texttt{undec}), (D, \texttt{undec}), (E, \texttt{undec})\}$ *and* $\{(A, \texttt{out}), (B, \texttt{in}), (C, \texttt{out}), (D, \texttt{in}), (E, \texttt{out})\}$ *are preferred labellings (because they are maximal w.r.t. set inclusion). Only* $\{(A, \texttt{out}), (B, \texttt{in}), (C, \texttt{out}), (D, \texttt{in}), (E, \texttt{out})\}$ *is a stable labelling and a semi-stable labelling.*

It can be proved that the various types of labellings correspond to the various kinds of argument semantics.

Theorem 2.1 (Theorem 1. in [26]). *Let* (Ar, def) *be an argumentation framework and let* $\mathcal{A}rgs \subseteq Ar$. $\mathcal{A}rgs$ *is an admissible set iff there exists an admissible labelling* $\mathcal{L}$ *with* $\texttt{in}(\mathcal{L}) = \mathcal{A}rgs$.

Based on Theorem 2.1, the following theorem is proved by Caminada [26].

Theorem 2.2 (Theorem 1. and Theorem 3. in [26]). *Let* (Ar, def) *be an argumentation framework and let* $\mathcal{A}rgs \subseteq Ar$.
$\mathcal{A}rgs$ *is a complete extension iff there exists a complete labelling* $\mathcal{L}$ *with* $\texttt{in}(\mathcal{L}) = \mathcal{A}rgs$.
$\mathcal{A}rgs$ *is a grounded extension iff there exists a grounded labelling* $\mathcal{L}$ *with* $\texttt{in}(\mathcal{L}) = \mathcal{A}rgs$.
$\mathcal{A}rgs$ *is a preferred extension iff there exists a preferred labelling* $\mathcal{L}$ *with* $\texttt{in}(\mathcal{L}) = \mathcal{A}rgs$.
$\mathcal{A}rgs$ *is a stable extension iff there exists a stable labelling* $\mathcal{L}$ *with* $\texttt{in}(\mathcal{L}) = \mathcal{A}rgs$.
$\mathcal{A}rgs$ *is a semi-stable extension iff there exists a semi-stable labelling* $\mathcal{L}$ *with* $\texttt{in}(\mathcal{L}) = \mathcal{A}rgs$.

As stated in [23, 32], complete labellings coincide with complete extensions in the sense of [40].

We now define two functions that, given an argumentation framework, allow a set of arguments to be converted to a labelling and vice versa. The function $\texttt{Ext2Lab}_{(Ar,def)}$ takes a set of arguments (possibly an extension) and converts it to a labelling. The function $\texttt{Lab2Ext}_{(Ar,def)}$ takes a labelling and converts it to a set of arguments (possibly an extension). Since a labelling is a function, it is possible to represent the labelling as a set of pairs.

Definition 2.26 (Definition 6 in [23]). *Let* (Ar, def) *be an argumentation framework,* $\mathcal{A}rgs \subseteq Ar$ *such that* $\mathcal{A}rgs$ *is conflict-free, and* $\mathcal{L} : Ar \rightarrow \{\texttt{in}, \texttt{out}, \texttt{undec}\}$ *a labelling. We define* $\texttt{Ext2Lab}_{(Ar,def)}(\mathcal{A}rgs)$ *as* $\{(A, \texttt{in}) \mid A \in \mathcal{A}rgs\} \cup \{(A, \texttt{out}) \mid \exists A' \in \mathcal{A}rgs :$

$A' \mathit{def}\, A\} \cup \{(A, \texttt{undec}) \mid A \notin \mathcal{A}rgs \wedge \neg \exists A' \in \mathcal{A}rgs : A' \mathit{def}\, A\}$. We define $\texttt{Lab2Ext}_{(Ar,def)}(\mathcal{L})$ as $\{A \mid (A, \texttt{in}) \in \mathcal{L})\}$.

When the associated argumentation framework is clear, we sometimes simply write `Ext2Lab` and `Lab2Ext` instead of $\texttt{Ext2Lab}_{(Ar,def)}$ and $\texttt{Lab2Ext}_{(Ar,def)}$.

It can be proved that the various types of labellings correspond to the various kinds of argument semantics [23, 26].

Theorem 2.3 (Theorem 1 in [23]). *Let (Ar, def) be an argumentation framework. If $\mathcal{L}$ is a complete labelling then $\texttt{Lab2Ext}(\mathcal{L})$ is a complete extension. If $\mathcal{A}rgs$ is a complete extension then $\texttt{Ext2Lab}(\mathcal{A}rgs)$ is a complete labelling.*

Proof. Please refer to [24]. □

When the domain and the range of `Lab2Ext` are restricted to complete labellings and complete extensions, and the domain and the range of `Ext2Lab` are restricted to complete extensions and complete labellings, then the resulting functions (call them $\texttt{Lab2Ext}^r$ and $\texttt{Ext2Lab}^r$) are bijective and are each other's inverse.

Theorem 2.4 (Theorem 2 in [23]). *Let $\texttt{Lab2Ext}^r_{(Ar,def)} : \{\mathcal{L} \mid \mathcal{L}$ is a complete labelling of $(Ar, \mathit{def})\} \rightarrow \{\mathcal{A}rgs \mid \mathcal{A}rgs$ is a complete extension of $(Ar, \mathit{def})\}$ be a function defined by $\texttt{Lab2Ext}^r_{(Ar,def)}(\mathcal{L}) = \texttt{Lab2Ext}_{(Ar,def)}(\mathcal{L})$.*
Let $\texttt{Ext2Lab}^r_{(Ar,def)} : \{\mathcal{A}rgs \mid \mathcal{A}rgs$ is a complete extension of $(Ar, \mathit{def})\} \rightarrow \{\mathcal{L} \mid \mathcal{L}$ is a complete labeling of $(Ar, \mathit{def})\}$ be a function defined by $\texttt{Ext2Lab}^r_{(Ar,def)}(\mathcal{A}rgs) = \texttt{Ext2Lab}_{(Ar,def)}(\mathcal{A}rgs)$. The functions $\texttt{Lab2Ext}^r_{(Ar,def)}$ and $\texttt{Ext2Lab}^r_{(Ar,def)}$ are bijective and are each other's inverse.

Proof. Please refer to [24]. □

From Theorem 2.4 it follows that complete labellings and complete extensions stand in a one-to-one relationship to each other. In essence, complete labellings and complete extensions are different ways to describe the same concept.

2.4 Grounded Game and Preferred Game

Essentially, a complete labelling can be seen as a subjective but reasonable position that an agent can take with respect to which arguments are accepted, rejected or undecided. In each such position the agent can use its own position to defend itself if questioned. It is possible to disagree with a position, but at least the position is internally coherent. The set of all complete labellings thus stands for all possible and reasonable positions an agent can take. Then for a particular argument (say A), there are two questions we are interested in:

1. Is there at least one reasonable position where A is accepted? That is, is there at least one complete labelling $\mathcal{L}$ such that $\mathcal{L}(A) = \texttt{in}$.

2. Is A accepted in every reasonable position? That is, does it hold that for each complete labelling $\mathcal{L}$, $\mathcal{L}(A) = \texttt{in}$.

The first question refers to the issue of credulous acceptance; the second question refers to the issue of sceptical acceptance.

2.4.1 Credulous Acceptance

In this section we treat a reformulated version of Vreeswijk and Prakken's argument game for preferred semantics [97]. Although there also exist other argument games for preferred semantics, like [35], we have chosen [97] for its relative simplicity and its easy adaptability to work with argument labellings. Our reformulation is aimed at slightly simplifying Vreeswijk and Prakken's approach, and also to allow for its easy adaptation to stable semantics, which will be treated in the next section.

In order to determine whether an argument (say A) is in an admissible set (say $\mathcal{A}rgs$), one can examine whether there exists an admissible labelling ($\mathcal{L}$) with $\mathcal{L}(A) = \texttt{in}$ (Theorem 2.2). The discussion game is then aimed at providing this admissible labelling. The game can be described as follows:

- proponent (P) and opponent (O) take turns; P begins.
- each move of O is a defeater of some (not necessarily the directly preceding) previous argument of P.
- each move of P (except the first one) is a defeater of the directly preceding argument of O.
- O is not allowed to repeat its own moves, but may repeat P's moves.
- P is not allowed to repeat O's moves, but may repeat its own moves.

The game is won by the proponent iff the opponent cannot move anymore. It is won by the opponent iff the proponent cannot move anymore, or if the opponent manages to repeat one of the proponent's moves.

Formally, the preferred discussion game can be described as follows.

Definition 2.27 (Preferred discussion game [33]). *Let (Ar, def) be an argumentation framework. A* preferred discussion *is a sequence of moves $[M_1, M_2, \ldots, M_n]$ $(n \geq 0)$ such that:*

- *each M_i $(1 \leq i \leq n)$ where i is odd (which is called a* proponent move*) is of the form $\texttt{in}(A)$, where $A \in Ar$.*
- *each M_i $(1 \leq i \leq n)$ where i is even (which is called an* opponent move*) is of the form $\texttt{out}(A)$ where $A \in Ar$.*
- *For each opponent move $M_i = \texttt{out}(A)$ $(1 \leq i \leq n)$ there exists a proponent move $M_j = \texttt{in}(B)$ $(j < i)$ where A defeats B.*
- *For each proponent move $M_i = \texttt{in}(A)$ $(3 \leq i \leq n)$ it holds that $M_{i-1} = \texttt{out}(B)$ where A defeats B.*
- *For each opponent move $M_i = \texttt{out}(A)$ $(1 \leq i \leq n)$ there does not exist an opponent move $M_j = \texttt{out}(A)$ with $j < i$.*
- *For each proponent move $M_i = \texttt{in}(A)$ $(1 \leq i \leq n)$ there does not exist an opponent move $M_j = \texttt{out}(A)$ with $j < i$.*

A preferred discussion $[M_1, M_2, \ldots, M_n]$ is said to be finished *iff there exists no M_{n+1} such that $[M_1, M_2, \ldots, M_n, M_{n+1}]$ is a preferred discussion, or if M_n is an opponent move of the form* `out`(A) *for which there exists a proponent move M_i ($1 \leq i \leq n$) of the form* `in`(A)*. A finished discussion is won by the proponent if the last move is a proponent move, and is won by the opponent if the last move is an opponent move.*

One good way to view the discussion game is as the proponent trying to build the set of `in`-labelled arguments and the opponent trying to build the set of `out`-labelled arguments. As an example, consider the argumentation framework illustrated in Figure 2.9.

Here, the proponent can win the discussion game for argument D in the following way:
P: `in`(D) "I have an admissible labelling in which D is labelled `in`."
O: `out`(C) "Then in your labelling it must also be the case that D's defeater C is labelled `out` (otherwise D would not be legally `in`). Based on which grounds?"
P: `in`(B) "C is labelled `out` because B is labelled `in`."
O: `out`(A) "Then in your labelling it must also be the case that B's defeater A is labelled `out` (otherwise B would not be legally `in`). Based on which grounds?"
P: `in`(B) "A is labelled `out` because B is labelled `in`."

The above example illustrates the need for the proponent to be able to repeat its own arguments. At the same time, the proponent should not be allowed to repeat the opponent's arguments, since these have to be labelled `out`, so the proponent cannot claim them to be labelled `in`.

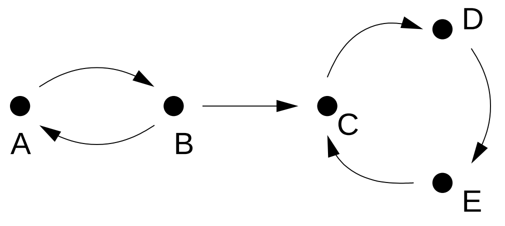


Figure 2.9: Same argumentation framework in Figure 2.4

The argumentation framework of Figure 2.9 can also be used for an example of a game won by the opponent:
P: `in`(E) "I have an admissible labelling in which E is labelled `in`."
O: `out`(D) "Then in your labelling it must be the case that E's defeater D is labelled `out`. Based on which grounds?"
P: `in`(C) "D is labelled `out` because C is labelled `in`."
O: `out`(E) "Then in your labelling it must be the case that C's defeater E is labelled `out`. This contradicts with your earlier claim that E is labelled `in`."

The above example illustrates that it is necessary to allow the opponent to repeat the proponent's arguments. Nevertheless, it would not be useful for the opponent to repeat its own arguments, since the reason why the particular argument is labelled out has already been explained by the proponent, so it makes no sense to ask again.

As a last illustration of the dialogue game for admissibility, consider the argumentation framework of Figure 2.10. Argument C is not in an admissible set. It is illustrative to see

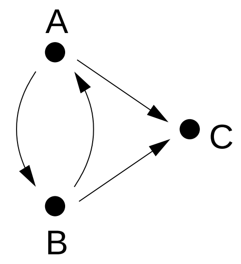


Figure 2.10: An argumentation framework with floating defeat

what happens if the proponent tries to defend C.
P: in(C) "I have an admissible labelling in which C is labelled in."
O: out(A) "Then in your labelling C's defeater A must be labelled out. Based on which grounds?"
P: in(B) "A is labelled out because B is labelled in."
O: out(B) "But from the fact that you hold C to be in, it follows that C's defeater B must be labelled out. This contradicts with your earlier claim that B is labelled in."

The above example illustrates the need for the opponent to be able to respond not only to the immediately preceding move, but to any past move of the proponent; in the example, out(B) is a response to in(C). This is because for an argument to be legally in, *all* its defeaters have to be out, so the opponent may need to respond to the proponent's argument with more than one move. At the other hand, it is not needed for the proponent to be able to respond to any move but the previous one. This is because for an argument to be legally out it is sufficient to have one defeater that is labelled in. If the proponent simply gives this defeater, then there is no need to give any additional defeaters. Hence, the proponent only needs to respond to the directly preceding move of the opponent.

The correlation between the thus described discussion game and the concept of admissibility can be described as follows.

Theorem 2.5 (Theorem 3 in [33]). *Let (Ar, def) be an argumentation framework and $A \in Ar$. There exists an admissible labelling $\mathcal{L}$ with $\mathcal{L}(A) =$ in iff there exists an admissible discussion for A that is won by the proponent.*

Since the concept of admissible labellings coincides with the concept of an admissible set (theorem 2.2) it holds that an argument is in an admissible set iff it is possible for the proponent to win the discussion for it. Moreover, it holds that an argument is in an admissible set iff it is in a preferred extension (or, alternatively, iff it is labelled in in a preferred labelling). Hence, the discussion game can be used as a basis for proof procedures for credulous preferred. Furthermore, it holds that an argument is in an admissible set iff it is in a complete extension (Proposition 2.2) (or, alternatively, iff it is labelled in in a complete labelling). Hence, the discussion game can also be used as a basis for proof procedures for credulous complete.

Vreeswijk and Prakken show that the discussion game can also be used as a basis for the decision problem of sceptical preferred semantics.

2.4.2 Sceptical Acceptance

The next thing is to determine which arguments are in every complete extension. Since the grounded extension is the intersection of all complete extensions, determing whether an argument is in every complete extension can be done by examining whether the argument is in the grounded extension.

The discussion whether an argument is in every complete extension (a discussion under grounded semantics, or simply g-discussion) can be described as follows [80, 75, 20]:

- proponent (P) and opponent (O) take turns; P begins
- each move of O is a defeater of the directly preceding argument of P.
- each move of P (except the first one) is a defeater of the directly preceding argument of O
- O is allowed to repeat its own moves.
- P is not allowed to repeat its own moves.

Formally, the grounded discussion game can be described as follows.

Definition 2.28 (Grounded discussion [80, 75, 20]). *Let (Ar, def) be an argumentation framework. A* grounded discussion *is a sequence of moves $[M_1, M_2, \ldots, M_n]$ $(n \geq 0)$ such that:*

- *each M_i $(1 \leq i \leq n)$ where i is odd (which is called a* proponent move*) is of the form $\texttt{in}(A)$, where $A \in Ar$.*
- *each M_i $(1 \leq i \leq n)$ where i is even (which is called an* opponent move*) is of the form $\texttt{out}(A)$ where $A \in Ar$.*
- *For each opponent move $M_i = \texttt{out}(A)$ $(2 \leq i \leq n)$ it holds that $M_{i-1} = \texttt{in}(B)$ where A defeats B.*
- *For each proponent move $M_i = \texttt{in}(A)$ $(3 \leq i \leq n)$ it holds that $M_{i-1} = \texttt{out}(B)$ where A defeats B.*
- *For each proponent move $M_i = \texttt{in}(A)$ $(1 \leq i \leq n)$ there does not exist a proponent move $M_j = \texttt{in}(A)$ $(1 \leq j \leq n)$ with $j \neq i$.*

A grounded discussion $[M_1, M_2, \ldots, M_n]$ is said to be finished *iff there exists no M_{n+1} such that $[M_1, M_2, \ldots, M_n, M_{n+1}]$ is a grounded discussion. A finished discussion is won by the proponent if the last move is a proponent move, and is won by the opponent if the last move is an opponent move.*

To determine whether an argument is in the grounded extension, it is not enough that there exists at least one g-discussion that is won by the proponent. What is need is the existence of a *winning strategy*. The idea of a winning strategy is to provide a road map for which moves the proponent should play, taking into account every possible counter-move of the opponent.

Definition 2.29 (Winning strategy (Definition 2.10 in[20])). *Let A be an argument. A winning strategy for A is a tree of which the nodes are associated with arguments, the root associated with A, such that:*

1. *each path from the root to a leaf corresponds to a discussion won by the proponent,*
2. *each opponent-move on such a path has exactly one child and,*
3. *the children of each proponent-move consist of all possible defeaters of the proponent-move.*

Theorem 2.6 (Theorem 2.4 in [20]). *Let (Ar, def) be an argumentation framework and $A \in Ar$. A is labelled* `in` *in every complete labelling iff the proponent has winning strategy for A in the discussion using g-discussion.*

Proof. Please refer to [20]. □

2.5 Other Semantics and Rationality Postulates

Caminada and Amgoud [31] specify the rationality postulates of *direct consistency*, *indirect consistency* and *closure*. In this section we restate the definitions of these three postulates.

The idea of closure is that the conclusions of an argumentation framework should be complete. If there exists a strict rule $a \to b$ and a is justified then b should be justified too.

An argumentation framework satisfies closure if its set of justified conclusions, as well as the set of conclusions supported by each extension are closed.

Definition 2.30 (Closure (Postulate 1 in [31])). *Let $\mathcal{T} = (\mathcal{S}, \mathcal{D})$ be a defeasible theory and AF be an argumentation framework (Definition 2.11) built from $\mathcal{T}$. Let $E_1, \dots, E_n$ be extensions of AF under a given semantics. Let* Output *be its set of justified conclusions such that* $\text{Output} = \bigcap_{i=1}^{n} E_i$. *AF satisfies closure iff:*

(1) $\texttt{Concs}(E_i) = \mathcal{C}l_{\mathcal{S}}(\texttt{Concs}(E_i))$ *for each* $1 \leq i \leq n$.

(2) $\text{Output} = \mathcal{C}l_{\mathcal{S}}(\text{Output})$.

The following Proposition shows that if the different sets of conclusions of the extensions are closed, then the set Output is also closed.

Proposition 2.5 (Proposition 4 in [31]). *Let $\mathcal{T} = (\mathcal{S}, \mathcal{D})$ be a defeasible theory and AF be an argumentation framework built from $\mathcal{T}$. Let $E_1, \dots, E_n$ be extensions of AF under a given semantics. Let* Output *be its set of justified conclusions. If* $\texttt{Concs}(E_i) = \mathcal{C}l_{\mathcal{S}}(\texttt{Concs}(E_i))$ *for each* $1 \leq i \leq n$ *then* $\text{Output} = \mathcal{C}l_{\mathcal{S}}(\text{Output})$.

An argumentation framework satisfies direct consistency if its set of justified conclusions is consistent and each set of conclusions corresponding to its extension is consistent.

Definition 2.31 (Direct consistency (Postulate 2 in [31])). *Let $\mathcal{T} = (\mathcal{S}, \mathcal{D})$ be a defeasible theory and AF be an argumentation framework built from $\mathcal{T}$. Let $E_1, \dots, E_n$ be extensions of AF under a given semantics. Let* Output *be its set of justified conclusions. AF satisfies direct consistency iff:*

(1) $\texttt{Concs}(E_i)$ *is consistent (according to Definition 2.4) for each* $1 \leq i \leq n$.

(2) Output *is consistent (according to Definition 2.4).*

If the closure of the set of justified conclusions is consistent and the closure of conclusions of each extension is consistent, then the argumentation framework satisfies indirect consistency.

Definition 2.32 (Indirect consistency (Postulate 3 in [31]))**.** *Let* $\mathcal{T} = (\mathcal{S}, \mathcal{D})$ *be a defeasible theory and AF be an argumentation framework built from* $\mathcal{T}$. *Let* $E_1, \ldots, E_n$ *be extensions of AF under a given semantics. Let* Output *be its set of justified conclusions. AF satisfies indirect consistency iff:*

(1) $\mathcal{C}l_{\mathcal{S}}(\texttt{Concs}(E_i))$ *is consistent (according to Definition 2.4) for each* $1 \leq i \leq n$.

(2) $\mathcal{C}l_{\mathcal{S}}(\text{Output})$ *is consistent (according to Definition 2.4).*

The following proposition shows that if the closure of conclusions of each extension is consistent, then the closure of the justified conclusions is consistent.

Proposition 2.6 (Proposition 5 in [31])**.** *Let* $\mathcal{T} = (\mathcal{S}, \mathcal{D})$ *be a defeasible theory and AF be an argumentation framework built from* $\mathcal{T}$. *Let* $E_1, \ldots, E_n$ *be extensions of AF under a given semantics. Let* Output *be its set of justified conclusions. If* $\mathcal{C}l_{\mathcal{S}}(\texttt{Concs}(E_i))$ *is consistent for each* $1 \leq i \leq n$ *then* $\mathcal{C}l_{\mathcal{S}}(\text{Output})$ *is consistent.*

Proposition 2.7 (Proposition 7 in [31])**.** *Let* $\mathcal{T} = (\mathcal{S}, \mathcal{D})$ *be a defeasible theory and AF be an argumentation framework built from* $\mathcal{T}$. *If AF satisfies closure and direct consistency, then it satisfies indirect consistency.*

It follows that if indirect consistency is satisfied by an argumentation framework, then the argumentation framework also satisfies direct consistency.

If we construct an argumentation framework and apply any admissibility-based semantics (semantics that each extension of arguments is an admissible set) according to Step 1, 2, 3 in Section 2.2, the set of conclusions yielded satisfies the postulates of direct consistency, indirect consistency and closure. For instance, in the example of Figure 2.3, the extension of conclusions of grounded semantics and the extensions of conclusions of preferred semantics satisfy direct consistency, indirect consistency and closure.

Steps 1,2 and 3 are not orthogonal. Some semantics do not work even with restricted rebut [31] (Definition 2.9) (naive, stage). Naive extensions are the maximal conflict-free sets of arguments. A stage extension [92, 15] is conflict-free set $\mathcal{A}rgs$ where $\mathcal{A}rgs \cup \mathcal{A}rgs^+$ is maximal. In the case of the argumentation framework in Figure 2.3, $\{A_1, A_2, A_3, A_4, A_5, A_6\}$ is a naive extension. Then it yields a set of conclusions $\{jw, mw, sw, jt, mt, st\}$. It puts three persons on a two-person tandem. Since $Cl_{\mathcal{S}}(\{jw, mw, sw, jt, mt, st\}) = \{jw, mw, sw, jt, mt, st, \neg jt, \neg mt, \neg st\}$, it does not satisfy indirect consistency and closure. Therefore naive semantics does not satisfy rationality postulates. The example in Figure 2.3 is not a counter example against rationality postulates under stage semantics because the stage extensions of the argumentation framework in Figure 2.3 coincide with the preferred extensions. Let us see the argumentation framework in Figure 2.11. $\{A_1, A_2, A_3, A_4, A_5, A_6\}$ is a stage extension. Then the set of conclusions yielded by $\{A_1, A_2, A_3, A_4, A_5, A_6\}$ violates indirect consistency and closure. So stage semantics does not satisfy rationality postulates by applying Step 1,2 and 3.

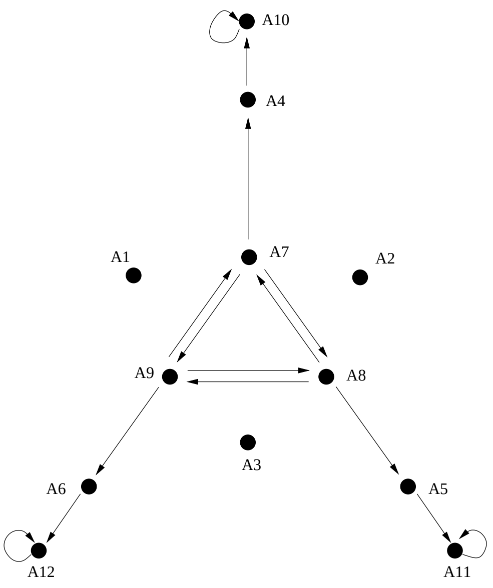


Figure 2.11: A counter-example for stage semantics.

Some purely abstract approaches do not guarantee that instantiations of them satisfy the postulates. Since the constructions of arguments are not concerned, the consistency of the collective conclusions cannot be guaranteed. Amgoud and Besnard [2, 3] and Gorogiannis and Hunter [50] also discuss these issues in the context of deductive instantiations of Dung's abstract argumentation frameworks. Since in Dung's abstract argumentation framework the defeat relation is binary, Amgoud and Besnard indicate that ternary (or more) conflicts can cause inconsistency by using an example.

Value-based argumentation is another example that does not guarantee satisfiability of postulates of instantiations. In value-based argumentation, each argument is assigned a value. Different audiences can assign different preferences regarding these values. They delete defeat relations before applying semantics. If the value of an argument A is smaller than the value of an argument B then the defeat from A to B will be deleted. Assume that an audience puts the values of A_4, A_5, A_6 above other arguments in Figure 2.3. Then the defeats from A_7 to A_4, the defeat from A_8 to A_5 and the defeat from A_9 to A_6 are deleted. Moreover, the audience puts the value of A_9 above the value of A_8 and the value of A_8 above the value of A_7. Then the defeat from A_7 to A_8, the defeat from A_8 to A_9 and the defeat from A_7 to A_9 are removed from the argumentation framework. Now we obtain the following argumentation framework in Figure 2.12.

In the argumentation framework in Figure 2.12, $\{A_1, A_2, A_3, A_4, A_5, A_6, A_9\}$ is a preferred extension. The corresponding set of conclusions is $\{jw, mw, sw, jt, mt, st, \neg st\}$. Apparently, it does not satisfy direct consistency, indirect consistency and closure.

2.6 Related Work

In this chapter, we have introduced only some classical semantics of argumentation, including admissible, complete, grounded, preferred, stable and semi-stable semantics, since they are the basic and the relatively easy semantics. However, there are also other semantics, e.g., ideal [41], eager [27], stage [92], and CF2 [12, 11, 10] semantics. A set of arguments is

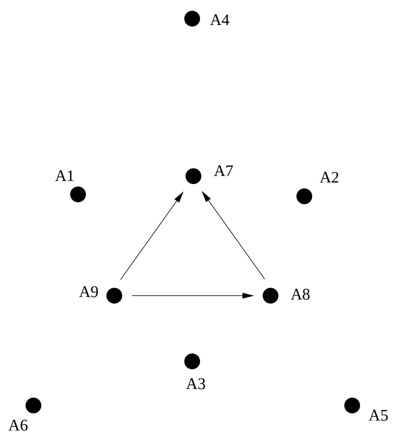


Figure 2.12: A counter-example for value-based argumentation.

ideal if and only if it is an admissible subset of each preferred extension. The *ideal extension* can then be defined as the (unique) maximal ideal set of arguments. Ideal semantics is proposed as a unique-status approach (each argumentation framework possesses exactly one extension) less skeptical than the grounded extension. Eager semantics is a unique-status semantics which is more credulous than ideal semantics. The eager extension is the greatest (with respect to set-inclusion) admissible set that is a subset of each semi-stable extension. The eager extension is a superset of the ideal extension and a complete extension. A stage extension is a stable extension of a maximal subgraph of the argumentation framework that has at least one stable extension. The idea of stage semantics is to take a maximal consistent subset of an inconsistent knowledge base. CF2 semantics is an instance of the SCC-recursive semantics [12] which is introduced in order to solve particular problems arising for AFs with odd-length cycles. CF2 semantics can symmetrically treat odd- and even-length cycles and ensures that attacks from self-defeating arguments have no influence on the selection of other arguments to be included in an extension. Ambiguity propagating and ambiguity blocking are two key defeasible logics. Governatori *et al.* [51] provide the first ambiguity blocking Dung-like argumentation system. The original defeasible logic is ambiguity blocking, but an ambiguity-propagating defeasible logic [8] can also be defined. The only argumentation semantics corresponds to ambiguity blocking is CF2. Governatori *et al.* [51] show that the ambiguity propagating variant is characterized by Dung's grounded semantics.

The basic argumentation framework of Dung has been extended into a number of ways, e.g., preference-based argumentation frameworks [5], value-based argumentation frameworks [14], bipolar argumentation frameworks [36] and collective argumentation [19]. In a preference-based argumentation framework, a preference relation over the arguments is imposed. The defeat relation of a preference-base argumentation framework is decided by a preference relation and a conflict relation together. A value-based argumentation framework extends Dung's abstract argumentation framework by adding a non-empty set of values V, a function *val* which maps from arguments to elements of V and a preference relation *valpref* on $V \times V$. If V contains a single value, the value-based argumentation framework becomes a standard argumentation framework. If each argument maps to a different value, we have a preference-based argument framework. An abstract bipolar argumentation framework is an extension of Dung's abstract argumentation framework by adding a new relation "support" between arguments. The support relation is totally independent of the attack relation. So in a bipolar argumentation framework, there are two relations between arguments. Collective argumentation is a 'disjunctive' generalization of Dung's argumentation theory. In this formalism, the defeat relation holds between sets of arguments, which makes it more complex than the forms of argumentation definable in Dung's argumentation theory. Although the above argumentation frameworks have their own advantages, in this thesis we focus on the standard argumentation frameworks since they are the basis of many extended argumentation frameworks.

From the example in Figure 2.12, we can see that abstract frameworks cannot prevent violation of the postulates. We also find that in reality arguments are hardly completely abstract. Informally, arguments consist of reasons that support claims during some kind of discussion. In AI, these discussions can be formalised in frameworks for structured argumentation such as ASPIC. The inference rules of ASPIC framework can be used as general inference rules. Argument schemes technically have the form of an inference rule [78] and argument schemes are defeasible. So argument schemes are defeasible inference rules [79, 74]. We can instantiate the underlying argument schemes by modelling the arguments in the form of inference rules. Therefore, the ASPIC framework is suitable for modelling reasoning with argument schemes [78].

2.7 Summary

In this chapter, we have restated some of the standard semantics for formal argumentation, including Dung's notations of complete, grounded, preferred and stable semantics, as well as notation of Caminada's semi-stable semantics. We have treated these semantics both in the original extension-based form, as well as in the form of argument labellings. For grounded and preferred semantics, we have introduced two discussion games. The grounded discussion game can determine whether an argument is in the grounded extension or not. The preferred discussion game can determine whether an argument is in at least one preferred extension. We have illustrated the three-step argumentation process by using an instantiation of Dung's abstract argumentation framework, which is introduced by Caminada and Amgoud [31]. Besides, we have presented three desirable postulates of argumentation system, namely direct consistency, indirect consistency and closure.

Given an argumentation framework, the grounded discussion game and preferred discussion game can determine whether an argument is in a complete extension, the grounded extension or a preferred extension. However, there is no discussion game for stable semantics. In Chapter 3, we fill the gap by presenting a discussion game for argumentation under stable semantics.

Furthermore, the approach of argument labellings provides more information than the traditional extension approach. We want to know all possible labels of an argument so that we can determine whether an argument is strongly accepted, weakly accepted, strongly rejected, weakly rejected or borderline case. In Chapter 4, we introduce a labelling-based justification status of arguments. Admissible labellings correspond to admissible sets, but the relationship is not one-to-one. Each admissible set is associated with one or more admissible labellings. A complete extension is a stronger condition than an admissible set. Moreover, complete labellings and complete extensions are one-to-one related. Therefore, in Chapter 4, we focus on the complete-labellings-based justification status of arguments.

Argumentation can be viewed as logic programming with negation as failure [40]. In

Chapter 5, we show that the complete extensions in abstract argumentation coincide with 3-valued stable models in logic programming.

In Chapter 6, we introduce an argumentation system ASPIC Lite. We show that the ASPIC Lite system does not satisfy the postulates of non-interference and crash-resistance. Then we identify conditions under which the ASPIC Lite system satisfies the postulates of direct consistency, indirect consistency and closure and the postulates of crash-resistance and non-interference.

Chapter 3

A Discussion Game for Credulous Stable Semantics

In this chapter, we present a discussion game for argumentation under stable semantics. Our work is inspired by Vreeswijk and Prakken, who have defined a similar game for preferred semantics. In Section 2.4.1, we restate Vreeswijk and Prakken's work using the approach of argument labellings and then in the current chapter we show how it can be adjusted for stable semantics. The nature of the resulting argument game is somewhat unusual, since stable semantics does not satisfy the property of *relevance*.

3.1 Introduction

Stable semantics, a concept that goes back to the work of von Neumann and Morgenstern [94], is one of the oldest semantics for argumentation and non-monotonic reasoning. Although Dung's landmark paper [40] was partially meant to argue against the use of it, stable semantics has remained an important concept in fields like default logic [86] and logic programming [46, 47]. In this chapter, we answer the research question 1: what is a discussion game for stable semantics? by providing a discussing game for stable semantics in argumentation.

During recent years, several new semantics have been proposed [12, 25, 41, 11]. What makes stable semantics unique, however, are two fundamental properties. First of all, there is the possible absence of stable extensions. When applying stable semantics in, for instance, answer set programming, this can in fact be a desirable property. If one encodes a problem such that the possible solutions correspond with the stable extensions, then the absence of stable extensions indicates the absence of solutions to the original problem. Secondly, stable semantics does not satisfy the property of *relevance* [25]. That is, it is possible for the status of an argument A to be influenced by a totally unrelated argument B. For instance, let (Ar, def) be the argumentation framework in Figure 3.1 where the set of arguments Ar is $\{A, B\}$ and the defeat relation def is $\{(B, B)\}$. Then A and B are totally unrelated in the sense that there does not exist an (undirected) defeat-path between A and B. Yet, the existence of argument B causes argument A not to be credulously accepted.

The invalidity of the property of relevance has implications for the possibilities of defining an argument game. For instance, for grounded and preferred semantics, both of which do satisfy relevance, it is possible to define argument games in which each move is a re-

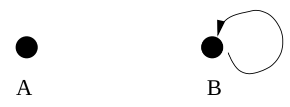


Figure 3.1: Stable semantics does not satisfy relevance.

sponse to a previous move [97, 80, 20]. For stable semantics, however, this is impossible. In the above example, argument B is the reason why argument A is not credulously accepted. Yet, it would be somewhat odd to reply to A with B, since no relation exists between these arguments.

In this chapter, we propose an argument game that can deal with the unique characteristics of stable semantics. The discussion game for credulous acceptance under stable semantics is given in Section 3.2, and an approach for sceptical acceptance under stable semantics is given in Section 3.3. Then, in Section 3.5, we finish with a discussion about some future research topics.

3.2 A Discussion Game for Credulous Stable Semantics

In the current section, we provide a discussion game for credulous stable semantics. Before doing so, it may be illustrative to see why the standard admissibility discussion game does not work for stable semantics. Consider the argumentation framework of Figure 3.2. Even though A is in an admissible set and in a preferred extension ($\{A\}$), A is not in a stable extension. To see why A is in an admissible set, consider the following discussion:

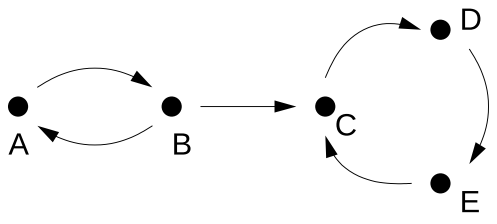


Figure 3.2: A is not in a stable extension.

P: `in`(A) “I have an admissible labelling where A is labelled `in`”
O: `out`(B) “Then in your labelling, argument B must be labelled `out`. Based on which grounds?”
P: `in`(A) “B is labelled `out` because A is labelled `in`”
The point is, however, that once it has been committed that A is labelled `in` and B is labelled `out`, it is impossible anymore to label the remaining arguments such that the final result will be a stable labelling. This can be seen as follows. Suppose C is labelled `in`. Then E must be labelled `out`, so D should be labelled `in`, which means that C would be labelled `out`. Contradiction. Similarly, suppose that C is labelled `out`. Then E must be labelled `in`, so D should be labelled `out`, so C should be labelled `in`. Again, contradiction.

Proposition 2.4 shows that there are many ways to characterize a stable extension. For our purposes, the most useful characterization is that of an admissible set which defeats every argument that is not in it. When one translates this to labellings, one obtains an admissible labelling where each argument is labelled either `in` or `out`, and no argument is left unlabelled.

It appears that a discussion game for stable semantics requires an additional type of move: `question`. To illustrate the role of this new move, imagine a politician being interviewed for TV. At first the discussion may be about financial matters (say, whether the banking system should be nationalized). Then, the discussion may be about the consequences of the politician's opinion ("If you accept to nationalize the banks, then you must reject the possibility to improve healthcare, because there will not be enough money left to do so."). However, at some moment, the interviewer could choose to totally change topic ("By the way, what are your opinions about abortion?"). It is this change of topic that is enabled by the `question` move.

For the discussion game for stable semantics, we use the `question` move to involve those arguments that have never been uttered before so that we can label all the elements of Ar. By questioning an argument (`question`(A)), the opponent asks the proponent to give an explicit opinion on whether A should be labelled `in` or `out`. If the proponent thinks that A should be labelled `in` then it should respond with `in`(A). If the proponent thinks that A should be labelled `out` then it should respond with `in`(B) where B is a defeater of A. The discussion game for stable semantics can thus be described as follows:

- The proponent (P) and opponent (O) take turns. The proponent begins.
- Each move of the opponent is either of the form `out`(A), where A is a defeater of some (not necessarily the directly preceding) move of the proponent, or of the form `question`(A), where A is an argument that has not been uttered in the discussion before (by either the proponent or the opponent).
- The first move of the proponent is of the form `in`(A), where A is the main argument of the discussion. The following moves of the proponent are also of the form `in`(A) although A no longer needs to be the main claim. If the directly preceding move of the opponent is of the form `out`(B) then A is a defeater of B. If the directly preceding move of the opponent is of the form `question`(B) then A is either equal to B or a defeater of B.
- The opponent may not repeat any of its `out` moves.
- The proponent is allowed to repeat its own moves, but may not do an `in`(A) move if the opponent has done some earlier `out`(A) move.

The opponent wins if it is able to do an `out`(A) move and the proponent has done an earlier `in`(A) move, or if the proponent cannot move anymore. The proponent wins if the opponent cannot move anymore.

To illustrate the use of the discussion game, consider the argumentation framework depicted in Figure 3.3.

Suppose the proponent would like to start a discussion about A.
P: `in`(A) "I have a stable labelling in which A is labelled `in`."
O: `out`(B) "Then in your labelling, A's defeater B must be labelled `out`. Based on which grounds?"
P: `in`(A) "B is labelled `out` because A is labelled `in`."

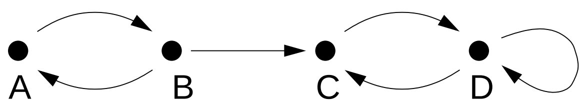


Figure 3.3: A is in a stable extension.

O: question(C) "What about C?"
P: in(C) "C is labelled in."
O: out(D) "Then C's defeater D must be labelled out. Based on which grounds?"
P: in(C) "D is labelled out because C is labelled in."
The proponent wins the discussion, since the opponent cannot move anymore.

The above example also shows that the outcome of a discussion may depend on P's response to a question move. For instance, if P would have replied to question(C) with in(D), then it would have lost the discussion, since O would then do out(D).

As an example of a game that cannot be won by the proponent, consider a game for argument B. This game has to be lost by the proponent since the argumentation framework of Figure 3.3 has only one stable extension: $\{A, C\}$, which does not include B.
P: in(B) "I have a stable labelling in which B is labelled in."
O: out(A) "Then in your labelling, B's defeater A must be labelled out. Based on which grounds?"
P: in(B) "A is labelled out because B is labelled in."
O: question(C) "What about C?"
P: in(D) "C is labelled out because its defeater D is labelled in."
O: out(D) "Then D's defeater D (itself) must be labelled out. Contradiction."
The proponent would still not have won the discussion if it had responded to question(C) with in(C) instead of with in(D). This is because then the opponent would have reacted with out(B) and would therefore still have won the discussion.

Formally, the stable discussion game can be described as follows.

Definition 3.1. *Let (Ar, def) be an argumentation framework and $Ar \neq \emptyset$. A* stable discussion *is a sequence of moves $[M_1, M_2, \ldots, M_n]$ $(n \geq 1)$ such that:*

- *each M_i $(1 \leq i \leq n)$ where i is odd (which is called a* proponent move*) is of the form* in(A)*, where $A \in Ar$.*
- *each M_i $(1 \leq i \leq n)$ where i is even (which is called an* opponent move*) is of the form* out(A) *where $A \in Ar$, or of the form* question(A) *where $A \in Ar$.*
- *For each opponent move $M_i =$* out(A) *$(2 \leq i \leq n)$ there exists a proponent move $M_j =$* in(B) *$(j < i)$ where A defeats B.*
- *For each proponent move $M_i =$* in(A) *$(3 \leq i \leq n)$ it either holds that (1) $M_{i-1} =$* out(B) *where A defeats B, or (2) $M_{i-1} =$* question(B) *where either $A = B$ or A defeats B.*
- *For each opponent move $M_i =$* out(A) *$(1 \leq i \leq n)$ there does not exist an opponent move $M_j =$* out(A) *with $j < i$.*
- *For each opponent move $M_i =$* question(A) *$(1 \leq i \leq n)$ there does not exist any move M_j $(j < i)$ of the form* in(A), out(A) *or* question(A).

- *For each proponent move $M_i =$ `in`(A) $(1 \leq i \leq n)$ there does not exist an opponent move $M_j =$ `out`(A) with $j < i$.*

A stable discussion $[M_1, M_2, \ldots, M_n]$ is said to be finished *iff there exists no M_{n+1} such that $[M_1, M_2, \ldots, M_n, M_{n+1}]$ is a stable discussion, or if M_n is an opponent move of the form* `out`(A) *for which there exists a proponent move M_i $(1 \leq i \leq n)$ of the form* `in`(A)*. A finished stable discussion is won by the proponent if the last move is a proponent move, and is won by the opponent if the last move is an opponent move.*

Definition 3.2. *Let A be an argument. A* stable discussion for A *is a stable discussion with $M_1 =$* `in`(A)*.*

For convenience, in the rest of the thesis, we use the term stable discussions for finished stable discussions.

It turns out that an argument is in at least one stable extension iff the proponent can win the stable discussion game for it.

Theorem 3.1. *Let (Ar, def) be an argumentation framework and $A \in Ar$. There exists a stable labelling $\mathcal{L}$ (Definition 2.25) with $\mathcal{L}(A) =$* `in` *iff there exists a stable discussion for A that is won by the proponent.*

Proof.
"$\Longrightarrow$" Suppose there exists a stable labelling $\mathcal{L}$ with $\mathcal{L}(A) =$ `in`.

At the first step the proponent labels A with `in`. Trivially, it now holds that we have a game in which all `in`-labelled moves are also labelled `in` in the stable labelling $\mathcal{L}$.

We now prove that any unfinished discussion where the proponent does the last move and where all proponent-moves are labelled `in` in $\mathcal{L}$ can be extended to a discussion with an additional opponent move and an additional proponent-move such that the result will again be a discussion in which the proponent does the last move, and all proponent moves are labelled `in` in $\mathcal{L}$.

Let $[M_1, \ldots, M_n]$ be an unfinished discussion where M_n is a proponent move and all proponent moves are labelled `in` in $\mathcal{L}$. From the fact that the discussion is unfinished, it follows that the opponent can do a move M_{n+1} which is either of the form `out`(B), where B is a defeater of some earlier proponent move (say `in`(A)), or of the form `question`(B). In the first case (`out`(B)), it holds that B is labelled `out` in $\mathcal{L}$, because A is labelled `in` in $\mathcal{L}$. It then follows that there exists an argument C which defeats B and is labelled `in` in $\mathcal{L}$, which makes it possible for the proponent to respond with `in`(C). In the second case (`question`(B)), the proponent can either respond with `in`(B) if B is labelled `in` in $\mathcal{L}$, or with `in`(C) if B is labelled `out` in $\mathcal{L}$, where C is a defeater of B. In any case, the resulting discussion will have a proponent move as the last move, and all proponent-moves labelled `in` in $\mathcal{L}$.

From the fact that the argumentation framework is finite, the fact that the opponent cannot repeat its moves and the fact that every unfinished discussion game can always be extended to a discussion game in which the last move is still move by the proponent, it follows that the discussion game can ultimately be extended in such a way that it is won by the proponent.

"$\Longleftarrow$" Suppose there exists a stable discussion game for argument A that is won by the proponent. Let $\mathcal{A}rgs$ be the set of the `in` labelled arguments. $\mathcal{A}rgs$ is conflict-free, otherwise the opponent would have labelled an argument `out` that was labelled `in` by proponent earlier and would have won the game. Furthermore, $\mathcal{A}rgs$ defeats each argument that is not in $\mathcal{A}rgs$. This can be seen as follows. Let $B \notin \mathcal{A}rgs$. Then either (1) the opponent made a move $\texttt{out}(B)$ which means the proponent labelled an argument C `in` such that C defeats B, or (2) the opponent made a move $\texttt{question}(B)$ which followed by $\texttt{in}(C)$ of proponent where C defeats B. In both cases, C is in $\mathcal{A}rgs$, so $\mathcal{A}rgs$ defeats B.

Since $\mathcal{A}rgs$ is conflict-free and defeats each argument not in it, $\mathcal{A}rgs$ is a stable extension. From Theorem 2.2, it follows that there exists a stable labelling with A is labelled `in`. □

For the discussion game for preferred semantics, it is quite straightforward to convert the resulting game to an admissible labelling: $\mathcal{L} = \{(A, \texttt{in}) \mid$ there exists a proponent-move $\texttt{in}(A)\} \cup \{(A, \texttt{out}) \mid$ there exists an opponent-move $\texttt{out}(A)\}$.

For the discussion game for stable semantics, converting the moves of the game to a stable labelling is slightly different. $\mathcal{L} = \{(A, \texttt{in}) \mid$ there exists a proponent-move $\texttt{in}(A)\} \cup \{(A, \texttt{out}) \mid$ there exists an opponent-move $\texttt{out}(A)\} \cup \{(A, \texttt{out}) \mid$ there exists an opponent-move $\texttt{question}(A)$ that was responded to with $\texttt{in}(B)$ where B is a defeater of $A\}$.

There are some possible optimizations for the above mentioned discussion game. As Vreeswijk and Prakken point out, the role of the opponent can also be seen as actually *helping* the proponent to find what it is looking for. If one takes this perspective, then it is quite reasonable to require the opponent to do a `question`-move only when it (temporarily) cannot do an `out` move anymore. There is, however, another way in which the opponent can help the proponent to construct a stable labelling. If the opponent has to do a `question`-move, because it (temporarily) ran out of `out`-moves, then it makes sense for the opponent to try to do a $\texttt{question}(A)$-move such that (1) A is an argument that has a defeater that is `in` (that is, there exists an argument B such that B defeats A and the proponent did an $\texttt{in}(B)$-move in the past) or (2) A is an argument that has all its defeaters `out` (that is, for each argument B such that B defeats A, the opponent did either an $\texttt{out}(B)$-move or a $\texttt{question}(B)$-move at which the proponent did not respond with an $\texttt{in}(B)$-move). In both cases, it is clear how the proponent should respond. In case (1), the proponent should respond with $\texttt{in}(B)$ (where B is a defeater of A that was already found to be `in`; basically, the proponent is repeating one of its earlier moves). In case (2), the proponent will respond with $\texttt{in}(A)$. In general, the opponent could adapt a strategy of trying to select questions that are relatively easy to answer for the proponent. Such a strategy does not influence the correctness and completeness of the discussion game as a proof theory for stable semantics because it just gives the priority to some arguments to be questioned first.

3.3 A Discussion Game for Sceptical Stable Semantics

Vreeswijk and Prakken [97] provide a procedure for determining if an argument is an element of every preferred extension. Their procedure, however, only works for argumentation frameworks where each preferred extension is also a stable extension.

Proposition 3.1 ((Proposition 2 in [97])). *In argument systems where each preferred extension is also stable, an argument is in all preferred extensions iff there exists a preferred discussion for A that is won by the proponent and there exists no preferred extension containing a defeater of A.*

This result in Proposition 3.1 [97] holds only in argumentation frameworks where each preferred extension is also a stable extension because if a preferred extension is not a stable extension then the arguments that are not in the preferred extension might have no defeaters in the preferred extension. So Vreeswijk and Prakken give the following example to show that the result does not hold in general.

Example 3.1. *Consider the argumentation framework in Figure 3.4. There are two preferred extensions $E_1 = \{A, C\}$ and $E_2 = \{D\}$. Since argument A is in E_1, there exists a preferred discussion for A that is won by the proponent. Argument A has only one defeater B and neither E_1 nor E_2 contains B. However, A is not in all preferred extensions since A is not in the preferred extension E_2. Although argument B is not in E_2, it is not defeated by any argument in E_2. This is because that in this argumentation framework, the preferred extension E_2 is not a stable extension.*

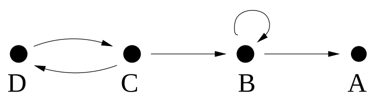


Figure 3.4: An argumentation framework with a preferred extension which is not stable

The discussion procedure for sceptical stable semantics that is proposed in this section does not have the restriction of the sceptical preferred semantics [97]. The idea is that an argument is in each stable extension iff there is no stable extension that contains one of its defeaters. From the definition of stable extensions (Definition 2.19), every argument that is not in a stable extension is defeated by the stable extension. So, if there is no stable extension contains one of the defeaters of an argument A, the defeaters of A are defeated by every stable extension, so A is defended by every stable extension. Therefore A is in every stable extension.

Theorem 3.2. *Let $AF = (Ar, def)$ be an argumentation framework, and let $A \in Ar$. A is an element of each stable extension iff there exists no stable extension containing a defeater of A.*

Proof.
"$\Longrightarrow$": Suppose A is an element of each stable extension. Then there exists no stable extension S that does not contain A. Therefore (since for each argument $A \in Ar$, a stable extension contains either A or a defeater of A), there exists no stable extension that contains a defeater of A.
"$\Longleftarrow$": Suppose there exists no stable extension containing a defeater of A. Then each stable extension does not contain a defeater of A. Therefore (since for each argument $A \in Ar$, a stable extension either contains A or a defeater of A) each stable extension contains A. □

So in order to examine whether an argument A is in each stable extension, one should examine the defeaters of A one by one. If one finds a defeater that is in a stable extension, then the question of whether A is in each stable extension can be answered with "no". If, however, it turns out that each defeater of A is not in any stable extension, then the answer is "yes". Therefore, one can simply apply the (credulous) stable discussion game for each defeater of A, to obtain the answer regarding sceptical stable. For instance, if we want to examine whether the argument C is in a stable extension in the argumentation framework in Figure 3.3, we can apply the credulous stable discussion game to the arguments B and D which are both defeaters of C. The fact is that there exists no stable discussion game for B or D that is won by the proponent. So there exists no stable extension containing B or D. Therefore, argument C is in every stable extension.

3.4 Related Work

There exist some discussion games for credulous argumentation semantics, for instance, the preferred discussion game of Vreeswijk and Prakken [97] for credulous preferred semantics. Our discussion game for stable semantics in this chapter is based on the work of Vreeswijk and Prakken [97]. Our discussion game is not the only approach that can be based on their work. The proof procedures of Dung, Mancarella and Toni for *ideal semantics* [41] can, for instance, also be described in terms of Vreeswijk and Prakken's argument game for preferred semantics. We recall that a set of arguments is *ideal* iff it is an admissible subset of each preferred extension. The *ideal extension* can then be defined as the (unique) maximal ideal set of arguments. It holds that an argument is in the ideal extension iff it is in an admissible set that is not defeated by any admissible set [41]. This means that one can first perform the dialogue game for the argument itself, and then the dialogue game against each argument in the thus obtained admissible set to see whether the main argument is in the ideal extension. Grossi [52] formalised all standard Dung's extension-based semantics in a second-order modal logic and showed that this logic can be used to provide adequate games for all known extension-based semantics. In Grossi's logic-based approach, positions consist of arguments, a formula and a valuation. The player can move by traveling along the defeat relation and selecting sets of arguments. The rules of Grossi's games are determined by the logical structure of the formula. In our dialogue game, positions only consist of arguments. Players move along the defeat relation. The rules are general and work for any argumentation framework. There are also a number of discussion games for sceptical argumentation semantics, for instance, the computational framework of Kakas and Toni [54] and the grounded game of Prakken and Sartor [80] for a particular system, a dialectical proof theory [72] for abstract frameworks and the algorithm of Vreeswijk [96] for grounded semantics.

3.5 Summary and Future Work

In this chapter, we have introduced a discussion game for credulous stable semantics. The discussion game can also be used as a basis for sceptical stable semantics. The stable discussion game is based on the preferred discussion game [97]. There is one more type of move (`question`) of the opponent in the stable game than in the preferred game. The move of `question` helps the stable game reach the arguments that are not uttered yet and are not connected to the arguments which are being discussed so that every argument in an argumentation framework can be discussed. Caminada shows that the preferred discussion game has a Socratic nature [29]. The nature of the stable discussion game is a bit different from the nature of the preferred discussion game since unlike the preferred discussion game, which does not need to construct the entire extension, our stable discussion game has to traverse the whole argumentation framework to check whether an argument is in one stable extension. Therefore we say that the stable discussion game is a Socratic traversal discussion.

A useful feature of most argument games, for instance, the preferred game [97], is that they allow to determine the status of an argument without having to construct the entire extension. This feature applies to games for semantics that satisfy relevance. For example, in Figure 3.1, the preferred discussion game for argument A has only one step: P: `in`(A). After the step P: `in`(A), we don't need to continue this game with discussing the argument B. We can immediately conclude that A is in at least one preferred extension after the first step in this game. Since preferred semantics satisfies the property of relevance and there exists no (undirected) defeat-path between A and B, argument B does not influence the status of argument A. Therefore, in this preferred game, we do not have to consider argument B in order to decide whether A is in one preferred extension. However, stable semantics does not satisfy the property of relevance. From the definition of a stable extension (Definition 2.19), if there exists a stable extension in an argumentation framework then every argument in the argumentation framework is either in the stable extension or is defeated by arguments in the stable extension. Therefore, in order to decide whether an argument is in a stable extension, we have to check the status of every argument in argumentation frameworks. Hence, our discussion game for stable semantics is that in every discussion all arguments are considered. In other words, the whole extension has to be constructed because we have to be sure that the extension constructed during the stable discussion game is a stable extension. We use again the argumentation framework in Figure 3.1 to illustrate why every argument has to be discussed in stable discussion games. There are two arguments in the argumentation framework in Figure 3.1. If the game for stable semantics does not have to define the entire extension then after the step P: `in`(A), argument B would not be discussed. The game stops after argument A is discussed. In this case, the stable discussion game is the same as the preferred discussion game. So the proponent wins the game. According to this discussion game, A is in one stable extension. In fact, $\{A\}$ is not a stable extension because A does not defeat every argument (namely B in this example) that is not in the set of $\{A\}$. Moreover, there exists no stable extension of the argumentation framework (Figure 3.1). Therefore no matter the proponent wins or not, argument A is not in any stable extension. Thus, without discussing the status of argument B, we cannot be sure that the extension that we construct partly is a stable extension. So we cannot decide whether A is in one stable extension without constructing the entire extension either. Hence, it is necessary to go through every argument in an argumentation framework in a stable discussion game.

Instead of going through every argument in an argumentation framework in order to decide whether an argument is in a stable extension, we can use the approach of Baumann [13] which splits an argumentation framework into parts. According to the work of Baumann, the union of the stable extensions of each part is the stable extension of the whole argumentation framework [13]. Then we can apply the stable discussion game only in one part of the argumentation framework which contains the argument we would like to discuss. However, this approach works only for argumentation frameworks that have at least one stable extension. For example, we cannot use this approach for the argumentation framework in Figure 3.1. We can split the argumentation framework in Figure 3.1 into two parts. One is $AF_1 = (\{A\}, \emptyset)$. The other one is $AF_2 = (\{B\}, \{B, B\})$. Then the stable discussion game for A in the argumentation framework AF_1 is won by the proponent. So argument A is in a stable extension in the argumentation framework AF_1. But argument A is not in any stable extension of the whole argumentation framework.

A topic that is currently left open is that of computational complexity. Although it is known that both credulous preferred and credulous stable are NP-complete problems [38], it still has to be examined how this theoretical complexity of the respective problems relates to the actual algorithms for credulous preferred and credulous stable.

Chapter 4

A Justification Status of Arguments

In this chapter, we define a labelling-based justification status of the arguments in an argumentation framework. Our proposal allows for a more fine-grained notion of a justification status than the traditional extensions-based approaches. In particular, we are able to distinguish different levels at which an argument can be accepted or rejected. Our approach is fully compatible with traditional abstract argumentation in the sense that it works on standard argumentation frameworks and can be implemented using existing argumentation-based proof procedures.

4.1 Introduction

The justification status of arguments that we propose in this chapter is based on the notion of a complete labelling (Definition 2.25). One of the main advantages of our proposal is that it allows for a more fine-grained notion of a justification status than is provided by the traditional extensions-based approaches as explained later in this section. In particular, it allows for six distinct justification statuses (strong accept, weak accept, strong reject, weak reject, undetermined border line and determined border line) which correspond with different levels of acceptance and rejection. Furthermore, our proposal works on standard argumentation frameworks and can be implemented using existing argumentation-based proof procedures [97, 65].

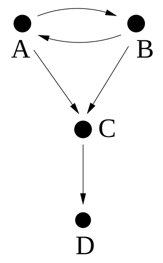


Figure 4.1: What are the possible complete labellings?

The justification status of an argument is the set of labels that are possibly assigned to the argument by complete labellings. We illustrate the justification statuses of arguments by using an example. There are three complete labellings of the argumentation framework in Figure 4.1,

$\mathcal{L}_1 = \{\texttt{in}(A), \texttt{out}(B), \texttt{out}(C), \texttt{in}(D)\}$ (Figure 4.2),
$\mathcal{L}_2 = \{\texttt{out}(A), \texttt{in}(B), \texttt{out}(C), \texttt{in}(D)\}$ (Figure 4.3),
$\mathcal{L}_3 = \{\texttt{undec}(A), \texttt{undec}(B), \texttt{undec}(C), \texttt{undec}(D)\}$ (Figure 4.4).

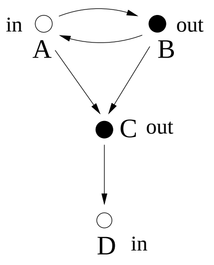


Figure 4.2: $\mathcal{L}_1$

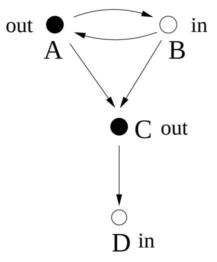


Figure 4.3: $\mathcal{L}_2$

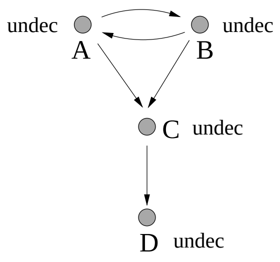


Figure 4.4: $\mathcal{L}_3$

The justification status of argument A is $\{\texttt{in}, \texttt{out}, \texttt{undec}\}$ because $\mathcal{L}_1(A) = \texttt{in}$, $\mathcal{L}_2(A) = \texttt{out}$ and $\mathcal{L}_3(A) = \texttt{undec}$. The justification statuses of B is $\{\texttt{in}, \texttt{out}, \texttt{undec}\}$ because $\mathcal{L}_1(B) = \texttt{out}$, $\mathcal{L}_2(B) = \texttt{in}$ and $\mathcal{L}_3(B) = \texttt{undec}$. The justification statuses of C is $\{\texttt{out}, \texttt{undec}\}$ because $\mathcal{L}_1(C) = \texttt{out}$, $\mathcal{L}_2(C) = \texttt{out}$ and $\mathcal{L}_3(C) = \texttt{undec}$. The justification statuses of D is $\{\texttt{in}, \texttt{undec}\}$ because $\mathcal{L}_1(D) = \texttt{in}$, $\mathcal{L}_2(C) = \texttt{in}$ and $\mathcal{L}_3(D) = \texttt{undec}$. The justification statuses of arguments A, B, C and D are indicated in Figure 4.5.

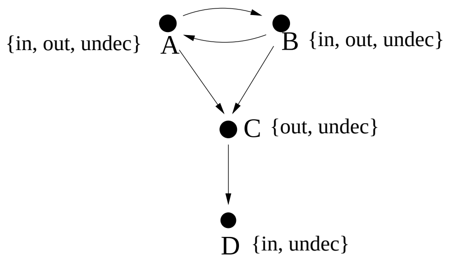


Figure 4.5: The justification statuses based on complete labellings

The notion of the complete labelling which we consider in this chapter is a three-valued semantics with three possible labels `in`, `out` and `undec` which can be assigned to arguments. This justification statuses of arguments cannot only be applied to the approach of the three-valued labelling (Definition 2.22) but also to the traditional extension approach. Since the traditional extension approach only identifies the set of arguments that are accepted, it can be expressed as a two-valued labellings approach in the following way: an argument is labelled `accepted` if it is in the extension, otherwise the argument is labelled `rejected`. The two-valued labelling is a simpler case than the three-valued labelling. In the argumentation framework of Figure 4.1, there are three complete extensions, $\{A, D\}$, $\{B, D\}$ and $\emptyset$. According to these three complete extensions we can label the arguments `accepted` or `rejected`. For the complete extension $\{A, D\}$, argument A and argument D are labelled `accepted` because A and D are in the complete extension, argument B and argument C are labelled `rejected` because they are not in the complete extension (Figure 4.6). For the complete extension $\{B, D\}$, argument B and argument D are labelled `accepted` because B and D are in the complete extension, argument A and argument C are labelled `rejected` because A and C are not in the complete extension (Figure 4.7).

For the complete extension $\emptyset$, since there is no argument in this complete extension, all arguments are labelled `rejected` (Figure 4.8).

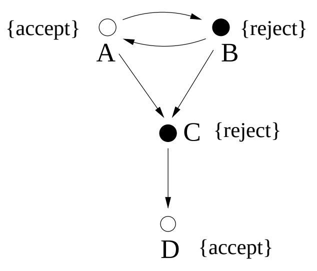


Figure 4.6: $E_1 = \{A, D\}$

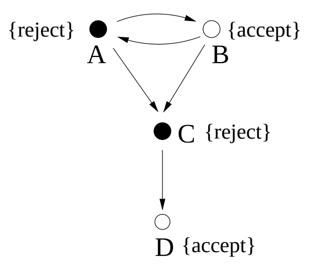


Figure 4.7: $E_2 = \{B, D\}$

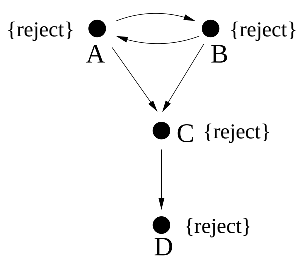


Figure 4.8: $E_3 = \emptyset$

In Figure 4.9, the justification statuses of the arguments are based on the notion of the complete extensions which can be expressed as two-valued labellings.

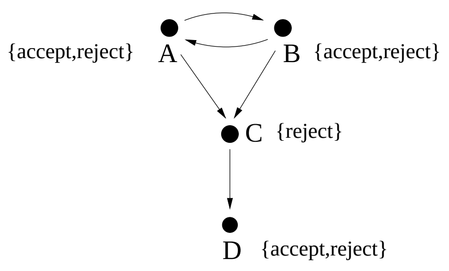


Figure 4.9: The justification statuses based on 2-valued labellings

Since there are two possible labels for the two-valued labelling, there are maximal four possible justification statuses, $\emptyset$, $\{\texttt{accepted}\}$, $\{\texttt{accepted}, \texttt{rejected}\}$ and $\{\texttt{rejected}\}$. Because there always exists at least one complete extension, $\emptyset$ is excluded. So there are three justification statuses for the two-valued labelling.

Although the justification statuses of arguments can be applied to the traditional extension approach, it captures less information than the approach of labellings does since there are only three justification statuses of arguments under the traditional extension approach. Besides, the labelling-based approaches are easier to be understood than the traditional extension-based approaches. Therefore we consider the approach of argument labellings in this chapter. Among the various types of labellings that capture different semantics, admissible labellings and complete labellings are both the bases of defining other argument labellings in abstract argumentation. We choose complete labellings because a complete labelling is a reasonable position that an agent can take and be able to defend itself. In an admissible labelling, every argument could be labelled `undec` making the admissible labelling useless. So in this chapter we propose justification statuses of arguments based on the notion of a complete labelling.

The remaining part of this chapter is organized as follows. We first define the justification status of an argument (Section 4.2) and describe a software implementation that is able to compute this status, given an argumentation framework (Section 4.3). The connection between our approach and the existing argumentation semantics is discussed in Section 4.4. We discuss the justification statuses of conclusions in Section 4.5. We then round up with a discussion of how our notion of a justification state relates to existing well-known approaches.

4.2 A Justification Status of Arguments

In this section we first define the justification statuses of arguments. Then we provide procedures to determine them. Intuitively, the justification status of an argument consists of the set of labels that could reasonably be assigned to the argument.

Definition 4.1 (Justification status of argument). *Let $AF = (Ar, def)$ be an argumentation framework and $A \in Ar$. The justification status of A is the outcome yielded by the function $\mathcal{JS} : Ar \to 2^{\{\texttt{in},\texttt{out},\texttt{undec}\}}$ such that $\mathcal{JS}(A) = \{\mathcal{L}ab(A) \mid \mathcal{L}ab$ is a complete labelling of $AF\}$.*

Given the above definition, one would expect there to be eight (2^3) possible justification statuses, $\emptyset$, $\{\texttt{in}\}$, $\{\texttt{out}\}$, $\{\texttt{undec}\}$, $\{\texttt{in}, \texttt{out}\}$?$\{\texttt{in}, \texttt{undec}\}$, $\{\texttt{out}, \texttt{undec}\}$ and $\{\texttt{in}, \texttt{out}, \texttt{undec}\}$, one for each subset of $\{\texttt{in}, \texttt{out}, \texttt{undec}\}$. However two of these subsets turn out not to be possible. First of all, it is impossible for a justification status to be $\emptyset$, because there always exists at least one complete labelling (the grounded labelling [32]). Furthermore, it is also impossible for a justification status to be $\{\texttt{in}, \texttt{out}\}$, because when in and out are both included in the justification status, then undec should also be included, as is stated by the following theorem.

Theorem 4.1. *Let $AF = (Ar, def)$ be an argumentation framework and $A \in Ar$. If AF has two complete labellings $\mathcal{L}ab_1$ and $\mathcal{L}ab_2$ such that $\mathcal{L}ab_1(A) = \texttt{in}$ and $\mathcal{L}ab_2(A) = \texttt{out}$ then there exists a complete labelling $\mathcal{L}ab_3$ such that $\mathcal{L}ab_3(A) = \texttt{undec}$.*

Proof. Let $CE_1 = \texttt{Lab2Ext}(\mathcal{L}ab_1)$ and $CE_2 = \texttt{Lab2Ext}(\mathcal{L}ab_2)$. From Theorem 3 of [32] it follows that CE_1 and CE_2 are complete extensions of AF. Let GE be the grounded extension of AF. From [40] it follows that GE is the intersection of all complete extensions of AF. From $\mathcal{L}ab_2(A) = \texttt{out}$, it follows that $A \notin CE_2$ which implies that $A \notin GE$. From $\mathcal{L}ab_1(A) = \texttt{in}$, it follows that $\forall B \in Ar.(B def A \supset \mathcal{L}ab_1(B) = \texttt{out})$. Therefore, $\forall B \in Ar.(B def A \supset B \notin GE)$. So $A \notin GE^+$. Let $\mathcal{L}ab_3 = \texttt{Ext2Lab}(GE)$. GE is a complete extension, so $\mathcal{L}ab_3$ is a complete labelling. Since $A \notin GE$ and $A \notin GE^+$, it holds that $A \in Ar \backslash (GE \cup GE^+)$. So $\mathcal{L}ab_3(A) = \texttt{undec}$. □

Since $\emptyset$ and $\{\texttt{in}, \texttt{out}\}$ are impossible as justification statuses, there are only 6 possible statuses left to be considered: $\{\texttt{in}\}$, $\{\texttt{out}\}$, $\{\texttt{undec}\}$, $\{\texttt{in}, \texttt{undec}\}$, $\{\texttt{out}, \texttt{undec}\}$ and $\{\texttt{in}, \texttt{out}, \texttt{undec}\}$. We now examine under which conditions these justification statuses occur.

First, we examine the conditions under which the justification status is $\{\texttt{in}\}$. From the definition of justification status of an argument, we know that if an argument's justification status is $\{\texttt{in}\}$ then it must be in every complete extension. Then the argument should be in the grounded extension.

Theorem 4.2. *Let $AF = (Ar, def)$ be an argumentation framework and $A \in Ar$. Then $\mathcal{JS}(A) = \{\texttt{in}\}$ iff A is in the grounded extension.*

Proof. "⇒": Suppose $\mathcal{JS}(A) = \{\texttt{in}\}$. Then A is labelled `in` by every complete labelling (Definition 4.1), so A is an element of each complete extension (Theorem 2.4) so A is in the grounded extension (Proposition 2.3).

"⇐": Suppose A is labelled $\{\texttt{in}\}$ by the grounded labelling. Then A is labelled $\{\texttt{in}\}$ by every complete extension. So $\mathcal{JS}(A) = \{\texttt{in}\}$. □

Example 4.1. *Consider the argumentation framework in Figure 4.10. Argument A is in the grounded extension because it is in every complete extension. Then the justification status of A is* $\{\texttt{in}\}$*.*

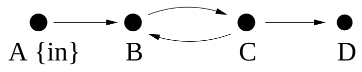


Figure 4.10: What is the justification status of A?

Next, we examine the conditions under which the justification status is $\{\texttt{out}\}$. From the definition of justification status of an argument, we know that an argument's justification status is $\{\texttt{out}\}$ if and only if it is labelled $\{\texttt{out}\}$ by every complete labelling.

Theorem 4.3. *Let $AF = (Ar, def)$ be an argumentation framework and $A \in Ar$. Then $\mathcal{JS}(A) = \{\texttt{out}\}$ iff A is defeated by the grounded extension.*

Proof. "⇒": Suppose $\mathcal{JS}(A) = \{\texttt{out}\}$. Then A is labelled `out` by every complete labelling (Definition 4.1). So in every complete labelling, there exists at least one defeater of A that is labelled `in` by this labelling (Definition 2.25). So every complete extension contains at least one defeater of A (Theorem 2.4). So also the grounded extension also contains a defeater of A. So A is defeated by the grounded extension.

"⇐": Suppose A is defeated by the grounded extension. Then there exists an argument B in the grounded extension such that B is a defeater of A. B is in every complete extension because B is in the grounded extension. So A is defeated by every complete extension since B defeats A. Then A is labelled $\{\texttt{out}\}$ by every complete labelling. So $\mathcal{JS}(A) = \{\texttt{out}\}$. □

Example 4.2. *Consider the argumentation framework in Figure 4.11. Argument B is defeated by argument A and argument A is in the grounded extension. Then the justification status of B is* $\{\texttt{out}\}$*.*

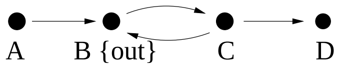


Figure 4.11: What is the justification status of B?

Next, we examine the conditions under which the justification status is $\{\texttt{undec}\}$. From the definition of justification status of an argument, we know that if an argument's justification status is $\{\texttt{undec}\}$ then it is labelled $\{\texttt{undec}\}$ by every complete extension.

Theorem 4.4. *Let $AF = (Ar, def)$ be an argumentation framework and $A \in Ar$. Then $\mathcal{JS}(A) = \{\texttt{undec}\}$ iff*

1. *A is not in any admissible set and*

2. *A is not defeated by any admissible set*

Proof. "$\Rightarrow$": Suppose $\mathcal{JS}(A) = \{\texttt{undec}\}$. Then it holds that

(1) A is not labelled `in` by any complete labelling and

(2) $A$is not labelled `out` by any complete labelling.

From (1) it follows that A is not an element of any complete extension (Theorem 2.4) so A is not an element of any admissible set (Proposition 2.2). From (2) it follows that no defeater of A is labelled `in` by any complete labelling (Definition 2.25) so no defeater of A is in any complete extension (Theorem 2.4) so no defeater of A is in any admissible set (Proposition 2.2) so A is not defeated by any admissible set. Notice that in this proof, we did not use the fact that A is labelled `undec` by at least one complete labelling, which after all is implied by (1) and (2) together with Theorem 4.1.

"$\Leftarrow$": Suppose that

(1) A is not in any admissible set and

(2) A is not defeated by any admissible set.

From (1) it follows that A is not in any complete extension (Proposition 2.2) so A is not labelled `in` by any complete labelling (Theorem 2.4). From (2) it follows that no defeater of A is in any admissible set, so no defeater of A is in any complete extension (Proposition 2.2) so no defeater of A is labelled `in` by any complete labelling (Theorem 2.4) so A is not labelled `out` by any complete labelling (Definition 2.25). This, together with the earlier observed fact that A is not labelled `in` by any complete labelling implies that A is labelled `undec` by every complete labelling. Due to the fact that there always exists at least one complete labelling (since there always exists at least one complete extension), this implies that $\mathcal{JS} = \{\texttt{undec}\}$. □

Example 4.3. *Consider the argumentation framework in Figure 4.12. Argument C and D are not in any admissible set. Argument C is defeated only by itself and argument B which are not in any admissible set. Argument D is defeated only by argument C which is not in any admissible set. Then the justification status of C and D is* $\{\texttt{undec}\}$.

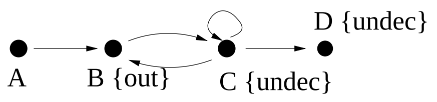


Figure 4.12: What is the justification status of C and D?

Next, we examine the conditions under which the justification status is $\{\texttt{in}, \texttt{undec}\}$.

Theorem 4.5. *Let $AF = (Ar, def)$ be an argumentation framework and $A \in Ar$. Then $\mathcal{JS}(A) = \{\texttt{in}, \texttt{undec}\}$ iff*

1. *A is not in the grounded extension,*
2. *A is in an admissible set, and*
3. *A is not defeated by any admissible set.*

Proof. "⇒": Suppose $\mathcal{JS}(A) = \{\texttt{in}, \texttt{undec}\}$. Then

(1) A is labelled `in` by at least one complete labelling and

(2) A is labelled `undec` by at least one complete labelling and

(3) there exists no complete labelling that labels A `out`.

From (1) it follows that A is contained in at least one complete extension (Theorem 2.4) and that therefore A is in at least one admissible set (Proposition 2.2).
From (2) it follows that there exists at least one complete extension that does not contain A (Theorem 2.4). So A is not in the grounded extension (Proposition 2.3).
From (3) it follows (Definition 2.25) that for all arguments B that defeat A, B is not labelled `in` by any complete labelling. Therefore, no argument B that defeats A is contained in any complete extension (Theorem 2.4). Therefore, no argument B that defeats A is in any admissible set (Proposition 2.2). That is, A is not defeated by any admissible set.

"⇐": Suppose that

(1) A is not in the grounded extension and

(2) A is in an admissible set and

(3) A is not defeated by any admissible set.

From (1) it follows that there exists a complete labelling where A is not labelled `in` (Proposition 2.3 and Theorem 2.4). This, together with the earlier observed fact that A is not labelled `out` by any complete labelling, implies that A is labelled `undec` by at least one complete labelling.
From (2) it follows that A is in a complete extension (Proposition 2.2) so A is labelled `in` by a complete labelling (Theorem 2.4).
From (3) it follows that no admissible set contains a defeater of A so also no complete extension contains any defeater of A (Proposition 2.2). So no complete labelling labels any defeater of A `in` (Theorem 2.4), so A is not labelled `out` by any complete labelling (Definition 2.25). □

Example 4.4. *Consider the argument D in the argumentation framework of Figure 4.13. (1) The argument D is not in the grounded extension since there is a complete extension $\emptyset$ which does not include argument D. (2) D is in the admissible set $\{A, D\}$. (3) D's only defeater C is not in any admissible set. Therefore the justification status of D is $\{\texttt{in}, \texttt{undec}\}$.*

Next, we examine the conditions under which the justification status is $\{\texttt{out}, \texttt{undec}\}$.

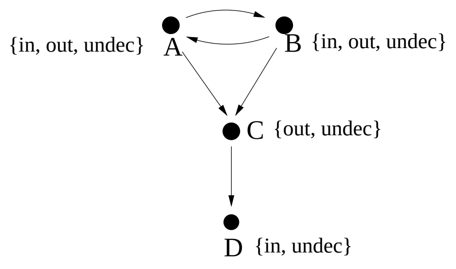


Figure 4.13: The justification status of D.

Theorem 4.6. *Let $AF = (Ar, def)$ be an argumentation framework and $A \in Ar$. Then $\mathcal{JS}(A) = \{\texttt{out}, \texttt{undec}\}$ iff*

1. *A is not in any admissible set,*
2. *A is defeated by an admissible set, and*
3. *A is not defeated by the grounded extension.*

Proof. "$\Rightarrow$": Suppose $\mathcal{JS}(A) = \{\texttt{out}, \texttt{undec}\}$. Then

(1) there exists no complete labelling that labels A `in` and

(2) there exists a complete labelling that labels A `out` and

(3) there exists a complete labelling that labels A `undec`.

From (1) it follows that A is not an element of any complete extension (Theorem 2.4) so A is not an element of any admissible set (Proposition 2.2).
From (2) it follows that A is defeated by at least one complete extension (Theorem 2.4) so A is defeated by at least one admissible set (Proposition 2.2).
From (3) it follows that there exists a complete labelling where A is not labelled `out`, so where none of the defeaters of A are labelled `in` (Definition 2.25). It then follows that there exists a complete extension that contains none of the defeaters of A (Theorem 2.4). So none of the defeaters of A are contained in the grounded extension (Proposition 2.3) so A is not defeated by the grounded extension.

"$\Leftarrow$": Suppose that

(1) there exists no admissible set that contains A and

(2) there is an admissible set that defeats A and

(3) A is not defeated by the grounded extension.

From (1) it follows that A is not an element of any complete extension (Proposition 2.2), so A is not labelled `in` by any complete labelling (Theorem 2.4).
From (2) it follows that A is defeated by a complete extension (Proposition 2.2) so A is labelled `out` by at least one complete labelling (Theorem 2.4).
From (3) it follows that no defeater of A is in the grounded extension. This implies that

there exists a complete extension that does not contain any defeater of A (Proposition 2.3). So there exists a complete labelling where no defeater of A is labelled `in` (Theorem 2.4), so where A is not labelled `out` (Definition 2.25). This, together with the earlier observed fact that A is not labelled `in` by any complete labelling, implies that A is labelled `undec` by at least one complete labelling. □

Example 4.5. *Consider the argument C in the argumentation framework of Figure 4.1. (1) The argument C is not in any admissible set. (2) C is defeated by argument A which is in the admissible set $\{A, D\}$. (3) All defeaters (A and B) of C are not in grounded extension. Therefore the justification status of C is $\{\texttt{out}, \texttt{undec}\}$.*

Next, we examine the conditions under which the justification status is $\{\texttt{in}, \texttt{out}, \texttt{undec}\}$.

Theorem 4.7. *Let $AF = (Ar, def)$ be an argumentation framework and $A \in Ar$. Then $\mathcal{JS}(A) = \{\texttt{in}, \texttt{out}, \texttt{undec}\}$ iff*

1. *A is in an admissible set*
2. *A is defeated by an admissible set*

Proof. "$\Rightarrow$": Suppose $\mathcal{JS}(A) = \{\texttt{in}, \texttt{out}, \texttt{undec}\}$. Then

(1) A is labelled `in` by at least one complete labelling and

(2) A is labelled `out` by at least one complete labelling.

From (1) it follows that A is an element of at least one complete extension (Theorem 2.4) so A is an element of at least one admissible set (Proposition 2.2).
From (2) it follows that there is a complete labelling that labels a defeater of A `in` (Definition 2.25). Therefore there exists a complete extension that contains a defeater of A (Theorem 2.4), so there exists an admissible set that contains a defeater of A (Proposition 2.2). That is, A is defeated by an admissible set.

"$\Leftarrow$": Suppose

(1) there exists an admissible set that contains A and

(2) there exists an admissible set that contains a defeater of A.

From (1) it follows that there exists a complete extension that contains A (Proposition 2.2). so there exists a complete labelling that labels A `in` (Theorem 2.4).
From (2) it follows that there exists a complete extension that contains a defeater of A (Proposition 2.2), so there exists a complete labelling that labels a defeater of A `in` (Theorem 2.4), so there exists a complete labelling where A is labelled `out`.
From the fact that there exists a complete labelling that labels A `in` and there exists a complete labelling that labels A `out` it follows that there also exists a complete labelling that labels A `undec` (Theorem 4.1). □

Example 4.6. *Consider the argument A in the argumentation framework of Figure 4.1. (1) The argument A is in the admissible set $\{A, D\}$. (2) A is defeated by argument B which is in the admissible set $\{B, D\}$. Therefore the justification status of A is $\{\texttt{in}, \texttt{out}, \texttt{undec}\}$.*

From the above theorems, it follows that membership of an admissible set and membership of the grounded extension, of the argument itself and of its defeaters, is sufficient to determine the argument's justification status. The overall procedure of doing so is shown in Figure 4.14. If an argument A is labelled `in` by the grounded labelling then it is labelled `in` by every complete labelling. So $\mathcal{JS}(A) = \texttt{in}$. Otherwise (1) if A is defeated by an argument B that is labelled `in` by the grounded labelling, $\mathcal{JS}(A) = \texttt{out}$ because A is labelled `out` in every complete labelling; (2) if A is not defeated by any argument that are labelled `in` by the grounded labelling then we continue to check whether it is labelled `in` by at least one admissible labelling. If the answer is positive then the justification status of A ends up with two possibilities that with `in` inside, $\{\texttt{in}, \texttt{out}, \texttt{undec}\}$ in which A has a defeater that is labelled `in` by at least one admissible labelling and $\{\texttt{in}, \texttt{undec}\}$ in which none of A defeaters is labelled `in` by every admissible labelling. If the answer is negative, there are also two possibilities that without `in` inside, $\{\texttt{out}, \texttt{undec}\}$ and $\{\texttt{undec}\}$. For the former, A has a defeater that is labelled `in` by at least one admissible labelling so that A is labelled `out` by at least one admissible labelling. For the latter, there exists no defeater of A that is labelled `in` by any admissible labelling so that A is not labelled `out` by any admissible labelling.

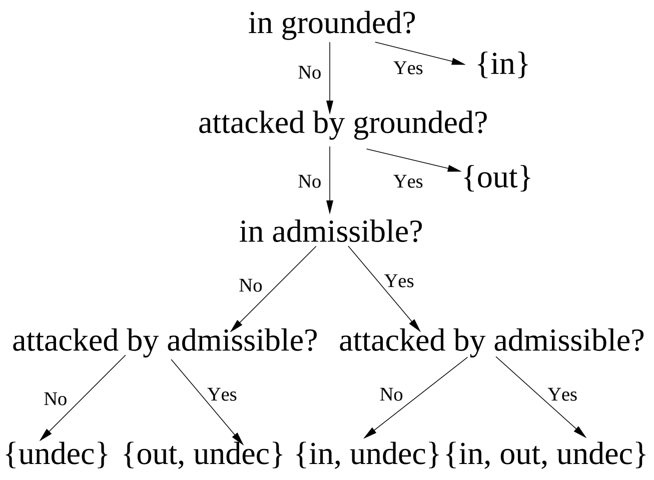


Figure 4.14: Determining the justification status of an argument.

4.3 An Implementation

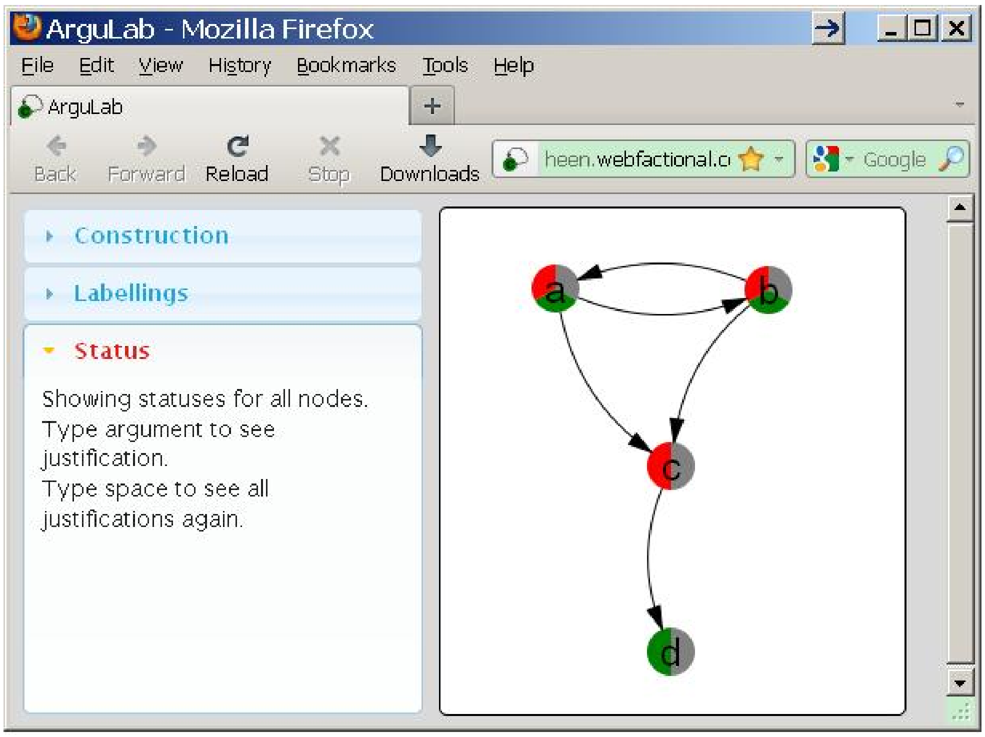
We now demonstrate the applicability of the theory developed in the previous sections by describing our software implementation [67] of it.[1]

In order to determine the justification status of an argument, our implementation follows the procedure of Figure 4.14. To determine whether an argument is in the grounded extension, the algorithm described in [65] is used. This algorithm is subsequently run for the argument's defeaters in order to determine whether the argument is defeated by the grounded extension. To determine whether an argument is in an admissible set, the algorithm described in [97, 33] is used. This algorithm is subsequently run for the argument's defeaters in order to determine whether the argument is defeated by an admissible set.

[1]Available online at http://heen.webfactional.com.

The software can determine the justification statuses and is able to defend them by entering the applicable discussion game with the user. Since the software can be expected to compute the correct justification status, the game is such that in the end the software always wins from the user. Hence, the software is able to explain why its answer is correct, by entering a discussion with the user. Again, to the best of our knowledge, the current simulator is the first implementation of labelling-based justification statuses.

Our software uses a graphical interface. Argumentation frameworks are displayed as graphs and arguments are labelled according to the labelling chosen by users, or according to the output of the software (given a particular semantics). The software consists of several components which (under GPL, i.e., General Public License) could be re-applied in other argumentation software. Hence, someone implementing, say, an argumentation-based on-line discussion forum could apply some of our components to provide a snapshot of, for instance, the justification status of a set of arguments given a particular state of the discussion.


ArguLab - Mozilla Firefox
File Edit View History Bookmarks Tools Help
ArguLab
Back Forward Reload Stop Downloads
heen.webfactional.c
Google
Construction
Labellings
Status
a
b
c
d

Showing statuses for all nodes.
Type argument to see justification.
Type space to see all justifications again.

Figure 4.15: An implementation of the justification status of an argument

4.4 Labelling-Based Justification Statuses vs. Existing Argumentation Semantics

We use the labelling-based approach for computing the justification statuses of arguments because it tends to yield more informative answers than the traditional extension-based approaches.

Consider the argumentation framework in Figure 4.16. A possible interpretation would be as follows:

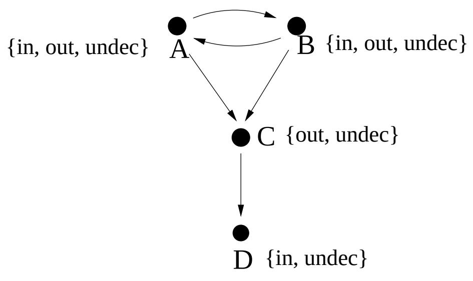


Figure 4.16: The same argumentation framework as in Figure 4.1

A: Carole does not get along with David according to Alice.
B: Carole does not know David according to Bob.
C: Carole says that David is unreliable.
D: David is trustworthy, since he has a general reputation of being so.
The grounded extension is $\emptyset$. None of the arguments is in the grounded extension. Grounded semantics treats all arguments (A, B, C and D) the same. $\{A, D\}$ and $\{B, D\}$ are preferred extensions. Credulous preferred semantics treats A, B and D the same (they are labelled `in` in at least one preferred labelling). Sceptical preferred semantics treats A, B and C the same (they are not labelled `in` in some preferred labellings). Also ideal semantics treats all arguments the same (they are not in the ideal extension ($\emptyset$)).

However, our labelling-based approach for computing the justification status of an argument allows for a more fine grained distinction between arguments. According to the hierarchy of the justification statuses in Figure 4.17, argument D is the strongest, argument C is the weakest, A and B are in between. Unlike sceptical preferred semantics, our labelling approach does not make D completely justified although it does give it a relatively strong status.

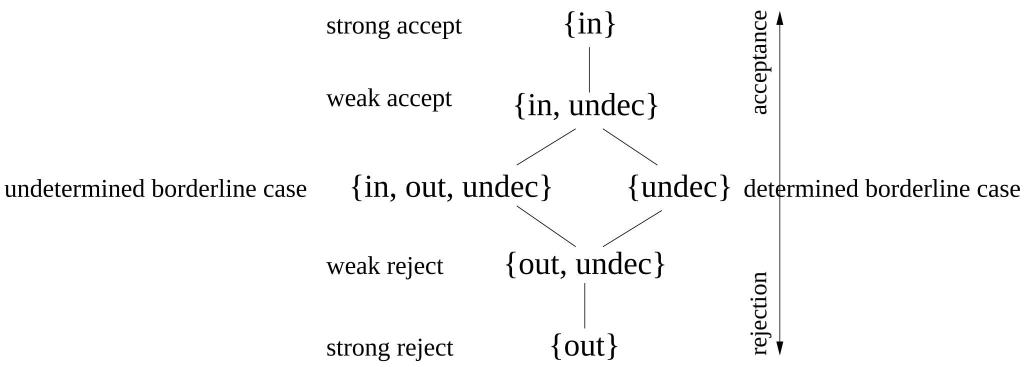


Figure 4.17: The hierarchy of justification statuses

We refer to the justification status $\{\texttt{in}\}$ as *strong accept*, to $\{\texttt{in}, \texttt{undec}\}$ as *weak accept*, to $\{\texttt{in}, \texttt{out}, \texttt{undec}\}$ as *undetermined borderline*, to $\{\texttt{undec}\}$ as *determined borderline*, to $\{\texttt{out}, \texttt{undec}\}$ as *weak reject* and to $\{\texttt{out}\}$ as *strong reject*.

Our labelling-based approach for defining justification statuses not only allows us to identify whether an argument is accepted according to grounded or credulous preferred semantics, it also helps to identify whether an argument is accepted according to ideal semantics. It is in an ideal set iff one can build an admissible set around it that consists only of strongly or weakly accepted arguments.

An ideal set in the sense of [41] is an admissible set that is a subset of each preferred extension. For example, in the argumentation framework of Figure 4.1, $\{D\}$ is an ideal set since $\{D\} = \{A, D\} \cap \{B, D\}$ where $\{A, D\}$ and $\{B, D\}$ are preferred extensions. It has been obtained that one can also describe an ideal set as an admissible set that is not defeated by any admissible set (Theorem 3.2 of [41]). This clears the way for proving the following lemma.

Lemma 4.1. *Let (Ar, def) be an argumentation framework and $\mathcal{A}rgs \subseteq Ar$. $\mathcal{A}rgs$ is an admissible set that is not defeated by any admissible set iff $\mathcal{A}rgs$ is an admissible subset of $\{A \mid \mathcal{JS}(A) = \{\texttt{in}\}\} \cup \{A \mid \mathcal{JS}(A) = \{\texttt{in}, \texttt{undec}\}\}$*

Proof. "$\Rightarrow$": Let $\mathcal{A}rgs$ be an admissible set that is not defeated by any admissible set. Let $A \in \mathcal{A}rgs$. From the fact that A is in an admissible set (and therefore also on a complete extension) it follows that A is labelled `in` in at least one complete labelling. From the fact that $\mathcal{A}rgs$ is not defeated by any admissible set, it follows that A is not defeated by any preferred extension and therefore not defeated by any complete extension. Hence, A is not labelled `out` by any complete labelling. This, together with the earlier observed fact that A is labelled `in` by at least one complete labelling implies that $A \in \{A \mid \mathcal{JS}(A) = \{\texttt{in}\}\} \cup \{A \mid \mathcal{JS}(A) = \{\texttt{in}, \texttt{undec}\}\}$.

"$\Leftarrow$": Let $\mathcal{A}rgs$ be an admissible subset of $\{A \mid \mathcal{JS}(A) = \{\texttt{in}\}\} \cup \{A \mid \mathcal{JS}(A) = \{\texttt{in}, \texttt{undec}\}\}$. Suppose that $\mathcal{A}rgs$ is defeated by an admissible set. That is, there is an argument $A \in \mathcal{A}rgs$ that is defeated by an admissible set. Then A is also defeated by a complete extension (since every admissible set is contained in a preferred extension, which is also a complete extension). This means that A is labelled `out` in at least one complete labelling. So $A \notin \{A \mid \mathcal{JS}(A) = \{\texttt{in}\}\} \cup \{A \mid \mathcal{JS}(A) = \{\texttt{in}, \texttt{undec}\}\}$. Contradiction. □

We now study some of the connections between our notion of justification status and a number of existing approaches. In particular, we examine some connection with grounded semantics [40], credulous preferred semantics [97], sceptical preferred semantics [35], semi-stable semantics [25] and ideal semantics [41].

Proposition 4.1. *Let (Ar, def) be an argumentation framework and $A \in Ar$.*

1. *A is in the grounded extension iff it is strongly accepted*

2. *A is in at least one preferred extension iff A is strongly accepted, weakly accepted, or undetermined borderline.*

3. *if A is in every preferred extension then A is strongly or weakly accepted*

4. *if A is strongly accepted then A is in every semi-stable extension*
   if A is weakly accepted then A is in at least one semi-stable extension

5. *A is in an ideal set iff A is member of an admissible set consisting only of strongly or weakly accepted arguments.*

Proof. Point 1 follows directly from Theorem 4.2. Point 2 follows from the fact that an argument is in a preferred extension iff it is in a complete extension, and therefore labelled `in` by a complete labelling. Point 3 follows from the fact that sceptical preferred rules out all justification statuses containing `out` (strong reject, weak reject and undetermined borderline) as well as the justification status $\{\texttt{undec}\}$ (determined borderline), which means only $\{\texttt{in}\}$ (strong accept) and $\{\texttt{in}, \texttt{out}\}$ (weak accept) remain. Point 4 follows from Theorem 5 of [25]. Point 5 follows from Lemma 4.1. □

In our current implementation (Section 4.3), we have used the discussion game of [97, 33] to determine membership of an admissible set, and the discussion game of [80, 65, 20] to determine membership of the grounded extension. An alternative would be to use the algorithm of [96], which determines both of these memberships in a single pass. Since our notion of justification status depends only on membership of an admissible set and membership of the grounded extension, one is free to apply any kind of algorithm that can determine these.

4.5 A Labelling-Based Justification Statuses of Conclusions

So far we only talked about justification statuses of arguments. However, what matters often are the *conclusions* of these arguments. In this section we treat a possible approach for defining justification statuses of conclusions. If A is an argument then following the ASPIC formalism [31, 78, 4] we write $\texttt{Conc}(A)$ for the conclusion of A (Definition 2.6). Every argument is assumed to have exactly one conclusion which is essentially a formula in some logical language. It is possible for different arguments to have the same conclusion.

We first define the notion of a conclusion labelling.

Definition 4.2 (Conclusion labelling). *Let $\mathcal{L}$ be a logical language. A* conclusion labelling *is a function $\mathcal{C}onc\mathcal{L}ab : \mathcal{L} \to \{\texttt{in}, \texttt{out}, \texttt{undec}\}$*

The next step is to define how to convert an argument labelling ($\mathcal{A}rg\mathcal{L}ab$) into a conclusion labelling ($\mathcal{C}onc\mathcal{L}ab$). For this we use a function $\mathcal{A}rg\mathcal{L}ab2\mathcal{C}onc\mathcal{L}ab$. Basically, the idea is to associate each conclusion with the label of the best possible argument that is able to produce this conclusion.[2] So if there are three arguments A_1, A_2 and A_3 with conclusion c, where $\mathcal{A}rg\mathcal{L}ab(A_1) = \texttt{in}$, $\mathcal{A}rg\mathcal{L}ab(A_2) = \texttt{out}$ and $\mathcal{A}rg\mathcal{L}ab(A_3) = \texttt{undec}$, then $\mathcal{C}onc\mathcal{L}ab(c) = \texttt{in}$, since the best argument with conclusion c (A_1) is labelled `in`. Similarly, if there are three arguments B_1, B_2 and B_3 with conclusion d, where $\mathcal{A}rg\mathcal{L}ab(B_1) = \texttt{out}$, $\mathcal{A}rg\mathcal{L}ab(B_2) = \texttt{undec}$ and $\mathcal{A}rg\mathcal{L}ab(B_3) = \texttt{out}$. Then $\mathcal{C}onc\mathcal{L}ab(d) = \texttt{undec}$, since the best argument with conclusion d (B_2) is labelled `undec`. If for a particular conclusion, there is no argument at all that produces it, then the conclusion is given the lowest possible label (`out`). Formally, the translation from an argument labelling to a conclusion labelling can be defined as follows.

Definition 4.3. *Let $AF = (Ar, def)$ be an argumentation framework whose conclusions belong to logical language $\mathcal{L}$. Let $\mathcal{A}rg\mathcal{L}abs$ be the set of all argument labellings of AF and $\mathcal{C}onc\mathcal{L}abs$ be the set of all conclusion labellings of $\mathcal{L}$. We define a function $\mathcal{A}rg\mathcal{L}ab2\mathcal{C}onc\mathcal{L}ab$:*

[2]We assume the labels to be ordered such that $\texttt{in} > \texttt{undec} > \texttt{out}$.

$\mathcal{A}rg\mathcal{L}abs \to \mathcal{C}onc\mathcal{L}abs$ such that, given a labelling $\mathcal{A}rg\mathcal{L}ab$ of AF, the associated conclusion labelling $\mathcal{C}onc\mathcal{L}ab = \mathcal{A}rg\mathcal{L}ab2\mathcal{C}onc\mathcal{L}ab(\mathcal{A}rg\mathcal{L}ab)$ is such that for every $c \in \mathcal{L}$ it holds that $\mathcal{C}onc\mathcal{L}ab(c) = \max(\{\mathcal{A}rg\mathcal{L}ab(A) \mid \texttt{Conc}(A) = c\} \cup \{\texttt{out}\})$.

We say that a conclusion labelling $\mathcal{C}onc\mathcal{L}ab$ is a *complete conclusion labelling* of AF iff there exists a complete argument labelling $\mathcal{A}rg\mathcal{L}ab$ of AF such that $\mathcal{C}onc\mathcal{L}ab = \mathcal{A}rg\mathcal{L}ab2\mathcal{C}onc\mathcal{L}ab(\mathcal{A}rg\mathcal{L}ab)$. This then allows us to define the justification status of a conclusion as the set of labels that can reasonably be assigned to it.

Definition 4.4 (Complete conclusion labelling). *Let $AF = (Ar, def)$ be an argumentation framework whose conclusions belong to logical language $\mathcal{L}$ and $c \in \mathcal{L}$. The* justification status *of c is the outcome yielded by the function $\mathcal{C}onc\mathcal{JS} : \mathcal{L} \to 2^{\{\texttt{in},\texttt{out},\texttt{undec}\}}$ such that $\mathcal{C}onc\mathcal{JS}(c) = \{\mathcal{C}onc\mathcal{L}ab(c) \mid \mathcal{C}onc\mathcal{L}ab$ is a complete conclusion labelling of $AF\}$.*

To illustrate the usefulness of justification statuses of conclusions, consider the following two examples taken from [73].

Example 4.7.
$\mathcal{P} = \{BornInHolland$, " Brygt Rykkje was born in Holland."
$NorwegianName\}$, "Brygt Rykkje has a Norwegian name."
$\mathcal{S} = \{Dutch \to \neg Norwegian; Norwegian \to \neg Dutch\}$.
$\mathcal{D} = \{BornInHolland \Rightarrow Dutch$, "Brygt Rykkje is Dutch since he was born in Holland."
$NorwegianName \Rightarrow Norwegian$, "Brygt Rykkje is Norwegian since he has a Norwegian name."
$Dutch \Rightarrow iceSkating$, "Brygt Rykkje likes ice skating since he is Dutch."
$Norwegian \Rightarrow iceSkating\}$, "Brygt Rykkje likes ice skating since he is Norwegian."

Consider the following arguments:
$A_1 :\ BornInHolland$
$A_2 :\ A_1 \Rightarrow Dutch$
$A_3 :\ A_2 \to \neg Norwegian$
$A_4 :\ A_2 \Rightarrow iceSkating$
$B_1 :\ NorwegianName$
$B_2 :\ B_1 \Rightarrow Norwegian$
$B_3 :\ B_2 \to \neg Dutch$
$B_4 :\ B_2 \Rightarrow iceSkating$

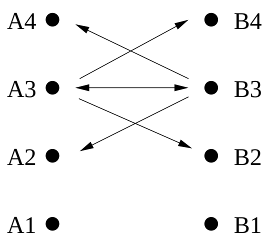


Figure 4.18: *iceSkating* is the floating conclusion

We first observe that in this example arguments A_4 and B_4 have the same conclusion iceSkating. For the argumentation framework in Figure 4.18, one can distinguish three different complete argument labellings:

$\mathcal{A}rg\mathcal{L}ab_1 = (\{A_1, A_2, A_3, A_4, B_1\}, \{B_2, B_3, B_4\}, \emptyset)$,
$\mathcal{A}rg\mathcal{L}ab_2 = (\{B_1, B_2, B_3, B_4, A_1\}, \{A_2, A_3, A_4\}, \emptyset)$,
$\mathcal{A}rg\mathcal{L}ab_3 = (\{A_1, B_1\}, \emptyset, \{A_2, A_3, A_4, B_2, B_3, B_4\})$.
Let us now consider how this situation is handled by our approach of conclusion-based justification statuses. Using the function $\mathcal{A}rg\mathcal{L}ab2\mathcal{C}onc\mathcal{L}ab$, one can then identify three different complete conclusion labellings.
$\mathcal{C}onc\mathcal{L}ab_1 = (\{BornInHolland, NorwegianName, Dutch, \neg Norwegian, iceSkating\}, \{Norwegian, \neg Dutch\}, \emptyset)$,
$\mathcal{C}onc\mathcal{L}ab_2 = (\{BornInHolland, NorwegianName, Norwegian, \neg Dutch, iceSkating\}, \{Dutch, \neg Norwegian\}, \emptyset)$,
$\mathcal{C}onc\mathcal{L}ab_3 = (\{BornInHolland, NorwegianName\}, \emptyset, \{Dutch, \neg Dutch, Norwegian, \neg Norwegian, iceSkating\})$.
Hence, the justification status of conclusion iceSkating is {`in`, `undec`} *(weak accept), even though the two arguments that produce conclusion iceSkating (A_4 and B_4) each have a justification status* {`in`, `out`, `undec`} *(undetermined borderline). The conclusion iceSkating is a floating conclusion because it is labelled* `in` *by both $\mathcal{C}onc\mathcal{L}ab_1$ and $\mathcal{C}onc\mathcal{L}ab_2$, but it is supported by different arguments A_4 and B_4 respectively. In this example, our approach of justification statuses makes floating conclusion* weakly accepted *but not* strongly accepted.

Example 4.8.
Let $unreliable(John) = \neg\lceil Says(John, stabbed) \Rightarrow stabbed\rceil$ and $unreliable(Bob) = \neg\lceil Says(Bob, shot) \Rightarrow shot\rceil$.

$\mathcal{P} = \{Says(John, stabbed)$, *"John says that the suspect stabbed the victim."*
$Says(Bob, shot)\}$, *"Bob says that the suspect shot the victim."*
$\mathcal{S} = \{stabbed \rightarrow \neg shot; shot \rightarrow \neg stabbed\}$.
$\mathcal{D} = \{Says(John, stabbed) \Rightarrow stabbed$, *"John says that the suspect stabbed the victim then probably the suspect stabbed the victim."*
$Says(Bob, shot) \Rightarrow shot$, *"Bob says that the suspect shot the victim then probably the suspect shot the victim."*
$stabbed \Rightarrow guilty$, *"The suspect might be guilty since the suspect stabbed the victim."*
$shot \Rightarrow guilty$, *"The suspect might be guilty since the suspect shot the victim."*
$\neg stabbed \Rightarrow unreliable(John)$, *"If the suspect shot the victim then John is unreliable."*
$\neg shot \Rightarrow unreliable(Bob)\}$, *"If the suspect stabbed the victim then Bob is unreliable."*

Consider the following arguments:
$A_1 :\ Says(John, stabbed)$
$A_2 :\ A_1 \Rightarrow stabbed$
$A_3 :\ A_2 \rightarrow \neg shot$
$A_4 :\ A_2 \Rightarrow guilty$
$A_5 :\ A_3 \Rightarrow unreliable(Bob)$
$B_1 :\ Says(Bob, shot)$
$B_2 :\ B_1 \Rightarrow shot$
$B_3 :\ B_2 \rightarrow \neg stabbed$
$B_4 :\ B_2 \Rightarrow guilty$
$B_5 :\ B_3 \Rightarrow unreliable(John)$

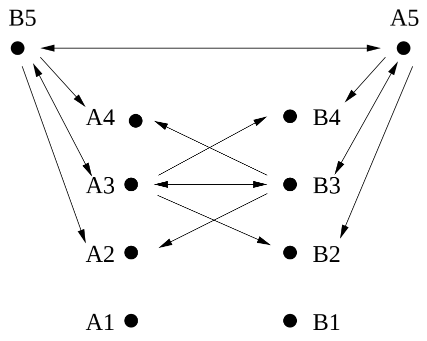


Figure 4.19: Floating Conclusion

We first observe that in this example arguments A_4 and B_4 have the same conclusion guilty. For the argumentation framework in Figure 4.19, one can distinguish three different complete argument labellings:

$\mathcal{A}rg\mathcal{L}ab_1 = (\{A_1, A_2, A_3, A_4, A_5, B_1\}, \{B_2, B_3, B_4, B_5\}, \emptyset)$,
$\mathcal{A}rg\mathcal{L}ab_2 = (\{B_1, B_2, B_3, B_4, B_5, A_1\}, \{A_2, A_3, A_4, A_5\}, \emptyset)$,
$\mathcal{A}rg\mathcal{L}ab_3 = (\{A_1, B_1\}, \emptyset, \{A_2, A_3, A_4, A_5, B_2, B_3, B_4, B_5\})$.

Let us now consider how this situation is handled by our approach of conclusion-based justification statuses. Using the function $\mathcal{A}rg\mathcal{L}ab2\mathcal{C}onc\mathcal{L}ab$, one can then identify three different complete conclusion labellings.

$\mathcal{C}onc\mathcal{L}ab_1 = (\{Says(John, stabbed), Says(Bob, shot), stabbed, \neg shot, guilty, unreliable(Bob)\}, \{shot, \neg stabbed, unreliable(John)\}, \emptyset)$,
$\mathcal{C}onc\mathcal{L}ab_2 = (\{Says(John, stabbed), Says(Bob, shot), shot, \neg stabbed, guilty, unreliable(John)\}, \{stabbed, \neg shot, unreliable(Bob)\}, \emptyset)$,
$\mathcal{C}onc\mathcal{L}ab_3 = (\{Says(John, stabbed), Says(Bob, shot)\}, \emptyset, \{stabbed, \neg stabbed, shot, \neg shot, guilty, unreliable(John), unreliable(Bob)\})$.

Hence, the justification status of conclusion guilty is $\{\mathtt{in}, \mathtt{undec}\}$ *(weak accept), even though the two arguments that produce conclusion guilty (A_4 and B_4) each have a justification status* $\{\mathtt{in}, \mathtt{out}, \mathtt{undec}\}$ *(undetermined borderline). The conclusion guilty is a floating conclusion because it is labelled* `in` *by both $\mathcal{C}onc\mathcal{L}ab_1$ and $\mathcal{C}onc\mathcal{L}ab_2$, but it is supported by different arguments A_4 and B_4 respectively. In this example, our approach of justification statuses makes floating conclusion* weakly accepted *but not* strongly accepted.

In [73], Prakken shows that these two examples do not have the precisely same logical form because in Example 4.8 there are in addition two undefeated undercutters of, respectively, A_4 and B_4, which have no counterpart in Example 4.7. However, if we ignore arguments A_5 and B_5 in Example 4.8 then we obtain the same argumentation frameworks. There are floating conclusions in both Example 4.7 and Example 4.8. Both of the floating conclusions are weakly accepted by using our approach of conclusion-based justification status. However, the final decisions of whether the floating conclusions should be accepted are different in these two cases.

Whether or not weak accept is sufficient to endorse a conclusion depends on the particular domain of reasoning. In criminal law, for instance, what matters is the suspect should be guilty beyond reasonable doubt. This can be interpreted as requiring that in every reasonable position that one can take given the information that is available (that is, in every complete labelling of the argumentation framework) the fact that the suspect is guilty is a valid conclusion (that is, there exists an accepted argument (labelled in) for the conclusion that the suspect is guilty). This condition is not fulfilled in the above example. Therefore the conclusion that the suspect is guilty should not be endured, at least not from a legal perspective.

Whether or not a conclusion is endorsed depends to a great extent on the proof standard [9] being applied. When deciding whether to take a friend to the ice rink or the cinema (Example 4.7), weak accept might be sufficient. However, when deciding whether or not to put someone in jail (Example 4.8), one may really require strong accept before doing so. In our approach, the system simply states how strongly each statement is accepted (or rejected). And it is up to the user to decide whether this level is sufficient to act upon. When this idea is applied to the legal notions of burdens and standards of proof, the accepted conclusion should be at least strongly accepted.

4.6 Related Work

The labelling-based approach for determining justification statuses is somewhat similar to the approach described by Baroni and Giacomin [10]. Baroni and Giacomin compared the skepticism among argumentation semantics using a weak and a strong comparison criterion. According to the weak comparison criteria, stable semantics and semi-stable semantics are equivalent and are the least skeptical. Grounded and complete semantics are equivalent and are the most skeptical. Preferred and ideal semantics are in the middle and preferred semantics is less skeptical than ideal skeptical. CF2 semantics is also in the middle and it is incomparable with preferred and ideal semantics. The ordering is different according to the strong criteria. The skepticism of stable and semi-stable semantics are still equivalent and are the least skeptical ones. Preferred and CF2 semantics are also the least skeptical and are not comparable with each other and with stable-related semantics. Ideal and complete semantics are in the middle and are incomparable with each other. Grounded semantics is the most skeptical and is not equivalent to complete semantics any more. However, Baroni and Giacomin [10] did not specify a concrete semantics with which to apply their approach to, and as a result of this, they did not provide any procedures regarding how to determine the justification status of an argument.

Our work only considers the complete semantics. Dvořák [42] generalized our work to arbitrary argumentation semantics. Particularly, Dvořák has shown the justification statuses with respect to grounded, admissible, complete, preferred, semi-stable and stage semantics and compared them. For instance, if $\mathcal{L}$ is the grounded labelling then the justification status of an argument A with respect to grounded semantics is $\{\mathcal{L}(A)\}$ since there is only one grounded labelling. However, Dvořák does not give any approach to determine and defend the justification statuses with respect to those argument semantics. Dvořák also provided the computational complexities of decision problems of the justification status of argument with respect to the above semantics. The complexity results of Dvořák [42] show that for grounded, admissible, complete, stable and preferred semantics, weak acceptance is easier than substantiating the justification statuses of an argument. Especially for preferred semantics, weak acceptance is D^P-c and easier than skeptical acceptance which is $\prod_2^P$-c. This could be an advantage of the cases where the floating conclusions should be accepted since the justification statuses of floating conclusions is $\{\mathtt{in}, \mathtt{undec}\}$ which means weak acceptance.

The idea of justification statuses can be applied to the notion of burdens and standards of proof. The justification statuses of arguments can be used to determine the acceptability of arguments. Arguments that are weakly accepted ($\{\mathtt{in}, \mathtt{undec}\}$) are not acceptable. Only arguments that are strongly accepted ($\{\mathtt{in}\}$) can be accepted in a court. Therefore, the justification statuses can be used as a proof standard. There are alternative formal modelings of burdens and standards of proof, for instance, the Carneades model of argument and burden of proof presented by Gordon *et al.* [48]. In the Carneades model, premise types and proof standards can be used to allocate the burdens of proof. The proof standard of SE (Scintilla of Evidence), BA (Best Argument) and DV (Dialectical Validity) have been defined in [48]. In [81], Prakken and Sartor used argument games to test an argument's dialectical status and formalised proof standards into a definition of the defeat relation between arguments. In the Survey of Prior Research of [49], Gordon and Walton conclude that their model can be seen as the model of Prakken and Sartor by substituting the non-monotonic logic with a structure that makes the evaluation of arguments tractable for common proof standards.

4.7 Summary and Future Work

In this chapter, we have presented a definition of justification statuses of arguments which indicates whether an argument has to be accepted, can be accepted, has to be rejected, can be rejected, etc. We then provided some concrete guidelines for determining these justification statuses, as well as for defending them using discussion games. We illustrated our demonstrator that implements the idea of justification statuses of arguments in Section 4.3. Then we compared the labelling-based justification statuses and the existing argumentation semantics. Finally, we provided an approach of defining the justification statuses of conclusions of arguments.

The justification statuses of arguments that we introduced in this chapter are the sets of labels that are possibly assigned to the arguments by complete labellings. There are six possible justification statuses of arguments. According to the justification status of an argument, the argument can be strongly accepted, weakly accepted, strongly rejected, weakly rejected or border line. The justification statuses of arguments can be determined under different conditions. Our demonstrator determines the justification statuses of arguments by using the grounded discussion game and preferred discussion game. The software implementation can be applied as component in other argumentation softwares. In Section 4.4, we have shown that the justification statuses of arguments can be used to identify whether an argument is accepted according to some existing semantics, i.e., grounded, preferred and ideal semantics. Moreover, since conclusions of arguments are the final results we want in argumentation framework of constructed arguments, we have defined labelling-based justification statuses of conclusions based on the notion of a complete labelling of conclusions which is defined according to the complete labellings of arguments.

We focus only on the justification statuses of arguments based on complete labellings.

The approaches to determine and defend the justification statuses of arguments based on other argument semantics, for instance, preferred, stable and semi-stable semantics, could be an interesting future work. For example, under stable semantics there are only four possible justification statuses of arguments: {`in`}, {`out`}, {`in`, `out`}, {} because stable semantics does not allow for an argument to be labelled `undec`.

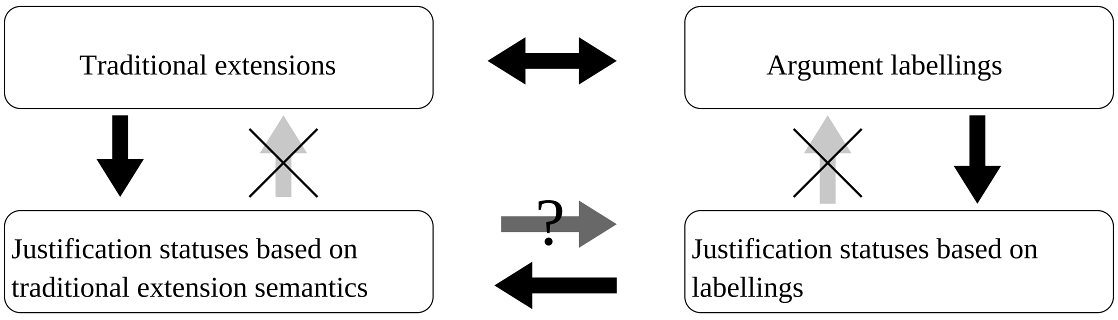

Traditional extensions
Argument labellings
?
Justification statuses based on traditional extension semantics
Justification statuses based on labellings

Figure 4.20: Interdefinability

As it is shown in Figure 4.20, the traditional extension semantics and the argument labellings are inter-definable. It means that if we have a complete extension of an argumentation framework then we can obtain a complete labelling with respect to the complete extension and vice versa. This is because that the complete extensions and the complete labellings are one to one related (Theorem 2.4). According to the definition of the justification statuses of arguments, we can draw the justification statuses out of argument labellings and extensions, but not vice versa. Moreover, if we know the justification statuses based on complete labellings then we can obtain the justification statuses based on the traditional extension semantics. What we don't know is that whether the justification statuses based on labellings can be obtained from the justification statuses based on traditional extension semantics. Although our guess is a "no", it could be another future work.

Chapter 5

Complete Extensions Coincide with 3-Valued Stable Models in Logic Programming

In this chapter, we prove the correspondence between complete extensions in abstract argumentation and 3-valued stable models in logic programming. We only consider finite argumentation frameworks and finite logic programs. This result is in line with earlier work of Dung [40] that identified the correspondence between the grounded extension in abstract argumentation and the well-founded model in logic programming, as well as between the stable extensions in abstract argumentation and the stable models in logic programming.

5.1 Introduction

Dung's theory of abstract argumentation has been shown to be suitable to express a whole range of logical formalisms for nonmonotonic reasoning, including logic programming with weak negation [40]. Many of the existing semantics for logic programing can be understood in a way using argumentation. The overlap between logic programming and abstract argumentation has been studied by Dung [40] in which he showed that the grounded extension in abstract argumentation corresponds to the well-founded model in logic programming, and that the stable extensions in abstract argumentation correspond to the stable models in logic programming.

At the right hand side of Figure 5.1 we find a number of logic programming semantics. The 3-valued stable semantics in logic programming was introduced by Przymusinski [84]. It has been used as the basis for describing other semantics in logic programming such as well-founded model [45], regular models [101, 102, 103], stable models [46] and L-stable models [88, 87, 43]. It has been proved that the well-founded model is also a 3-valued stable model [84]. Eiter *et al.* [43] show that every L-stable model is also a regular model, that every regular model is also a 3-valued stable model and that every (2-valued) stable model is also an L-stable model. At the left hand side of Figure 5.1 we can find the same number of argument semantics as the logic programming semantics that are at the right hand side of the figure. The relation between these argument semantics is shown in Figure 5.1 and stated in Chapter 2. The correspondence between the grounded extension in argumentation and the well-founded model in logic programming and the correspondence between

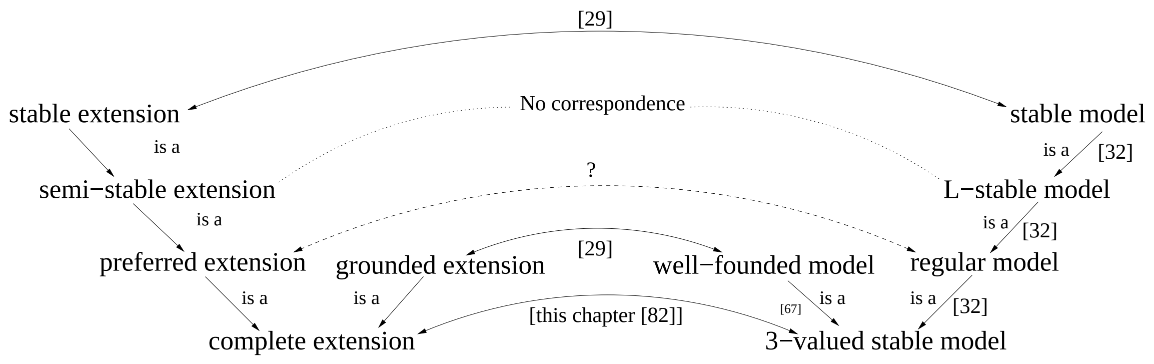


Figure 5.1: An overview of the different semantics

the stable extensions in argumentation and the stable models in logic programming that have been shown by Dung give rise to the interest in the correspondences between other semantics in argumentation and models in logic programming, for instance, in Figure 5.1, the correspondence between complete extensions and 3-valued stable models and the correspondence between preferred extensions and regular models. Recently, Wolfgang Dvorák (private communication) proved that the correspondence between semi-stable extensions and L-stable models do not exit. Since the complete semantics and the 3-valued stable models are both used as basis of defining other semantics in abstract argumentation and logic programming respectively, in this chapter we examine the correspondence between 3-valued stable models in logic programming and complete extensions in abstract argumentation.

We expect that this correspondence can lead to other overlaps between abstract argumentation semantics and logic programming semantics. For example, the correspondence between preferred semantics and regular model has not been studied yet. The overlaps between abstract argumentation semantics and logic programming semantics can make it possible to apply already existing algorithms and computational results of one area to the other area.

The remaining part of this chapter is organized in the following way. Section 5.2 states some preliminaries on logic programing. Section 5.3 demonstrates the equivalence between complete labellings (which coincide with complete extensions [23]) and 3-valued stable models. Finally in Section 5.5 we discuss the main results of the chapter and identify some possibilities for further research.

5.2 3-Valued Stable Models in Logic Programming

We first summarize some basic concepts and terminologies in the field of logic programming. In this chapter, we consider the simplified, propositional case in which a logic program contains no variables.

A logic program is a finite set of rules of the form $A \leftarrow A_1, \ldots, A_n, \text{not } B_1, \ldots, \text{not } B_m$, where $n, m \geq 0$ and A, A_i, B_j $(1 \leq i \leq n, 1 \leq j \leq m)$ are atoms. A is called the *head* of the rule. $A_1, \ldots, A_n, \text{not } B_1, \ldots, \text{not } B_m$ is the *body* of the rule. A program P is positive if there is no negative literals (eg. not B_i) in the body of every rule (i.e., $m = 0$).

Given a logic program P, $\mathcal{A}_P$ is the set of all atoms occurring in P. An *interpretation*

$I = \langle T; F\rangle$ for a program P can be viewed as a mapping from $\mathcal{A}_P$ to the set of truth values $\{t, f, u\}$, denoted by:

$$I(A) = \begin{cases} t & \text{if } A \in T, \\ u & \text{if } A \in \overline{I} \\ f & \text{if } A \in F \end{cases}$$

where $T \cap F = \emptyset$ and $\overline{I} = \mathcal{A}_P - (T \cup F)$. t, f, u denote *true*, *false* and *undefined* respectively, ordered as $f < u < t$. We use this order to define the satisfiability of a rule with respect to a 3-valued interpretation.

Definition 5.1 (Definition 2.3 in [84]). *Let $r = A \leftarrow L_1, \ldots, L_n$ be a rule in a logic program and M be a 3-valued interpretation. M* satisfies *r if $M(A) \geq \min\{M(L_i) \mid 1 \leq i \leq n\}$.*

Example 5.1. *Suppose that r = work $\leftarrow$ not tired. Let $M = \langle\{\textit{work, angry}\}; \{\}\rangle$. Then $M(\textit{work}) = t$ and $M(\textit{not angry}) = f$. So $M(\textit{work}) \geq \min\{M(\textit{not angry})\}$ because $t \geq f$. Therefore M satisfies r.*

If a 3-valued interpretation satisfies every rule in a logic program then the 3-valued interpretation is a 3-valued model of the logic program.

Definition 5.2 (Model (Definition 2.3 in [84])). *Let P be a logic program and M be a 3-valued interpretation for P. M is a* 3-valued model *for P if every rule in P is satisfied by M.*

There is a standard ordering ($\preceq$) between interpretations. Models which are $\preceq$-least are called least models.

Definition 5.3 ([83]). *Let $I = \langle T; F\rangle$ and $J = \langle T'; F'\rangle$ be two 3-valued interpretations. We say that $I \preceq J$ if $T \subseteq T'$ and $F \supseteq F'$.*

Let P be a logic program and I be any 3-valued interpretation. The *Gelfond-Lifschitz-transformation* of P w.r.t. I, denoted by $\frac{P}{I}$, is obtained by replacing in the body of every rule of P all negative literals which are *true* (respectively *undefined*, *false*) by t (respectively u, f).

Example 5.2 ([84]). *Suppose that P is :*

$$\begin{array}{rcl} \textit{work} & \leftarrow & \textit{not tired} \\ \textit{sleep} & \leftarrow & \textit{not work} \\ \textit{tired} & \leftarrow & \textit{not sleep} \\ \textit{angry} & \leftarrow & \textit{work, not paid} \\ \textit{paid} & \leftarrow & \end{array}$$

Let $I = \langle\{\textit{paid}\}; \{\textit{angry}\}\rangle$. Then the transformed program $\frac{P}{I}$ is:

$$\begin{array}{rcl} \textit{work} & \leftarrow & u \\ \textit{sleep} & \leftarrow & u \\ \textit{tired} & \leftarrow & u \\ \textit{angry} & \leftarrow & \textit{work, } f \\ \textit{paid} & \leftarrow & \end{array}$$

The resulting program $\frac{P}{I}$ of the Gelfond-Lifschitz-transformation is positive. According to the Kowalski-Van Emden Theorem [91], every positive program has a unique least 3-valued model. So $\frac{P}{I}$ has a least model.

Definition 5.4 ([84]). *Let $\mathcal{P}$ be a set of logic programs and $\mathcal{I}$ be a set of 3-valued interpretations. We define a function $\Gamma^* : \mathcal{P} \times \mathcal{I} \to \mathcal{I}$ such that $\Gamma^*(I)$ is the least model of $\frac{P}{I}$ where $P \in \mathcal{P}$ and $I \in \mathcal{I}$.*

A 3-valued interpretation M is a 3-valued stable model of a logic program P if and only if it is the least model of $\frac{P}{M}$, i.e., $M = \Gamma^*(M)$.

Definition 5.5 (3-valued stable model (Definition 3.2 in[84])). *A 3-valued interpretation M of a logic program P is a* 3-valued stable model *of P iff $\Gamma^*(M) = M$.*

Example 5.3. *Continue with Example 5.2. $M = \langle\{paid\}; \{angry\}\rangle$ satisfies $\frac{P}{I}$ because for the first three rules, $u \geq u$, for the forth rule, $f \geq \min\{u, f\}$ and for the last rule, $t \geq f$.*

$$\begin{array}{rcl} work & \leftarrow & u \\ sleep & \leftarrow & u \\ tired & \leftarrow & u \\ angry & \leftarrow & work,\ f \\ paid & \leftarrow & \end{array} \quad \Rightarrow \quad \begin{array}{rcl} u & \leftarrow & u \\ u & \leftarrow & u \\ u & \leftarrow & u \\ f & \leftarrow & u,\ f \\ t & \leftarrow & \end{array}$$

$M_1 = \langle\{paid, work\}; \{angry\}\rangle$ is also a model of $\frac{P}{I}$, but $M \preceq M'$. Therefore M' is not the least model of $\frac{P}{I}$. M is the least model of $\frac{P}{M}$ since $M \preceq M'$ for each model M' of $\frac{P}{M}$. Therefore M is a 3-valued stable model of P.

5.3 Complete Labellings Coincide with 3-Valued Stable Models

In this section we first transform argumentation frameworks into logic programs and prove that the complete labellings of an argumentation framework coincide with the 3-valued stable models of the associated logic program (Section 5.3.1). Then, in Section 5.3.2 we transform logic programs into argumentation frameworks and prove that the 3-valued stable models of a logic program coincide with the complete labellings of the associated argumentation framework. The transformation from argumentation frameworks into logic programs is relatively easier than the transformation from logic programs into argumentation frameworks. The reason is that the we do not consider the internal structures of arguments when we transform argumentation frameworks into logic programs. However, when we transform a logic program into an argumentation framework, we have to consider how to construct arguments from the rules of the logic program.

5.3.1 Transforming Argumentation Frameworks into Logic Programs

We use the approach of Gabbay and Garcez [44] to transform abstract argumentation frameworks into logic programs.

An argumentation framework can be transformed into a logic program by generating a rule for each argument in the argumentation framework such that the argument itself is in the head of the rule and the negations of all its defeaters are in the body of the rule.

Definition 5.6 (Associated logic program ([44])). *Let $AF = (Ar, def)$ be an argumentation framework. We define the associated logic program P_{AF} as follows,*
$P_{AF} = \{A \leftarrow not\ B_1, \dots, not\ B_n \mid A, B_1, \dots, B_n \in Ar\ (n \geq 0)$ *and* $\{B_i \mid B_i def A\} = \{B_1, \dots, B_n\}\}$.

The following example shows how to transform an argumentation framework to a logic program.

Example 5.4. *Consider the argumentation framework AF in Figure 5.2.*

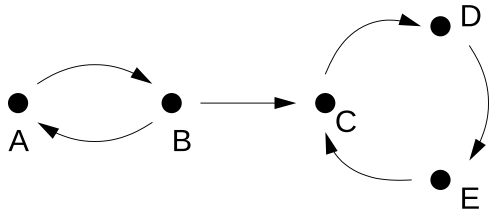


Figure 5.2: The same argumentation framework as in Figure 2.4.

The associated logic program P_{AF} is:

$$\begin{array}{lcl} A & \leftarrow & not\ B \\ B & \leftarrow & not\ A \\ C & \leftarrow & not\ B,\ not\ E \\ D & \leftarrow & not\ C \\ E & \leftarrow & not\ D \end{array}$$

We now define two functions that, given an argumentation framework AF, allow a labelling to be converted to a 3-valued interpretation of P_{AF} and vice versa.

Definition 5.7. *Let $\mathcal{L}abellings$ be the set of all labellings of AF and $\mathcal{M}odels$ be all the 3-valued interpretations of P_{AF}. Let $\mathcal{L} \in \mathcal{L}abellings$. We introduce a function* $\texttt{Lab2Mod} : \mathcal{L}abellings \rightarrow \mathcal{M}odels$. *Let* $\texttt{Lab2Mod}(\mathcal{L}) = \mathcal{M}$. *Then* $\mathcal{M} = \langle \texttt{in}(\mathcal{L}); \texttt{out}(\mathcal{L}) \rangle$ *and* $\overline{\mathcal{M}} = \texttt{undec}(\mathcal{L})$.

Example 5.5. *Consider the argumentation framework AF in Figure 5.2 and the the associated logic program P_{AF} in Example 5.4. Let* $L = \{(A, \texttt{in}), (B, \texttt{out}), (C, \texttt{undec}), (D, \texttt{undec}), (E, \texttt{undec})\}$ *be a labelling of AF. Then* $\texttt{Lab2Mod}(L) = \langle \{A\}; \{B\} \rangle$ *and* $\overline{\texttt{Lab2Mod}(L)} = \{C, D, E\}$.

Definition 5.8. *Let $\mathcal{L}abellings$ be the set of all labellings of AF and $\mathcal{M}odels$ be all the 3-valued interpretations of P_{AF}. Let $I \in \mathcal{M}odels$ and $I = \langle T, F \rangle$. We define a function* $\texttt{Mod2Lab} : \mathcal{M}odels \rightarrow \mathcal{L}abellings$ *such that* $\texttt{in}(\texttt{Mod2Lab}(I)) = T$ *and* $\texttt{out}(\texttt{Mod2Lab}(I)) = F$ *and* $\texttt{undec}(\texttt{Mod2Lab}(I)) = \overline{I}$.

Example 5.6. *Consider again the argumentation framework AF in Figure 5.2 and the associated logic program P_{AF} in Example 5.4. Let $I = \langle \{A\}; \{B\} \rangle$ be a 3-valued interpretation. Then* $\texttt{Mod2Lab}(I) = \{(A, \texttt{in}), (B, \texttt{out}), (C, \texttt{undec}), (D, \texttt{undec}), (E, \texttt{undec})\}$.

If $\mathcal{L}$ is a complete labelling of an argumentation framework, then $\texttt{Lab2Mod}(\mathcal{L})$ is a 3-valued stable model of the associated logic program, as is stated by the following theorem.

Theorem 5.1. *Let $AF = (Ar, def)$ be an argumentation framework and $\mathcal{L}$ be a complete labelling of AF. Then* $\texttt{Lab2Mod}(\mathcal{L})$ *is a 3-valued stable model of P_{AF}.*

Proof. In order to prove $\texttt{Lab2Mod}(\mathcal{L})$ is a 3-valued stable model of P_{AF} we have to verify that $\texttt{Lab2Mod}(\mathcal{L})$ is a fixed point of Γ^*. We first examine $\frac{P_{AF}}{\texttt{Lab2Mod}(\mathcal{L})}$ (the reduct of P_{AF} under $\texttt{Lab2Mod}(\mathcal{L})$).

Let $A \leftarrow \text{not } B_1, \ldots, \text{not } B_n$ be a rule of P_{AF} (corresponding with an argument A that has defeaters $B_1, \ldots, B_n$). We distinguish three cases.

1. A is labelled $\texttt{in}$ by $\mathcal{L}$.
   Then from the fact that $\mathcal{L}$ is a complete labelling it follows that $B_1, \ldots, B_n$ are labelled $\texttt{out}$ by $\mathcal{L}$. The reduct of the rule is therefore $A \leftarrow t$, so in the smallest model of $\frac{P_{AF}}{\texttt{Lab2Mod}(\mathcal{L})}$, A will be *true* in $\Gamma^*(\texttt{Lab2Mod}(\mathcal{L}))$.

2. A is labelled $\texttt{out}$ by $\mathcal{L}$.
   Then, from the fact that $\mathcal{L}$ is a complete labelling it follows that there is a B_i $(1 \leq i \leq n)$ that is labelled $\texttt{in}$. The reduct of the rule is therefore $A \leftarrow v_1, \ldots, f, \ldots, v_n$ $(v_i \in \{t, f, u\})$ which is equivalent to $A \leftarrow f$. Since there is no other rule with A in the head, this means that in the smallest model of $\frac{P_{AF}}{\texttt{Lab2Mod}(\mathcal{L})}$, A will be *false* in $\Gamma^*(\texttt{Lab2Mod}(\mathcal{L}))$.

3. A is labelled $\texttt{undec}$ by $\mathcal{L}$.
   Then from the fact that $\mathcal{L}$ is a complete labelling it follows that not each $B_1, \ldots, B_n$ is labelled $\texttt{out}$ by $\mathcal{L}$ and there is no B_i $(i \leq i \leq n)$ that is labelled $\texttt{in}$ by $\mathcal{L}$. It also follows that there is at least one B_i labelled $\texttt{undec}$. The reduct of the rule is therefore $A \leftarrow, v_1, \ldots, u, \ldots, v_n$ $(v_i \in \{t, u\})$. Since this is the only rule that has A in the head, A will be *undefined* in $\Gamma^*(\texttt{Lab2Mod}(\mathcal{L}))$.

Since for any arbitrary argument A, it holds that $\texttt{Lab2Mod}(\mathcal{L})(A) = \Gamma^*(\texttt{Lab2Mod}(\mathcal{L}))(A)$, it follows that $\texttt{Lab2Mod}(\mathcal{L}) = \Gamma^*(\texttt{Lab2Mod}(\mathcal{L}))$. Hence $\texttt{Lab2Mod}(\mathcal{L})$ is a fixed point of Γ^*, so $\texttt{Lab2Mod}(\mathcal{L})$ is a 3-valued stable model of P_{AF}. □

Example 5.7. *Consider Example 5.4. Let $\mathcal{L} = \{(A, \texttt{in}), (B, \texttt{out}), (C, \texttt{undec}), (D, \texttt{undec}), (E, \texttt{undec})\}$. Then $\mathcal{L}$ is a complete labelling of the argumentation framework AF in Figure 5.2. Let $\mathcal{M} = \texttt{Lab2Mod}(\mathcal{L}) = \langle\{A\}; \{B\}\rangle$. Then the transformed program $\frac{P_{AF}}{\mathcal{M}}$ is:*

$$\begin{array}{lcl} A & \leftarrow & \textit{not } f \\ B & \leftarrow & \textit{not } t \\ C & \leftarrow & \textit{not } f, \textit{ not } u \\ D & \leftarrow & \textit{not } u \\ E & \leftarrow & \textit{not } u \end{array}$$

The least model of $\frac{P_{AF}}{\mathcal{M}}$ coincides with $\mathcal{M}$ which shows that $\mathcal{M}$ is a 3-valued stable model of P.

The next thing to be proved is that when an argumentation framework is transformed into a logic program, and $\mathcal{M}$ is a 3-valued stable model of this logic program, then $\texttt{Mod2Lab}(\mathcal{M})$ is a complete labelling of the original argumentation framework.

Theorem 5.2. *Let $AF = (Ar, def)$ be an argumentation framework and $\mathcal{M}$ be a 3-valued stable model of P_{AF}. Then* `Mod2Lab`$(\mathcal{M})$ *is a complete labelling of AF.*

Proof. Let $\mathcal{M}$ be a 3-valued stable model of P_{AF}. Then $\mathcal{M}$ is a fixed point of Γ^*, that is $\Gamma^*(\mathcal{M}) = \mathcal{M}$. We now prove that `Mod2Lab`$(\mathcal{M})$ is a complete labelling of AF.
Let A be an arbitrary argument in Ar. We distinguish three cases.

1. $\mathcal{M}(A) = t$.
   From the fact that $\Gamma^*(\mathcal{M}) = \mathcal{M}$ it follows that the reduct of the rule $A \leftarrow$ not $B_1, \dots,$ not B_n is equivalent to $A \leftarrow t$. This means that each B_i $(1 \leq i \leq n)$ is labelled `out` in `Mod2Lab`$(\mathcal{M})$. So A is legally `in` in `Mod2Lab`$(\mathcal{M})$.

2. $\mathcal{M}(A) = f$.
   From the fact that $\Gamma^*(\mathcal{M}) = \mathcal{M}$ it follows that the reduct of the rule $A \leftarrow$ not $B_1, \dots,$ not B_n is equivalent to $A \leftarrow f$. This implies that there exists a B_i $(1 \leq i \leq n)$ that is labelled `in` in `Mod2Lab`$(\mathcal{M})$. So A is legally `out` in `Mod2Lab`$(\mathcal{M})$.

3. $\mathcal{M}(A) = u$.
   From the fact that $\Gamma^*(\mathcal{M}) = \mathcal{M}$ it follows that the reduct of the rule $A \leftarrow$ not $B_1, \dots,$ not B_n is equivalent to $A \leftarrow u$. This implies that there exists a B_i $(1 \leq i \leq n)$ that is *undefined* in M and that each of the B_j $(1 \leq j \leq n, j \neq i)$ is either *false* or *undefined* in M. Hence, A has no defeaters that is labelled `in` by `Mod2Lab`$(\mathcal{M})$ and not all its defeaters are labelled `out` by `Mod2Lab`$(\mathcal{M})$. Thus A is legally `undec` in `Mod2Lab`$(\mathcal{M})$.

Since this holds for any arbitrary argument A, it follows that each argument that is `in` is legally `in`, each argument that is `out` is legally `out`, and each argument that is `undec` is legally `undec`. Hence, `Mod2Lab`$(\mathcal{M})$ is a complete labelling of AF. □

Example 5.8. *Consider Example 5.4. Let $\mathcal{M} = \langle\{A\};\{B\}\rangle$ be a 3-valued interpretation. Then the transformed program $\frac{P_{AF}}{M}$ is:*

$$\begin{array}{lcl} A & \leftarrow & \textit{not } f \\ B & \leftarrow & \textit{not } t \\ C & \leftarrow & \textit{not } f, \textit{ not } u \\ D & \leftarrow & \textit{not } u \\ E & \leftarrow & \textit{not } u \end{array}$$

The least model of $\frac{P_{AF}}{\mathcal{M}}$ coincides with M which shows that M is a 3-valued stable model of P. Let $\mathcal{L} =$ `Mod2Lab`$(\mathcal{M}) = \{(A, \texttt{in}), (B, \texttt{out}), (C, \texttt{undec}), (D, \texttt{undec}), (E, \texttt{undec})\}$. *Then we can see that $\mathcal{L}$ is a complete labelling of the argumentation framework AF in Figure 5.2.*

When `Lab2Mod` and `Mod2Lab` are restricted to work only on complete labellings and 3-valued stable models, they turn out to be bijective and each other's inverse.

Theorem 5.3. *Let $AF = (Ar, def)$ be an argumentation framework.*
Let `Lab2Mod`^{r} : $\{\mathcal{L} \mid \mathcal{L}$ *is a complete labelling of AF*$\} \rightarrow \{\mathcal{M} \mid \mathcal{M}$ *is a 3-valued stable model of P_{AF}*$\}$ *be a function defined by* `Lab2Mod`$^r(\mathcal{L}) =$ `Lab2Mod`$(\mathcal{L})$.
Let `Mod2Lab`^{r} : $\{\mathcal{M} \mid \mathcal{M}$ *is a 3-valued stable of model of P_{AF}*$\} \rightarrow \{\mathcal{L} \mid \mathcal{L}$ *is a complete labelling of AF*$\}$ *be a function defined by* `Mod2Lab`$^r(\mathcal{M}) =$ `Mod2Lab`$(\mathcal{M})$.
`Lab2Mod`^{r} *and* `Mod2Lab`^{r} *are bijective and are each other's inverse.*

Proof. As every function that has an inverse is bijective, we only need to prove that $\texttt{Lab2Mod}^r$ and $\texttt{Mod2Lab}^r$ are each other's inverse, meaning that $(\texttt{Lab2Mod}^r)^{-1} = \texttt{Mod2Lab}^r$ and $(\texttt{Mod2Lab}^r)^{-1} = \texttt{Lab2Mod}^r$. Let $AF = (Ar, \mathit{def})$ be an argumentation framework. We prove the following two things:

1. For every complete labelling $\mathcal{L}$ of AF
   it holds that $\texttt{Mod2Lab}^r(\texttt{Lab2Mod}^r(\mathcal{L})) = \mathcal{L}$.
   Let $\mathcal{L}$ be a complete labelling of AF and $A \in Ar$.
   If $\mathcal{L}(A) = \texttt{in}$ then A is t in $\texttt{Lab2Mod}^r(\mathcal{L})$,
   so $\texttt{Mod2Lab}^r(\texttt{Lab2Mod}^r(\mathcal{L}))(A) = \texttt{in}$.
   If $\mathcal{L}(A) = \texttt{out}$ then A is f in $\texttt{Lab2Mod}^r(\mathcal{L})$,
   so $\texttt{Mod2Lab}^r(\texttt{Lab2Mod}^r(\mathcal{L}))(A) = \texttt{out}$.
   If $\mathcal{L}(A) = \texttt{undec}$ then A is u in $\texttt{Lab2Mod}^r(\mathcal{L})$,
   so $\texttt{Mod2Lab}^r(\texttt{Lab2Mod}^r(\mathcal{L}))(A) = \texttt{undec}$.

2. For every 3-valued stable model $\mathcal{M}$ of P_{AF}
   it holds that $\texttt{Lab2Mod}^r(\texttt{Mod2Lab}^r(\mathcal{M})) = \mathcal{M}$.
   Let $\mathcal{M}$ be a 3-valued stable model $\mathcal{M}$ of P_{AF}.
   If $\mathcal{M}(A) = t$ then $\texttt{Mod2Lab}^r(A) = \texttt{in}$,
   so A is t in $\texttt{Lab2Mod}^r(\texttt{Mod2Lab}^r(\mathcal{M}))$.
   If $\mathcal{M}(A) = f$ then $\texttt{Mod2Lab}^r(A) = \texttt{out}$,
   so A is f in $\texttt{Lab2Mod}^r(\texttt{Mod2Lab}^r(\mathcal{M}))$.
   If $\mathcal{M}(A) = u$ then $\texttt{Mod2Lab}^r(A) = \texttt{undec}$,
   so A is u in $\texttt{Lab2Mod}^r(\texttt{Mod2Lab}^r(\mathcal{M}))$.

□

From Theorem 5.3, it follows that complete labellings of an argumentation framework and 3-valued stable models of the associated logic program are one-to-one related.

5.3.2 Transforming Logic Programs into Argumentation Frameworks

We show that the complete labellings coincide with 3-valued stable models if we transform logic programs into argumentation frameworks.

We follow the approach of structured arguments (which is taken in [80, 31, 78]) to do the transformation. This approach treats all rules equally in logic programs, i.e., there is no defeasible rule. The argumentation frameworks transformed from logic programs do not contain defeasible rules as they exist in ASPIC framework [78]. Hence, only rebutting defeat relation is considered in the transformed argumentation frameworks. Similarly, preferences among arguments are not considered either. So the following definition of arguments is a special case of the structured arguments of the ASPIC framework with only strict rules, only assumption-type premises and only assumption attack defeat relation.

In Definition 5.9 the reason for including the condition that each rule occurs no more than once in each root-originated branch is to make sure that a finite program will yield a finite number of arguments.[1]

[1]Without this condition the logic program $\{a \leftarrow;\ a \leftarrow a\}$ would yield an infinite number of arguments.

Definition 5.9. *Let P be a logic program.*

- *An argument A based on P is a finite tree of rules from P such that (1) each node (of the form $c \leftarrow a_1, \dots, a_n,$ not $b_1, \dots,$ not b_m with $n \geq 0$ and $m \geq 0$) has exactly n children, each having a different head $a_i \in \{a_1, \dots, a_n\}$ and (2) no rule occurs more than once in each root-originated branch of the tree. The conclusion of A* ($\texttt{Conc}(A)$) *is the head of its root.*
- *An argument a_1 defeats an argument a_2 iff a_1 has conclusion c and a_2 has a rule containing* `not` *c.*

We say that argument A is a subargument *of argument B iff A is a subtree of B.*

Definition 5.10 (Associated argumentation framework)**.** *Let P be a logic program. We define the associated argumentation framework $AF_P = (Ar, def)$ where Ar is the set of arguments that can be constructed using P, and def is the defeat relation under P.*

The procedure of transforming logic programs to argumentation frameworks corresponds with the first step (Section 2.2.1) of the three-step argumentation process (Section 2.2). It takes the logic programs as the underlying knowledge bases to generate a set of arguments and the defeat relation between these arguments according to Definition 5.9. The result of the first step is an argumentation framework (Definition 5.10)

Let's consider the logic program in Example 5.2.

Example 5.9. *We transform the logic program P in Example 5.2 into an argumentation framework to illustrate how to transform logic programs to argumentation frameworks. The argumentation framework in Figure 5.3 is the associated argumentation framework AF_P of the logic program P in Example 5.2 where*

$$\begin{array}{lrcl}
A: & paid & \leftarrow & \\
B: & angry & \leftarrow & work,\ not\ paid \\
 & & & \quad | \\
 & & & work \leftarrow not\ tired \\
C: & tired & \leftarrow & not\ sleep \\
D: & work & \leftarrow & not\ tired \\
E: & sleep & \leftarrow & not\ work
\end{array}$$

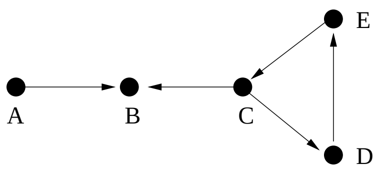


Figure 5.3: The associated argumentation framework of P in Example 5.2

After the first step of the three-step argumentation process, we obtain an argumentation framework from a logic program. The next thing is to determine the sets of arguments that can be accepted (step 2 (Section 2.2.2) of the three-step argumentation process) and then identify the sets of accepted conclusions (step 3 (Section 2.2.3) of the three-step argumentation process). In this chapter, we focus on the complete semantics. So in the second step, we determine the complete extensions of the argumentation frameworks. Then we identify the label of each conclusion in the third step. Since there could be more than one argument having the same conclusion in the argumentation frameworks transformed from logic programs, we define that the label of a conclusion is the "best" label of all arguments with the same conclusion with respect to the strict order on the labels $\{\texttt{out}, \texttt{undec}, \texttt{in}\}$ such that $\texttt{out} < \texttt{undec} < \texttt{in}$.

In order to convert labellings to 3-valued interpretations, we define a function that assigns a label to each atom. The idea is that the label of an atom should be the label of the best argument that yields the atom as a conclusion (or be `out` if there is no argument at all that yields this atom as a conclusion).

Definition 5.11. *Let P be a logic program and A_P be the set of all ground atoms that occur in P. Let $AF_P = (Ar, def)$ be the associated argumentation framework and $\mathcal{L}$ be a labelling of AF_P. We define a function $W(\mathcal{L}) : A_P \to \{\texttt{in}, \texttt{undec}, \texttt{out}\}$ such that $W(\mathcal{L})(c) = \max(\{\mathcal{L}(A) \mid A \in Ar \wedge \texttt{Conc}(A) = c\} \cup \{\texttt{out}\})$.*

Example 5.10. *Consider the following logic program P:*

$$\begin{array}{lll} a & \leftarrow & \\ b & \leftarrow & \\ c & \leftarrow & a, not\ b \\ c & \leftarrow & b \end{array}$$

The argumentation framework in Figure 5.4 is the associated argumentation framework of P where:

$$\begin{array}{llll} A: & a & \leftarrow & \\ B: & b & \leftarrow & \\ C: & c & \leftarrow & a,\ not\ b,\ not\ d \\ & & & \quad | \\ & & & a \leftarrow \\ D: & c & \leftarrow & b \\ & & & | \\ & & & b \leftarrow \end{array}$$

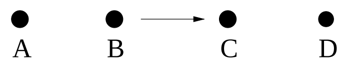


Figure 5.4: D is the best argument

There is a complete labelling $L = \{(A, \texttt{in}), (B, \texttt{in}), (C, \texttt{out}), (D, \texttt{in})\}$. Then $W(L)(a) = \texttt{in}$ and $W(L)(b) = \texttt{in}$. $W(L)(c) = \max(\texttt{in}, \texttt{out}) = \texttt{in}$ because $\texttt{Conc}(C) = c$ and $L(C) = \texttt{out}$ and $\texttt{Conc}(D) = c$ and $L(D) = \texttt{in}$. $W(L)(d) = \texttt{out}$ because there is no argument with conclusion d.

We now define two functions that, given a logic program P, allow a 3-valued interpretation to be converted to a labelling of AF_P and vice versa.

Definition 5.12. *Let $\mathcal{M}odels$ be all the 3-valued interpretations of P and $c \in A_P$. Let $\mathcal{L}abellings$ be the set of all labellings of AF_P and $\mathcal{L} \in \mathcal{L}abellings$. We introduce a function $\mathcal{L}ab2\mathcal{M}od : \mathcal{L}abellings \rightarrow \mathcal{M}odels$ such that $\mathcal{L}ab2\mathcal{M}od(\mathcal{L}) = \mathcal{M} = \langle T; F\rangle$ where*

- $T = \{c \mid c \in A_P \text{ and } W(\mathcal{L})(c) = \texttt{in}\}$;
- $F = \{c \mid c \in A_P \text{ and } W(\mathcal{L})(c) = \texttt{out}\}$;
- $\overline{\mathcal{M}} = \{c \mid c \in A_P \text{ and } W(\mathcal{L})(c) = \texttt{undec}\}$.

Example 5.11. *Consider the logic program P and the associated argumentation framework AF_P in Example 5.10. Let $L = \{(A, \texttt{in}), (B, \texttt{in}), (C, \texttt{out}), (D, \texttt{in})\}$ be a labelling of AF_P. Then $\mathcal{L}ab2\mathcal{M}od(L) = \langle\{a, b, c\}; \{d\}\rangle$.*

Definition 5.13. *Let $\mathcal{M}odels$ be all the 3-valued interpretations of P and $\mathcal{L}abellings$ be the set of all labellings of AF_P. Let $M \in \mathcal{M}odels$ and $M = \langle T, F\rangle$ and A be an argument in Ar. We define a function $\mathcal{M}od2\mathcal{L}ab : \mathcal{M}odels \rightarrow \mathcal{L}abellings$ such that*

1. *$\mathcal{M}od2\mathcal{L}ab(\mathcal{M})(A) = \texttt{in}$ if for each defeater B of A, $\texttt{Conc}(B) \in F$.*
2. *$\mathcal{M}od2\mathcal{L}ab(\mathcal{M})(A) = \texttt{out}$ if there is a defeater B of A such that $\texttt{Conc}(B) \in T$.*
3. *$\mathcal{M}od2\mathcal{L}ab(\mathcal{M})(A) = \texttt{undec}$ if not each defeater of A has a conclusion that is in F and there is no defeater B of A such that $\texttt{Conc}(B) \in T$.*

Example 5.12. *Consider the logic program P and the associated argumentation framework AF_P in Example 5.10. Let $M = \langle\{a, b, c\}; \{d\}\rangle$ be a 3-valued interpretation of P. Then $\mathcal{M}od2\mathcal{L}ab(\mathcal{M})(A) = \texttt{in}$, $\mathcal{M}od2\mathcal{L}ab(\mathcal{M})(B) = \texttt{in}$ and $\mathcal{M}od2\mathcal{L}ab(\mathcal{M})(D) = \texttt{in}$ because arguments A, B and D do not have any defeater. $\mathcal{M}od2\mathcal{L}ab(\mathcal{M})(C) = \texttt{out}$ because B defeats C and $\texttt{Conc}(B) = c \in T$.*

When a logic program is transformed into an argumentation framework, and $\mathcal{L}$ is a complete labelling of this argumentation framework, then $\mathcal{L}ab2\mathcal{M}od(\mathcal{L})$ is a 3-valued stable model of the logic program.

Theorem 5.4. *Let P be a logic program and $\mathcal{L}$ be a complete labelling of AF_P. Then $\mathcal{L}ab2\mathcal{M}od(\mathcal{L})$ is a 3-valued stable model of P.*

Proof. Let $\mathcal{M} = \mathcal{L}ab2\mathcal{M}od(\mathcal{L})$. In order to prove $\mathcal{M}$ is a 3-valued stable model of P we have to verify that $\mathcal{M}$ is a fixed point of Γ^*. We first examine $\frac{P}{\mathcal{M}}$ (the reduct of P under $\mathcal{M}$).

Let $A \in Ar$ and $c \leftarrow a_1, \ldots, a_n, \text{not } b_1, \ldots, \text{not } b_m$ $(n, m \geq 0)$ be the root of A. We distinguish three cases.

1. $c \in T$.
   This means that $W(\mathcal{L})(c) = \texttt{in}$. It follows that there exists an argument A such that A is labelled $\texttt{in}$ and $\texttt{Conc}(A) = c$. Then all defeaters of A are labelled $\texttt{out}$. Let $c' \leftarrow a'_1, \ldots, a'_k, \text{not } b'_1, \ldots, \text{not } b'_l$ $(k, l \geq 0)$ be an arbitrary rule of A. It follows that $W(\mathcal{L})(b'_j) = \texttt{out}$ $(0 \leq j \leq l)$. Then b'_j is *false* in $\mathcal{M}$. We prove by induction that all the conclusions of subarguments of A are *true* in $\Gamma^*(\mathcal{M})$.

- Basis. Let $c''' \leftarrow \text{not } b'''_1, \ldots, \text{not } b'''_{m'''}$ $(m''' \geq 0)$ be an arbitrary leaf in A such that the distance between the leaf and the root of A is the furthest. The distance between two nodes is the vertical distance between the two nodes in the direction from root down to leaves. Since $b'''_{j'''}$ $(0 \leq j''' \leq m''')$ is *false* in $\mathcal{M}$, then the reduct of the leaf is $c''' \leftarrow t$. So in the least model of $\frac{P}{\mathcal{M}}$, c''' will be *true* in $\Gamma^*(\mathcal{M})$.
- Step. Let $a''_1, \ldots, a''_{n''}$ $(n'' \geq 0)$ be heads of nodes such that the distance between them and the furthest leaf is n'. Assume that $a''_1, \ldots, a''_{n''}$ are *true* in $\Gamma^*(\mathcal{M})$. Let $c'' \leftarrow a''_{i''}, \ldots, a''_{j''}, \text{not } b''_1, \ldots, \text{not } b''_{m''}$ $(0 \leq i'', j'' \leq n'', m'' \geq 0)$ be a node that is $n' + 1$ distance from the furthest leaf in the tree of A. Since $b''_1, \ldots,\ b''_{m''}$ are *false* in $\mathcal{M}$ and $a''_{i''}, \ldots, a''_{j''}$ are *true* in $\mathcal{M}$, the reduct of the node is $c'' \leftarrow t$. So in the least model of $\frac{P}{\mathcal{M}}$, c'' will be *true* in $\Gamma^*(\mathcal{M})$.

So all the conclusions of subarguments of A are *true* in $\Gamma^*(\mathcal{M})$. Therefore in the least model of $\frac{P}{\mathcal{M}}$, c will be *true* in $\Gamma^*(\mathcal{M})$.

2. $c \in F$.
This means that $W(\mathcal{L})(c) = \texttt{out}$. It follows that for all arguments A such that $\texttt{Conc}(A) = c$, A is labelled `out`. Then there exists a defeater of A that is labelled `in`. It follows that there exists a rule $c' \leftarrow a'_1, \ldots, a'_k, \text{not } b'_1, \ldots, \text{not } b'_l$ $(k \geq 0, l \geq 1)$ in A such that $W(\mathcal{L})(b'_j) = \texttt{in}$ $(1 \leq j \leq l)$. Then b'_j is *true* in $\mathcal{M}$. Then the reduct of the root of A is the $c \leftarrow f$. So in the least model of $\frac{P}{\mathcal{M}}$, c will be *false* in $\Gamma^*(\mathcal{M})$.

3. $c \in \overline{\mathcal{M}}$.
This means that $W(\mathcal{L})(c) = \texttt{undec}$. It follows that there exists an argument A such that $\texttt{Conc}(A) = c$ and $\mathcal{L}(A) = \texttt{undec}$ and there is no argument A such that $\texttt{Conc}(A) = c$ and $\mathcal{L}(A) = \texttt{in}$. Then not each defeater of A is labelled `out` and there is no defeater that is labelled `in`. It follows that there exists a rule $c'''' \leftarrow a''''_1, \ldots, a''''_{k'}, \text{not } b''''_1, \ldots, \text{not } b''''_{l'}$ $(k', l' \geq 0)$ in A such that $W(\mathcal{L})(b''''_j) = \texttt{undec}$ $(1 \leq j \leq l')$. Then b''''_j is *undefined* in $\mathcal{M}$. Let $c' \leftarrow a'_1, \ldots, a'_k, \text{not } b'_1, \ldots, \text{not } b'_l$ $(k \geq 0, l \geq 1)$ be an arbitrary rule in A. Since there is no defeater that is labelled `in`, then for each b'_i $(0 \leq i \leq l)$, b'_i is not *true* in $\mathcal{M}$. We prove that the conclusions of subarguments of A are *undefined* in $\Gamma^*(\mathcal{M})$ if they have defeaters that are labelled `undec` by induction.

- Basis. Let $c''' \leftarrow \text{not } b'''_1, \ldots, \text{not } b'''_{m'''}$ $(m''' \geq 0)$ be an arbitrary leaf in A such that the distance between the leaf and the root of A is the furthest. Since $b'''_{j'''}$ $(0 \leq j''' \leq m''')$ is either *false* or *undefined* in $\mathcal{M}$, then the reduct of the leaf is $c''' \leftarrow v_1, \ldots, v_{m'''}$ $(v_i \in \{t, u\}, 0 \leq i \leq m''')$. If the leaf has no defeater that is labelled `undec` the reduct of the leaf is $c''' \leftarrow t$. So in the least model of $\frac{P}{\mathcal{M}}$, c''' will be *true* in $\Gamma^*(\mathcal{M})$. If the leaf has a defeater that is labelled `undec` the reduct of the leaf is $c''' \leftarrow v_1, \ldots, u, \ldots, v_{m'''}$ $(v_i \in \{t, u\}, 0 \leq i \leq m''')$. So in the least model of $\frac{P}{\mathcal{M}}$, c''' will be *undefined* in $\Gamma^*(\mathcal{M})$.
- Step. Let $A''_1, \ldots, A''_{n''}$ $(n'' \geq 0)$ be subarguments of A and their roots are n' distance from the furthest leaf. Let $a''_1, \ldots, a''_{n''}$ be conclusions of $A''_1, \ldots, A''_{n''}$ respectively. Assume that a''_i $(0 \leq i \leq n'')$ is *true* in $\Gamma^*(\mathcal{M})$ if A''_i does not have a defeater that is labelled `undec` and a''_i is *undefined* in $\Gamma^*(\mathcal{M})$ if A''_i has a

defeater that is labelled `undec`.
Let $c'' \leftarrow a''_{i''}, \ldots, a''_{j''}, \text{not } b''_1, \ldots, \text{not } b''_{m''}$ $(0 \leq i'', j'' \leq n'', m'' \geq 0)$ be a node that is $n' + 1$ distance from the furthest leaf in the tree of A. $b''_1, \ldots, b''_{m''}$ are either *true* or *undefined* in $\mathcal{M}$ and $a''_{i''}, \ldots, a''_{j''}$ are either *true* or *undefined* in $\mathcal{M}$. Then the reduct of the node is $c'' \leftarrow v_1, \ldots, v_{j''-i''+1+m''}$ $(v_i \in \{t, u\}, 0 \leq i \leq j'' - i'' + 1 + m'')$. If there is a defeater that is labelled `undec` the reduct of the node is $c'' \leftarrow v_1, \ldots, u, \ldots, v_{j''-i''+1+m''}$ $(v_i \in \{t, u\}, 0 \leq i \leq j''-i''+1+m'')$. So in the least model of $\frac{P}{\mathcal{M}}$, c'' will be *undefined* in $\Gamma^*(\mathcal{M})$ if the subargument has a defeater that is labelled `undec`.

A has a defeater that is labelled `undec` and A is a subargument of A. Then in the least model of $\frac{P}{\mathcal{M}}$, c will be *undefined* in $\Gamma^*(\mathcal{M})$.

Since for any arbitrary atom c, it holds that $\mathcal{M}(c) = \Gamma^*(\mathcal{M})(c)$, it follows that $\mathcal{M} = \Gamma^*(\mathcal{M})$. Hence $\mathcal{M}$ is a fixed point of Γ^*, so $\mathcal{M}$ is a 3-valued stable model of P. □

Let's consider the logic program P in Example 5.2 and the associated argumentation framework AF_P in Example 5.9.

Example 5.13. *Let $\mathcal{L}$ be one of the complete labellings of the argumentation framework in Example 5.9 and* $\mathcal{L} = \{(A, \texttt{in}), (B, \texttt{out}), (C, \texttt{undec}), (D, \texttt{undec}), (E, \texttt{undec})\}$. *We transform $\mathcal{L}$ to a 3-valued stable interpretation of the logic program P. Let $\mathcal{M} = \langle T; F \rangle = \mathcal{L}ab2\mathcal{M}od(\mathcal{L})$. Then $T = \{\texttt{Conc}(A)\} = \{paid\}$ and $F = \{\texttt{Conc}(B)\} = \{angry\}$. So $\mathcal{M} = \langle$ paid, angry $\rangle$. From Example 5.2, $\mathcal{M}$ is a 3-valued stable model of P.*

When $\mathcal{M}$ is a 3-valued stable model of a logic program, then $\mathcal{M}od2\mathcal{L}ab(\mathcal{M})$ is a complete labelling of the associated argumentation framework, as is stated by the following theorem.

Theorem 5.5. *Let P be a logic program and $\mathcal{M} = \langle T; F \rangle$ be a 3-valued stable model of P. Let $\mathcal{L} = \mathcal{M}od2\mathcal{L}ab(\mathcal{M})$ and c be a ground atom. Then $\mathcal{L}$ is a complete labelling of AF_P such that $W(\mathcal{L})(c) = \texttt{in}$ if $c \in T$, $W(\mathcal{L})(c) = \texttt{out}$ if $c \in F$ and $W(\mathcal{L})(c) = \texttt{undec}$ if $c \in \overline{\mathcal{M}}$.*

Proof. $\mathcal{M}$ is a 3-valued stable model of P. Then $\mathcal{M}$ is a fixed point of Γ^*, that is $\Gamma^*(\mathcal{M}) = \mathcal{M}$. Let A be an argument in Ar. We now prove that $\mathcal{L}$ is a complete labelling of AF_P. We distinguish three cases.

1. $\mathcal{L}(A) = \texttt{in}$.
   According to Definition 5.13, for each defeater B of A, $\texttt{Conc}(B) \in F$. Let $c \leftarrow a_1, \ldots, a_m, \text{not } b_1, \ldots, \text{not } b_n$ be the root of B. Then $c \in F$. From the fact that $\Gamma^*(\mathcal{M}) = \mathcal{M}$ it follows that reduct of the rule is equivalent to $c \leftarrow f$. Then there exists a rule $c' \leftarrow a'_1, \ldots, a'_k, \text{not } b'_1, \ldots, \text{not } b'_l$ $(k \geq 0, l \geq 1)$ in B such that there is a $b'_j \in T$ $(1 \leq j \leq l)$. It follows that B has a defeater whose conclusion is in T which implies B is labelled `out`. Since this holds for an arbitrary defeater B of A it follows that each defeater of A is labelled `out`. So A is legally `in` in $\mathcal{M}od2\mathcal{L}ab(\mathcal{M})$.

2. $\mathcal{L}(A) = \texttt{out}$.
   According to Definition 5.13, there exists a defeater B of A such that $\texttt{Conc}(B) \in T$. Then from the fact that $\Gamma^*(\mathcal{M}) = \mathcal{M}$ it follows that the reduct of the root of B is $\texttt{Conc}(B) \leftarrow t$. Let $c \leftarrow a_1, \ldots, a_m, \text{not } b_1, \ldots, \text{not } b_n$ be the root of B and

$c' \leftarrow a'_1, \ldots, a'_k, \text{not } b'_1, \ldots, \text{not } b'_l$ $(k, l \geq 0)$ be an arbitrary rule of B. So $c \in T$ and $c \leftarrow t$. Then for all b'_j $(0 \leq j \leq l)$, $b'_j \in F$. It follows that the conclusions of all defeaters of B are in F which implies B is labelled `in`. So A is legally `out` in $\mathcal{M}od2\mathcal{L}ab(\mathcal{M})$.

3. $\mathcal{L}(A) = \texttt{undec}$.
According to Definition 5.13, there is no defeater of A that has a conclusion that is in T and not all defeaters of A have a conclusion that is in F.

   (a) Assume there is a defeater B of A that is labelled `in`. Let $c \leftarrow a_1, \ldots, a_m, \text{not } b_1, \ldots, \text{not } b_n$ $(m, n \geq 0)$ be the root of B and $c' \leftarrow a'_1, \ldots, a'_k, \text{not } b'_1, \ldots, \text{not } b'_l$ $(k, l \geq 0)$ be an arbitrary rule of B. According to Definition 5.13, the conclusion of each defeater of B is *false* in $\mathcal{M}$. Then for all b'_j $(0 \leq j \leq l)$, $b'_j \in F$. So the reduct of the root of B is equivalent to $c \leftarrow t$. From the fact that $\Gamma^*(\mathcal{M}) = \mathcal{M}$ it follows that $c \in T$. Contradiction.

   (b) Assume all defeaters of A are labelled `out`. Let B be an arbitrary defeater of A and $c \leftarrow a_1, \ldots, a_m, \text{not } b_1, \ldots, \text{not } b_n$ be the root of B. According to Definition 5.13, there exists a defeater of B whose conclusion is *true* in $\mathcal{M}$. Then there exists a rule $c' \leftarrow a'_1, \ldots, a'_k, \text{not } b'_1, \ldots, \text{not } b'_l$ $(k \geq 0, l \geq 1)$ in B such that there is a $b'_j \in T$ $(1 \leq j \leq l)$. Then the reduct of the root of B is equivalent to $c \leftarrow f$. From the fact that $\Gamma^*(\mathcal{M}) = \mathcal{M}$ it follows that $c \in F$. Then each defeater of A has a conclusion that is in F. Contradiction.

Therefore there is no defeater of A that is labelled `in` and not all defeaters of A are labelled `out`. So A is legally `undec` in $\mathcal{M}od2\mathcal{L}ab(\mathcal{M})$.

Since this holds for any arbitrary argument A, it follows that each argument that is `in` is legally `in`, each argument that is `out` is legally `out`, and each argument that is `undec` is legally `undec`. Hence, $\mathcal{L}$ is a complete labelling of AF_P.

The next things to be proved is that (1) if $c \in T$ then $W(\mathcal{L})(c) = \texttt{in}$, (2) if $c \in F$ then $W(\mathcal{L})(c) = \texttt{out}$ and (3) if $c \in \overline{\mathcal{M}}$ then $W(\mathcal{L})(c) = \texttt{undec}$.

1. If $c \in T$.
From the fact that $\Gamma^*(\mathcal{M}) = \mathcal{M}$ it follows that there is a rule whose reduct is equivalent to $c \leftarrow t$. Let this rule be the root of an argument A which implies that $\texttt{Conc}(A) = c$. Let $c \leftarrow a_1, \ldots, a_m, \text{not } b_1, \ldots, \text{not } b_n$ $(m, n \geq 0)$ be the root of A and $c' \leftarrow a'_1, \ldots, a'_k, \text{not } b'_1, \ldots, \text{not } b'_l$ $(k, l \geq 0)$ be an arbitrary rule of A. Then for all b'_j $(0 \leq j \leq l)$, $b'_j \in F$. It follows that the conclusions of all defeaters of A are in F which implies that A is labelled `in`. So $\mathcal{L}(A) = \texttt{in} = \max(\{\mathcal{L}(A') \mid A' \in Ar \wedge \texttt{Conc}(A') = c\} \cup \{\texttt{out}\})$. Then $W(\mathcal{L})(c) = \texttt{in}$.

2. If $c \in F$.
From the fact that $\Gamma^*(\mathcal{M}) = \mathcal{M}$ it follows that each rule with c in the head has the reduct $c \leftarrow f$. Let $A \in Ar$ be an arbitrary argument such that $\texttt{Conc}(A) = c$ and $c \leftarrow a_1, \ldots, a_m, \text{not } b_1, \ldots, \text{not } b_n$ $(m, n \geq 0)$ be the root of A. Then the reduct of the root of A is equivalent to $c \leftarrow f$. Then there exists a rule $c' \leftarrow a'_1, \ldots, a'_k, \text{not } b'_1, \ldots, \text{not } b'_l$ $(k \geq 0, l \geq 1)$ in A such that there is a $b'_j \in T$ $(1 \leq j \leq l)$. It follows that A has a defeater whose conclusion is in T which implies A

is labelled $\texttt{out}$. It follows that each argument A such that $\texttt{Conc}(A) = c$ is labelled $\texttt{out}$. So $\mathcal{L}(A) = \texttt{out} = \max(\{\mathcal{L}(A') \mid A' \in Ar \wedge \texttt{Conc}(A') = c\} \cup \{\texttt{out}\})$. Then $W(\mathcal{L})(c) = \texttt{out}$.

3. If $c \in \overline{\mathcal{M}}$ then there is no rule whose reduct is $c \leftarrow t$ and there is a rule whose reduct is $c \leftarrow u$. It follows that there is no argument A such that $\texttt{Conc}(A) = c$ is labelled $\texttt{in}$ and there is an argument A such that $\texttt{Conc}(A) = c$ is labelled $\texttt{undec}$. So $\mathcal{L}(A) = \texttt{undec} = \max(\{\mathcal{L}(A') \mid A' \in Ar \wedge \texttt{Conc}(A') = c\} \cup \{\texttt{out}\})$. Then $W(\mathcal{L})(c) = \texttt{undec}$.

$\square$

Let's consider the logic program P in Example 5.2 and the associated argumentation framework AF_P in Example 5.9.

Example 5.14. *Let $\mathcal{M} = \langle$ paid, angry $\rangle$ be a 3-value stable interpretation of the logic program in Example 5.2. From Example 5.2, $\mathcal{M}$ is a 3-valued stable model of P. We transform $\mathcal{M}$ to a labelling $\mathcal{L}$ of the associated argumentation framework AF_P of P. Let $\mathcal{L} = \mathcal{M}od2\mathcal{L}ab(\mathcal{M})$. Then $\mathcal{L} = \{(A, \texttt{in}), (B, \texttt{out}), (C, \texttt{undec}), (D, \texttt{undec}), (E, \texttt{undec})\}$. We can see that $\mathcal{L}$ is a complete labelling of AF_P.*

When $\mathcal{L}ab2\mathcal{M}od$ and $\mathcal{M}od2\mathcal{L}ab$ are restricted to work only on complete labellings and 3-valued stable models, they turn out to be bijective and each other's inverse.

Theorem 5.6. *Let P be a logic program and AF_P be the associated argumentation framework. Let $\mathcal{L}ab2\mathcal{M}od^r : \{\mathcal{L} \mid \mathcal{L}$ is a complete labelling of $AF_P\} \to \{\mathcal{M} \mid \mathcal{M}$ is a 3-valued stable model of $P\}$ be a function defined by $\mathcal{L}ab2\mathcal{M}od^r(\mathcal{L}) = \mathcal{L}ab2\mathcal{M}od(\mathcal{L})$. Let $\mathcal{M}od2\mathcal{L}ab^r : \{\mathcal{M} \mid \mathcal{M}$ is a 3-valued stable of model of $P\} \to \{\mathcal{L} \mid \mathcal{L}$ is a complete labelling of $AF_P\}$ be a function defined by $\mathcal{M}od2\mathcal{L}ab^r(\mathcal{M}) = \mathcal{M}od2\mathcal{L}ab(\mathcal{M})$. $\mathcal{L}ab2\mathcal{M}od^r$ and $\mathcal{M}od2\mathcal{L}ab^r$ are bijective and are each other's inverses.*

Proof. As every function that has an inverse is bijective, we only need to prove that $\mathcal{L}ab2\mathcal{M}od^r$ and $\mathcal{M}od2\mathcal{L}ab^r$ are each other's inverses. That is $(\mathcal{L}ab2\mathcal{M}od^r)^{-1} = \mathcal{M}od2\mathcal{L}ab^r$ and $(\mathcal{M}od2\mathcal{L}ab^r)^{-1} = \mathcal{L}ab2\mathcal{M}od^r$. Let $AF_P = (Ar, def)$ be an argumentation framework. We prove the following two things:

1. For every 3-valued stable model $\mathcal{M}$ of P it holds that
   $\mathcal{L}ab2\mathcal{M}od^r(\mathcal{M}od2\mathcal{L}ab^r(\mathcal{M})) = \mathcal{M}$.
   Let $\mathcal{M}$ be a 3-valued stable model $\mathcal{M}$ of P.
   If $\mathcal{M}(c) = t$ then $W(\mathcal{M}od2\mathcal{L}ab^r)(c) = \texttt{in}$ (Theorem 5.5),
   so c is *true* in $\mathcal{L}ab2\mathcal{M}od^r(\mathcal{M}od2\mathcal{L}ab^r(\mathcal{M}))$.
   If $\mathcal{M}(c) = f$ then $W(\mathcal{M}od2\mathcal{L}ab^r)(c) = \texttt{out}$ (Theorem 5.5),
   so c is *false* in $\mathcal{L}ab2\mathcal{M}od^r(\mathcal{M}od2\mathcal{L}ab^r(\mathcal{M}))$.
   If $\mathcal{M}(c) = u$ then $W(\mathcal{M}od2\mathcal{L}ab^r)(c) = \texttt{undec}$ (Theorem 5.5),
   so c is *undefined* in $\mathcal{L}ab2\mathcal{M}od^r(\mathcal{M}od2\mathcal{L}ab^r(\mathcal{M}))$.

2. For every complete labelling $\mathcal{L}$ of AF_P it holds that
   $\mathcal{M}od2\mathcal{L}ab^r(\mathcal{L}ab2\mathcal{M}od^r(\mathcal{L})) = \mathcal{L}$.
   Let $\mathcal{L}$ be a complete labelling of AF_P and let $A \in Ar$.
   If $\mathcal{L}(A) = \texttt{in}$ then each defeater of A is labelled $\texttt{out}$. Let B be an arbitrary defeater

of A and $\texttt{Conc}(B) = b$, then $\mathcal{L}(B) = \texttt{out} = \max(\{\mathcal{L}(B') \mid B' \in Ar \wedge \texttt{Conc}(B') = b\} \cup \{\texttt{out}\})$. Then $W(\mathcal{L})(b) = \texttt{out}$. So $\mathcal{L}ab2\mathcal{M}od^r(\mathcal{L})(b) = f$. Then for each defeater B of A, $\texttt{Conc}(B) = b$ is *false* in $\mathcal{L}ab2\mathcal{M}od^r(\mathcal{L})$. It follows from Definition 5.13 that A is labelled $\texttt{in}$ in $\mathcal{M}od2\mathcal{L}ab^r(\mathcal{L}ab2\mathcal{M}od^r(\mathcal{L}))$. So $\mathcal{M}od2\mathcal{L}ab^r(\mathcal{L}ab2\mathcal{M}od^r(\mathcal{L}))(A) = \texttt{in}$.

If $\mathcal{L}(A) = \texttt{out}$ then there is a defeater B of A that is labelled $\texttt{in}$. Let $\texttt{Conc}(B) = b$, then $\mathcal{L}(B) = \texttt{in} = \max(\{\mathcal{L}(B') \mid B' \in Ar \wedge \texttt{Conc}(B') = b\} \cup \{\texttt{out}\})$. Then $W(\mathcal{L})(b) = \texttt{in}$. So $\mathcal{L}ab2\mathcal{M}od^r(\mathcal{L})(b) = t$. Then there is a defeater B of A such that $\texttt{Conc}(B) = b$ is *true* in $\mathcal{L}ab2\mathcal{M}od^r(\mathcal{L})$. It follows from Definition 5.13 that A is labelled $\texttt{out}$ in $\mathcal{M}od2\mathcal{L}ab^r(\mathcal{L}ab2\mathcal{M}od^r(\mathcal{L}))$. So $\mathcal{M}od2\mathcal{L}ab^r(\mathcal{L}ab2\mathcal{M}od^r(\mathcal{L}))(A) = \texttt{out}$.

If $\mathcal{L}(A) = \texttt{undec}$ then there is a defeater B of A that is labelled $\texttt{undec}$ and there is no defeater of A that is labelled $\texttt{in}$. Let $\texttt{Conc}(B) = b$, then $\mathcal{L}(B) = \texttt{undec} = \max(\{\mathcal{L}(B') \mid B' \in Ar \wedge \texttt{Conc}(B') = b\} \cup \{\texttt{out}\})$. Then $W(\mathcal{L})(b) = \texttt{undec}$. So $\mathcal{L}ab2\mathcal{M}od^r(\mathcal{L})(b) = u$. Then there is a defeater B of A such that $\texttt{Conc}(B) = b$ is *undefined* in $\mathcal{L}ab2\mathcal{M}od^r(\mathcal{L})$.
Assume there is a defeater C of A such that $\texttt{Conc}(C)$ is *true* in $\mathcal{L}ab2\mathcal{M}od^r(\mathcal{L})$. Let $\texttt{Conc}(C) = b'$, then $W(\mathcal{L})(b') = \texttt{in}$. It follows that $\texttt{in} = \max(\{\mathcal{L}(B') \mid B' \in Ar \wedge \texttt{Conc}(B') = b'\} \cup \{\texttt{out}\})$. Then there is a defeater (C) of A such that $\texttt{Conc}(C) = b'$ and $\mathcal{L}(C) = \texttt{in}$. Then from the fact that $\mathcal{L}$ is a complete labelling it follows that $\mathcal{L}(A) = \texttt{out}$. Contradiction.
So there is no defeater of A has a conclusion that is *true* in $\mathcal{L}ab2\mathcal{M}od^r(\mathcal{L})$ and not each defeater of A has a conclusion that is *false* in $\mathcal{L}ab2\mathcal{M}od^r(\mathcal{L})$. So $\mathcal{M}od2\mathcal{L}ab^r($ $\mathcal{L}ab2\mathcal{M}od^r(\mathcal{L}))(A) = \texttt{undec}$.

□

From Theorem 5.3 and Theorem 5.6, it follows that complete labellings and 3-valued stable models are one-to-one related. Since Theorem 2.4 states that complete extensions and complete labellings are one-to-one related, it follows that complete extensions, complete labellings and 3-valued stable models are different ways of describing essentially the same concept.

5.4 Related Work

Dung [40] has introduced a number of argumentation semantics, including complete, grounded, preferred and stable semantics. Then Verheij [92] and Caminada [25] has introduced semi-stable semantics. Dung [40] has shown that the stable extensions in abstract argumentation coincide with the stable models in logic programming and that the grounded extension in abstract argumentation coincides with the well-founded model in logic programming. Dung's work shed a light on the connection between abstract argumentation and logic programming. On the other hand, Eiter *et al.* [43] show the relation between various semantics of logic programming. Every stable model [46] is a L-stable model [88, 87]. Every L-stable model is a regular model [101, 102, 103]. Every regular model is a 3-valued stable model and each well-founded model [45] is a 3-valued stable model. The relation between these semantics in logic programming is similar to the relation between semantics in abstract argumentation. The above works lay a solid foundation for our work in this chapter.

5.5 Summary and Future Work

The results presented in this chapter show that the complete labellings are semantically equivalent to 3-valued stable models.

We transformed argumentation frameworks into logic programs and proved that the complete labellings of an argumentation framework coincide with 3-valued stable models of the associated logic programs. We can obtain the same correspondence between complete labellings and 3-valued stable models if we transform logic programs into argumentation frameworks.

Since complete extensions and 3-valued stable models are both used as bases for describing other semantics in abstract argumentation and logic programming, the currently proved equivalence between complete semantics and 3-valued stable model semantics could perhaps be used to prove other equivalences as well, between argumentation and logic programming semantics.

One topic for further study would be the possible correspondence between the preferred extensions in abstract argumentation [25] and the regular models [43] in logic programming.

Definition 5.14 (Regular model ([103])). *A 3-valued stable model $M = \langle T; F\rangle$ of a logic program P is a regular model of P if there exists no 3-valued stable model $M' = \langle T'; F'\rangle$ of P such that $\overline{M'} \subset \overline{M}$, $T \subseteq T'$ and $F \subseteq F'$.*

From the definition of a regular model, we see that the regular models and the preferred extension share a common property which is that they are both the maximal sets. The preferred extensions are maximal complete extensions. A regular model is a 3-valued stable model that has a maximal set of T and a maximal set of F. We have shown that the complete extensions coincide with 3-valued stable model. Then it would be convenient to prove that the regular models in logic programming coincide with the preferred extensions in abstract argumentation.

Chapter 6

Implementing Crash-resistance and Non-interference in ASPIC Lite

Over the last decennia, many systems for formal argumentation have been defined. The problem, however, is that these systems do not always satisfy reasonable properties. In the current chapter, we focus on the particular property that a conflict between two arguments cannot keep other unrelated arguments from becoming justified. Although this property appears obvious, it is in fact violated by several existing argumentation formalisms. In this chapter we examine what exactly goes wrong and how things can be improved.

6.1 Introduction

As illustrated in Figure 6.1 (the same figure as 2.2), the argumentation approach provides a graph based way of performing non-monotonic reasoning. An interesting phenomenon is that the non-monotonicity is isolated purely in step 2 of the process. Step 1 is monotonic (having additional information in the knowledge base yields an argumentation framework with zero or more additional vertices and edges), just like step 3 is monotonic (having additional arguments in an argument-based extension yields an associated conclusion-based extension with zero or more additional conclusions). Step 2, however, is non-monotonic because adding new arguments and defeats can change the status of arguments that were already present in the argumentation framework when it comes to determining the argument-based extensions. That is, when adding new arguments and defeats it is by no means guaranteed that the resulting argument-based extensions will be supersets of the previous argument-based extensions. Apart from isolating non-monotonicity in step 2, the argumentation approach to non-monotonic reasoning also offers the advantage of different levels of abstraction. The field of abstract argumentation, for instance, only studies step 2 of the overall argumentation process and has now become one of the most popular topics in argumentation research.

Despite its advantages, the argumentation approach to non-monotonic reasoning also has important difficulties that are often overlooked by those studying purely abstract argumentation. The point is that in step 1 of the overall argumentation process, one constructs arguments that have a logical content. Yet, in step 2, one selects the sets of accepted arguments (argument-based extensions) purely based on some topological principle of the resulting graph, without looking what is actually inside of the arguments. The abstract

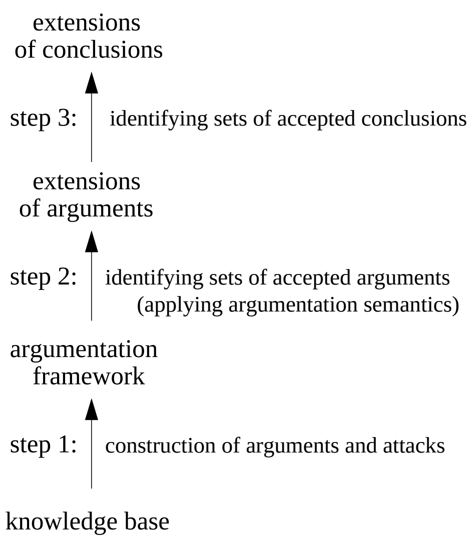

extensions of conclusions
step 3:
identifying sets of accepted conclusions
extensions of arguments
step 2:
identifying sets of accepted arguments (applying argumentation semantics)
argumentation framework
step 1:
construction of arguments and attacks
knowledge base

Figure 6.1: Argumentation for inference

level (step 2) is essentially about how to apply a semantics "blindly", without looking at the logical content of the arguments. But if one cannot see what is inside of the arguments, then how can one make sure that the selected set of arguments makes sense from a logical perspective? For instance, how can one be sure that the conclusions yielded by these sets of arguments (step 3) will be consistent? Or, alternatively, how does one know that these conclusions will actually be closed under logical entailment?

Issues like that of consistency and closure of argumentation-based entailment cannot be handled purely at the level of any of the individual three steps in the overall argumentation process. Instead, they require a carefully selected *combination* of how to carry out *each* of these individual steps. For instance, Caminada and Amgoud [30, 31] point out that when applying the argumentation process to a knowledge base consisting of strict and defeasible rules, one can obtain closure and consistency of the resulting conclusions by applying transposition and restricted rebut when constructing the argumentation framework (step 1), in combination with any admissibility-based argumentation semantics (step 2). Under these conditions, the conclusions associated with the argument-based extensions (step 3) will be consistent and closed under the strict rules [31].

Caminada and Amgoud introduce three postulates that they aim to satisfy for argument-based entailment: *direct consistency*, *indirect consistency* and *closure* [31]. The current chapter extends this line of research by examining two additional postulates: *crash resistance* and *non-interference*. It is explained why these postulates matter, and how they are in fact violated by several well-known formalisms for argument-based entailment (including Pollock's OSCAR system [71]). Our findings are relevant for other formalisms that aim to combine classical logic with defeasible rules, such as [71, 78, 86]. Furthermore, we provide a general way of satisfying the postulates of Caminada and Amgoud [30, 31] as well as the additional postulates introduced in this chapter, in the context of argumentation formalisms that apply defeasible argument schemes in combination with classical logic.

The current chapter is structured as follows. First, in Section 6.2, we introduce the ASPIC Lite system. Then in Section 6.3, the postulates of non-interference, contamination, non-triviality and crash resistance are introduced, and it is examined how these are violated by formalisms like ASPIC Lite, OSCAR [71] and ASPIC [78]. In Section 6.4, we provide a general solution to satisfy both the postulates introduced in [30, 31] (direct consistency, indirect consistency and closure) and the postulates introduced in the current chapter (non-trivial, crash resistance and non-interference [34]). The discussion is then rounded off with some concluding remarks in Section 6.6.

6.2 The ASPIC Lite System

In this section, we introduce the ASPIC Lite system.

Let $\mathcal{L}$ be a propositional language. Based on Definition 2.2, we extend defeasible rules by explicitly showing all undercutters of the defeasible rules.

Definition 6.1. *A* defeasible rule *is of the form* $\varphi_1, \ldots, \varphi_n \underset{U}{\Rightarrow} \psi$ *where* $U \subseteq \mathcal{L}$*.*

This form of defeasible rules indicates that if $\neg u$ $(u \in U)$ holds, then the defeasible rule $\varphi_1, \ldots, \varphi_n \underset{U}{\Rightarrow} \psi$ is inapplicable.

In this chapter, we use defeasible rules of the form in Definition 6.1. The definition of undercutting becomes the following.

Definition 6.2. *An argument A* undercuts *argument B on B' iff* $\texttt{Conc}(A) = \neg u$ *where* $u \in U$ *for some* $B' \in \texttt{Sub}(B)$ *of the form* $B''_1, \ldots, B''_n \underset{U}{\Rightarrow} \psi$*.*

In this chapter, we use undercutting of Definition 6.2 instead of the one of Definition 2.8.

The first step of the three-step argumentation process is constructing an argumentation framework from a given knowledge base.

Definition 6.3 (Knowledge base)**.** *A* knowledge base *in the ASPIC Lite system is a pair* $(\mathcal{P}, \mathcal{D})$ *where* $\mathcal{P} \subseteq \mathcal{L}$ *and* $\mathcal{P}$ *is propositionally consistent and* $\mathcal{D}$ *is a set of defeasible rules.*

In the rest of the chapter we use $\texttt{Atoms}(\mathcal{F})$ for the atoms that occur in a set of formulas $\mathcal{F}$ or the atoms that occur in a set of sets of formulas $\mathcal{F}$. For instance: $\texttt{Atoms}(\{a \supset b; b \supset c\}) = \{a, b, c\}$. Furthermore, if At is a set of atoms and $\mathcal{F}$ is a set of formulas, then we write $\mathcal{F}_{|At}$ for formulas in $\mathcal{F}$ that contain only atoms from At. For instance: $\{a \supset b; b \supset c\}_{|\{a,b\}} = \{a \supset b\}$. We say that two sets of formulas $\mathcal{F}_1$ and $\mathcal{F}_2$ are *syntactically disjoint* iff $\texttt{Atoms}(\mathcal{F}_1) \cap \texttt{Atoms}(\mathcal{F}_2) = \emptyset$. Furthermore, we extend $\texttt{Atoms}$ to a set of sets of formulas F point-wise: $\texttt{Atoms}(F) = \{\texttt{Atoms}(\mathcal{F}) \mid \mathcal{F} \in F\}$. For a strict rule $s = \varphi_1, \ldots, \varphi_n \rightarrow \psi$, $\texttt{Atoms}(s) = \texttt{Atoms}(\varphi_1) \cup \ldots \cup \texttt{Atoms}(\varphi_n) \cup \texttt{Atoms}(\psi)$. Similarly, for a defeasible rule $d = \varphi_1, \ldots, \varphi_n \underset{U}{\Rightarrow} \psi$, $\texttt{Atoms}(d) = \texttt{Atoms}(\varphi_1) \cup \ldots \cup \texttt{Atoms}(\varphi_n) \cup \texttt{Atoms}(\psi) \cup \texttt{Atoms}(U)$. $\texttt{Atoms}(\mathcal{S}) = \texttt{Atoms}(s_1) \cup \ldots \cup \texttt{Atoms}(s_n)$ where $\mathcal{S} = \{s_1, \ldots, s_n\}$ is a set of strict rules. Similarly for a set of defeasible rules $\mathcal{D} = \{d_1, \ldots, d_n\}$, $\texttt{Atoms}(\mathcal{D}) = \texttt{Atoms}(d_1) \cup \ldots \cup \texttt{Atoms}(d_n)$. Then for a knowledge base $\mathcal{B} = (\mathcal{P}, \mathcal{D})$, $\texttt{Atoms}(\mathcal{B}) = \texttt{Atoms}(P) \cup \texttt{Atoms}(D)$. For an argument A, $\texttt{Atoms}(A) = \texttt{Atoms}(\texttt{StrictRules}(A)) \cup \texttt{Atoms}(\texttt{DefRules}(A))$ and for a set of arguments $\mathcal{A}rgs = \{A_1, \ldots, A_n\}$, $\texttt{Atoms}(\mathcal{A}rgs) = \texttt{Atoms}(A_1) \cup \ldots \cup \texttt{Atoms}(A_n)$.

Definition 6.4 (Defeasible theory)**.** *A* defeasible theory *associated with a knowledge base* $(\mathcal{P}, \mathcal{D})$ *is a pair* $(\mathcal{S}(\mathcal{P}), \mathcal{D})$ *such that* $\mathcal{S}(\mathcal{P}) = \{\rightarrow \varphi \mid \varphi \in \mathcal{P}\} \cup \{\varphi_1, \ldots, \varphi_n \rightarrow \psi \mid \varphi_1, \ldots, \varphi_n, \psi \in \mathcal{L}$ *and* $\varphi_1, \ldots, \varphi_n \vdash \psi$ *and* $\texttt{Atoms}(\{\varphi_1, \ldots, \varphi_n, \psi\}) \subseteq \texttt{Atoms}(\mathcal{P}) \cup \texttt{Atoms}(\mathcal{D})\}$.

Definition 6.5 (Argumentation framework)**.** *An* abstract argumentation framework *AF built from a knowledge base* $\mathcal{B} = (\mathcal{P}, \mathcal{D})$ *is a pair* (Ar, def) *such that:*

- *Ar is the set of arguments on the basis of $\mathcal{T} = (\mathcal{S}(\mathcal{P}), \mathcal{D})$ as defined by Definition 2.6,*
- *def is the relation on Ar given by Definition 2.10.*

In the ASPIC Lite system, the definitions except the definitions of defeasible rules, undercutting, defeasible theory and argumentation framework are the same as the definitions in Chapter 2.

6.3 Problems and Postulates

In this section we introduce the definitions of non-interference and crash resistance [34] based on the notion of argumentation frameworks.

Although closure, direct consistency and indirect consistency are three important properties of the ASPIC Lite system, they are not the only properties that matter. There are other postulates that an argumentation framework should satisfy to make the system stable and prevent the system from crashing because of some problematic piece of information. So non-interference and crash-resistance are two more postulates we will introduce in this section.

In order to illustrate the kinds of problems that we are interested in, let us first take a look at Example 6.1 which is essentially equivalent to an example in [70].

Example 6.1 ([22]).
$\mathcal{P} = \{Says(J,s)$, "John says the cup of coffee contains sugar."
$Says(M,\neg s)$, "Mary says the cup of coffee does not contain sugar."
$Says(WF,r)\}$, "The weather forecaster predicts rain today."
$\mathcal{D} = \{Says(X,y) \underset{\{rel(X)\}}{\Rightarrow} y\}$, "People usually tell the truth."
We assume that the set of strict rules coincide with classical entailment (that is, we have $\phi_1, \ldots, \phi_n \rightarrow \psi$ whenever $\phi_1, \ldots, \phi_n \vdash \psi$) Consider the following arguments:
$J_1: \ Says(J,s) \underset{\{rel(J)\}}{\Rightarrow} s$
$M_1: \ Says(M,\neg s) \underset{\{rel(M)\}}{\Rightarrow} \neg s$
$JM: \ J_1, M_1 \rightarrow \neg r$
$W_1: \ Says(WF,r) \underset{\{rel(WF)\}}{\Rightarrow} r$
where $rel(J)$ means that John is reliable and $rel(M)$ means that Mary is reliable.
The resulting argumentation framework is partly illustrated in Figure 6.2.

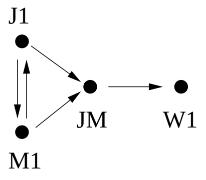


Figure 6.2: Arguments J_1 and M_1 contaminate argument W_1.

In the argumentation framework of Figure 6.2, arguments J_1 and M_1 defeat each other. As one may expect, a particularly troublesome argument is JM. It is composed of J_1

and M_1, together with the strict rule $s, \neg s \rightarrow \neg r$ (which exists because of the fact that $s, \neg s \vdash \neg r$). JM defeats (rebuts) W_1 and is defeated (rebutted) by J_1 and M_1. Figure 6.2 illustrates a general problem when trying to embed classical logic into rule-based argumentation formalisms: whenever there are two arguments that rebut each other (like J_1 and M_1) it is possible to combine them into an argument with any arbitrary conclusion (like JM) that can then be used to defeat any defeasible or plausible argument (like W_1).

Simply forbidding rules with inconsistent antecedents does not provide a solution. For instance, in the case of Example 6.1, classical logic also generates the strict rules $s \rightarrow s \vee \neg r$ and $s \vee \neg r, \neg s \rightarrow \neg r$ which can then be used (together with J_1 and M_1) to construct an argument for $\neg r$, even in the absence of $s, \neg s \rightarrow \neg r$ or any other rule with inconsistent antecedent.

The argumentation framework of Figure 6.2 is particularly troublesome under grounded semantics. Since there are no undefeated arguments, the grounded extension is empty. Hence, under grounded semantics, the weather forecast is not justified because John and Mary are having a disagreement about a cup of coffee. This is of course absurd.

However, as has been observed by Prakken [78], the problem seems to go away when using preferred semantics. In the example of Figure 6.2, there exist two preferred extensions: $\{J_1, W_1\}$ and $\{M_1, W_1\}$. Each of these extension contains W_1. A similar observation can be made for stable [40], semi-stable [25, 92], ideal [41] and eager [27] semantics. So if the problem of Figure 6.2 only becomes serious under grounded semantics, then why not for instance use preferred semantics instead? The problem, however, is that although preferred semantics "solves" the problem of Figure 6.2, there exist other more complex situations where preferred semantics alone does *not* provide a solution. One of these situations is described in Example 6.2 (taken from [22]).

Example 6.2 ([22]). *Given a knowledge base $\mathcal{B} = (\mathcal{P}, \mathcal{D})$ where*

$\mathcal{P} = \{Says(J, s)$, *"John says the cup of coffee contains sugar."*
$Says(M, \neg s)$, *"Mary says the cup of coffee does not contain sugar."*
$Says(J, \neg rel(J))$, *"John says John is unreliable."*
$Says(M, \neg rel(M))$, *"Mary says Mary is unreliable."*
$Says(WF, r)\}$, *"The weather forecaster predicts rain today."*

$\mathcal{D} = \{Says(J, s) \underset{\{rel(J)\}}{\Rightarrow} s$, *"John says the cup of coffee contains sugar then the cup of coffee probably contains sugar."*
$Says(M, \neg s) \underset{\{rel(M)\}}{\Rightarrow} \neg s$, *"Mary says the cup of coffee does not contain sugar then the cup of coffee probably does not contain sugar."*
$Says(J, \neg rel(J)) \underset{\{rel(J)\}}{\Rightarrow} \neg rel(J)$, *"John says John is unreliable then John probably is unreliable."*
$Says(M, \neg rel(M)) \underset{\{rel(M)\}}{\Rightarrow} \neg rel(M)$, *"Mary says Mary is unreliable then Mary probably is unreliable."*
$Says(WF, r) \underset{\{rel(WF)\}}{\Rightarrow} r\}$, *"The weather forecaster predicts rain today then it probably rains today."*

We assume that the set of strict rules coincides with classical entailment. Consider the

following arguments:
$J_0: \ Says(J, s)$
$J_1: \ Says(J, \neg rel(J))$
$J_2: \ J_1 \underset{\{rel(J)\}}{\Rightarrow} \neg rel(J)$
$J_3: \ J_0 \underset{\{rel(J)\}}{\Rightarrow} s$
$M_0: \ Says(M, \neg s)$
$M_1: \ Says(M, \neg rel(M))$
$M_2: \ M_1 \underset{\{rel(M)\}}{\Rightarrow} \neg rel(M)$
$M_3: \ M_0 \underset{\{rel(M)\}}{\Rightarrow} \neg s$
$W_0: \ Says(WF, r)$
$W_1: \ W_0 \underset{\{rel(WF)\}}{\Rightarrow} r$
$JM: \ J_7, M_7 \rightarrow \neg r$

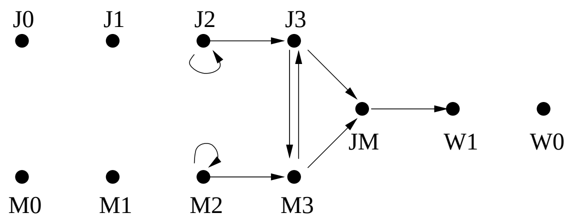


Figure 6.3: Switching from grounded to preferred semantics still does not always provide a solution

When applying the defeat relation specified by Definition 2.10, the argumentation framework AF in Figure 6.3 can be built (In Figure 6.3, we give only the arguments that can be captured by intuition. We can use a part of the argumentation framework to illustrate the contamination of the ASPIC Lite system since all arguments can be defeated by arguments that are obtained by substituting the conclusion of the argument JM). In this argumentation framework, only one complete extension exists: $\{J_0, J_1, M_0, M_1, W_0\}$. *So it is also a preferred extension and a stable extension. Therefore, there exists only one preferred extension in this argumentation framework. So we have that the weather forecast is not justified because unreliable John and unreliable Mary are having a disagreement about a cup of coffee.*

Here the argument JM *is the contaminating information. Without argument* JM*, the argumentation framework in Figure 6.3 can be split into two syntactically disjoint argumentation frameworks. The argumentation framework* AF_r *on the right side part consists of two arguments* W_0 *and* W_1. $\{W_0, W_1\}$ *is the only complete extension of the argumentation framework* AF_r. *The only complete extension of the whole argumentation framework* $\{J_0, J_1, M_0, M_1, W_0\}$ *does not contain* W_1. *The argument* W_1 *which is supposed to be accepted in the small argumentation framework is rejected in the big argumentation framework. It means that the combination of knowledge bases interferes with the reasoning results of systems. In this case, we say that this system does not satisfy the postulate of non-interference under complete semantics. Besides, there are arguments can be con-*

structed by replacing the conclusion of argument JM with arbitrary formula. Therefore all arguments could be defeated by those arguments. The consequence is that no argument can be accepted. Then the argumentation system crashes because it can only supply the empty set as the result under complete semantics.

The aim of the current chapter is to come up with a general solution to problems like those illustrated in Figure 6.2 and Figure 6.3. However, in order to claim any general solution, we first need to precisely define what it actually is that is violated in the outcome of Example 6.1 (Figure 6.2) and Example 6.2 (Figure 6.3). To do so, we will now proceed to describe the postulates of *non-interference* and *crash-resistance* [34]. The first notion to be defined is that of a logical formalism.

Definition 6.6 (Logical formalism (Definition 1 in [34]))**.** *A* logical formalism *is a triple* $(\mathcal{A}toms, \mathcal{F}ormulas, Cn)$ *where* $\mathcal{A}toms$ *is a countable (finite or infinite) set of atoms,* $\mathcal{F}ormulas$ *is the set of all well-formed formulas that can be constructed using* $\mathcal{A}toms$*, and* $Cn : 2^{\mathcal{F}ormulas} \to 2^{2^{\mathcal{F}ormulas}}$ *is a consequence function.*

The original formulation of non-interference, crash resistance, contamination, and non-triviality as stated by Caminada *et al.* [34] was done in a very general way for an arbitrary logical formalism. We can regard a particular argumentation system as a logical formalism satisfying Definition 6.6. In the following, we first restate the original definitions of the postulates of non-interference, crash resistance, contaminating, and non-trivial for an arbitrary logical formalism. Then we define non-interference, contamination and non-triviality in the specific case of the ASPIC Lite formalism respectively. Finally we give a theorem that states that non-trivial and non-interference implies crash resistance.

The idea of non-interference is that for two completely independent knowledge bases $\mathcal{F}_1$ and $\mathcal{F}_2$, $\mathcal{F}_1$ does not influence the outcome with respect to the language of $\mathcal{F}_2$ and vice versa.

The definition of non-interference for an arbitrary logical formalism stated in [34] is as follows.

Definition 6.7 (Non-interference (Definition 2 in [34]))**.** *A logical formalism* $(\mathcal{A}toms, \mathcal{F}ormulas, Cn)$ *satisfies* non-interference *iff for every* $\mathcal{F}_1, \mathcal{F}_2 \subseteq \mathcal{F}ormulas$ *such that* $\mathcal{F}_1$ *and* $\mathcal{F}_2$ *are syntactically disjoint it holds that* $Cn(\mathcal{F}_1)_{|\mathcal{A}toms(\mathcal{F}_1)} = Cn(\mathcal{F}_1 \cup \mathcal{F}_2)_{|\mathcal{A}toms(\mathcal{F}_1)}$ *and* $Cn(\mathcal{F}_2)_{|\mathcal{A}toms(\mathcal{F}_2)} = Cn(\mathcal{F}_1 \cup \mathcal{F}_2)_{|\mathcal{A}toms(\mathcal{F}_2)}$*.*

In order to define the postulate of non-interference, we need the union of knowledge bases.

Definition 6.8 (Union of knowledge bases)**.** *Let* $\mathcal{B}_1 = \langle \mathcal{P}_1, \mathcal{D}_1 \rangle$ *and* $\mathcal{B}_2 = \langle \mathcal{P}_2, \mathcal{D}_2 \rangle$ *be two knowledge bases. The union of* $\mathcal{B}_1$ *and* $\mathcal{B}_2$ *(denoted* $\mathcal{B}_1 \cup \mathcal{B}_2$*) is a knowledge base* $\mathcal{B} = \langle \mathcal{P}, \mathcal{D} \rangle$ *such that* $\mathcal{P} = \mathcal{P}_1 \cup \mathcal{P}_2$ *and* $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2$*.*

We say that two knowledge bases $\mathcal{B}_1$ and $\mathcal{B}_2$ are *syntactically disjoint* if $\mathtt{Atoms}(\mathcal{B}_1) \cap \mathtt{Atoms}(\mathcal{B}_2) = \emptyset$.

For complete semantics of argumentation system, non-interference means that when irrelevant information is added into the knowledge base from which an argumentation framework is built, the complete extensions of the argumentation framework essentially remain the same as that of the original argumentation framework.

In order to define non-interference specifically in the context of ASPIC Lite, we extend the function Cn to a function Cn_{ext} such that $Cn_{ext}(\mathcal{B})$ is a set of sets of conclusions under certain argumentation semantics.

Definition 6.9. *Let $\mathcal{B}$ be a knowledge base and $AF = (Ar, def)$ be the argumentation framework built from $\mathcal{B}$. $Cn_{ext} : \mathcal{B} \to 2^{2^{\texttt{Concs}(Ar)}}$ is a function such that $Cn_{ext}(\mathcal{B}) = \{\texttt{Concs}(\mathcal{A}rgs_1), \ldots, \texttt{Concs}(\mathcal{A}rgs_n)\}$ where $\mathcal{A}rgs_1, \ldots, \mathcal{A}rgs_n$ ($n \geq 0$) are the sets of arguments under certain semantics of AF.*

In this chapter, $\texttt{Atoms}(\mathcal{B})$ represents the atoms that occur in a knowledge base $\mathcal{B}$ and $\texttt{Atoms}(Ar)$ represents the atoms that occur in a set of arguments Ar.

In the specific case of the ASPIC Lite formalism, non-interference can be described as follows.

Definition 6.10 (Non-interference). *The ASPIC Lite system satisfies non-interference under a given semantics iff for every pair of syntactically disjoint knowledge bases $\mathcal{B}_1$ and $\mathcal{B}_2$ it holds that $Cn_{ext}(\mathcal{B}_1 \cup \mathcal{B}_2)_{|\texttt{Atoms}(\mathcal{B}_1)} = Cn_{ext}(\mathcal{B}_1)$.*

Example 6.3. *Let $\mathcal{B}$ be the knowledge base in Example 6.2. $\mathcal{B}$ is the union of the following two syntactically disjoint knowledge bases $\mathcal{B}_1$ and $\mathcal{B}_2$.*
$\mathcal{B}_1$:
$\mathcal{P}_1 = \{Says(J, s)$, "John says the cup of coffee contains sugar."
$Says(M, \neg s)$, "Mary says the cup of coffee does not contain sugar."
$Says(J, \neg rel(J))$, "John says John is unreliable."
$Says(M, \neg rel(M))$, "Mary says Mary is unreliable."

$\mathcal{D} = \{Says(J, s) \underset{\{rel(J)\}}{\Rightarrow} s$, "John says the cup of coffee contains sugar then the cup of coffee probably contains sugar."
$Says(M, \neg s) \underset{\{rel(M)\}}{\Rightarrow} \neg s$, "Mary says the cup of coffee does not contain sugar then the cup of coffee probably does not contain sugar."
$Says(J, \neg rel(J)) \underset{\{rel(J)\}}{\Rightarrow} \neg rel(J)$, "John says John is unreliable then John probably is unreliable."
$Says(M, \neg rel(M)) \underset{\{rel(M)\}}{\Rightarrow} \neg rel(M)$, "Mary says Mary is unreliable then Mary probably is unreliable."

$\mathcal{B}_2$:
$\mathcal{P}_2 = \{Says(WF, r)\}$, "The weather forecaster predicts rain today."
$\mathcal{D}_2 = \{Says(WF, r) \underset{\{rel(WF)\}}{\Rightarrow} r\}$, "The weather forecaster predicts rain today then it probably rains today."
We assume that the set of strict rules coincides with classical entailment. Consider the following arguments:
$W_0 :\ Says(WF, r)$
$W_1 :\ W_0 \underset{\{rel(WF)\}}{\Rightarrow} r$

There is one complete extension $\{W_0, W_1\}$ of the argumentation framework AF_2 which is built from $\mathcal{B}_2$. There is only one complete extension of AF built from $\mathcal{B}$ (Example 6.2) which

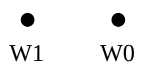


Figure 6.4: The argumentation framework AF_2 built from $\mathcal{B}_2$.

is $\{J_0, J_1, M_0, M_1, W_0\}$. *Then* $Cn_{ext}(\mathcal{B})_{|\texttt{Atoms}(\mathcal{B}_2)} = \{\{Says(WF, r)\}\}$ *and* $Cn_{ext}(\mathcal{B}_2) = \{\{Says(WF, r), r\}\}$. *In this case* $Cn_{ext}(\mathcal{B})_{|\texttt{Atoms}(\mathcal{B}_2)} \neq Cn_{ext}(\mathcal{B}_2)$. *Therefore, the ASPIC Lite system does not satisfy the postulate of non-interference.*

From Definition 6.10 and Example 6.3, the argumentation formalism of Caminada and Amgoud [31] does not satisfy the postulate of non-interference.

Some formulas can cause the knowledge base that is obtained by merging it with another set of formulas still yield the same outcome even though the other set of formulas is completely independent. We call this phenomenon contamination. The definition of contaminating for an arbitrary logical formalism stated in [34] is as follows.

Definition 6.11 (Contamination (Definition 3 in [34])). *Let* $(Atoms, \mathcal{F}ormulas, Cn)$ *be a logical formalism. A set* $\mathcal{F}_1 \subseteq \mathcal{F}ormulas$, *with* $Atoms(\mathcal{F}_1) \subsetneq \mathcal{A}toms$, *is called* contaminating *iff for every* $\mathcal{F}_2 \subseteq \mathcal{F}ormulas$ *such that* $\mathcal{F}_1$ *and* $\mathcal{F}_2$ *are syntactically disjunct it holds that* $Cn(\mathcal{F}_1) = Cn(\mathcal{F}_1 \cup \mathcal{F}_2)$.

Crash resistance is defined based on the concept of contamination. The definition of crash resistance for an arbitrary logical formalism stated in [34] is as follows.

Definition 6.12 (Crash resistance (Definition 4 in [34])). *A logical formalism satisfies* crash resistance *iff there does not exist a set of formulas* $\mathcal{F}$ *that is contaminating.*

A system that satisfies crash resistance cannot be affected by totally unrelated factors.

The definition of non-trivial for an arbitrary logical formalism stated in [34] is as follows.

Definition 6.13 (Non-trivial (Definition 5 in [34])). *A logical formalism* $(\mathcal{A}toms, \mathcal{F}ormulas, Cn)$ *is* non-trivial *iff for each* $\mathcal{A} \subseteq \mathcal{A}toms$ *such that* $\mathcal{A} \neq \emptyset$ *there exists* $\mathcal{F}_1, \mathcal{F}_2 \subseteq \mathcal{F}ormulas$ *such that* $Atoms(\mathcal{F}_1) = Atoms(\mathcal{F}_2) = \mathcal{A}$ *and* $Cn(\mathcal{F}_1)_{|\mathcal{A}} \neq Cn(\mathcal{F}_2)_{|\mathcal{A}}$.

In the specific case of the ASPIC Lite formalism, non-trivial can be described as follows.

Definition 6.14 (Non-trivial). *A formalism for argument-based inference is called* non-trivial *iff for each nonempty set* $\mathcal{A}$ *of atoms there exist knowledge bases* $\mathcal{B}_1$ *and* $\mathcal{B}_2$ *such that* $\texttt{Atoms}(\mathcal{B}_1) = \texttt{Atoms}(\mathcal{B}_2) = \mathcal{A}$ *and* $Cn_{ext}(\mathcal{B}_1)_{|\mathcal{A}} \neq Cn_{ext}(\mathcal{B}_2)_{|\mathcal{A}}$.

For any non-trivial formalism, non-interference implies crash resistance.

Theorem 6.1 (Theorem 1 in [34]). *Each non-trivial logical formalism* $(\mathcal{A}toms, \mathcal{F}ormulas, Cn)$ *that satisfies non-interference also satisfies crash resistance.*

In the specific case of the ASPIC Lite formalism, Theorem 6.1 can be described as follows.

Theorem 6.2. *Each non-trivial argumentation system that satisfies non-interference also satisfies crash resistance.*

Proof. Follows from Theorem 6.1. □

6.4 Solution

In this section we provide a solution to the contamination of the ASPIC Lite system. In Figure 6.3, the argument JM combines the two unconnected graphs. It makes argument W_1 being affected by completely irrelevant information which is the reason of the contamination. We avoid this by deleting the inconsistent arguments. We build an argumentation framework AF from a knowledge base. Then the argumentation framework obtained by deleting all inconsistent arguments from AF is the resulting framework we want.

An argument is inconsistent iff the set of conclusions of all its sub-arguments is not propositionally consistent.

Definition 6.15 (Consistent argument)**.** *An argument $A \in Ar$ is* consistent *iff* $\{\mathtt{Conc}(A') \mid A' \in \mathtt{Sub}(A)\}$ *is (propositionally) consistent. Otherwise, the argument is* inconsistent*.*

A set of arguments is consistent if the set of conclusions of all subarguments of arguments in the set is consistent.

Definition 6.16 (Consistent set of arguments)**.** *A set of arguments $\mathcal{A}rgs = \{A_1, \ldots, A_n\}$ is consistent if* $\mathtt{Concs}(\mathtt{Sub}(A_1)) \cup \ldots \cup \mathtt{Concs}(\mathtt{Sub}(A_n))$ *is (propositionally) consistent, otherwise $\mathcal{A}rgs$ is inconsistent.*

We propose that if inconsistent arguments are not included in argumentation frameworks then the ASPIC Lite system satisfies postulates of closure, direct consistency, indirect consistency, non-interference and crash resistance.

Definition 6.17 (Inconsistency cleaned argumentation framework)**.** *Let (Ar, def) be an argumentation framework built from a knowledge base $\mathcal{B}$. We define Ar_c as $\{A \mid A \in Ar$ and A is consistent $\}$, and $def_c = def \cap (Ar_c \times Ar_c)$. We refer to (Ar_c, def_c) as the inconsistency cleaned argumentation framework built from $\mathcal{B}$.*

Now let's see whether the problem in Example 6.2 can be fixed in the inconsistency cleaned argumentation framework.

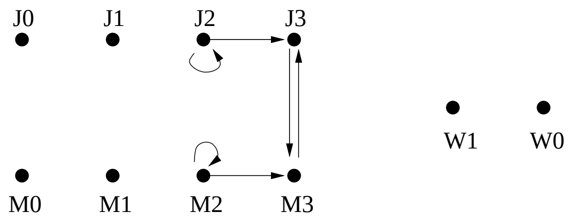


Figure 6.5: Repaired argumentation framework

Example 6.4. *Consider the argumentation framework generated by Example 6.2. We delete the argument JM from the argumentation framework because it is an inconsistent argument. We then obtain the argumentation framework of Figure 6.5. In this new argumentation framework, there is only one complete extension, and W_1 is justified under any mainstream argumentation semantics (including grounded and preferred). The weather forecast is no longer affected by a quarrel between other two persons on a cup of coffee.*

Since our approach is to delete the class of inconsistent arguments, we have to make sure that this does not cause any problem. Hence, we need to prove not only that our approach satisfies the postulates of non-interference and crash-resistance, but that also it continues to satisfy the previously satisfied postulates of closure, direct consistency and indirect consistency.

In the following, we show that without inconsistent arguments the ASPIC Lite system satisfies postulates of closure, direct consistency, indirect consistency, non-interference and crash resistance under complete semantics.

The following lemma shows that the closure under subarguments is satisfied for ASPIC Lite system under complete semantics.

Lemma 6.1 (Subargument closure). *Let E be a complete extension of an inconsistency cleaned argumentation framework AF. Then for each argument $A \in E$, $\texttt{Sub}(A) \subseteq E$.*

Proof. Let $B \in Sub(A)$. B is consistent because A is consistent and subset of a consistent set is consistent. Therefore B was not deleted from AF. B is defended by E because A is defended and any argument that defeats B defeats A as well. E is complete so includes all defended arguments. □

We extend $Cl_{\mathcal{S}}$ (Definition 2.3) to a set of arguments. Given a set of arguments $\mathcal{A}rgs$, the function $Cl_{\mathcal{S}}$ returns a set of arguments whose conclusions are logical consequences of $\texttt{Concs}(\mathcal{A}rgs)$.

Definition 6.18. *Let $\mathcal{A}rgs = \{A_1, ..., A_n\}$ $(n \geq 1)$ be a set of arguments and $\mathcal{S}$ be a set of strict rules. $Cl_{\mathcal{S}}(\mathcal{A}rgs)$ is a smallest set such that*

1. *$\mathcal{A}rgs \subseteq Cl_{\mathcal{S}}(\mathcal{A}rgs)$;*
2. *if $A_1, \ldots, A_n \in Cl_{\mathcal{S}}(\mathcal{A}rgs)$ and $\texttt{Conc}(A_1), \ldots, \texttt{Conc}(A_n) \rightarrow c \in \mathcal{S}$ then argument $A_1, \ldots, A_n \rightarrow c \in Cl_{\mathcal{S}}(\mathcal{A}rgs)$.*

If a set of arguments $\mathcal{A}rgs = \{A_1, \ldots, A_n\}$ is consistent then every argument in $Cl_{\mathcal{S}}(\mathcal{A}rgs)$ is also a consistent argument. Furthermore, if $\mathcal{A}rgs$ is a subset of a complete extension then $Cl_{\mathcal{S}}(\mathcal{A}rgs)$ is also a subset of the complete extension.

Lemma 6.2 (Consequence closure). *Let $\mathcal{A}rgs = \{A_1, \ldots, A_n\}$ be a consistent set of arguments and E a complete extension.*

1. *$Cl_{\mathcal{S}}(\mathcal{A}rgs)$ is a consistent set of arguments and*
2. *for each $B \in Cl_{\mathcal{S}}(\mathcal{A}rgs)$, B is consistent and*
3. *if $\mathcal{A}rgs \subseteq E$ then $Cl_{\mathcal{S}}(\mathcal{A}rgs) \subseteq E$.*

Proof. 1. From Definition 6.18, $\texttt{Concs}(\mathcal{A}rgs) \vdash \texttt{Concs}(Cl_{\mathcal{S}}(\mathcal{A}rgs))$. $\texttt{Concs}(\mathcal{A}rgs)$ is consistent because $\mathcal{A}rgs$ is a consistent set of argument. Therefore $\texttt{Concs}(Cl_{\mathcal{S}}(\mathcal{A}rgs))$ is also consistent. $\texttt{Concs}(\texttt{Sub}(\mathcal{A}rgs)) \vdash \texttt{Concs}(Cl_{\mathcal{S}}(\mathcal{A}rgs))$ because $\texttt{Concs}(\mathcal{A}rgs) \subseteq \texttt{Concs}(\texttt{Sub}(\mathcal{A}rgs))$. Then $\texttt{Concs}(\texttt{Sub}(\mathcal{A}rgs)) \cup \texttt{Concs}(Cl_{\mathcal{S}}(\mathcal{A}rgs))$ is consistent. Then $\texttt{Concs}(\texttt{Sub}(Cl_{\mathcal{S}}(\mathcal{A}rgs)))$ is consistent because $\texttt{Concs}(\texttt{Sub}(Cl_{\mathcal{S}}(\mathcal{A}rgs))) = \texttt{Concs}(\texttt{Sub}(\mathcal{A}rgs)) \cup \texttt{Concs}(Cl_{\mathcal{S}}(\mathcal{A}rgs))$. Therefore $Cl_{\mathcal{S}}(\mathcal{A}rgs)$ is a consistent set of arguments.

2. Let $B = B_1, \ldots, B_m \to c \in Cl_{\mathcal{S}}(\mathcal{A}rgs)$. Then $\texttt{Conc}(\texttt{Sub}(B)) \subseteq \texttt{Concs}(\texttt{Sub}(Cl_{\mathcal{S}}(\mathcal{A}rgs)))$. $\texttt{Concs}(\texttt{Sub}(Cl_{\mathcal{S}}(\mathcal{A}rgs)))$ is consistent from item 1. Then $\texttt{Conc}(\texttt{Sub}(B))$ is consistent. So B is a consistent argument.

3. Let $\mathcal{A}rgs = \{A_1, \ldots, A_n\} \subseteq E$ and $B \in Cl_{\mathcal{S}}(\mathcal{A}rgs)$. $\texttt{DefRules}(B) \subseteq \texttt{DefRules}(A_1) \cup \ldots \cup \texttt{DefRules}(A_n)$ from Definition 6.18. So for each defeasible rule $d \in \texttt{DefRules}(B)$, $d \in \texttt{DefRules}(A_1) \cup \ldots \cup \texttt{DefRules}(A_n)$. B can only be attacked on defeasible rules. For each argument C that defeats B, we assume C defeats B on B' such that B' is a subargument of B and $\texttt{TopRule}(B') = d \in \texttt{DefRules}(B)$. Then there exists an argument A_i $(1 \leq i \leq n)$ such that $d \in \texttt{DefRules}(A_i)$ and C also defeats A_i. So there exists an argument D such that D defeats C because $A_i \in E$. Therefore B is defended by E. B is consistent and E is a complete extension. Therefore, $B \in E$. So $Cl_{\mathcal{S}}(\mathcal{A}rgs) \subseteq E$.

□

Considering an argument as a tree, the height of an argument is the number of defeasible rules on the path that uses the most defeasible rules.

Definition 6.19 (Height of argument)**.** *The height of an argument A (denoted $h_d(A)$) is*

- 0 *if A consists of a single strict rule with empty antecedent.*
- 1 *if A consists of a single defeasible rule with empty antecedent.*
- $1 + \max\{h_d(B_1), \ldots, h_d(B_n)\}$ *if $A = B_1, \ldots, B_n \Rightarrow c$ $(n \geq 1)$.*
- $\max\{h_d(B_1), \ldots, h_d(B_n)\}$ *if $A = B_1, \ldots, B_n \to c$ $(n \geq 1)$.*

The following theorem shows that every complete extension of an inconsistency cleaned argumentation framework is propositionally consistent.

Theorem 6.3. *Let $\mathcal{B} = (\mathcal{P}, \mathcal{D})$ be a knowledge base and $AF_c = (Ar_c, def_c)$ be an inconsistency cleaned argumentation framework built from $\mathcal{B}$. Let E be a complete extension of AF_c. $\texttt{Concs}(E)$ is propositionally consistent.*

Proof. Let E be an arbitrary complete extension of AF_c. We can partition E into sets of arguments that have different heights as follows:

$E = \bigcup_{i=0..\infty} E_i$ where $E_i = \{A \in E \mid h_d(A) = i\}$.

Additionally let us denote partial sums:

$S_k = \bigcup_{0 \leq i \leq k} E_i$.

Observations:

- Any argument $A \in E$ is finite so $h(A)$ is defined and it falls into exactly one E_i.
- If for some i, $E_i = \emptyset$ then $E_j = \emptyset$ for all $j > i$. This follows from Lemma 6.1.
- If for some i, $\texttt{Concs}(S_i) = \texttt{Concs}(S_{i+1})$ then $\texttt{Concs}(S_i) = \texttt{Concs}(S_j)$ for all $j > i$. It is because no new defeasible rules can be applied. This will be always the case if $\mathcal{D}$ is finite.

- S_i is closed under subarguments because E is closed under subarguments and for each $A \in S_i$, it holds that if $A' \in \mathtt{Sub}(A)$ then $h_d(A') \leq h_d(A)$. So $A' \in S_i$.

In order to prove E is propositionally consistent, it is enough to show that for any i, $\mathtt{Concs}(S_i)$ is consistent.

We prove consistency of S_i by induction on the heights of arguments.

Basis step: $S_0 = E_0$. Let $P = \{\rightarrow c \mid c \in \mathcal{P}\}$. Then $S_0 = E_0 \subseteq Cl_{\mathcal{S}}(P)$. P is a consistent set of arguments since we assume that $\mathcal{P}$ is propositional consistent. So S_0 is consistent from Lemma 6.2.

Induction step: Assume S_k is consistent. We show that S_{k+1} is consistent. We prove by contradiction. Assume S_{k+1} is inconsistent.

Let $Elem(S_{k+1}) = \{A \in S_{k+1} \mid \mathtt{TopRule}(A) \in \mathcal{D}$ or $\mathtt{TopRule}(A) \in P\}$. Note that if S_{k+1} is inconsistent then $Elem(S_{k+1})$ is also inconsistent because S_{k+1} is closed under subarguments and each argument in S_{k+1} is constructed from P and $\mathcal{D}$ by propositional inferences.

Let $\{c_1, \ldots, c_n\} \subseteq \mathtt{Concs}(Elem(S_{k+1}))$ be a minimal set of conclusions such that $\{c_1, \ldots, c_n\}$ is inconsistent. Let $\mathcal{C} = \{C_1, \ldots, C_n\} \subseteq S_{k+1}$ be a minimal set such that for each i $(1 \leq i \leq n)$,

(1) $\mathtt{Conc}(C_i) = c_i$ and

(2) if $C_i \notin S_k$ then there is no argument $C \in S_k$ such that $\mathtt{Conc}(C) = \mathtt{Conc}(C_i)$ and

(3) $C_i \in Elem(S_{k+1})$.

Let $\mathcal{C}_1 = \{A_1, ..., A_l\} = \mathcal{C} \cap S_k$. Let $\mathcal{C}_2 = \{B_1, ..., B_m\} = \mathcal{C} \backslash S_k$. $m \geq 1$ since S_k is consistent and $\mathcal{C}$ is inconsistent. Then $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$. So $\{\mathtt{Conc}(A_1), \ldots, \mathtt{Conc}(A_l), \mathtt{Conc}(B_1), \ldots, \mathtt{Conc}(B_m)\} = \{c_1, \ldots, c_n\}$. Then $\mathtt{Conc}(A_1), \ldots, \mathtt{Conc}(A_l), \mathtt{Conc}(B_1), \ldots, \mathtt{Conc}(B_m) \vdash \perp$ because $\{c_1, \ldots, c_n\}$ is inconsistent. So the strict rule $\mathtt{Conc}(A_1), \ldots, \mathtt{Conc}(A_l), \mathtt{Conc}(B_1), \ldots, \mathtt{Conc}(B_{i-1}), \mathtt{Conc}(B_{i+1}), \ldots, \mathtt{Conc}(B_m) \rightarrow \neg\mathtt{Conc}(B_i)$ $(1 \leq i \leq m)$ is in $\mathcal{S}(\mathcal{P})$. Let argument $K = A_1, \ldots, A_l, B_1, \ldots, B_{i-1}, B_{i+1}, \ldots, B_m \rightarrow \neg\mathtt{Conc}(B_i)$. K is consistent, otherwise $\mathcal{C} \setminus \{B_i\}$ is inconsistent and $\mathcal{C}$ is not minimal.

Now we prove that K is defended by E. We use proof by contradiction. Assume that E does not defend K. Then there is an argument $D \in AF_c$ such that D defeats K and there exists no argument $D' \in E$ such that D' defeats D. D defeats K on a subargument $K' \in \mathtt{Sub}(K)\backslash\{K\}$ since $\mathtt{TopRule}(K)$ is strict. $K' \in E$ since $\mathtt{Sub}(K)\backslash\{K\} \in E$. Then there is an argument $D' \in E$ such that D' defeats D because E is a complete extension. Contradiction. So K is defended by E.

Then K is in E because K is consistent and is defended by E. K rebuts B_i because the top rule of B_i is defeasible. Then K defeats B_i. $B_i \in E_{k+1} \subseteq E$. Then E is not conflict-free. Contradiction. So S_{k+1} is consistent.

So by the induction, we have proven that S_k is consistent for all $k \in \mathbb{N}$. So E is consistent.

□

From the fact that every complete extension is propositionally consistent, it follows that the ASPIC Lite system without inconsistent arguments satisfies direct consistency.

Theorem 6.4. *The inconsistency cleaned version of ASPIC Lite system satisfies direct consistency under complete semantics.*

Proof. Let $\mathcal{B}$ be a knowledge base and $AF_c = (Ar_c, def_c)$ be the inconsistency cleaned argumentation framework built from $\mathcal{B}$. Let E be a complete extension of AF_c. Then $\texttt{Concs}(E)$ is propositionally consistent. Therefore $\texttt{Concs}(E)$ does not contain a formula ϕ and a formula $\neg\phi$. □

The following theorem shows that inconsistency cleaned argumentation frameworks satisfy closure under complete semantics.

Theorem 6.5. *The inconsistency cleaned version of ASPIC Lite system satisfies closure under complete semantics.*

Proof. Let $\mathcal{B}$ be a knowledge base $\mathcal{T} = (\mathcal{S}, \mathcal{D})$ be a defeasible theory associated with $\mathcal{B}$ and $AF_c = (Ar_c, def_c)$ be the inconsistency cleaned argumentation framework built from $\mathcal{B}$. Let E be a complete extension of AF_c. We need to prove that $\texttt{Concs}(E) = Cl_{\mathcal{S}}(\texttt{Concs}(E))$.

"$\subseteq$": Let $c \in \texttt{Concs}(E)$. Then $c \in Cl_{\mathcal{S}}(\texttt{Concs}(E))$ from Definition 2.3. So $\texttt{Concs}(E) \subseteq Cl_{\mathcal{S}}(\texttt{Concs}(E))$.

"$\supseteq$": Let $c \in Cl_{\mathcal{S}}(\texttt{Concs}(E))$. Then $c \in \texttt{Concs}(Cl_{\mathcal{S}}(E))$ from Definition 6.18. Then there exists an argument $A \in Cl_{\mathcal{S}}(E)$ such that $\texttt{Conc}(A) = c$. E is a consistent set of arguments from Theorem 6.3. Then A is consistent and $A \in E$ from Lemma 6.2. Therefore $c \in \texttt{Conc}(E)$. So $\texttt{Concs}(E) \supseteq Cl_{\mathcal{S}}(\texttt{Concs}(E))$. □

Since inconsistency cleaned argumentation frameworks satisfy closure and direct consistency under complete semantics, from Proposition 2.7, indirect consistency is satisfied under complete semantics.

Theorem 6.6. *The inconsistency cleaned version of ASPIC Lite system satisfies indirect consistency under complete semantics.*

Proof. It follows from Theorem 6.5 and Theorem 6.4. □

Now that we have proved that the inconsistency cleaned ASPIC Lite system satisfies direct and indirect consistency, as well as closure, the next step is to prove that the inconsistency cleaned ASPIC Lite system satisfies non-interference and crash resistance. For this, we first notice that in inconsistency cleaned argumentation frameworks that are constructed from unions of syntactically disjoint knowledge bases, for each argument A whose conclusion contains atoms from only one knowledge base, we can always find an argument A' such that A and A' have the same conclusions, A' contains atoms from only one knowledge base and the sets of arguments defending A also defend A'.

Lemma 6.3. *Let $\mathcal{B}$ be a knowledge base such that $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$ where $\mathcal{B}_1 = (\mathcal{P}_1, \mathcal{D}_1)$ and $\mathcal{B}_2 = (\mathcal{P}_2, \mathcal{D}_2)$ are syntactically disjoint. Let $AF = (Ar, def)$ and $AF_1 = (Ar_1, def_1)$ be the inconsistency cleaned argumentation frameworks built from $\mathcal{B}$ and $\mathcal{B}_1$ respectively. For each argument $C \in Ar$ such that $\texttt{Atoms}(\texttt{Conc}(C)) \subseteq \texttt{Atoms}(\mathcal{B}_1)$, there is an argument $C' \in Ar_1$ such that $\texttt{DefRules}(C') \subseteq \texttt{DefRules}(C)$ and $\texttt{Conc}(C') = \texttt{Conc}(C)$.*

Proof. We prove by induction on depths of arguments.

Basis step: $\texttt{depth}(C) = 1$. Then C is an atomic argument. $C \in Ar_1$ since $\texttt{Atoms}(\texttt{Conc}(C)) \subseteq \texttt{Atoms}(\mathcal{B}_1)$ and C has empty antecedent. Let $C' = C$. Then $C' \in Ar_1$, $\texttt{DefRules}(C') \subseteq \texttt{DefRules}(C)$ and $\texttt{Conc}(C') = \texttt{Conc}(C)$.

Induction step: Assume it holds for each argument C such that $\texttt{depth}(C) \leq k$. We show that for each argument C such that $\texttt{depth}(C) = k + 1$ it also holds.

Let $M(C)$ consists of the arguments $C' \in \texttt{Sub}(C)$ such that:

1. $\texttt{TopRule}(C') \in \mathcal{D}$ or C' is an argument with empty antecedent; and
2. there is no $C'' \in \texttt{Sub}(C)$ such that $C'' \neq C'$ and $C' \in \texttt{Sub}(C'')$ and C'' satisfies condition 1.

For each element $c \in M(C)$, either $\texttt{Atoms}(\texttt{TopRule}(c)) \subseteq \texttt{Atoms}(\mathcal{B}_1)$ or $\texttt{Atoms}(\texttt{TopRule}(c)) \subseteq \texttt{Atoms}(\mathcal{B}_2)$ because defeasible rules and arguments with empty antecedent do not contain atoms from both knowledge bases. There are two possible cases:

1. $\texttt{TopRule}(C)$ is defeasible. Then $M(C) = C$. Let $C = C_1, \ldots, C_n \Rightarrow c$. Then $\texttt{Atoms}(\texttt{Conc}(C_i)) \subseteq \texttt{Atoms}(\mathcal{B}_1)$ $(1 \leq i \leq n)$ because $\texttt{TopRule}(C) \in \mathcal{D}_1$. Then we can construct an argument $C' = C'_1, \ldots, C'_n \rightarrow \texttt{Conc}(C)$ from $\mathcal{B}_1$ where $C'_i \in Ar_1$ $(1 \leq i \leq n)$ and $\texttt{DefRules}(C'_i) \subseteq \texttt{DefRules}(C_i)$ and $\texttt{Conc}(C'_i) = \texttt{Conc}(C_i)$ because $\texttt{depth}(C_i) \leq k$ and $\texttt{Atoms}(\texttt{Conc}(C_i)) \subseteq \texttt{Atoms}(Ar_1)$. So $\texttt{Atoms}(C') \subseteq \texttt{Atoms}(\mathcal{B}_1)$. Therefore $C' \in Ar_1$ and $\texttt{DefRules}(C') \subseteq \texttt{DefRules}(C)$ and $\texttt{Conc}(C') = \texttt{Conc}(C)$.
2. $\texttt{TopRule}(C)$ is strict. Let $M(C) = \{A_1, \ldots, A_n, B_1, \ldots, B_m\}$ such that $\texttt{Atoms}(\texttt{TopRule}(A_i)) \subseteq \texttt{Atoms}(\mathcal{B}_1)(1 \leq i \leq n)$ and $\texttt{Atoms}(\texttt{TopRule}(B_j)) \subseteq \texttt{Atoms}(\mathcal{B}_2)$ $(1 \leq j \leq m)$. Then $\texttt{Concs}(\{A_1, \ldots, A_n\}), \texttt{Concs}(B_1, \ldots, B_m) \vdash \texttt{Conc}(C)$. $\texttt{Atoms}(\texttt{Conc}(C)) \cap \texttt{Atoms}(\texttt{Concs}(B_1, \ldots, B_m)) = \emptyset$ since $\texttt{Atoms}(\texttt{Conc}(C)) \subseteq \texttt{Atoms}(\mathcal{B}_1)$. So $\texttt{Concs}(\{A_1, \ldots, A_n\}) \vdash \texttt{Conc}(C)$. So there is a strict rule $\texttt{Conc}(A_1), \ldots, \texttt{Conc}(A_n) \rightarrow \texttt{Conc}(C)$ in $\mathcal{S}(\mathcal{P}_1)$. Then we can construct an argument $C' = A'_1, \ldots, A'_n \rightarrow \texttt{Conc}(C)$ from $\mathcal{B}_1$ where $A'_i \in Ar_1$ $(1 \leq i \leq n)$ and $\texttt{DefRules}(A'_i) \subseteq \texttt{DefRules}(A_i)$ and $\texttt{Conc}(A'_i) = \texttt{Conc}(A_i)$. There exist such A'_i since $\texttt{Atoms}(\texttt{Conc}(A_i)) \subseteq \texttt{Atoms}(\mathcal{B}_1)$ and $\texttt{depth}(A_i) \leq k$. So $\texttt{Atoms}(C') \subseteq \texttt{Atoms}(\mathcal{B}_1)$. Therefore $C' \in Ar_1$ and $\texttt{DefRules}(C') \subseteq \texttt{DefRules}(C)$ and $\texttt{Conc}(C') = \texttt{Conc}(C)$.

□

Let $\mathcal{B}_1$ and $\mathcal{B}_2$ be two syntactically disjoint knowledge bases. Let $AF = (Ar, def)$, $AF_1 = (Ar_1, def_1)$ and $AF_2 = (Ar_2, def_2)$ be the inconsistency cleaned argumentation frameworks built from $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$, $\mathcal{B}_1$ and $\mathcal{B}_2$ respectively. Let E_1 and E_2 be complete extensions of AF_1 and AF_2 respectively. In order to show that the inconsistency cleaned version of the ASPIC Lite system satisfies non-interference under complete semantics, we need to show that (1) for each complete extension E of AF, $E \cap Ar_1$ is a complete extension of AF_1; (2) for each complete extension E_1 of AF_1, there exists a complete extension E of AF such that $E_1 = E \cap Ar_1$. To show (2), we construct a set of arguments $F_{AF}(E_1 \cup E_2)$ where E_1 and E_2 are complete extensions of AF_1 and AF_2 respectively. Then first we show that $Ar_1 \cap F_{AF}(E_1 \cup E_2) = E_1$ (Lemma 6.4). Second we show that $Ar_1 \cap F_{AF}(E_1 \cup E_2) = E_1$ is complete (Lemma 6.5).

Figure 6.6 illustrates the following lemma such that if E_1 and E_2 are complete extensions of two inconsistency cleaned argumentation frameworks AF_1 and AF_2 that are built from two syntactically disjoint knowledge bases $\mathcal{B}_1$ and $\mathcal{B}_2$, then the set of arguments in AF_1 (AF_2) that are defended by $E_1 \cup E_2$ under the inconsistency cleaned argumentation framework built from knowledge base $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$ is exactly E_1 (E_2).

Lemma 6.4. *Let $\mathcal{B}_1$ and $\mathcal{B}_2$ be two syntactically disjoint knowledge bases. Let $AF = (Ar, def)$, $AF_1 = (Ar_1, def_1)$ and $AF_2 = (Ar_2, def_2)$ be the inconsistency cleaned argumentation frameworks built from $\mathcal{B}_1 \cup \mathcal{B}_2$, $\mathcal{B}_1$ and $\mathcal{B}_2$ respectively. Let E_1 and E_2 be*

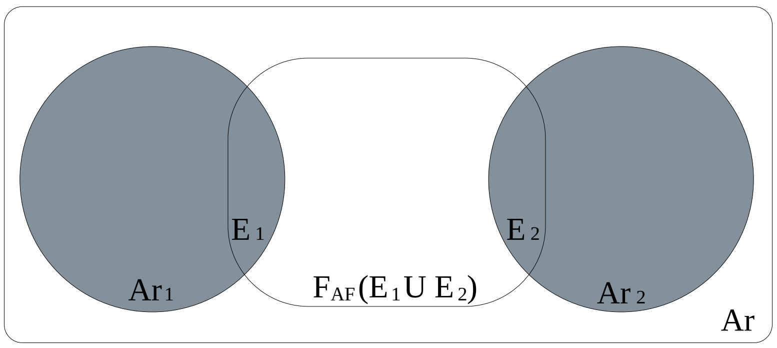


Figure 6.6: $Ar_1 \cap F_{AF}(E_1 \cup E_2) = E_1$ and $Ar_2 \cap F_{AF}(E_1 \cup E_2) = E_2$.

complete extensions of AF_1 and AF_2 respectively. Then $Ar_1 \cap F_{AF}(E_1 \cup E_2) = E_1$ (F_{AF} is the defense function of Definition 2.12).

Proof. "$\subseteq$": Let $A \in Ar_1$ and $A \in F_{AF}(E_1 \cup E_2)$. We use proof by contradiction. Assume that $A \notin E_1$. Then $A \notin F_{AF_1}(E_1)$ since $E_1 = F_{AF_1}(E_1)$. So there exists an argument $B \in Ar_1$ such that B defeats A and there exists no argument $C \in E_1$ such that C defeats B. From the fact that $A \in F_{AF}(E_1 \cup E_2)$ it follows that for all $B' \in Ar$ such that $B' \mathit{def} A$ there exists $C' \in E_1 \cup E_2$ such that $C' \mathit{def} B'$. Then there exists an argument $C' \in E_1 \cup E_2$ such that $C' \mathit{def} B$. Then $C' \in E_2$ because there exists no argument $C \in E_1$ such that C defeats B. From the fact that $B \in Ar_1$ and $\mathtt{Atoms}(Ar_1) \cap \mathtt{Atoms}(Ar_2) = \emptyset$ it follows that for all C' such that $C' \mathit{def} B$, $C' \notin Ar_2$. So $C' \notin E_2$. Contradiction. So $Ar_1 \cap F_{AF}(E_1 \cup E_2) \subseteq E_1$.

"$\supseteq$": Let $A \in E_1$. We use proof by contradiction. Assume that $A \notin F_{AF}(E_1 \cup E_2)$. Then there exists an argument $B \in Ar$ such that $B \mathit{def} A$ and there exists no argument $C \in E_1 \cup E_2$ such that $C \mathit{def} B$. Then $\mathtt{Atoms}(\mathtt{Conc}(B)) \subseteq \mathtt{Atoms}(Ar_1)$ because $A \in E_1$ and $B \mathit{def} A$. Then from Lemma 6.3 there is an argument $B' \in Ar_1$ such that $\mathtt{DefRules}(B') \subseteq \mathtt{DefRules}(B)$ and $\mathtt{Conc}(B') = \mathtt{Conc}(B)$. Then $B' \mathit{def} A$. Then there exists an argument $C' \in E_1$ such that $C' \mathit{def} B'$ because E_1 is a complete extension of AF_1 and $A \in E_1$. It follows that $C' \mathit{def} B$ because $C' \mathit{def} B'$ and $\mathtt{DefRules}(B') \subseteq \mathtt{DefRules}(B)$. Contradiction. So $Ar_1 \cap F_{AF}(E_1 \cup E_2) \supseteq E_1$.

□

Figure 6.7 illustrates Lemma 6.5 such that the set that is defended by the union of two complete extensions of two inconsistency cleaned argumentation frameworks built from two syntactically disjoint knowledge bases is a complete extension of the inconsistency cleaned argumentation framework built from the union of the two syntactically disjoint knowledge bases.

Lemma 6.5. *Let $\mathcal{B}_1$ and $\mathcal{B}_2$ be two syntactically disjoint knowledge bases. Let $AF = (Ar, \mathit{def})$, $AF_1 = (Ar_1, \mathit{def}_1)$ and $AF_2 = (Ar_2, \mathit{def}_2)$ be the inconsistency cleaned argumentation frameworks built from $\mathcal{B}_1 \cup \mathcal{B}_2$, $\mathcal{B}_1$ and $\mathcal{B}_2$ respectively. Let E_1 and E_2 be complete extensions of AF_1 and AF_2 respectively. Then $E = F_{AF}(E_1 \cup E_2)$ is a complete extension of AF.*

Proof. Let $Ar_{comb} \subseteq Ar$ such that $Ar_{comb} = \{A \in Ar \mid \mathtt{Atoms}(A) \cap \mathtt{Atoms}(AF_1) \neq \emptyset$ and $\mathtt{Atoms}(A) \cap \mathtt{Atoms}(AF_2) \neq \emptyset\}$. Then $Ar_1 \cap Ar_2 = \emptyset$, $Ar_1 \cap Ar_{comb} = \emptyset$, $Ar_2 \cap Ar_{comb} =$

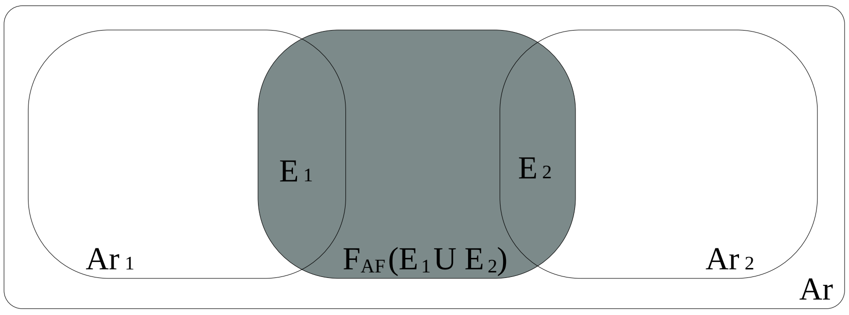


Figure 6.7: $F_{AF}(E_1 \cup E_2)$ is a complete extension of AF.

$\emptyset$ and $Ar = Ar_1 \cup Ar_2 \cup Ar_{comb}$. Let $E_{comb} = E \cap Ar_{comb}$. From Lemma 6.4 we know that $E_1 = E \cap Ar_1$ and $E_2 = E \cap Ar_2$. Then $E = E_1 \cup E_2 \cup E_{comb}$. We now prove that E is a complete extension of AF. First we prove that E is conflict-free. $E_1 \cup E_2$ is conflict-free since no argument in E_1 defeats any argument in E_2 and vice versa (this is because AF_1 and AF_2 are syntactically disjoint). We need to prove that

1. There is no argument in E_1 that defeats any argument in E_{comb} and vice versa.
2. There is no argument in E_2 that defeats any argument in E_{comb} and vice versa.
3. E_{comb} is conflict-free.

We now prove the first property (the proof of the second property is similar).

(1) Assume that $\exists A \in E_1$ and $\exists B \in E_{comb}$ such that $A def B$. Then $B \in F_{AF}(E_1 \cup E_2)$. So $\exists C \in E_1 \cup E_2$ such that $C def A$. Then $C \notin Ar_2$ otherwise C cannot defeat A because A_1 and A_2 are syntactically disjoint and inconsistent arguments have been deleted. So $C \in E_1$. Then E_1 is not conflict-free. Contradiction.

(2) Assume that $\exists A \in E_{comb}$ and $\exists B \in E_1$ such that $A def B$. $\texttt{Atoms}(\texttt{Conc}(A)) \subseteq \texttt{Atoms}(Ar_1)$ since otherwise A cannot defeat B. From Lemma 6.3 there is an argument $A' \in Ar_1$ such that $\texttt{Conc}(A') = \texttt{Conc}(A)$ and $\texttt{DefRules}(A') \subseteq \texttt{DefRules}(A)$. Then $A' def B$ because $A def B$ and $\texttt{Conc}(A') = \texttt{Conc}(A)$. Then from the fact that E_1 is admissible, it follows that there exists an argument $C \in E_1$ such that $C def A'$. Then $C def A$ since $\texttt{DefRules}(A') \subseteq \texttt{DefRules}(A)$. From the fact that $A \in F_{AF}(E_1 \cup E_2)$ it follows that there exists an argument $D \in E_1 \cup E_2$ such that $D def C$. So $D \in Ar_1 \cup Ar_2$. Then $D \in Ar_1$ because $C \in Ar_1$ and otherwise D cannot defeat C because Ar_1 and Ar_2 are syntactically disjoint and inconsistent arguments have been deleted. So $D \in E_1$. Then E_1 is not conflict-free. Contradiction.

Now we prove the third property. Assume that E_{comb} is not conflict-free. Then $\exists A, B \in E_{comb}$ such that $A def B$. From the fact that $B \in F_{AF}(E_1 \cup E_2)$ it follows that $\exists C \in E_1 \cup E_2$ such that $C def A$. Then there exists an argument in $E_1 \cup E_2$ defeats an argument in E_{comb}. It contradicts with (1) and (2).

The next thing to prove is that E is a fixpoint of F under AF.

$E \subseteq F_{AF}(E)$:: Let $A \in E$. Then $A \in F_{AF}(E_1 \cup E_2)$. So $\forall B \in Ar$ such that $B def A$ there exists an argument $C \in E_1 \cup E_2$ such that $C def B$. Then $C \in E$ because $E_1 \cup E_2 \subseteq E$.

Therefore $\forall B \in Ar$ such that $B def A$ there exists an argument $C \in E$ such that $C def B$. So $A \in F_{AF}(E)$.

$F_{AF}(E) \subseteq E$:: Let $A \in F_{AF}(E)$. Then $\forall B \in Ar$ such that $B def A$ there exists an argument $C \in E_1 \cup E_2 \cup E_{comb}$ such that $C def B$. Assume that $A \notin E$. Then $A \notin F_{AF}(E_1 \cup E_2)$. So there exists an argument $B' \in Ar$ that $B' def A$ and there exists no argument $C \in E_1 \cup E_2$ such that $C def B'$. Then it follows that there exists an argument $C' \in E_{comb}$ such that $C' def B'$. It follows that either $\texttt{Atoms}(\texttt{Conc}(C')) \cap \texttt{Atoms}(Ar_1) = \emptyset$ or $\texttt{Atoms}(\texttt{Conc}(C')) \cap \texttt{Atoms}(Ar_2) = \emptyset$ because otherwise C' cannot defeat any arguments. Then either $\texttt{Atoms}(\texttt{Conc}(C')) \subseteq \texttt{Atoms}(Ar_1)$ or $\texttt{Atoms}(\texttt{Conc}(C')) \subseteq \texttt{Atoms}(Ar_2)$. Assume $\texttt{Atoms}(\texttt{Conc}(C')) \subseteq \texttt{Atoms}(Ar_1)$ (similar to the case of $\texttt{Atoms}(\texttt{Conc}(C')) \subseteq \texttt{Atoms}(Ar_2)$). From Lemma 6.3, there is an argument $C'' \in Ar_1$ such that $\texttt{Conc}(C'') = \texttt{Conc}(C')$ and $\texttt{DefRules}(C'') \subseteq \texttt{DefRules}(C')$. Then $C'' def B'$ since $\texttt{Conc}(C'') = \texttt{Conc}(C')$. E defends C'' because $\texttt{DefRules}(C'') \subseteq \texttt{DefRules}(C')$ and $C' \in E$. Then $C'' \in E$ because E is a complete extension. Then $C'' \in E \cap Ar_1$. So $C'' \in E_1$. Contradiction. So $A \in E$.

From the fact that E is a conflict-free set and the fact that $E = F_{AF}(E)$ it then follows that E is a complete extension of AF. □

Given a knowledge base, CompExt gives a set of complete extensions of an argumentation framework built from the knowledge base.

Definition 6.20. *Let $\mathcal{B}$ be a knowledge base and $AF_c = (Ar_c, def_c)$ be the inconsistency cleaned argumentation framework built from $\mathcal{B}$.* CompExt : $\mathcal{B} \rightarrow 2^{2^{Ar_c}}$ *is the function such that* CompExt$(\mathcal{B}) = \{\mathcal{A}rgs_1, \ldots, \mathcal{A}rgs_n\}$ *where $\mathcal{A}rgs_1, \ldots, \mathcal{A}rgs_n$ ($n \geq 1$) are the complete extensions of AF_c.*

Given a knowledge base, $Cn_{complete}$ gives a set of sets of conclusions of complete extensions of an argumentation framework built from the knowledge base.

Definition 6.21. *Let $\mathcal{B}$ be a knowledge base and $AF_c = (Ar_c, def_c)$ be the inconsistency cleaned argumentation framework built from $\mathcal{B}$. $Cn_{complete} : \mathcal{B} \rightarrow 2^{2^{\texttt{Concs}(Ar_c)}}$ is the function such that $Cn_{complete}(\mathcal{B}) = \{\texttt{Concs}(\mathcal{A}rgs_1), \ldots, \texttt{Concs}(\mathcal{A}rgs_n)\}$ where $\mathcal{A}rgs_1, \ldots, \mathcal{A}rgs_n$ ($n \geq 1$) are the complete extensions of AF_c.*

The following lemma shows that if Ar_{c1} and Ar_{c2} are built from two syntactically disjoint knowledge bases $\mathcal{B}_1$ and $\mathcal{B}_2$ and E is a complete extension under the argumentation framework built from $\mathcal{B}_1 \cup \mathcal{B}_2$, then the conclusions of the arguments in $E \cap Ar_{c1}$ ($E \cap Ar_{c2}$) are equal to the conclusions of $E_{|Atoms(Ar_{c1})}$ ($E_{|Atoms(Ar_{c2})}$).

Lemma 6.6. *Let $\mathcal{B}_1$ and $\mathcal{B}_2$ be two syntactically disjoint knowledge bases. Let $AF = (Ar, def)$, $AF_1 = (Ar_1, def_1)$ and $AF_2 = (Ar_2, def_2)$ be the inconsistency cleaned argumentation frameworks built from $\mathcal{B}_1 \cup \mathcal{B}_2$, $\mathcal{B}_1$ and $\mathcal{B}_2$ respectively. Let E be a complete extension of AF. Then* $\texttt{Concs}(E \cap Ar_1) = \texttt{Concs}(E) \cap \texttt{Concs}(Ar_1)$.

Proof. "$\subseteq$": Let $c \in \texttt{Concs}(E \cap Ar_1)$. Then $\exists A \in E \cap Ar_1$ such that $\texttt{Conc}(A) = c$. So $A \in E$ and $A \in Ar_1$. Then $c \in \texttt{Concs}(E)$ and $c \in \texttt{Concs}(Ar_1)$. Therefore $c \in \texttt{Concs}(E) \cap \texttt{Concs}(Ar_1)$.

"$\supseteq$": Let $c \in \texttt{Concs}(E) \cap \texttt{Concs}(Ar_1)$. So $c \in \texttt{Concs}(E)$ and $c \in \texttt{Concs}(Ar_1)$. Then $\texttt{Atoms}(c) \subseteq \texttt{Atoms}(Ar_1)$. Then $\exists A \in E$ such that $\texttt{Conc}(A) = c$. Then there exists an argument $A' \in Ar_1$ such that $\texttt{DefRules}(A') \subseteq \texttt{DefRules}(A)$ and $\texttt{Conc}(A') = \texttt{Conc}(A) = c$

from Lemma 6.3. E defends A' since $\texttt{DefRules}(A') \subseteq \texttt{DefRules}(A)$ and E defends A. Then $A' \in E$ because E is a complete extension. Therefore $A' \in E \cap Ar_1$. Then $\texttt{Conc}(A') \in \texttt{Concs}(E \cap Ar_1)$. Hence, $c \in \texttt{Concs}(E \cap Ar_1)$.

□

Now we show that inconsistency cleaned argumentation frameworks satisfy non-interference under complete semantics.

Theorem 6.7. *The inconsistency cleaned version of ASPIC Lite system satisfies non-interference under complete semantics.*

Proof. Let $\mathcal{B}$ be a knowledge base such that $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$ where $\mathcal{B}_1$ and $\mathcal{B}_2$ are syntactically disjoint. Let $AF_c = (Ar_c, def_c)$, $AF_1 = (Ar_1, def_1)$ and $AF_2 = (Ar_2, def_2)$ be the inconsistency cleaned argumentation frameworks built from $\mathcal{B}_1$, $\mathcal{B}_2$ and $\mathcal{B}$ respectively.

Let $Ar_{comb} \subseteq Ar_c$ such that $Ar_{comb} = \{A \in Ar_c \mid \texttt{Atoms}(A) \cap \texttt{Atoms}(AF_1) \neq \emptyset$ and $\texttt{Atoms}(A) \cap \texttt{Atoms}(AF_2) \neq \emptyset\}$. Then $Ar_1 \cap Ar_2 = \emptyset$, $Ar_1 \cap Ar_{comb} = \emptyset$, $Ar_2 \cap Ar_{comb} = \emptyset$ and $Ar_c = Ar_1 \cup Ar_2 \cup Ar_{comb}$.

We first show: $\displaystyle\bigcup_{E_1 \in \text{CompExt}(\mathcal{B}_1)} E_1 = \bigcup_{E \in \text{CompExt}(\mathcal{B})} (E \cap Ar_1)$.

- "$\subseteq$": Let E_1 be a complete extension of AF_1. We now prove that there exists a complete extension E of AF_c such that $E \cap Ar_1 = E_1$. Let E_2 be a complete extensions of AF_2. Let $E = F_{AF_c}(E_1 \cup E_2)$. From Lemma 6.4 we know that $E_1 = E \cap Ar_1$. From Lemma 6.5, it then follows that E is a complete extension of AF_c.

- "$\supseteq$": We prove that for each complete extension E of AF_c, $E \cap Ar_1$ is a complete extension of AF_1. Let $E_1 = E \cap Ar_1$. E_1 is conflict-free because $E_1 \subseteq E$ and E is conflict-free. Now we prove that $E_1 = F_{AF_1}(E_1)$. Let $E_{comb} = E \cap Ar_{comb}$.

  - "$\subseteq$": Let $A \in E_1$. We prove by contradiction. Assume $A \notin F_{AF_1}(E_1)$. Then there is an argument $B \in Ar_1$ such that $B\,def\,A$ and there exists no argument $C \in E_1$ such that $C\,def\,B$. $A \in E$ since $E_1 \subseteq E$. Then $A \in F_{AF_c}(E)$. Then there exists an argument $C \in E$ such that $C\,def\,B$. Then $\texttt{Atoms}(\texttt{Conc}(C)) \subseteq \texttt{Atoms}(Ar_1)$ and $C \notin Ar_2$ because Ar_1 and Ar_2 are syntactically disjoint. Then $C \in E_{comb}$ because $C \notin E_1$ and $C \notin Ar_2$. Then from Lemma 6.3, there is an argument $C' \in Ar_1$ such that $\texttt{DefRules}(C') \subseteq \texttt{DefRules}(C)$ and $\texttt{Conc}(C) = \texttt{Conc}(C')$. C' is also defended by E since $\texttt{DefRules}(C') \subseteq \texttt{DefRules}(C)$ and C is in E. Then $C' \in E$ because E is complete. Then $C' \in E_1$ since $C' \in E \cap Ar_1$. $C'\,def\,B$ because $\texttt{Conc}(C) = \texttt{Conc}(C')$. Contradiction. So $A \in F_{AF_1}(E_1)$.
  - "$\supseteq$": Let $A \in Ar_1$ and $A \in F_{AF_1}(E_1)$. We prove by contradiction. Assume $A \notin E_1$. Then $A \notin E$ because $A \in Ar_1$ and $E_1 \subseteq E \cap Ar_1$. Then there is an argument $B \in Ar$ such that $B\,def\,A$ and there is no argument $C \in E$ such that $C\,def\,B$. So $B \notin Ar_1$ because for each argument $B \in Ar_1$ such that $B\,def\,A$ there is an argument $C \in E_1$ such that $C\,def\,B$ since $A \in F_{AF_1}(E_1)$. $B \notin Ar_2$ because Ar_1 and Ar_2 are syntactically disjoint. So $B \in Ar_{comb}$. From Lemma 6.3, there is an argument $B' \in Ar_1$ such that $\texttt{DefRules}(B') \subseteq \texttt{DefRules}(B)$ and $\texttt{Conc}(B) = \texttt{Conc}(B')$. Then $B'\,def\,A$ because $\texttt{Conc}(B) = \texttt{Conc}(B')$. Then there is an argument $C \in E_1$ such that $C\,def\,B'$ because $A \in F_{AF_1}(E_1)$. Then $C \in E$

since $E_1 \subseteq E$. $C\,def\,B$ because $\texttt{DefRules}(B') \subseteq \texttt{DefRules}(B)$. Contradiction. So $A \in E_1$.

From the fact that E_1 is a conflict-free set and the fact that $E_1 = F_{AF_1}(E_1)$ it then follows that E_1 is a complete extension of AF_1.

We have proven that $\bigcup_{E_1 \in \text{CompExt}(\mathcal{B}_1)} E_1 = \bigcup_{E \in \text{CompExt}(\mathcal{B})} (E \cap Ar_1)$.

We extend $\texttt{Concs}$ to a set of extensions $\mathcal{E}$ point-wise: $\texttt{Concs}(\mathcal{E}) = \{\texttt{Concs}(E) \mid E \in \mathcal{E}\}$. Then $\texttt{Concs}(\bigcup_{E_1 \in \text{CompExt}(\mathcal{B}_1)} E_1) = \texttt{Concs}(\bigcup_{E \in \text{CompExt}(\mathcal{B})} (E \cap Ar_1))$.

From the definition of $Cn_{complete}$, $Cn_{complete}(\mathcal{B}_1) = \texttt{Concs}(\bigcup_{E_1 \in \text{CompExt}(\mathcal{B}_1)} E_1)$.

From Lemma 6.6, $Cn_{complete}(\mathcal{B})_{|Atoms(\mathcal{B}_1)} = \texttt{Concs}(\bigcup_{E \in \text{CompExt}(\mathcal{B})} (E \cap Ar_1))$.

So $Cn_{complete}(\mathcal{B}_1) = Cn_{complete}(\mathcal{B})_{|Atoms(\mathcal{B}_1)}$. Therefore the ASPIC Lite system satisfies non-interference.

□

An inconsistency cleaned argumentation framework is non-trivial under complete semantics.

Theorem 6.8. *The inconsistency cleaned version of ASPIC Lite system satisfies non-triviality under complete semantics.*

Proof. Let $\mathcal{A}t$ be a non-empty set of atoms. We have to prove that there exist two consistency cleaned argumentation frameworks $AF_1 = (Ar_1, def_1)$ and $AF_2 = (Ar_2, def_2)$ built from $\mathcal{B}_1 = (\mathcal{P}_1, \mathcal{D}_1)$ and $\mathcal{B}_2 = (\mathcal{P}_2, \mathcal{D}_2)$ respectively such that $\texttt{Atoms}(\mathcal{B}_1) = \texttt{Atoms}(\mathcal{B}_2) = \mathcal{A}t$ and $Cn_{complete}(AF_1) \neq Cn_{complete}(AF_2)$.

Let $\mathcal{A}t = \{a_1, \ldots, a_n\}$ $(n \geq 1)$. Let $\texttt{Atoms}(\mathcal{B}_1) = \texttt{Atoms}(\mathcal{B}_2) = \mathcal{A}t$ such that $\mathcal{B}_1 = (\emptyset, \{\Rightarrow a_1, \ldots, \Rightarrow a_n\})$ and $\mathcal{B}_2 = (\emptyset, \{a_1 \Rightarrow a_1, \ldots, a_n \Rightarrow a_n\})$. Then $Cn_{complete}(AF_1) = \{\{a_1, \ldots, a_n\}\}$ and $Cn_{complete}(AF_2) = \{\emptyset\}$. $Cn_{complete}(AF_1) \neq Cn_{complete}(AF_2)$. □

From the fact that any non-trivial formalism, non-interference implies crash resistance (Theorem 6.1), it follows that without inconsistent arguments, argumentation frameworks satisfy crash resistance under complete semantics.

Theorem 6.9. *The inconsistency cleaned version of ASPIC Lite system satisfies crash resistance under complete semantics.*

Proof. It follows from Theorem 6.7 and Theorem 6.8. □

6.5 Related Work

There are other solutions proposed, for instance, deleting self-defeating arguments discussed by Pollock [70] and Caminada and Amgoud [31]. We show that deleting self-defeating arguments does not solve the contamination of ASPIC Lite system by the following example.

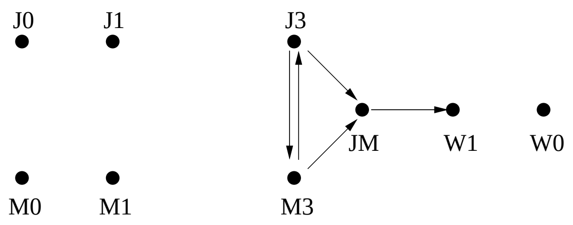


Figure 6.8: Deleting self-defeating arguments from the AF of Figure 6.3

Example 6.5. *Consider the argumentation framework AF in Figure 6.3. There are two self-defeating arguments J_2 and M_2 in AF. We obtain the argumentation framework in Figure 6.8 after delete the self-defeating arguments J_2 and M_2 of AF. We can see that in Figure 6.8, the argument W_1 is still not necessarily labelled* `in`. *The weather forecast is still contaminated by completely irrelevant information.*

In general one has to be extremely careful when starting to remove arguments from an argumentation framework. To illustrate the perils, let us examine what happens when one starts deleting self-undercutting arguments.

Example 6.6. *[70] Let $\mathcal{D} = \{p \underset{U}{\Rightarrow} q\}$, $\{q \rightarrow \neg a\} \subseteq \mathcal{S}$. where $a \in U$ and $\mathcal{S}$ contains all propositionally inferences. Then:*

$A_1 : p \quad A_2 : A_1 \Rightarrow q \quad A_3 : A_2 \rightarrow \neg a$

(A possible interpretation: p: John says that he is unreliable and q: John is unreliable.)

In this argumentation framework (Figure 6.9), A_3 is a self-undercutting argument. If we deleted A_3 from the argumentation framework, A_1 and A_2 would be in the grounded extension and preferred extension. Then the postulate of closure is violated because $q \in$ $\texttt{Concs}(E)$ *but $\neg a \notin$* $\texttt{Concs}(E)$.

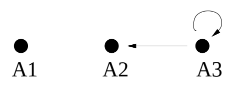


Figure 6.9: The argumentation framework of Example 6.6

What Example 6.6 illustrates is that if one starts deleting particular classes of arguments, then one may end up violating some of the rationality postulates (in this case: closure). Our contribution is that we have proved that these problems do *not* occur when deleting inconsistent arguments. That is, unlike for instance deleting the class of self-undercutting arguments, deleting the class of inconsistent arguments does not cause any violations of for instance, the postulates of closure and (direct/indirect) consistency.

6.6 Summary and Future Work

In this chapter we identified conditions under which the ASPIC Lite system can avoid being affected by contaminating information.

We have introduced the ASPIC Lite system which is similar to the argumentation formalism treated by Caminada and Amgoud in [31] in this chapter. We consider the universal order on the set of defeasible rules and premises since the solution of deleting inconsistent arguments does not work for the last-link argument ordering. The following is an example (found by Leon van der Torre) showing that after applying the current solution to an ASPIC Lite framework with last-link principle, the postulate of closure can be violated.

Example 6.7. *Given the knowledge base $\mathcal{B} = (\mathcal{P}, \mathcal{D})$ with $\mathcal{P} = \emptyset$ and $\mathcal{D} = \{\Rightarrow p; p \Rightarrow q; \Rightarrow \neg p \vee \neg q\}$. Assume that $\Rightarrow p$ has priority 1 (lowest), $\Rightarrow \neg p \vee \neg q$ has priority 2 (middle) and $p \Rightarrow q$ has priority 3 (highest). In that case, we can construct the following arguments with associated (lastlink principle) preferences.*

		preference
A_1:	$\Rightarrow p$	(1)
A_2:	$\Rightarrow \neg p \vee \neg q$	(2)
A_3:	$A_1 \Rightarrow q$	(3)
A_4:	$A_1, A_2 \rightarrow p \wedge \neg q$	(1)
A_5:	$A_1, A_3 \rightarrow p \wedge q$	(1)
A_6:	$A_2, A_3 \rightarrow \neg p \wedge q$	(2)

The resulting argumentation framework AF is depicted in Figure 6.10.

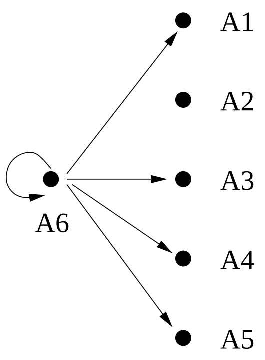


Figure 6.10: An example for last link principle

In AF, argument A_6 is an inconsistent argument. So according to the solution proposed in this chapter, we delete A_6 from the argumentation framework. Then we obtain the argumentation framework AF' in Figure 6.11. There is a complete extension $E = \{A_1, A_2, A_3, A_4, A_5\}$. It does not satisfy closure because A_2 and A_3 are in E and A_6 is not in E.

Therefore this solution does not work for any arbitrary reasonable argument ordering in the sense of [78].

In Section 6.3 we defined the postulates of non-interference and crash-resistance in the specific case of the ASPIC Lite formalism. Then we illustrated how these postulates were violated by the ASPIC Lite system. Then a solution, namely deleting inconsistent arguments, has been proposed. We showed that without inconsistent arguments, the ASPIC Lite system satisfies the postulates of closure, direct consistency, indirect consistency, non-interference and crash resistance under complete semantics.

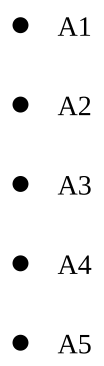


Figure 6.11: A counter example for last link principle

Argumentation frameworks are built from knowledge bases. Some knowledge bases can be regarded as unions of syntactically disjoint knowledge bases. The consequences of each small argumentation framework should not influence other argumentation frameworks when they are logically unrelated. The union of the consequences of sub-frameworks should also be in the consequence of the whole argumentation framework. The inconsistent arguments can connect the graphs together and change the logical consequences of the frameworks. It makes the unrelated arguments affect each other so that the consequence becomes unreasonable. The argumentation systems that have this problem violate the postulates of non-interference and crash-resistance. Several formalisms for argument-based entailment violate these two postulates, for instance, Pollock's OSCAR system [71] and some instantiations of the ASPIC framework [78]. The ASPIC Lite system also violates the postulates of non-interference and crash resistance. In order to solve this problem in the ASPIC Lite system we delete inconsistent arguments from argumentation frameworks. Then the ASPIC Lite system under complete semantics satisfies the five postulates of closure, direct consistency, indirect consistency, non-interference and crash resistance. Those postulates ensure that the system does not crash and can produce logically reasonable results even when potentially contaminating information is being inputted into the system. The five postulates are satisfied by the ASPIC Lite system under complete semantics. So the troublesome behavior that occurs in formalisms like that of Prakken [78] and that of Pollock [70] is avoided in this particular argumentation system.

A potential topic for future research would be how our solution (deleting inconsistent arguments) behaves in the context of semantics like grounded, preferred, semi-stable and ideal. Another topic would be the search for alternative solutions, for instance, forbidding strict rules feeding their consequents into the antecedents of other strict rules and disallowing the application of strict rules to an indirectly inconsistent set of formulas. To which extent is it possible to satisfy all five rationality postulates without having to delete all inconsistent arguments?

Chapter 7

Conclusions

This work contributes to the theoretical state of the art of formal argumentation in "Dung-base"(i.e., abstract argument semantics, using argumentation frameworks as defined by Dung [40]) as well as in "Dung-instantiated" (i.e., instantiations of abstract argumentation with structured arguments). As for Dung-base, we have first provided an argumentation discussion game under stable semantics. Secondly, we have defined a justification status of arguments based on the complete semantics. Then we have proved that the complete extensions in abstract argumentation coincide with 3-valued stable models in logic programming. As for "Dung-instantiated", we have provided a general way of satisfying the postulates of closure, direct consistency, indirect consistency, crash resistance and non-interference for the ASPIC Lite system under complete semantics.

The research questions that were raised in the introduction chapter have been answered in Chapter 3, 4, 5 and 6 respectively.

Question 1: *What is a discussion game for stable semantics?*

To answer Question 1, we introduced a credulous discussion game for argumentation under stable semantics in Chapter 3. In this discussion game, the proponent and the opponent can make moves of `in`, `out` and `question` where `question` is an additional move that does not exist in the work of Vreeswijk and Prakken [97]. The move of `question` enables the game to continue with the arguments that have not been discussed before so that all arguments can be discussed in the game. By applying this credulous discussion game, we can examine whether an argument is in at least one stable extension. The proponent argues with the opponent in favor of the argument which we want to examine following the rules of the credulous stable discussion game. If there exists one stable discussion game for an argument that is won by the proponent then the argument is in at least one stable extension. Besides credulous acceptance of stable semantics, we have also examined sceptical acceptance under stable semantics. We can use the credulous stable discussion game to check whether an argument is in every stable semantics by applying the credulous stable discussion game on the defeaters of the argument. If none of the defeaters of the argument is in any stable extension then the argument is in every stable extensions.

Due to the fact that the stable semantics does not satisfy relevance, the stable discussion game proposed here has to go through every argument in an argumentation framework. Therefore, the complexity of the stable discussion game would not be less than that of the preferred discussion game.

In Chapter 4, we defined the justification status of arguments based on the notion of a complete labelling and given the approach to determine and defend the justification status, which is the answer to Question 2.

Question 2: *How to define justification statuses of arguments, what are its properties, and how can we compute them?*

A justification status of an argument is a subset of $\{\texttt{in}, \texttt{out}, \texttt{undec}\}$. For complete semantics, there are six possible justification statuses $\{\texttt{in}\}$, $\{\texttt{out}\}$, $\{\texttt{undec}\}$, $\{\texttt{in}, \texttt{undec}\}$, $\{\texttt{out}, \texttt{undec}\}$ and $\{\texttt{in}, \texttt{out}, \texttt{undec}\}$ which indicate whether an argument has to be accepted, has to be rejected, cannot be accepted and cannot be rejected either, can be accepted, can be rejected etc. The justification status is easy to explain since it basically answers the question whether an argument has to be accepted, can be accepted, has to be rejected, can be rejected, etc. In Chapter 4 we also defined a justification status of conclusions. We transfered complete labellings of arguments into complete labellings of conclusions. Similarly, the justification status of a conclusion is the set of labels that could be assigned by the complete labellings of conclusions. This labelling based approach for computing the justification status of arguments yields more informative answers than the traditional extensions approaches. It allows for a more fine grained distinction between arguments. Besides, we can apply the justification status to trust.

When arguing about the trustworthiness of a certain trustee, if the justification status of the trustworthiness is strongly accepted ($\{\texttt{in}\}$) then trustors can decide to rely on the trustee. If the justification status of the trustworthiness is strongly rejected ($\{\texttt{out}\}$) then trustors would choose to give up on this trustee. If the justification status of the trustworthiness is weakly accepted ($\{\texttt{in}, \texttt{undec}\}$) then trustors have to be aware of the possibility that the trustee might disappoint trustors. Otherwise if the justification status of the trustworthiness is weakly rejected or borderline case ($\{\texttt{out}, \texttt{undec}\}$, $\{\texttt{undec}\}$ and $\{\texttt{in}, \texttt{out}, \texttt{undec}\}$) then the possibility of being disappointed is quite big.

Our current work considers only the justification status based on complete semantics. Dvořák [42] generalized our work to grounded, admissible, complete, preferred, semi-stable and stage semantics and compared them. The approaches to determine and defend the justification statuses of arguments based on other argument semantics, for instance, preferred, stable and semi-stable semantics, could be an interesting future work.

Question 3: *Which argumentation semantics corresponds to 3-valued stable semantics in logic programming?*

In Chapter 5 we answered Question 3 by showing the correspondence between the complete extensions in argumentation and the 3-valued stable models in logic programming. First, we transfered argumentation frameworks into logic programming and proved the correspondence between complete extensions and 3-valued stable extensions. Then we transformed logic programming into argumentation frameworks and proved that the 3-valued stable models of the original logic programming coincide with the complete extensions of the associated argumentation framework.

So we have proved that the 3-valued stable models in logic programming are equal to the sets of arguments obtained after step 3 (applying abstract argumentation semantics) of the 3-step argumentation process (Section 2.2) when we apply the complete semantics. This shows that the 3-value stable model semantics of logic programming can be modelled by using the 3-step argumentation process.

Many semantics of argumentation are described based on complete semantics. From Figure 5.1, stable extensions, semi-stable extensions, preferred extensions and the grounded extension are all complete extensions. 3-valued stable models are also the basic semantics in logic programming. Stable models, L-stable models, regular models and well founded models are defined based on the notion of 3-valued stable models.

So the results in Chapter 5 allow for new possible correspondences to be identified. For instance, the correspondence between the preferred extensions in abstract argumentation and the regular models in logic programming.

Question 4: *How to make instantiations of the ASPIC Lite system satisfy crash resistance and non-interference?*

In Chapter 6, we answered Question 4. We identified a set of conditions under which the ASPIC Lite system satisfies the postulates of direct consistency, indirect consistency, closure, non-interference and crash resistance. Argumentation systems that satisfy these postulates obtain consistent entailments that are not only based on arguments but also on the conclusions associated with the arguments. Furthermore, systems that satisfy the postulates of crash-resistance and non-interference will not be disturbed by completely irrelevant information.

Following the 3-step process, inconsistent arguments could appear in the argumentation framework when we construct an argumentation framework. These inconsistent arguments could contaminate the whole argumentation framework, which causes a crash of the argument-based entailment. Briefly, the solution is getting rid of the inconsistent arguments. We have proved that the ASPIC Lite system without inconsistent arguments satisfies the postulates of direct consistency, indirect consistency, closure, non-interference and crash resistance under complete semantics. Since many other semantics in argumentation are described based on complete semantics, we leave the solution for grounded, preferred and semi-stable semantics as future work.

The postulates of non-interference and crash-resistance increase the trustworthiness and the stability of an argumentation system. If some irrelevant information is accidentally or intentionally input into a knowledge base and an argumentation framework is constructed from this knowledge base, then this argumentation system will crash since all the entailments can be empty. Using our solution, we do not need to care whether the information from a source is irrelevant or not. Therefore the argumentation frameworks without inconsistent arguments are more trustworthy and stable than the original ones.

Topics for future research have been discussed at the end of each chapter.

Overall, our work has enhanced today's generation of argumentation formalisms and made it become suitable for a wider variety of real-life applications, such as trust management.

Bibliography

[1] Sergio J. Alvarado, Michael G. Dyer, and Margot Flowers. Argument comprehension and retrieval for editorial text. *Knowl.-Based Syst.*, 3(3):139–162, 1990.

[2] Leila Amgoud and Philippe Besnard. Bridging the gap between abstract argumentation systems and logic. In *Proceedings of the 3rd International Conference on Scalable Uncertainty Management (SUM)*, pages 12–27, 2009.

[3] Leila Amgoud and Philippe Besnard. A formal analysis of logic-based argumentation systems. In *Proceedings of the 4th International Conference on Scalable Uncertainty Management (SUM)*, pages 42–55, 2010.

[4] Leila Amgoud, L. Bodenstaff, M. Caminada, P. McBurney, S. Parsons, H. Prakken, J. van Veenen, and G.A.W. Vreeswijk. 2006. "final review and report on formal argumentation system". 2006.

[5] Leila Amgoud and Claudette Cayrol. On the acceptability of arguments in preference-based argumentation. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 1–7, 1998.

[6] Leila Amgoud and Claudette Cayrol. A reasoning model based on the production of acceptable arguments. *Ann. Math. Artif. Intell.*, 34(1-3):197–215, 2002.

[7] Leila Amgoud, Claudette Cayrol, and Marie-Christine Lagasquie-Schiex. On the bipolarity in argumentation frameworks. In *Proceedings of the 10th International Workshop on Non-Monotonic Reasoning (NMR)*, pages 1–9, 2004.

[8] Grigoris Antoniou, David Billington, Guido Governatori, and Michael J. Maher. A flexible framework for defeasible logics. In *Proceedings of the 12th Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI)*, pages 405–410, 2000.

[9] Katie Atkinson and Trevor J. M. Bench-Capon. Argumentation and standards of proof. In *Proceedings of the 11th International Conference on Artificial Intelligence and Law (ICAIL)*, pages 107–116, 2007.

[10] Pietro Baroni and Massimiliano Giacomin. Comparing argumentation semantics with respect to skepticism. In *Proceedings of the 9th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQArU)*, pages 210–221, 2007.

[11] Pietro Baroni and Massimiliano Giacomin. On principle-based evaluation of extension-based argumentation semantics. *Artif. Intell.*, 171(10-15):675–700, 2007.

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

[13] Ringo Baumann. Splitting an argumentation framework. In *Proceedings of the 11th International Conference on Logic Programming and Nonmonotonic Reasoning (LP-NMR)*, pages 40–53, 2011.

[14] Trevor J. M. Bench-Capon. Persuasion in practical argument using value-based argumentation frameworks. *J. Log. Comput.*, 13(3):429–448, 2003.

[15] Trevor J. M. Bench-Capon, Katie Atkinson, and Alison Chorley. Persuasion and value in legal argument. *J. Log. Comput.*, 15(6):1075–1097, 2005.

[16] Philippe Besnard and Anthony Hunter. A logic-based theory of deductive arguments. *Artif. Intell.*, 128(1-2):203–235, 2001.

[17] Lawrence Birnbaum. Argument molecules: A functional representation of argument structure. In *Proceedings of the 2nd National Conference on Artificial Intelligence (AAAI)*, pages 63–65, 1982.

[18] Lawrence Birnbaum, Margot Flowers, and Rod McGuire. Towards an AI model of argumentation. In *Proceedings of the 1st National Conference on Artificial Intelligence (AAAI)*, pages 313–315, 1980.

[19] Alexander Bochman. Collective argumentation and disjunctive logic programming. *J. Log. Comput.*, 13(3):405–428, 2003.

[20] Martin Caminada. *For the sake of the Argument. Explorations into argument-based reasoning.* Doctoral dissertation Free University Amsterdam, 2004.

[21] Martin Caminada. Collapse in formal argumentation systems. Technical Report UU-CS-2005-023, Utrecht University, 2005.

[22] Martin Caminada. Contamination in formal argumentation systems. In *Proceedings of the 17th Benelux Conference on Artificial Intelligence (BNAIC)*, pages 59–65, 2005.

[23] Martin Caminada. On the issue of reinstatement in argumentation. In *Proceedings of the 10th European Conference on Logics in Artificial Intelligence (JELIA)*, pages 111–123, 2006.

[24] Martin Caminada. On the issue of reinstatement in argumentation. Technical Report UU-CS-2006-023, Institute of Information and Computing Sciences, Utrecht University, 2006.

[25] Martin Caminada. Semi-stable semantics. In *Proceedings of the 1st Conference on Computational Model of Arguments (COMMA)*, pages 121–130, 2006.

[26] Martin Caminada. An algorithm for computing semi-stable semantics. In *Proceedings of the 9th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*, number 4724 in Springer Lecture Notes in AI, pages 222–234, 2007.

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

[28] Martin Caminada. Project proposal: Advanced argumentation services for trust management, May 2007.

[29] Martin Caminada. Preferred semantics as Socratic discussion. In *Proceedings of the 11th AI*IA symposium on artificial intelligence*, pages 209–216. 2010.

[30] Martin Caminada and Leila Amgoud. An axiomatic account of formal argumentation. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI)*, pages 608–613, 2005.

[31] Martin Caminada and Leila Amgoud. On the evaluation of argumentation formalisms. *Artif. Intell.*, 171(5-6):286–310, 2007.

[32] Martin Caminada and Dov Gabbay. A logical account of formal argumentation. *Studia Logica*, 93(2-3):109–145, 2009.

[33] Martin Caminada and Yining Wu. An argument game for stable semantics. *Logic Journal of the IGPL*, 17(1):77–90, 2009.

[34] Martin W. A. Caminada, Walter.A. Carnielli, and Paul.E. Dunne. Semi-stable semantics. *Journal of Logic and Computation*, pages 1–45, 2011.

[35] Claudette Cayrol, Sylvie Doutre, and Jérôme Mengin. On decision problems related to the preferred semantics for argumentation frameworks. *Journal of Logic and Computation*, 13(3):377–403, 2003.

[36] Claudette Cayrol and Marie-Christine Lagasquie-Schiex. On the acceptability of arguments in bipolar argumentation frameworks. In *Proceedings of the 8th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*, pages 378–389, 2005.

[37] Sylvie Coste-Marquis, Caroline Devred, and Pierre Marquis. Prudent semantics for argumentation frameworks. In *Proceedings of the 17th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 568–572, 2005.

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

[39] Phan Minh Dung. Negations as hypotheses: An abductive foundation for logic programming. In *Proceedings of the 8th International Conference on Logic Programming (ICLP)*, pages 3–17, 1991.

[40] 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.

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

[42] Wolfgang Dvorák. On the complexity of computing the justification status of an argument. In *Proceedings of the 1st International Workshop on the Theory and Applications of Formal Argumentation (TAFA)*, pages 32–49, 2011.

[43] Thomas Eiter, Nicola Leone, and Domenico Saccà. On the partial semantics for disjunctive deductive databases. *Ann. Math. Artif. Intell.*, 19(1-2):59–96, 1997.

[44] Dov M. Gabbay and Artur S. d'Avila Garcez. Logical modes of attack in argumentation networks. *Studia Logica*, 93(2-3):199–230, 2009.

[45] Allen Van Gelder, Kenneth A. Ross, and John S. Schlipf. The well-founded semantics for general logic programs. *J. ACM*, 38(3):620–650, 1991.

[46] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *Proceedings of the 5th International Conference on Logic Programming/Symposium on Logic Programming (ICLP/SLP)*, pages 1070–1080, 1988.

[47] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9(3/4):365–386, 1991.

[48] Thomas F. Gordon, Henry Prakken, and Douglas Walton. The carneades model of argument and burden of proof. *Artif. Intell.*, 171(10-15):875–896, 2007.

[49] Thomas F. Gordon and Douglas Walton. Proof burdens and standards. In *Argumentation in Artificial Intelligence*. Springer Publishing Company, Incorporated, 2009.

[50] Nikos Gorogiannis and Anthony Hunter. Instantiating abstract argumentation with classical logic arguments: Postulates and properties. *Artif. Intell.*, 175(9-10):1479–1497, 2011.

[51] Guido Governatori, Michael J. Maher, Grigoris Antoniou, and David Billington. Argumentation semantics for defeasible logic. *J. Log. Comput.*, 14(5):675–702, 2004.

[52] Davide Grossi. An application of model checking games to abstract argumentation. In *Proceedings of the 3rd International Workshop on Logic, Rationality and Interaction (LORI)*, pages 74–86, 2011.

[53] Hadassa Jakobovits and Dirk Vermeir. Robust semantics for argumentation frameworks. *J. Log. Comput.*, 9(2):215–261, 1999.

[54] Antonis C. Kakas and Francesca Toni. Computing argumentation in logic programming. *J. Log. Comput.*, 9(4):515–562, 1999.

[55] Laurent Keiff. Dialogical logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Summer 2011 edition, 2011.

[56] Kurt Konolige. Defeasible argumentation in reasoning about events. In *Proceedings of the 3rd International Syposium on Methodologies for Intelligent Systems (ISMIS)*, pages 380–390, 1988.

[57] Fangzhen Lin and Yoav Shoham. Argument systems: A uniform basis for nonmonotonic reasoning. In *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 245–255, 1989.

[58] Kuno Lorenz. *Arithmetik und Logik als Spiele*. PhD thesis, Kiel, 1961.

[59] Kuno Lorenz. Basic objectives of dialogue logic in historical perspective. *Synthese*, 127:255–263, 2001.

[60] Paul Lorenzen. Dialectical foundations of logical calculi. In *Constructive Philosophy*. University of Massachusetts Press, 1987. translated by K.R.Pavlovic.

[61] Paul Lorenzen and Kuno Lorenz. Dialogische logik. *Wissenschaftliche Buchgesellschaft, Darmstadt*, 1978.

[62] Ronald Prescott Loui. Defeat among arguments: a system of defeasible inference. *Computational Intelligence*, 3:100–106, 1987.

[63] Drew V. McDermott and Jon Doyle. Non-Monotonic Logic I. *Artif. Intell.*, 13(1-2):41–72, 1980.

[64] Rod McGuire, Lawrence Birnbaum, and Margot Flowers. Opportunistic processing in arguments. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 58–60, 1981.

[65] Sanjay Modgil and Martin W.A. Caminada. Proof theories and algorithms for abstract argumentation frameworks. In Iyad Rahwan and Guillermo R. Simari, editors, *Argumentation in Artificial Intelligence*, pages 105–129. Springer, 2009.

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

[67] Mikolaj Podlaszewski, Yining Wu, and Martin Caminada. An implementation of basic argumentation components. Presented in The Conference on Computational Model of Arguments (COMMA), 2010.

[68] John L. Pollock. Defeasible reasoning. *Cognitive Science*, 11(4):481–518, 1987.

[69] John L. Pollock. How to reason defeasibly. *Artif. Intell.*, 57(1):1–42, 1992.

[70] John L. Pollock. Justification and defeat. *Artif. Intell.*, 67(2):377–407, 1994.

[71] John L. Pollock. *Cognitive Carpentry. A Blueprint for How to Build a Person.* MIT Press, Cambridge, MA, 1995.

[72] Henry Prakken. Dialectical proof theory for defeasible argumentation with defeasible priorities (preliminary report). In *ModelAge Workshop*, pages 202–215, 1997.

[73] Henry Prakken. Intuitions and the modelling of defeasible reasoning: some case studies. In *Proceedings of the 9th International Workshop on Non-Monotonic Reasoning (NMR)*, pages 91–102, 2002.

[74] Henry Prakken. Analysing reasoning about evidence with formal models of argumentation. *Law, Probability & Risk*, 3:1:33–50, 2004.

[75] Henry Prakken. Commonsense reasoning. Technical report, Institute of Information and Computing Sciences, Utrecht University, 2004. Course material.

[76] Henry Prakken. Coherence and flexibility in dialogue games for argumentation. *J. Log. Comput.*, 15(6):1009–1040, 2005.

[77] Henry Prakken. A study of accrual of arguments, with applications to evidential reasoning. In *Proceedings of the 10th International Conference on Artificial Intelligence and Law (ICAIL)*, pages 85–94, 2005.

[78] Henry Prakken. An abstract framework for argumentation with structured arguments. *Argument and Computation*, 1:93–124, 2010.

[79] Henry Prakken, Chris Reed, and Douglas Walton. Argumentation schemes and generalizations in reasoning about evidence. In *Proceedings of the 9th International Conference on Artificial Intelligence and Law (ICAIL)*, pages 32–41, 2003.

[80] Henry Prakken and Giovanni Sartor. Argument-based extended logic programming with defeasible priorities. *Journal of Applied Non-Classical Logics*, 7(1), 1997.

[81] Henry Prakken and Giovanni Sartor. A logical analysis of burdens of proof. In *Legal evidence and proof: statistics, stories, logic*. Ashgate Publishing, Ltd., 2009.

[82] Teodor C. Przymusinski. On the declarative semantics of deductive databases and logic programs. In Jack Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 193–216. Morgan Kaufmann, 1988.

[83] Teodor C. Przymusinski. Every logic program has a natural stratification and an iterated least fixed point model. In *Proceedings of the 8th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 11–21, 1989.

[84] Teodor C. Przymusinski. The well-founded semantics coincides with the three-valued stable semantics. *Fundam. Inform.*, 13(4):445–463, 1990.

[85] Iyad Rahwan and Guillermo R. Simari, editors. *Argumentation in Artificial Intelligence*. Springer Publishing Company, Incorporated, 1st edition, 2009.

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

[87] Domenico Saccà. Deterministic and non-deterministic stable model semantics for unbound datalog queries. In *Proceedings of the 5th International Conference on Database Theory (ICDT)*, pages 353–367, 1995.

[88] Domenico Saccà. The expressive powers of stable models for bound and unbound datalog queries. *J. Comput. Syst. Sci.*, 54(3):441–464, 1997.

[89] Guillermo Ricardo Simari and Ronald Prescott Loui. A mathematical treatment of defeasible reasoning and its implementation. *Artif. Intell.*, 53(2-3):125–157, 1992.

[90] Stephen Toulmin. *The Uses of Argument.* Cambridge University Press, 1958.

[91] Maarten H. van Emden and Robert A. Kowalski. The semantics of predicate logic as a programming language. *J. ACM*, 23(4):733–742, 1976.

[92] Bart Verheij. Two approaches to dialectical argumentation: admissible sets and argumentation stages. In J.-J.Ch. Meyer and L.C. van der Gaag, editors, *Dutch Conference on Artificial Intelligence*, pages 357–368, Utrecht, 1996. Utrecht University.

[93] Bart Verheij. A labeling approach to the computation of credulous acceptance in argumentation. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 623–628, 2007.

[94] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior.* Princeton University Press, 1944.

[95] Gerard Vreeswijk. *Studies in Defeasible Argumentation.* PhD thesis, Dept. of Mathematics and Computer Science,Vrije Universiteit Amsterdam, 1993.

[96] Gerard Vreeswijk. An algorithm to compute minimally grounded and admissible defence sets in argument systems. In *Proceedings of the 1st Conference on Computational Model of Arguments (COMMA)*, pages 109–120, 2006.

[97] Gerard Vreeswijk and Henry Prakken. Credulous and sceptical argument games for preferred semantics. In *Proceedings of the 7th European Conference on Logics in Artificial Intelligence (JELIA)*, pages 239–253, 2000.

[98] John Hayden Woods, Andrew Irvine, and Douglas N. Walton. *Argument - critical thinking, logic and the fallacies (2. ed.).* Pearson Education, 2004.

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

[100] Yining Wu, Martin Caminada, and Dov M. Gabbay. Complete extensions in argumentation coincide with 3-valued stable models in logic programming. *Studia Logica*, 93(2-3):383–403, 2009.

[101] Jia-Huai You and Li-Yan Yuan. Three-valued formalization of logic programming: Is it needed? In *Proceedings of the 17th Symposium on Principles of Database Systems (PODS)*, pages 172–182, 1990.

[102] Jia-Huai You and Li-Yan Yuan. A three-valued semantics for deductive databases and logic programs. *J. Comput. Syst. Sci.*, 49(2):334–361, 1994.

[103] Jia-Huai You and Li-Yan Yuan. On the equivalence of semantics for normal logic programs. *J. Log. Program.*, 22(3):211–222, 1995.