Formal Methods of Argumentation as Models of Engineering Design Decisions and Processes

Jann Müller

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Department of Computer Science

University College London

December 4, 2016

I, Jann Müller, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

Complex engineering projects comprise many individual design decisions. As these decisions are made over the course of months, even years, and across different teams of engineers, it is common for them to be based on different, possibly conflicting assumptions. The longer these inconsistencies go undetected, the costlier they are to resolve. Therefore it is important to spot them as early as possible. There is currently no software aimed explicitly at detecting inconsistencies in interrelated design decisions.

This thesis is a step towards the development of such tools. We use formal methods of argumentation, a branch of artificial intelligence, as the foundation of a logical model of design decisions capable of handling inconsistency. It has three parts. First, argumentation is used to model the pros and cons of individual decisions and to reason about the possible worlds in which these arguments are justified. In the second part we study sequences of interrelated decisions. We identify cases where the arguments in one decision invalidate the justification for another decision, and develop a measure of the impact that choosing a specific option has on the consistency of the overall design. The final part of the thesis is concerned with non-deductive arguments, which are used in design debates, for example to draw analogies between past and current problems. Our model integrates deductive and non-deductive arguments side-by-side.

This work is supported by our collaboration with the engineering department of Queen's University Belfast and an industrial partner. The thesis contains two case studies of realistic problems and parts of it were implemented as software prototypes. We also give theoretical results demonstrating the internal consistency of our model.

Acknowledgements

There are a number of people who, in various ways, are just as responsible for the successful completion of this project as I am. I would like to extend my sincere gratitude to everyone who has supported my work on this project over the last few years.

First I would like to thank Prof. Anthony Hunter who supervised my thesis at UCL. Besides being the most knowledgeable mentor he has always shown enthusiasm and willingness to discuss new ideas, as well as a great deal of patience. He was always willing to dedicate time to this project, both in personal meetings and when reviewing the latest drafts. I do not think there could be a better supervisor.

Then I would like to thank my wife Mary-Anne, who thought I would submit "a couple of months" after we first met. She too has been very patient with this thesis, and her encouragement (and help with the children) greatly contributed to its successful completion.

I want to thank my former coworkers at SAP: Phil, Alan, Fergal, Mary, and our friends from East Germany, for creating a stimulating environment for my research, and for being good colleagues all around. Further I want to thank David Payne at QUB for sharing the results of his experiments, and Fiona Browne for good advice throughout the project. With my old friend Christian Hoffmann I had many conversations that helped shaped both our views on uncertainty and I thank him for that. I am also grateful to my second supervisor Prof. John Dowell for his insights on engineering design, and to Tobias Trapp for working with me on an interesting application of the theory.

Finally I want to thank Pat and Carmel for support and proof-reading, and my parents for supporting me in my academic endeavours.

This work is supported by SAP AG and the Invest NI Collaborative Grant for R&D - RD1208002.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

In this chapter we identify the problem solved in this thesis and give an overview of the solution. We start by characterising the engineering design process in Section 1.2.1. We then derive a list of requirements for a formal model of design processes (1.2.2), which establishes the scope of the thesis. We discuss some traditional tools for decision analysis (1.2.3) and justify our choice of argumentation as the foundation of our model of design processes (1.2.4). The chapter is completed by an overview of the thesis (1.3), a list of contributions of this thesis to the state of the art (1.4) and a list of peer-reviewed publications that form parts of the work presented here (1.5).

1.2 Decision Processes

Decisions are ubiquitous in our professional and private lifes. Some decisions are made in isolation – what car to buy, where to go for lunch, whether to take an umbrella. However, many decisions are made in the context of *decision processes*. The term describes sequences of related decisions with a product as the overall result. We use the word product rather loosely here and mean any kind of design, specification etc. that is the outcome of a number of individual decisions.

1.2.1 Decisions in Context: Motivating Examples

Decision theory, the science of decision-making, is concerned primarily with isolated decisions. However, decisions in practice rarely exist without context: They are influenced by earlier decisions and in turn will impact later decisions. This is especially true in engineering design, where decisions are part of a long-running process that can span

months or years.

The theory of engineering design distinguishes two kinds of thought processes; intuitive thinking and discursive thinking [1, page 48]. The former is characterised by sudden, unforeseeable insights resulting from unconscious "thinking". The latter, on the other hand, "is a conscious process that can be communicated and influenced" [1, page 48f], characterised by its systematic structure comprising a series of steps. Since intuition has the disadvantage of being unpredictable, a stronger reliance on discursive methods is advisable [1, page 55]. Further, the discursive method stimulates intuition, indicating that the two are symbiotic.

The systematic approach requires a precise definition of goals and boundary conditions, which is a prerequisite for breaking down a large problem into a sequence of smaller tasks. If requirements and constraints are clearly stated, then the systematic approach may help in minimising errors resulting from prejudice (experience) of the designers [1, page 55f]. A crucial part of both kinds of thought processes is documenting any decisions that were made.

The value of documentation does not only lie in the support it provides for discursive thought processes. Accurate project documentation provides a degree of protection from litigation and, more importantly, constitutes a competetitive advantage to its owners [2, page 118]. Design documentation is often incomplete [3]: Decisions are justified retroactively, so the documentation contains mostly arguments in favour of the option that was chosen, and does not represent the full picture of the pros and cons of all options available at the time. As a result of this bias, design documents do not give comprehensive evidence of the design process - they only document the outcome of that process.

Engineering design processes consist of sequences of decisions, ranging from small to complex, which collectively determine the final design. At this point we will introduce some terminology. Every decision requires a set of at least two distinct options; the set of options is the outcome of the previous step in the process, namely the actual design, in which different possibilities are systematically produced. The decision making itself then is rather mechanical, as it is an evaluation of the design options according to some clearly defined goals and requirements. Decisions thus mark the boundaries of various stages in the design process.

We can find further evidence for this view in the literature. Pahl and Beitz [1] state that "without decisions there can be no progress" [1, page 54]. In their methodology, each of the successive phases of the design process is ended by a decision. This decision determines whether to advance to the next stage, or to repeat the current stage with an adapted list of requirements, if no satisfactory solution has been found [1, page 66]. The "Total Design" approach by Pugh [4] exhibits a similar structure, although this method only involves a single decision phase at the transition from concept design to detailed design [4, page 11]. The design process by Ertas and Jones [5] features decisions at two distinct stages, first after the feasibility assessment of the initial conceptualisation, and again after the preliminary and detailed design phases. At both of those stages, the decision determines whether to proceed to the next step or to repeat the previous steps. The need for automated tools to support decision making in engineering design has long been recognised [6].

It is clear that while there is no universal definition of "the" engineering design process, a consensus exists that decision making plays a crucial role in transitioning from one phase to the next, and thus in making progress towards the eventual design. It is also clear that knowledge is re-used throughout the decision process, be it in written form (design documents, requirements, experimental data) or in the minds of engineers.

1.2.2 Requirements for a Formal Model of Decision Processes

Decisions are made within a set of constraints that change regulary. Within those constraints, options are chosen intuitively, not by purely mechanical means. Further, decision making processes are distributed across several teams or individuals. Earlier decisions influence later ones, both in their range of available options and in their constraints. Knowledge is re-used throughout the process in two forms: As concrete artifacts (specifications, requirements, experimental data, etc.) and intangibly as the knowledge of people involved in the decision making process. In addition, there are trade-offs that have to be made in enineering design, as one tries to balance cost and, broadly, quality (including aesthetics, quality control, margins of error).

There have been efforts to support the knowledge-specific tasks of decision processes in order to increase product quality and to avoid redundant work, for example as part of knowledge management in engineering [7]. Previous approaches that recorded

the reasoning behind design decisions have not found widespread industry acceptance [8, 9], broadly because the effort required for their implementation was unproportional to the benefits they provided. The survey by Lee [8] suggests that this is due in part to their lack of automated reasoning facilities, which made it more difficult to apply the knowledge stored in those systems.[1]

Based on this discussion we identify the following requirements for a knowledge-based model of engineering design processes.

RQ1 Represent design decisions with the pros and cons for each of their options, including the reasoning that was applied to arrive at the pros and cons and possible worlds in which the underlying assumptions hold.

RQ2 Reason about decisions so represented, specifically by characterising the decision rules used to arrive at the decision, and determining the effect that "choosing an option" has on the knowledge base.

RQ3 Formulate sequences of decision problems in which decisions made at one stage influence the range of options and constraints for decisions made later in the process, and assess the impact of changing a previous decision.

RQ4 Combine various forms of reasoning such as deductive arguments, empirical evidence, intuition and heuristics.

The individual decisions mentioned in RQ1 and RQ2 are the smallest "building blocks" of design processes. By requiring that the effects of choosing an option should be discoverable through inference (RQ2) we address the central weakness of previous approaches to modeling engineering design processes, which were entirely syntactical in nature - no inferences could be made. As a result, they required a high amount of user input. Through the ability to make automated inferences, any knowledge expressed in our model becomes reusable. RQ3 is the central requirement, placing individual decisions (as in RQ1) in the context of the engineering processes that we are trying to model. And finally RQ4 addresses the fact that most arguments made by human beings are not expressible in classical logic, because they rely on a number of implicit assumptions.

[1] cf. Krause *et al.*: "(...) the application of knowledge is the most essential task of knowledge management." [7, p.215]

1.2.3 Decision Models

In this section we discuss the traditional approach to decision modelling and point out where it fails to meet the requirements set in Section 1.2.2.

Decision theory is a rich research area on its own. We will attempt to briefly characterise some of the assumptions made by most of the popular decision models, bearing in mind that all of those methods focus on individual decisions only and thus fail to satisfy requirements RQ2 and RQ3.

1.2.3.1 Decision Making Under Uncertainty

The first axiomatic account of decision theory is given by Savage [10]. It breaks with the dominating, frequentist interpretation of probabilities and introduces a subjective, bayesian alternative. The debate over the two interpretations is still on-going, but we can make some observations about decision theory that are acceptable regardless of one's interpretation of probability.

In the probabilistic approach to decision theory, the decision maker's goal is to maximise *utility* [11]. The abstract concept of utility is expressed as a function from possible outcomes to the real numbers. In financial decisions, utility is often identified with returns, even though this does not appear to be an accurate reflection of human preferences [12].

The third fundamental concept in decision theory, besides utility and possible outcomes, is the set of possible worlds. This is an expression of the uncertainty inherent in most decisions. Each possible world is associated with a probability. The three concepts are tied together by a function that relates options and possible worlds to outcomes.

An analysis of classical decision theory from the viewpoint of our requirements (Section 1.2.2) reveals three shortcomings. First, the assumption that the probabilities of possible worlds can be quantified has already been shown to be problematic, and in many debates no quantitative information about the proabilities of possible worlds is available.

The second problem is with the mapping of options and possible worlds to outcomes in the form of real-valued functions. Such functions are in general not a reusable representation of knowledge since the only way that two real-valued functions

can be combined is by combining their results. Hence, there is no meaningful inference that can be performed automatically. In order to assess the impact of, say, changed assumptions on the utility functions, the thought processes that led to the original assignments have to be performed again (by a human being), because they are lost as soon as the number is decided on. The probabilistic-quantitative approach to decision theory therefore fails to meet requirement RQ2.

Thirdly, decision theory as outlined above does not have a notion of decision processes (E.g. as in 1.2.1) and thus cannot meet requirement RQ3. Because of the models' lack of support for automated inferences, any concept of decision processes would offer no additional value, no knowledge that is not already apparent by simply viewing each of the decisions on its own.

1.2.3.2 Multi-Criteria Decision Making

A complementary approach to decision theory is multi-criteria decision making. Here, the decision maker's preferences are given on multiple dimensions rather than with a single utility function. The key question is how to choose an option if no option completely dominates (outperforms) the others. In this approach, uncertainty does not play as prominent a role as it does in the utility approach discussed above. However, multi-criteria decision making also does not meet requirements RQ1 to RQ3, for the same reasons as the utility approach.

1.2.3.3 Summary & Critique

Regardless of its usefulness for our particular purpose, decision theory has been questioned on general grounds also. Quantitative decision theory as a descriptive model of human decision behaviour has drawn some criticism because of the assumptions it makes. The tendency of people to apply heuristics in decision making [12] (for example to overvalue outcomes that are certain over those that are probable) inspired the development of prospect theory (Kahneman and Tversky [13]). The assumption that decision makers act rationally has been drawn into question by studies in cognitive psychology, e.g. Stewart *et al.* [14] and Gigerenzer [15]. Certain phenomena such as the purchase of insurance policies and lottery tickets as well as the high popularity of government bonds cannot adequately be described with expected utility models [16].

In conclusion, decision theory as described above does not have a representation

of the knowledge from which its utilities, outcomes etc. are derived, which prevents its models from being re-used. Because of that, neither decision making under uncertainty nor multi-criteria decision making are suitable starting points for our search of a model that meets the requirements we listed above.

1.2.4 Formal Methods of Argumentation

In this section we introduce formal methods of argumentation and sketch informally how they might address our requirements. An overview of relevant work in argumentation can be found in Chapter 2.

Argumentation is a field within Artificial Intelligence (AI) research that uses the concept of arguments (consisting of claims and support) to reason about inconsistent knowledge bases [17, 18]. Apart from being a research area in its own right, argumentation has been applied to problems in agent-based communication [19, 20, 21, 22], satellite image analysis [23], medicine [24, 25], law [26, 27] and others.

Common uses ofw argumentation systems include: To identify conflict-free (acceptable) subsets of inconsistent knowledge bases; to organise conflicting knowledge in a way that highlights how conflicting conclusions can be inferred; to reason with alternative notions of conflict (i.e. ones that do not necessarily lead to absurdity in the form of $a \wedge \neg a$).

The suitability of argumentation-based methods for our requirements will be discussed in detail below, but first we will explain how argumentation addresses one critical requirement that is not met by other decision models (in Section 1.2.3), namely the re-usability of knowledge implied by (RQ2). The key to reusability is being able to combine two elements of the same type (the one that is re-used and the new one) into a new element of the same type – that is to have an associative binary operation. In argumentation, this operation is simply the set union of knowledge bases. Any inconsistencies that emerge from the combination of two sets of propositions will be dealt with by the argumentation system. We will return to this point on several occasions throughout this thesis, but for now we can summarise that argumentation systems fulfil requirement RQ2 automatically.

Having identitified formal methods of argumentation as a potential foundation for a model of decision processes, we can now assess what previous work exists in this

area, and whether it meets our requirements. There exists a large body of work on decision making with argumentation [28, 29, 22, 24, 30, 31, 32, 33], and some of those contributions are aimed explicitly at representing design debates. This covers the first part of RQ1, but none of the existing approaches combine the idea of "possible worlds" in which assumptions hold (cf. Section 1.2.3.1) with the weighing up of pros and cons for each option (cf. Section 1.2.3.2). This is an area where argumentation-based approaches to decision making can learn from decision theory.

A common characteristic of all argumentation-based approaches to decision making is that they focus on individual decisions. As a result, none of them meet requirements RQ2 and RQ3, which have to be addressed on the level of decision processes.

The last requirement RQ4 is somewhat orthogonal to the previous three, because it relates to the nature of arguments used in by model, rather than the model itself. There are two distinct areas of previous work: meta-argumentation and argument schemes. While argument schemes are more directly related to RQ4 ("integrating different kinds of reasoning"), we claim that they are an example of meta-argumentation and thus should be treated the same way as meta-arguments.

To summarise, prior art in argumentation covers parts of RQ1 and RQ4, and for RQ2 and RQ3 no previous work exists. We intend to fill that gap with the work presented here.

1.3 Thesis Overview

The thesis contains the following chapters:

1. *Introduction* This chapter

2. *Background* A review of the foundations formal methods of argumentation as far as they are required to read the thesis, as well as other definitions that are required. Each of the three following chapters has a separate review of the literature relevant to it.

3. *Decision making with argumentation* An argument-based model for multi-criteria decision making and decision making with uncertainty. The model is based on a realistic use case developed with an industrial partner. It also includes a characterisation of decision rules, and a method for adjusting the knowledge base after

making a decision

4. *An argumentation-based model of decision processes* A model of decision sequences in which it is possible to assess the impact of choosing a specific option on the justifications of other decisions in the sequence.

5. *Argument schemes for decision making* Non-deductive arguments (argument schemes) cast as an instance of meta-argumentation. A special emphasis is on using argumentation to interpret experimental data (by way of analogies), which is illustrated by an extensive use case.

6. *Conclusion & Discussion* A summary and review of the results and some discussion of future work

1.4 Contributions

The contributions of this thesis to the state of the art are

- A formal model of decision making using the ASPIC+ argumentation system. Our model combines multi-criteria decision making with decision making under uncertainty, by mapping possible worlds (caused by uncertainty in decision making) to preferred extensions of argument graphs (a manifestation of conflicting inferences).

- A method for comparing decision rules in argument-based decision making, e.g. for their optimism or their decisiveness

- A method for adjusting a knowledge base to reflect the fact that a decision has been made

- A method for describing and analysing sequences of argument-based decisions, and measuring the impact that changing a previous decision has on the overall consistency of the designs

- A framework in which non-deductive arguments, such as analogies and references to authority, can be expressed side-by-side with deductive (logical) arguments, as well as arguments not about the domain but about other arguments (i.e. meta-argumentation)

1.5 Publications

Parts of this thesis have been published in peer-reviewed conference proceedings.

1.5.1 Main authorship

The following papers are directly based on this thesis:

1. Jann Müller and Anthony Hunter: *An Argumentation-Based Approach for Decision Making.* 2012 IEEE 24th International Conference on Tools with Artificial Intelligence (ICTAI), ppage 564–571 [34]
   This work is on a system for multi-criteria decision making using the ASPIC+ argumentation system [35]. The paper is the foundation of Chapter 3, Section 1.

2. Jann Müller and Anthony Hunter: *Comparing Decision Rules in Argument-Based Decision Making with Uncertainty.* (preparing for re-submission after receiving helpful feedback from reviewers)
   Another paper on decision making with argumentation; the focus is on formal criteria for the comparison of decision rules. This paper is the foundation of Chapter 3 Section 2.

3. Jann Müller, Anthony Hunter, Philip S. Taylor: *Meta-level Argumentation with Argument Schemes.* Scalable Uncertainty Management - 7th International Conference (SUM) 2013, Proceedings: LNCS 8078, ppage 92–105 [36]
   A publication on meta-argumentation with argument schemes. This paper is a short version of Chapter 5.

4. Jann Müller and Anthony Hunter: *Deepflow: Using argument schemes to query relational databases.* Computational Models of Argument - Proceedings of COMMA 2014 in: Frontiers in Artificial Intelligence and Applications 266, ppage 469-470 [37]
   This demo paper describes the implementation of a system for using argument schemes (Chapter 5) to generate queries of SQL databases.

A journal article based on Chapter 4 (theory of decision processes) is in preparation.

1.5.2 Co-authorship

This thesis constitutes a part of project DEEPFLOW, an R&D collaboration between SAP AG (my employer during the project), the Queen's University Belfast's Mechanical Engineering Department and the Computer Science department at the University of Ulster. This collaboration provided me with a realistic understanding of the decision documentation problem in engineering design, and also gave me the opportunity to contribute to other publications in related areas. The findings from this research influenced my understanding of argumentation within engineering design processes and, by extension, this thesis.

1. Niall Rooney, Hui Wang, Fiona Browne, Fergal Monaghan, Jann Müller, Alan Sergeant, Zhiwei Lin, Philip S. Taylor, Vladimir Dobrynin: *An Exploration into the Use of Contextual Document Clustering for Cluster Sentiment Analysis.* Recent Advances in Natural Language Processing (RANLP) 2011: 140-145 [38]

2. Fiona Browne, David A. Bell, Weiru Liu, Yan Jin, Colm Higgins, Niall Rooney, Hui Wang, Jann Müller: *Application of Evidence Theory and Discounting Techniques to Aerospace Design.* Advances in Computational Intelligence - 14th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, IPMU 2012 Part III: 543-553 [39]

3. Jann Müller and Tobias Trapp: *Using Argumentation to Develop a Set of Rules for Claims Classification.* 7th International KES Conference on Intelligent Decision Technologies (KES-IDT): 459-469 [40]

Chapter 2

Background

2.1 Introduction

In this chapter we will review the background in formal methods of argumentation upon which the rest of the thesis is built. There are two main parts. We first introduce abstract argument graphs (in Section 2.2) and then define the ASPIC+ argumentation system [35, 41] which we will use for concrete instantiations of arguments (Section 2.3). We then present additional notation and definitions (Section 2.4) and conclude the chapter with a brief discussion (Section 2.5).

Abstract argument graphs allow us to model the relationships between arguments. Unlike dialectical trees (an alternative method for computing the acceptability of arguments), graphs make it possible to evaluate all arguments in a debate at the same time, instead of one argument at a time. However, abstract arguments do not contain any information about the content of arguments. In order to be able to identify all relationships between arguments – and not just those that are explicitly mentioned in a design document and can thus be mapped directly to an attack in the argument graph – we need a formal language to give structure to the arguments, or to *instantiate* them. Several instantiations have been proposed in the literature, and we will discuss them in Section 2.5. For this thesis we decided to use the ASPIC+ argumentation system. ASPIC+ is quite flexible, allowing us to leave the choice of underlying logical language to the eventual application of our model.

2.2 Abstract Argumentation

Dung's 1995 article [42] introduced argument graphs with arguments as nodes and their "attacks" (a notion of conflict) as directed edges. Argument graphs are given an interpretation using one of several extension semantics.

The internal structure of arguments is not specified, so they are only defined in their relationships with other arguments. We will present the concept of argument graphs in some detail here, because it is referenced in almost every single publication discussed in this chapter.

Definition 1 (Argument Graph)**.** *An* argument graph *is tuple* (A, Att) *where* A *is a set and* $Att \subseteq A \times A$.

In an argument graph $G = (A, Att)$, elements of the set A are called arguments and Att is the "attacks" relation. Given two arguments $a, b \in A$, a attacks b if $(a, b) \in Att$. We will display argument graphs visually by drawing circles for arguments and arrows for attacks, as shown in Figure 2.1.

Example 1. *Imagine we are faced with the problem of designing a structure that has two components which are joined using a bracket fixed by bolts. Our task is to decide what type of bolts to use, whether or not to use a shim (a thin sheet of metal) between the bracket and the components, and how many layers of varnish to apply. Some requirements are to maintain the structural integrity of the component and to achieve a high resistance to corrosion. The following arguments might be put forward:*

a_1 *Not using a shim means that the structure remains balanced. Therefore, it will not be damaged.*

a_2 *Steel/titanium bolts cause microscopic fractures in the two parts, leading to damage to the structure.*

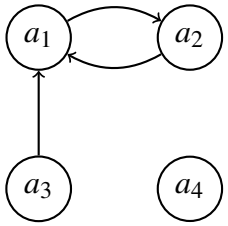


Figure 2.1: Graph for Example 1

a_3 *Steel/titanium bolts are too strong for the material that the structure is made of, so there will be microscopic fractures.*

a_4 *The structure is highly corrosion resistant, because two layers of varnish are used.*

The argument graph for this example is given by (A, Att) *with* $A = \{a_1, \ldots, a_4\}$ *and* $Att = \{(a_1, a_2), (a_2, a_1), (a_3, a_1)\}$ *and visualised in Figure 2.1.*

There are several acceptability semantics, based on the idea that a set of arguments is *admissible* if it defends itself against any attacks, and does not attack itself. An acceptability semantics is a way of computing certain admissible sets of arguments, called extensions, from an argument graph. The semantics differ in their degree of credulity, that is, whether arguments can stand for themselves (and hence may defend themselves against attackers) or not (and hence must be defended by other arguments).

Acceptability semantics select conflict-free sets of arguments. Sets of arguments are conflict-free if there is no attack between any two of their arguments. A set defends an argument a if it can attack any attacker of a.

Definition 2 (Conflict-free, defence)**.** *Let* (A, Att) *be an argument graph and let* $B \subseteq A$*.*

- *B is* conflict-free *iff there exist no a, b in B such that a attacks b.*
- *B* defends *an argument a iff for each argument* $b \in A$*: If b attacks a, then there exists an argument c in B such that c attacks b.*

Note that if an argument has no attackers then it is defended by and acceptable to any set of arguments.

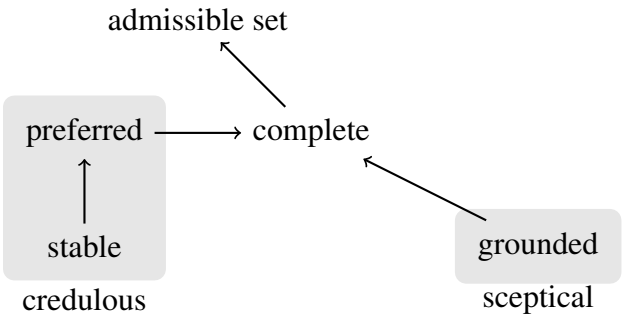


Figure 2.2: Extension semantics

Example 2. *The graph in Figure 2.1 on page 24 has several conflict-free sets, for example* $\{a_3, a_2, a_4\}$ *and* $\{a_4, a_1\}$.

Semantics in abstract argumentation make use of the following notions of acceptability and admissibility.

Definition 3 (Acceptable, admissible)**.** *Let* $G = (A, Att)$ *be an argument graph.*

1. *An argument* $a \in A$ *is* acceptable *with respect to a set* $S \subseteq A$ *of arguments if and only if for each argument* $b \in A$*: If* b *attacks* a *then* b *is attacked by some argument in* S.

2. *A conflict-free set of arguments* $S \subseteq A$ *is* admissible *iff each argument in* S *is acceptable with respect to* S

The following result by Dung [42] makes a connection between acceptability and admissibility.

Proposition 1 ([42])**.** *Let* S *be an admissible set, and* a *and* a' *be arguments that each are acceptable with respect to* S*. Then*

1. $S' = S \cup \{a\}$ *is admissible*

2. a' *is acceptable with respect to* S'.

Semantics in Dung's framework are sceptical or credulous. Sceptical semantics result in a single extension called the grounded extension, whereas credulous semantics may result in more than one extension, called preferred extensions. We start with the grounded extension, which requires us to introduce the characteristic function of an argument graph. This function maps a set of arguments to the set of arguments acceptable to it.

Definition 4 (Characteristic Function)**.** *The* characteristic function *of an argument graph* $G = (A, Att)$ *is* $\mathscr{F}_G : 2^{\mathscr{A}} \to 2^{\mathscr{A}}$ *such that* $\mathscr{F}_G(B) = \{a \in A \mid a \text{ is acceptable with respect to } B\}$.

Example 3. *For the graph* $G = (\{a_1,\ldots,a_4\},\{(a_1,a_2),(a_2,a_1),(a_3,a_1)\})$ *from Figure 2.1, we can compute for example*

$$\mathscr{F}_G(\emptyset) = \{a_3,a_4\} \qquad \mathscr{F}_G(\{a_1\}) = \{a_3,a_4\}$$
$$\mathscr{F}_G(\{a_3\}) = \{a_2,a_3,a_4\} \qquad \mathscr{F}_G(\{a_2\}) = \{a_2,a_3,a_4\}$$

Definition 5 (Sceptical Acceptability)**.** *The* grounded extension *of an argument graph G is the least fixed point of* $\mathscr{F}_G$

Example 4. *The characteristic function of the graph* $G = (\{a_1,\ldots,a_4\}, \{(a_1,a_2), (a_2,a_1), (a_3,a_1)\})$ *from Figure 2.1 reaches a fixed point after one iteration:*

$$\mathscr{F}_G(\emptyset) = \{a_3,a_4\}$$
$$\mathscr{F}_G(\{a_3,a_4\}) = \{a_2,a_3,a_4\}$$
$$\mathscr{F}_G(\{a_2,a_3,a_4\}) = \{a_2,a_3,a_4\}$$

Every argument graph has a single grounded extension. This is not the case for preferred extensions, of which there can be more than one.

Definition 6 (Credulous Acceptability)**.** *A* preferred extension *of an argument graph G is a maximal admissible set.*

Besides grounded and preferred extensions, Dung introduced two additional extension semantics. A conflict-free set of arguments is a *stable extension* if it attacks every argument that does not belong to it, and an admissible set of arguments S is a *complete extension* iff each argument which is acceptable with respect to S belongs to S. The relationships between the four semantics are shown in Figure 2.2 (arrows indicate the "is-a" relationship). In this thesis however we will use grounded and preferred semantics.

The function $\Sigma_s(G)$ computes the set of s-extensions of an argument graph G, with $s \in \{\mathsf{pr},\mathsf{gr}\}$ (preferred and grounded, respectively). Σ_{pr} may return an empty set and Σ_{gr} returns a set with exactly one element (which in turn may be the empty set). Other semantics besides preferred and grounded have been discussed in the literature, but in this thesis we will only use the two.

The following result by Dung will be useful in proofs about the acceptability of arguments.

Proposition 2 (Dung 1995, [42])**.** *Let $G = (A, Att)$ be an argument graph. Then*

1. *The set of all admissible sets of G forms a complete partial order with respect to set inclusion*

2. *For each admissible set S of G, there exists a preferred extension $E \in \Sigma_{\mathsf{pr}}(G)$ such that $S \subseteq E$*

3. *Every argumentation framework posseses at least one preferred extension*

Argument graphs can tell us which sets of arguments are acceptable, but they do not contain any information about the structure of arguments. This gap will be filled by the ASPIC+ framework.

2.3 Aspic+

In this section we introduce the ASPIC+ argumentation system which will be used throughout the thesis. ASPIC+ [35, 41] is a framework for structured argumentation that combines strict and defeasible inference rules. Defeasible rules, unlike their strict counterparts, can only be applied in contexts where they are not attacked by exceptions. Exceptions are arguments whose conclusion is a contrary of the rule itself (as opposed to a contrary of one of the rule's premises). The applicability of defeasible rules in ASPIC+ is therefore determined locally for each application.

2.3.1 Formal Definition of Aspic+

The ASPIC+ framework assumes an unspecified logical language $\mathscr{L}$. Inference rules can be strict or defeasible. If the conditions of a rule hold, then its conclusion must be accepted unconditionally if it is a strict rule, but if it is a defeasible rule then there may be exceptions that render the rule unapplicable.

Rules in ASPIC+ can be used to encode domain-specific knowledge or general patterns of reasoning [41]. An example of a domain-specific rule is $\texttt{bird}(X) \Rightarrow \texttt{fly}(X)$, and an example of a general pattern is $(a \rightarrow b), a \Rightarrow b$. In this thesis we use ASPIC+ rules for domain-specific inferences, and rely on ASPIC+ arguments (Definition

9 below) as the sole general reasoning pattern. It would also be possible to encode domain-specific rules in the logical language $\mathscr{L}$ and add a general inference rule for *modus ponens* to the ASPIC+ system [35].

In ASPIC+ the choice of underlying logic is left to the user. An argumentation system therefore contains as parameters both the logical language used inside arguments ($\mathscr{L}$) and a means of determining the relationship *between* arguments ($\bar{\cdot}$ and $\mathscr{R}$).

Definition 7 (ASPIC+ argumentation system)**.** *An* argumentation system *is a four-tuple* $AS = (\mathscr{L}, \bar{\cdot}, \mathscr{R}, \leq)$ *where*

- $\mathscr{L}$ *is a logical language*
- $\bar{\cdot}$ *is a contrariness function with* $\bar{\cdot} : \mathscr{L} \to 2^{\mathscr{L}}$
- $\mathscr{R} = \mathscr{R}_s \cup \mathscr{R}_d$ *is a set of strict* ($\mathscr{R}_s$) *and defeasible* ($\mathscr{R}_d$) *inference rules of the form* $\varphi_1, \ldots, \varphi_n \to \varphi$ *and* $\varphi_1, \ldots, \varphi_k \Rightarrow \varphi$ *respectively (where* φ_i, φ *are meta-variables ranging over* $\mathscr{L}$ *and* $k, n \geq 0$*), and such that* $\mathscr{R}_s \cap \mathscr{R}_d = \emptyset$
- $\leq$ *is a partial order*[1] *on* $\mathscr{R}_d$

We will give a comprehensive example on page 34, when all notation has been introduced. The function name assigns a name to each defeasible rule, so its signature is $\mathsf{name} : \mathscr{R}_d \to \mathscr{L}$. We will use an infix notation $\langle r \rangle$ to mean $\mathsf{name}(r)$. Note that $\langle \cdot \rangle$ is a *meta-level* operator. A set $S \subseteq \mathscr{L}$ is *consistent* iff $\nexists \psi, \varphi \in S$ such that $\psi \in \overline{\varphi}$, or *inconsistent* otherwise.

In ASPIC+, argumentation systems set out the general rules for arguments – what constitutes an argument and what determines an attack. To actually use these rules, one requires a knowledge base that contains knowledge about a particular problem.

Definition 8 (ASPIC+ knowledge base)**.** *A* knowledge base *in an argumentation system* $(\mathscr{L}, \bar{\cdot}, \mathscr{R}, \leq)$ *is a pair* $(\mathscr{K}, \leq')$ *where*

- $\mathscr{K} \subseteq \mathscr{L}$ *and* $\mathscr{K} = \mathscr{K}_n \cup \mathscr{K}_p \cup \mathscr{K}_a$ *such that the three constituent subsets of* $\mathscr{K}$ *are disjoint*
- $\leq'$ *is a partial order on* $\mathscr{K} \setminus \mathscr{K}_n$

[1] A definition for partial order is given on page 36, along with definitions for other general concepts.

In a knowledge base, the set $\mathscr{K}_n$ contains axioms, the set $\mathscr{K}_p$ contains premises and the set $\mathscr{K}_a$ contains assumptions. The intuition behind this stratification of $\mathscr{K}$ is that the three sets represent different kinds of certainty: Axioms are similar to strict rules in that they always apply and cannot be attacked. Formulae in $\mathscr{K}_p$ and $\mathscr{K}_a$ can be attacked. Attacks on premises in $\mathscr{K}_p$ are resolved using the preference relations $\leq$ and $\leq'$, whereas attacks on assumptions $\mathscr{K}_a$ always succeed. Arguments combine knowledge from a knowledge base with rules from an argumentation system. They are defined in Definition 9 below. Please note that each of the two conditions defines the functions prem, conc, sub, rules and topRule which return the premises, conclusion, sub-arguments, rules and top rule of an argument. Further, Definition 9 Cond. 2 covers both strict rules ($\rightarrow$) and defeasible rules ($\Rightarrow$), separated by the / symbol.

Definition 9 (ASPIC+ arguments)**.** *An* argument *"a" on the basis of a knowledge base* $(\mathscr{K}, \leq')$ *in an argumentation system* $(\mathscr{L}, \bar{}, \mathscr{R}, \leq)$ *is:*

1. φ *if* $\varphi \in \mathscr{K}$ *with* $\mathsf{prem}(a) = \{\varphi\}$; $\mathsf{conc}(a) = \varphi$; $\mathsf{sub}(a) = \{a\}$; $\mathsf{rules}(a) = \emptyset$; $\mathsf{topRule}(a) = \mathsf{undefined}$

2. $a_1, \ldots, a_n \rightarrow / \Rightarrow \psi$ *if* $a_1, \ldots, a_n$ *are finite arguments such that there exists a strict/defeasible rule* $\mathsf{conc}(a_1), \ldots, \mathsf{conc}(a_n) \rightarrow / \Rightarrow \psi$ *in* $\mathscr{R}_s/\mathscr{R}_d$
   $\mathsf{prem}(a) = \mathsf{prem}(a_1) \cup \ldots \cup \mathsf{prem}(a_n)$
   $\mathsf{conc}(a) = \psi$
   $\mathsf{sub}(a) = \mathsf{sub}(a_1) \cup \ldots \cup \mathsf{sub}(a_n) \cup \{a\}$
   $\mathsf{topRule}(a) = a_1, \ldots, a_n \rightarrow / \Rightarrow \psi$

We use the symbol $\mathscr{A}$ for the set of all arguments, and use capital letters $A, B, \ldots$ to denote sets of arguments. The set of arguments for a given knowledge base KB is denoted with $\mathsf{args}(KB)$.

Once a number of arguments have been established, their inconsistency can be analysed in an argument graph, by determining attacks between arguments. In ASPIC+, three kinds of attack are possible: Rebuttal, undermining and undercut. A rebuttal occurs if an argument is attacked on its conclusion. Undermining is an attack on an argument's premises and undercut is an attack on a defeasible inference step (application of a defeasible rule).

Definition 10 (ASPIC+ Attack)**.** *Let a, b be two arguments on the basis of a knowledge base $(\mathscr{K}, \leq')$ in an argumentation system $(\mathscr{L}, \bar{\cdot}, \mathscr{R}, \leq)$. a* attacks *b if and only if*

1. $\mathsf{conc}(a) \in \overline{\mathsf{conc}(b)}$ *(rebuttal) or*

2. $\mathsf{conc}(a) \in \overline{\langle \mathsf{topRule}(b) \rangle}$ *(undercut) or*

3. $\exists b' \in \mathsf{sub}(b)$ *such that a attacks b' (undermining on b')*

To determine which attacks result in defeats (i.e. are valid), we assume a partial order $\preceq$ on the constructed arguments. In [35], a method is presented for deriving a definition of $\preceq$ from $\leq$ and $\leq'$, the orderings of non-axiom premises and defeasible rules. However, any partial order of arguments $\preceq$ is acceptable for the following definition:

Definition 11 (ASPIC+ defeat)**.** *Let a, b be two* ASPIC+ *arguments on the basis of a knowledge base $(\mathscr{K}, \leq')$ in an argumentation system $(\mathscr{L}, \bar{\cdot}, \mathscr{R}, \leq)$. a* defeats *b iff*

1. *a undercuts b or*

2. *a rebuts/undermines b on b', and either a contrary rebuts/undermines b or $a \not\prec b'$.*

The above definitions allow us to compute the argument graph of an ASPIC+ system, by taking its set of arguments and their defeats (Definition 11):

Definition 12 (Argument graph of an ASPIC+ system)**.** *Given a knowledge base $KB = (\mathscr{K}, \leq')$ in an argumentation system $AS = (\mathscr{L}, \bar{\cdot}, \mathscr{R}, \leq)$, the* argument graph *of KB and AS is (A, Att) where*

1. *$A = \mathsf{args}(KB)$ and*

2. *$Att = \{(a,b) \in A \mid a \text{ defeats } b\}$ (Definition 11)*

Definition 12 is the connection between ASPIC+, described here, and argument graphs, described in Section 2.2. In this thesis, the function of the ASPIC+ framework is to construct arguments, and the function of argument graphs is to select acceptable subsets of arguments.

2.3.2 ASPIC+ Conventions

As we mentioned before, we will adopt some conventions for ASPIC+ knowledge bases in this thesis. The first set of conventions aims to streamline the presentation of our ideas by simplifying definitions and proofs, and the second set of conventions enables us to reason about arguments about rules, by restricting the arguments we can potentially form.

At the same time, all knowledge bases considered in this thesis are valid ASPIC+ knowledge bases, so all related work on ASPIC+ applies to our system without adjustments. However, not all of our results transfer to general ASPIC+ systems.

In the following paragraphs, remember that ASPIC+ systems are four-tuples $(\mathscr{L}, \bar{}, \mathscr{R}_s \cup \mathscr{R}_d, \leq)$ and ASPIC+ knowledge bases are pairs $(\mathscr{K}, \leq')$ with $\mathscr{R} = \mathscr{R}_d \cup \mathscr{R}_s$, $\leq$ a partial order of $\mathscr{R}_d$, $\mathscr{K} = \mathscr{K}_n \cup \mathscr{K}_p \cup \mathscr{K}_a$ with $\leq'$ a partial order of $\mathscr{K} \setminus \mathscr{K}_n$.

2.3.2.1 Simplifications

As this thesis is about modeling human argumentation, we regard all knowledge to be defeasible and consequently all our ASPIC+ rules will be defeasible. Concretely, we require that $\mathscr{R}_s = \emptyset$ (no strict rules). This means that $\mathscr{R} = \mathscr{R}_d$. We also make no assumptions about the ordering of rules in $\mathscr{R}$, so we define $\leq$ to contain only the minimum information necessary to make it a partial order - that is, $\leq$ only contains pairs of rules (r, r) so that it satisfies the property of reflexivity: $\leq = \{(r, r') \in \mathscr{R}_d \times \mathscr{R}_d \mid r = r'\}$.

Regarding knowledge bases, we assume there are no ASPIC+-assumptions (in the set $\mathscr{K}_a$). Whenever the behaviour of assumptions (literals that can be disproved) is required, we will use a defeasible rule with empty antecedent instead. For example, we write $\Rightarrow \texttt{bird}(\texttt{Tweety})$ to express the assumption that Tweety is a bird. This convention allows us to introduce attacks on assumptions that always succeed, through a literal $\neg\langle \Rightarrow \texttt{bird}(\texttt{Tweety})\rangle$, which results in an asymmetric attack on the argument $[\Rightarrow \texttt{bird}(\texttt{Tweety})]$. We also assume there are no ASPIC+-axioms. As a result, we always set $\mathscr{K}_n = \mathscr{K}_a = \emptyset$.

The definition of $\leq'$ follows the familiar pattern of containing only the minimum of information required for it to be a partial order: $\leq' = \{(a, a') \in \mathscr{K}_p \times \mathscr{K}_p \mid a = a'\}$. The reason for this restriction is that non-minimal definitions of $\leq$ and $\leq'$ disable some

attacks (so they do not become defeats and do not appear in the argument graph of the ASPIC+ system). $\leq$ and $\leq'$ therefore solve the same problem as Dung's extension semantics – both decide which attacks hold and which attacks are to be rejected. By restricting the two relations we ensure that all attacks are evaluated within the argument graph that is generated from an ASPIC+ system, and not at the stage where attacks are turned into defeats inside ASPIC+ (cf Definition 11).

Argument graphs in ASPIC+ arise from a combination of a general ASPIC+ system $(\mathscr{L}, \bar{\cdot}, \mathscr{R}_s \cup \mathscr{R}_d, \leq)$ with concrete knowledge bases $(\mathscr{K}, \leq')$. Due to the the restrictions we make, we can drop $\leq$ and $\leq'$ from these definitions because they are directly determined by $\mathscr{K}$ and $\mathscr{R}_s$. In addition, we assume that $\preceq$ (the argument ordering required to decide which attacks are defeats, Definition 11) is defined in the same, minimal way, which allows us to elide it too (because in this setting, all ASPIC+-attacks are ASPIC+-defeats). Further, we will assume the logical language $\mathscr{L}$ to consist only of atoms, so all inferences will be made via rules in $\mathscr{R}$. $\mathscr{L}$ is therefore just a set of symbols, and we will not mention it explicitly in our definitions and examples. The same goes for the naming function $\langle\cdot\rangle$. As a result we can drastically simplify our presentation: Instead of specifying a four-tuple and a pair, we only need to give a single component, $\mathscr{R}$ (defeasible rules) explicitly. We will therefore adopt the notation $KB = \mathscr{R}$ to specify both the ASPIC+ system $(\mathscr{L}, \bar{\cdot}, \mathscr{R}, \leq)$ and the knowledge base $(\mathscr{K}, \leq')$ with $\mathscr{K} = \emptyset$, where KB stands for "knowledge base". Since KB is a set we can use $\subseteq$ to determine whether one knowledge base contains another one, and $\cup$ to describe the union of two knowledge bases. We will use the function $\mathsf{argGraph}(KB)$ to refer to the argument graph of a knowledge base KB.

2.3.2.2 Other Restrictions

In chapters 3 and 4 we will develop a model of decision making in which decisions made at an earlier stage in the process can impact decisions made at a later stage. Conversely, decisions at a later stage can override decisions made earlier. In order to ensure that our definitions have the desired properties for overriding knowledge, we will assume that all ASPIC+ knowledge bases meet the following two conditions:

Condition 1: No Unused Rules In a given knowledge base $KB = \mathscr{R}$, for every rule $r \in \mathscr{R}$, there exists at least one argument $a \in \mathsf{args}(KB)$ such that $\mathsf{topRule}(\mathsf{a}) = r$.

Condition 2: Injective Rule Labels The function $\langle\cdot\rangle$ is injective, so each rule is assigned a different name.

We are now going to define a number of common functions that operate on objects in ASPIC+. These functions will make it easier to refer to specific parts of an argument or a set of arguments. We start with the attackers function, which restricts a set of arguments A to those that attack a given argument a.

Definition 13 (Attackers)**.** *For all* ASPIC+ *arguments a, and* $A \subseteq \mathscr{A}$*:* $\mathsf{attackers}(a,A) = \{b \in A \mid b \text{ attacks } a\}$

We also use the function $\mathsf{attacks}(A)$ to return all pairs of arguments $(a,b) \in A \times A$ such that a attacks b.

In the context of argumentation systems, specifically for ASPIC+, a knowledge base KB infers a literal l, short $KB \vdash l$, if there exists an argument with conclusion l that is acceptable under a given semantics, for example grounded or preferred. The inference relation is therefore parameterised over the semantics, and we use an index pr to indicate preferred semantics and gr for grounded semantics. The following definition is based on Definition 14 in [41]:

Definition 14 (ASPIC+ Inference)**.** *Let KB be an* ASPIC+ *knowledge base and let* $l \in \mathscr{L}$*. Let* $s \in \{\mathsf{gr},\mathsf{pr}\}$*. KB s-*infers *l, short* $KB \vdash_s l$*, if and only if there exists an extension* $E \in \Sigma_s(\mathsf{argGraph}(KB))$ *such that* $a \in E$*.*

In addition to the conventions on ASPIC+ systems and knowledge bases just discussed, we will also use the following syntax for arguments, which is easy to read and conveys all the information contained in an argument.

To refer to arguments we use variables $a, b, \ldots$. When talking about the structure of an argument a we write $a = [l]$ if a is a literal argument (Definition 9 case 1) and we write $a = [b_1, \ldots, b_n; r; c]$ if a is the application of a rule r with sub-arguments b_1 to b_n and conclusion c. In both cases, the conclusion is immediately visible (either l or c) and in the second case both sub-arguments and the rule are included. The following example demonstrates our syntax conventions.

Example 5. *Let* $KB = \{r_1, r_2, \Rightarrow \mathtt{a}, \Rightarrow \mathtt{b}, \Rightarrow \mathtt{c}\}$ *be a knowledge base with* $r_2 = \mathtt{d}, \mathtt{b} \Rightarrow$

$\neg$c. *KB gives rise to the following arguments.* $\mathsf{args}(KB) = \{a_1, \ldots, a_5\}$ *with*

$$a_1 = [\Rightarrow \mathsf{a}] \qquad a_2 = [\Rightarrow \mathsf{b}]$$
$$a_3 = [\Rightarrow \mathsf{c}] \qquad a_4 = [a_1; r_1; \mathsf{d}]$$
$$a_5 = [a_4, a_2; r_2; \neg\mathsf{c}]$$

We get $\mathsf{attacks}(\mathsf{args}(KB)) = \{(a_5, a_3), (a_3, a_5)\}$. *In other words,* a_5 *rebuts* a_3 *and vice versa.*

2.3.3 Discussion

Engineering design processes are iterative, involving several refinements and adjustments before settling on a final design [1]. The iterations serve not only to increase the design's specificity, but also to revisit and change earlier decisions, in order to adapt to information (for example, when requirements have changed). It is important to keep a record of those changes, including the reasoning behind them, and not just of the latest state of the design, so that repeated mistakes can be avoided and knowledge can be re-used. Since we use a logical language to model engineering design processes, the logical language should support changes to earlier decisions (conclusions). This means that we should be able to invalidate knowledge additively, by adding new rules to our knowledge base, without deleting the invalidated formula.

This property will be a major concern in Chapter 3, and the restrictions we imposed on ASPIC+ systems in this chapter prepare the ground for our developments there, because they ensure that our model of decision processes has the desired property of being able to invalidate knowledge solely through the addition of new rules.

An immediate consequence of Condition 1 (no unused rules) on page 33 is the following:

Proposition 3. *If* $\mathsf{attackers}(a, \mathsf{args}(KB_1)) = \emptyset$ *and* $\mathsf{attackers}(a, \mathsf{args}(KB_2)) = \emptyset$ *then* $\mathsf{attackers}(a, \mathsf{args}(KB_1 \cup KB_2)) = \emptyset$.

Proof. Proof by contradiction. Let KB_1, KB_2 be two ASPIC+ knowledge bases such that $\mathsf{attackers}(a, \mathsf{args}(KB_1)) = \emptyset$ and $\mathsf{attackers}(a, \mathsf{args}(KB_2)) = \emptyset$. Assume there exists an argument $b \in \mathsf{attackers}(a, \mathsf{args}(KB_1 \cup KB_2))$ with $\mathsf{conc}(b) = l$. Then by Definition 9 and condition 1 on page 33, b is either an application of a rule with empty antecedent

argument ($b = [\Rightarrow l]$) or arises from the application of a rule r witn non-empty antecedent, $b = [\ldots;r;l]$. Assume $b = [\Rightarrow l]$. Then $\Rightarrow l \in KB_1 \cup KB_2$, so $\Rightarrow l \in KB_1$ or $l \in KB_2$. In the first case, $[\Rightarrow l] \in \mathsf{args}(KB_1)$, so $[\Rightarrow l] \in \mathsf{attackers}(a, \mathsf{args}(KB_1))$, and in the second case $[\Rightarrow l] \in \mathsf{args}(KB_2)$, so $[\Rightarrow l] \in \mathsf{attackers}(a, \mathsf{args}(KB_2))$. Both contradict the assumption.

Now assume $b = [\ldots;r;l]$. In that case, $r \in KB_1 \cup KB_2$, so $r \in KB_1$ or $r \in KB_2$. Assume without loss of generality $r \in KB_1$. Then, by Cond. 1 on page 33, there exists an argument $b' \in \mathsf{args}(KB_1)$ such that $\mathsf{topRule}(b') = r$. Therefore, $\mathsf{conc}(b) = \mathsf{conc}(b')$ so b' attacks a, and $b' \in \mathsf{attackers}(a, \mathsf{args}(KB_1))$, contradicting the assumption. □

This result will play a role in Chapter 3, as will the rest of the restrictions. With ASPIC+ we can define arguments and their attacks, and with argument graph semantics we can select acceptable sets of arguments.

2.4 Additional Definitions

Before we conclude this chapter with a brief discussion of alternatives to ASPIC+, we will define a number of mathematical concepts that are used throughout the thesis. The definitions are not argumentation-specific and will be given without much comment. For a list of all symbols and function names the reader may refer to Table A.1 on page 233.

Powerset

Definition 15. *Let S be a set. The* powerset *of S, short $\mathscr{P}(S)$, is defined as*

$$\mathscr{P}(S) = \{T \mid T \subseteq S\}$$

Partial Order

Definition 16. *A binary relation $R \subseteq S \times S$ over a set S is a* partial order *if and only if it is*

Reflexive *For all $s \in S$: sRs*

Antisymmetric *For all $s,t \in S$: If sRt and tRs then $s = t$*

Transitive *For all $s,t,u \in S$: If sRt and tRu then sRu.*

If there is a partial order R for a set S, then we may call S a partially ordered set, or a poset.

Equivalence Relation

Definition 17. *A binary relation $R \subseteq S \times S$ over a set S is an* equivalence relation *if and only if it is*

Reflexive *For all $s \in S$: sRs*

Symmetric *For all $s,t \in S$: If sRt then tRs*

Transitive *For all $s,t,u \in S$: If sRt and tRu then sRu.*

If S is a set and $\sim$ is an equivalence relation over S then $S/\sim$ is the set of equivalence classes of S. Formally, $S/\sim = \{[a] \mid a \in S\}$ where $[a] = \{b \in S \mid b \sim a\}$.

Lattice

Definition 18 (Lattice)**.** *A* lattice *is a partially ordered set L equipped with two binary operations $\sqcup, \sqcap : (L \times L) \to L$ satisfying the following equations:*

$$
\begin{aligned}
&a \sqcup b = b \sqcup a,\ a \sqcap b = b \sqcap a && \textit{(commutative)}\\
&a \sqcup (b \sqcup c) = (a \sqcup b) \sqcup c,\ a \sqcap (b \sqcap c) = (a \sqcap b) \sqcap c && \textit{(associative)}\\
&a \sqcup (a \sqcap b) = a,\ a \sqcap (a \sqcup b) = a && \textit{(absorption)}
\end{aligned}
$$

The powerset $\mathscr{P}(S)$ of a set S is a lattice with $\sqcup = \cup$ and $\sqcap = \cap$. We use the symbol $\bigsqcap C$ to mean $c_1 \sqcap c_2 \sqcap \ldots \sqcap c_n$ for a set $C = \{c_1, \ldots, c_n\}$.

A lattice is **join irreducible** (meet irreducible) if $x = a \sqcup b$ ($x = a \sqcap b$) implies $x = a$ or $x = b$

Symmetric Difference

The symmetric difference between two sets A and B consists of exactly those elements that are either in A or in B but not in both.

Definition 19 (Symmetric Difference)**.** *The* symmetric difference *of two sets A and B is*

$$A \Delta B = (A \setminus B) \cup (B \setminus A)$$

Kleene Closure

Definition 20 (Kleene Closure)**.** *The* Kleene closure *(or free monoid) of a set S, denoted by S^*, is the set of all words w over S.*

S^* includes the empty word ε. If w and $w' \in \mathscr{C}^*$ then $w \circ w' \in \mathscr{C}^*$ is their concatenation. The prefix relation of words is denoted by $\sqsubseteq$. $|w|$ stands for the length of word w with $|\varepsilon| = 0$. To distinguish an element of the set $c \in S$ from a word in S we write $[c]$ for the latter, where required.

Lexicographic Order

If we have a set S with a partial order $\leq$, then we can derive from $\leq$ a partial order on the set S^* as follows (again $\sqsubseteq$ denotes the prefix relation).

Definition 21 (Lexicographic Order)**.** *Let S be a set with a partial order $\leq$, and let $w, w' \in S^*$. $w \preceq w'$ if and only if*

1. *$w \sqsubseteq w'$ or*
2. *$w = vau$ and $w' = vbu'$ where v is the longest common prefix of w, w', $a \leq b$ and $u, u' \in S^*$.*

Linear Extension

In Chapter 4 we will compare different versions of a knowledge base over time, in order to describe the evolution of acceptable sets of arguments over time. To this end we introduce the concept of linear extensions, a relationship between sets of sets (in our case, sets of sets of acceptable arguments). If the preferred extensions of an argument graph G' are a linear extension of the preferred extensions of an argument graph G, then all arguments that were credulously acceptable in G are credulously acceptable in G', but there may be additional, credulously acceptable arguments in G' that did not exist in G. Further, every preferred extension in G is subsumed by a preferred extension in G', and every preferred extension in G' subsumes a preferred extension in G.

Formally, a linear extension of a set of sets P is a set P' in which each of the original sets in P is a subset of one of the sets in P', and every set in P' is a superset of a set in P.

Definition 22 (Linear Extension)**.** *Let P, P' be two sets of sets. P' is a **linear extension** of P iff*

1. *For all $p \in P$, there is a $p' \in P'$ such that $p \subseteq p'$.*
2. *For all $p' \in P'$, there is a $p \in P$ such that $p \subseteq p'$.*

If the preferred extensions of G' are a linear extension of G's preferred extensions, then (a) $G \sqsubseteq G'$ (subgraph) and (b) G' does not contain any sceptically acceptable attackers for arguments that were credulously acceptable in G.

Example 6. *Consider the set $P = \{p_1, p_2\}$. P has two elements, the sets $p_1 = \{a, b\}$ and $p_2 = \{d\}$. A linear extension P' is given by $P' = \{p_1, p_2'\}$ with $p_2' = \{d, e\}$. While Definition 22 itself works on any sets of sets, it makes most sense when P, P' are sets of arguments. If P is the set of preferred extensions of an argument graph G and P' of a graph G', then the extensions in P are still acceptable in G', but may contain additional arguments (such as e in the example). Conversely, for each preferred extension E of G' there exists a preferred extension E' of G such that $E' \subseteq E$.*

For any set S, if $P \subseteq \mathscr{P}(S)$ then $P' = \{\bigcup_{p \in P} p\}$ is a (trivial) linear extension of P.

Proposition 4. *For any two argument graphs G, G', if $\Sigma_{\mathsf{pr}}(G')$ is a linear extension of $\Sigma_{\mathsf{pr}}(G)$, then the grounded extension of G is a subset of the grounded extension of G'.*

Proof. Let G, G' be two argument graphs such that $\Sigma_{\mathsf{pr}}(G')$ is a linear extension of $\Sigma_{\mathsf{pr}}(G)$, and let E be the grounded extension of G. Let E' be the grounded extension of G', and let $a \in E$. We will show that $a \in E'$.

Let $F' \in \Sigma_{\mathsf{pr}}(G')$ a preferred extension of G'. By Definition 22 Cond. 2, there exists a preferred extension $F \in \Sigma_{\mathsf{pr}}(G)$ such that $F \subseteq F'$. Since a is in the grounded extension of G, a is also in every preferred extension of G, and specifically $a \in F$ so $a \in F'$. Since a is in all preferred extensions of G', a is also in the grounded extension of G'. □

Although the definition of linear extensions (Definition 22) does not make any reference to argument graphs, the concept is particularly useful in an argumentation setting. It can be used to express that an argument graph G' *contains* an argument graph G, by stating that the preferred extensions of G' are a linear extension of the

preferred extensions of G. This ensures not only that all of G's credulously acceptable arguments are credulously acceptable in G', but also that any credulously acceptable set of arguments in G is still credulously acceptable *as a set* in G'. In particular, this definition ensures that G' does not introduce any attacks between arguments that are credulously acceptable *together*, that is as part of the same extension, in G.

This is a much stronger statement than saying that an argument graph G is a subgraph of an argument graph G'. If we only knew that G is a subgraph of G', then G' could introduce arbitrary attacks between any two arguments that were conflict-free in G. If the preferred extensions of G' are a linear extension of the preferred extensions of G, then all arguments that share a preferred extension in G also share a preferred extension in G' (and are therefore conflict-free in G'). Linear extensions of preferred extensions thus preserve not only the acceptability status but also the context of arguments.

2.5 Discussion

In this chapter we gave formal definitions of abstract argument graphs (Section 2.2) and ASPIC+ (Section 2.3), the two systems we will use to construct and evaluate arguments. Before we begin the development of our decision making framework in Chapter 3, we will briefly discuss possible alternatives to the chosen systems.

2.5.1 Grounded and Preferred Semantics

One of the developments in the first part of this thesis is enforce, a function that elevates the status of a set of arguments in an argument graph from credulously to sceptically acceptable. enforce is an important building block for the model of decision processes developed in Chapter 4. We choose preferred and grounded semantics as representatives of credulous and sceptical acceptability, because they were defined in the original article by Dung [42], and because their definitions are relatively convenient to work with (maximum admissible sets and least fixed point of the characteristic function, respectively).

An open question for future work is whether these two semantics are good models of the semantics used in design debates, or whether other semantics such as robust [43], naive [44] or ranking-based [45] would be more realistic, and how the choice of

different semantics affects our results, particularly on enforcement (see Section 3.4.3 on page 94).

2.5.2 Alternatives to ASPIC+

ASPIC+ is a framework for structured argumentation. It provides definitions for creating and evaluating arguments based on a knowledge base, and a method for identifying attacks between those arguments. A number of other approaches to structured argumentation have been proposed (see e.g. [46] for an introduction), and we will briefly review them here.

2.5.2.1 ABA

Assumption-based argumentation (ABA [47]), like ASPIC+, is a general framework that instantiates abstract argument graphs. Every ABA-system has an underlying deductive system, a subset of which is designated as assumptions, and a set of inference rules. An ABA-argument is a set of assumptions from which a claim can be inferred by applying the inference rules. Attacks are determined by a contrariness-relation. In ABA, all inference rules are strict and defeasibility is expressed through assumptions.

2.5.2.2 DeLP

Defeasible Logic Programming (DeLP, [48]) is a logic-programming inspired system for argumentation. In DeLP there are two kinds of negation, strong negation and default negation. Unlike ABA and ASPIC+, DeLP uses dialectical tree semantics. A DeLP system can respond to queries of the status of a claim with one of four answers: Yes (if the claim is supported by an undefeated argument), no (if the complement of the claim is supported), undecided (neither yes or no), and unknown (if the query is not known to the program).

2.5.2.3 Deductive Argumentation

In deductive argumentation (e.g. [49, 50]) arguments are pairs of support and conclusion such that the conclusion is entailed (inference in the chosen base logic) by the claim. In deductive argumentation, the only inference rules are those of the base logic. Deductive argumentation has a great degree of flexibility, for example in the choice of base logic and how an argument graph is constructed from a knowledge base.

2.5.2.4 Discussion

Our primary reason for choosing ASPIC+ is that its syntax is appropriate for our needs, and – with the adjustments discussed on page 32 – its knowledge bases lead to a straightforward interpretation as argument graphs.

However, the model we develop in this thesis does not inherently require ASPIC+, and we presume that it could be translated to the other approaches without much difficulty. Such a translation of our model into other approaches to structured argumentation would help to draw out the distinction between characteristics that are intrinsic of decision processes, and those that are specific to the concrete expression of decision processes in ASPIC+. We leave this study for future work.

Chapter 3

Argument-Based Decision Making

3.1 Introduction

Individual decisions are the smallest building block of decision processes, so they are a good starting point for our model of decision processes. In decision analysis, the two fields multi-criteria decision making (MCDM) and decision making with uncertainty (DM) are different areas of research, each with their own methodologies, research questions, models, etc. One can say – with quite a bit of simplification – that MCDM is DM minus uncertainty plus multiple conflicting utility functions, and likewise DM is MCDM with uncertainty and with only a single criterion (i.e. utility).

In argumentation, Dung's argument graphs represent uncertainty non-numerically, namely as conflict between arguments. The same mechanism is used to represent multiple conflicting utilities (preferences) of decision outcomes. It should thus be possible for an argumentation-based model of decision making to combine DM's focus on uncertainty with MCDM's focus on multiple criteria. In this chapter we develop such a moel.

In our approach, the pros and cons of different options in a decision form an argument graph. In this graph, multi-criteria decision problems are analysed using the grounded extension of an argument graph, and decision problems with uncertainty are analysed using preferred extensions. In this model, credulous acceptability is an expression of "what-if" hypothetical reasoning, while sceptical acceptability amounts to evaluating one or more dialectical trees of arguments. Unlike in earlier proposals, decisions in our framework are modeled as sets of literals, rather than as single literals. This means that they can partially overlap, resulting in a more finely tuned set of deci-

sions. The need for this kind of analysis was identified during our collaboration with an aerospace manufacturer.

Our system also provides output that can be used to create decision documentation. It uses formal logic to reason with arguments and counterarguments. Because these arguments are generated from structured knowledge (in the form of rules), they can with little additional effort be transformed into an ontology-based format. This is because ontology standards are based upon a formal foundation of description logics [51].

We begin with a review and precise definition of problems in multi-criteria decision making and decision making under uncertainty, in Section 3.2. This section establishes our understanding of existing, non-argumentative methods that will act as a guideline to frame the necessary (but not sufficient) features our system should have. In the same section we look at additional requirements beyond the two approaches discussed. These requirements are direct consequences of the fact that the decisions we are interested in are parts of a larger process, and any artifacts (in the form of formal models) should be reusable.

In Section 3.3 we present our own argumentation-based decision model. After some initial definitions and examples, we evaluate it from the perspective of MCDM, DM, und the additional, reusability-related requirements established in the previous section. We pay specific consideration to decision rules, functions that establish a ranking of options in a decision. Decision rules serve two different purposes. First, they describe how the best option can be identified automatically, which is how they are used most commonly. Second, they can be used to characterise a decision after it has been made (by checking which of a number of decision rules would have lead to the outcome that was actually chosen). It is therefore important to develop a good understanding not just of a single decision rule, but of the differences between a number of decision rules.

In the next Section, 3.4, we investigate what it means to *accept* a decision - this is the process of adjusting one's knowledge base after choosing an option, in order to promote the arguments supporting this option from credulously to sceptically acceptable. This step is not usually part of decision models, but it is crucial for the remaining chapters of this thesis, in which we will look at sequences of decisions. Accepting a decision has implications for subsequent decisions, and those implications should be

reflected in the formal model.

We conclude the chapter with a review of previous work in the field of decision-making in argumentation (Section 3.5) and a discussion of our results (Section 3.6).

3.2 Two Models of Decision Making

Engineering design processes, such as those practised in the aerospace industry, are often complex and long-running. They involve a multitude of decisions, many of which affect subsequent steps in the process. Automated methods are commonly used to handle the complexity and interrelatedness of the decisions. The formal foundation of these methods has been studied widely in the literature (see [52] for a survey). However, while the quantitative analysis of decisions helps to manage the complexity of individual decisions, it does not address requirements that result from the long duration of the overall process. These requirements relate to decision documentation and include justifiability of decisions and traceability of the impact decisions have on one another. Information of this kind is usually recorded in a less rigorous manner, for example as text documents. The two tasks of analysing and documenting decisions are solved using two different methods. Therefore, a single human reasoning process (making a particular design decision) results in the production not only of two different artifacts, but of two entirely different models of that decision. We claim that an argumentation-based model can be the common foundation for both analysis and documentation of decisions.

3.2.1 Multi-Criteria Decision Making

A variety of formal definitions exists for multi criteria decision making. We will focus our analysis on the class of problems characterised by the definition below. The criteria Cr are represented by functions that map options onto numerical values, representing the quality of an option w.r.t a criterion. The values for each criterion are then aggregated to obtain an overall result (or a ranking) that determines the most favourable option.

Definition 23 (Multi criteria decision problem)**.** *A* multi criteria decision problem $P = (O, Cr, agg)$ *consists of*

1. *A set of options* $O = \{o_1, \ldots, o_n\}$ *with* $n \geq 1$

2. *A set of criteria* $Cr = \{c_1, \ldots, c_k\}$ *with* $k \geq 1$ *such that each* $c_i \in Cr$ *is a function* $c_i : O \to \mathbb{R}$

3. *An aggregation function* $agg : \mathbb{R}^{|O|*|Cr|} \to \mathbb{R}^{|O|}$

The set of all multi criteria decision problems is called MCD. We denote with V_P *the two-dimensional vector of the criteria for each decision:*

$$V_P = \begin{bmatrix} c_1(o_1) & \ldots & c_k(o_1) \\ \vdots & \ddots & \\ c_1(o_n) & & c_k(o_n) \end{bmatrix}$$

Definition 23 corresponds to the decision matrix, a popular decision method in engineering [4]. The actual rating of an option for a particular criterion is assigned by the decision maker who creates the table. In order to achieve consistency and accountability in the decision making process, additional documentation is required to justify decisions for a later verification. The numerical model alone does not explain *why* the criteria were assigned their values.

Example 7. *Table 3.1 on page 47 illustrates the problem of choosing a material for a wing component of an airplane. There are four possible options, aluminium, plastics, steel, and composite materials. The two criteria are weight and cost. The example demonstrates how the choice of the aggregation function agg influences the results: If one considers the sum of the criteria, aluminium is the first choice, but if one is instead interested in maximising the best criterion, then composites and steel are tied for first place, and aluminium is last. Aluminium has better results for both criteria than plastics, therefore plastics is dominated by aluminum.*

A *preferred option* is a decision that is as good as or better than all other options.

Definition 24 (Preferred Option)**.** *Given a multi criteria decision problem* $P = (O, Cr, agg)$*, with* $O = \{o_1, \ldots, o_n\}$ *an option* $o_i \in O$ *is* preferred *iff for all* $o_j \in O$

$$agg(V_P)_j \leq agg(V_P)_i$$

Example 8. *In the example given in Table 3.1, aluminium is preferred if we choose* Σ *as the aggregation. Otherwise, steel or composites would be preferred.*

Table 3.1: Multi Criteria Decision Making

	Criteria[a]		*Aggregations*	
	Weight	*Cost*	Σ[b]	max[c]
Aluminium	0.4	0.7	1.1	0
Plastics	0.3	0.6	0.9	0
Steel	0.2	0.8	1.0	1
Composites	0.7	0.2	0.9	1

[a] The higher the value, the better this criterion is met, e.g. low weight will result in a high value for *weight*.
[b] Σ is the sum of all criteria for one option.
[c] $\max(d)$ is the number of criteria in which d has the highest value.

3.2.2 Decision Making under Uncertainty

We will now develop a formal definition of problems in decision making under uncertainty, in the same way we did for multi-criteria decision making problems in the previous section. It aims to capture the essence of the axiomatic approach by Savage [10], although we do not repeat the axioms here and jump directly to the definition commonly found in the literature [53].

Definition 25 (Decision Making Problem under Uncertainty)**.** *A* decision making under uncertainty problem *is a three-tuple* (S,X,F) *where S is a set of states of the world, X is a set of consequences and* $F = \{f_1,\ldots,f_n\}$ *with* $n \geq 1$ *is the set of possible acts with* $f_i : S \to X$*. Further, there is a probability distribution p over S, and a utility measure* $u : X \to \mathbb{R}$*.*

Definition 25 makes some strong assumptions about what can be quantified: Both the probabilities of states of the world S, and the utilities of consequences X. As with MCDM, it is our aim to relax those assumptions by replacing them with weaker algebraic structures. As we will see in Section 3.3.4, we make no assumptions about the likelihood of possible worlds (and thus consider all possible worlds to be equally likely), and only assume a partial order of S, equipped with a lattice.

Example 9. *Table 3.2 shows the utilities for a decision similar to the one in Example 7, but with uncertainty about the future price developments of commodities. Aluminium*

Table 3.2: Decision Making Problem under Uncertainty (Example 9)

Option	*Stagnation*	*Increase*	*Decrease*
Aluminium	3	1	4
Plastics	3	3	1
Steel	3	1	3
Composites	2	2	2

and steel are much more affected by price movements, so their utilities vary more in the three scenarios. Composites are completely unaffected, and plastics are somewhere in the middle.

This example is related to the previous example: Depending on market prices, the criterion Cost may take different values for the same material. However, this relationship does not appear in the formal definitions of multi-criteria decision making and decision making under uncertainty problems. In our argumentation-based approach we will be able to express relationships between criteria and uncertainty about the state of the world.

Table 3.2 shows the decision making under uncertainty problem (S,X,F) *based on the following values:*

$$S = \{Stagnation, Increase, Decrease\}$$
$$X = \{1,2,3,4\}$$
$$F = \{f_{al}, f_{pl}, f_{st}, f_{co}\} \text{ with}$$
$$f_{al} = Stagnation \mapsto 3, Increase \mapsto 1, Decrease \mapsto 4$$
$$f_{pl} = Stagnation \mapsto 3, Increase \mapsto 3, Decrease \mapsto 1$$
$$f_{st} = Stagnation \mapsto 3, Increase \mapsto 1, Decrease \mapsto 3$$
$$f_{co} = Stagnation \mapsto 2, Increase \mapsto 2, Decrease \mapsto 2$$

In this example, $X \subseteq \mathbb{R}$*, so the utility measure u is simply the identity function* $u(x) = x$*. Finally, assuming all three states in S are equally likely, the probability distribution p is given by* $p(Stagnation) = p(Increase) = p(Decrease) = \frac{1}{3}$.

In DMU, options are compared using *decision rules*. Decision rules rank options by evaluating their consequences in different possible worlds. A common example is

the *maxmin* decision rule, which maximises the outcome in the worst case. Decision rules are functions $d : F \to B$ from the set of actions F to a set B with a total order $\leq$, so that the ordering of actions $f \in F$ is determined by the ordering of their results $d(f) \in B$. To illustrate the concept we now give a definition of *maxmin*, followed by an example.

Definition 26 (Maxmin for DMU)**.** *Let $P = (S, X, F)$ be a decision making problem with uncertainty. $maxmin_P : F \to \mathbb{R}$ is defined as*

$$maxmin(f) = \min_{s \in S} f(s)$$

Note that the "max" part of *maxmin* is determined by the ordering of $\mathbb{R}$, so it is not present in the definition of *maxmin* itself.

Example 10. *For the decision making under uncertainty problem in Table 3.2, the maxmin rule recommends composites, because with composites we are guaranteed a utility of 2 in the worst case. For the other options, the worst case has a utility of 1.*

The example also shows why maxmin is a rather pessimistic rule: By choosing composites we forego the possibility of much higher utilities than 2. For example, we could get a utility of 4 by choosing aluminium.

Another common decision rule in DMU is to compute the expected utility for each option, by multiplying the probability of each possible world with its payoff. This requires both probabilities and payoffs to be quantifiable. In this thesis we do not make either of those assumptions, so we will not calculate expected utilities.

There is another use for decision rules, besides identifying the best option. They can be used to characterise decisions after they have been made. For example, if we have an optimistic decision rule and a pessimistic decision rule, we can see which of the two would have selected the option that was actually chosen. This application of decision rules benefits from having more rules, and it is a motivation for our analysis of decision rules in Section 3.3.5 of this chapter.

3.2.3 Documentation of Decisions

As outlined above, the primary reasons for documenting design decisions are consistency and accountability. The design decision process needs to be consistent throughout

organisations and, with regards to regulations, the entire industry. Consistency means that given a specific problem, any decision maker would ideally come to the same conclusion. Records need to be kept in order to verify that the decision making process is consistent.

Design decisions in the aerospace industry are often complex. They are also part of an iterative design process, which means that decisions may need to be revised to account for previously unconsidered factors or for changed requirements.

In our collaboration with one major aerospace manufacturer, we found that discussions about design questions were primarily carried out in emails and personal meetings. Only when a decision was made, was it documented in a central repository used to manage the design process. To create this documentation, some of the information contained in the emails had to be duplicated, while the rest remained only on the email server and was thus not readily available to a search of the structured repository. Having to duplicate information carries the risk of introducing errors, sometimes simply by using a slightly different wording.

This process also entails that alternative options which had been discussed would only be documented informally. Later on in the process, a changing requirement might lead to a re-evaluation of previously made decisions. In this case, the alternatives have to be retrieved from the unstructured documentation. This is a labour intensive process. It is also error prone, especially after several iterations of the design. The retrieval of decision rationale could therefore benefit greatly from a formal, structured documentation.

Contract and claims management are another use case for decision documentation. At the beginning of each phase in the project life cycle, it needs to be shown that all of the requirements of the previous stage have been fulfilled. Design documentation is used to show how each requirement is addressed. Here, the same issues that were described in the previous paragraph arise.

3.2.4 Problems with Current Approach

The two problems described at the beginning of this chapter are the modeling and the documentation of decisions. Having presented common solutions to each of those problems, we are now going to highlight their shortcomings.

1, Opaque resoning process For each of the potential decisions, a set of criteria has to be evaluated. Assigning values to decisions accounts for a large part of the effort involved in making decisions with an MCDM or DMU approach. It usually represents the outcome of some reasoning process, which itself does not appear in the final model and needs to be documented separately.

2, Local optimum An MCDM/DMU model can only identify the best of a predefined set of options. It is possible that there exists a better decision that was not part of the model. Therefore, MCDM models have the inherent risk of producing only a local optimum.

3, Proprietary documentation formats There is no standard method of documenting decisions. This prevents the development of standard tools to support decision documentation.

4, Manual analysis If decisions are documented informally, there is no underlying model on which an automated analysis of decisions could be performed. With an automated approach, one could, for example, immediately spot conflicting assumptions made by two different engineers, or characterise past decisions by matching them to decision rules.

5, Costly retrieval of documentation Most of the documentation relating to the process of decision making is documented in an unstructured way. The effort required to find a particular piece of information in an unstructured repository is much higher than that of finding it in a structured repository. If, at the end of a project, there are claims that some requirements have not been met, the entire documentation related to that subcomponent has to be read by an engineer in order to verify that either the requirement has actually been met or that the requirement was defined differently in the contract.

Both of the problems associated with the formal models of DMU and MCDM (1-2) relate to the fact that creating the model is the actual challenge in those approaches. Once a model has been created, the actual evaluation consists of mechanically applying a set of predefined mathematical operations. The informal documentation of decisions, on the other hand, is limited in terms of automated processing of information (3-5).

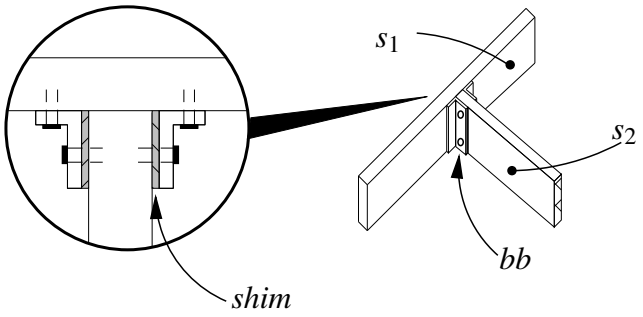


Figure 3.1: Joining Two Structures

3.3 Argument-Based Decision Framework (ADF)

3.3.1 Overview

We present a framework for decision making using argumentation. More precisely, given a set of options and a set of goals, we are going to use argumentation to construct arguments in favour of and against each option, with the aim of identifying the best option. The best option is the one that satisfies the most goals. However, this should not be the only criterion for selecting an option: Any formal method of decision making depends on a model, a formalised representation of the problem. The quality of the model affects the quality of the decision. There are two factors which determine the quality of the model: The quality of the information that the model was built on, and the quality (correctness) of the model itself. Because of these two inherent risk factors, we argue that any formal method of decision making should not only identify the best option, but it should also present a justification for the outcome, so that the influence on the decision making process of potential flaws in the model can be traced.

Formal methods of argumentation are well suited for this purpose. They produce not only a claim, but also a description of how the available information was used to arrive at that claim, and how any counterarguments were addressed. Our decision making framework uses argumentation to reason about the possible outcomes of decisions.

Example 11. *Imagine we are faced with the problem of designing a structure (t) that has two components, s_1 and s_2 as illustrated in Figure 3.1. s_1 and s_2 are joined with a bracket fixed by bolts, bb. Our task is to decide which bolts to use, whether or not to use a shim, a thin sheet of metal, between the bracket and the components, and how many*

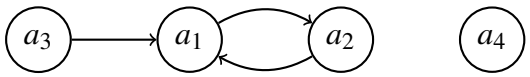


Figure 3.2: Argument graph for Example 11.

layers of varnish to apply. Our requirements are to maintain the structural integrity of t and to achieve a high resistance to corrosion. The first option we consider, O_1, is to use steel/titanium bolts, not to use a shim, and to apply two layers of varnish to the product. The following arguments might be put forward:

a_1 *Not using a shim means that the structure remains balanced. Therefore, it will not be damaged.*

a_2 *Steel/titanium bolts cause microscopic fractures in s_1 and s_2, resulting in damage to the structure.*

a_3 *Steel/titanium bolts are too strong for the material that s_1 is made of, so there will be microscopic fractures.*

a_4 *The structure has a high corrosion resistance, because two layers of varnish are used.*

A possible argument graph for this example is shown in Figure 3.2, with the grounded extension $\{a_2, a_3, a_4\}$. a_3 is a sub-argument of a_2 (giving the reason for microscopic fractures), so a_3 and a_2 both attack a_1, but a_1 only attacks a_2 because their conclusions directly contradict. a_4 is not in conflict with any arguments.

3.3.2 Decision Frames

A core concept of our argumentation-based decision system is the decision frame, defined below. It captures the context of a decision, that is, the background knowledge, requirements, and any other possible consequences. Background knowledge is simply a set of ASPIC+ rules, and conseqences and requirements are represented as literals of the language $\mathscr{L}$. To show that a decision has good consequences, we will construct arguments that have a goal as their claim. We then check which of those arguments are part of a grounded extension.

Table 3.3: Rules for Example 12

Rule name	Content
r_1	$\texttt{mat}(\texttt{bb},\texttt{st}) \Rightarrow \texttt{fractured}(\texttt{s}_1)$
r_2	$\texttt{fractured}(\texttt{s}_1) \Rightarrow \texttt{damaged}(\texttt{t})$
r_3	$\neg\texttt{shim}(\texttt{t}) \Rightarrow \texttt{balanced}(\texttt{t})$
r_4	$\neg\texttt{shim}(\texttt{t}) \Rightarrow \neg\texttt{fractured}(\texttt{s}_1)$
r_5	$\neg\texttt{fractured}(\texttt{s}_1), \texttt{balanced}(\texttt{t}) \Rightarrow \neg\texttt{damaged}(\texttt{t})$
r_6	$\texttt{varnish}(\texttt{t},2) \Rightarrow \texttt{coRes}(\texttt{t},\texttt{high})$
r_7	$\texttt{mat}(\texttt{al}) \Rightarrow \neg\texttt{fractured}(\texttt{s}_1)$
r_8	$\texttt{mat}(\texttt{al}), \texttt{shim}(\texttt{t}) \Rightarrow \texttt{balanced}(\texttt{t})$

Example 12. *The background knowledge from Example 11 can be formalised as shown in Table 3.3.* r_1 *shows that using steel/titanium bolts (*$\texttt{mat}(\texttt{bb},\texttt{st})$*) will cause microscopic fractures in* s_1*.* r_2 *says that if* s_1 *is fractured, then the whole structure t is damaged. Not using a shim results in a good overall balance of t, as well as in the absence of fractures (*r_3, r_4*). If the structure is balanced and free of fractures, then it is not damaged (*r_5*). Applying two layers of varnish results in high resistance to corrosion (*r_6*). The final two rules,* r_7 *and* r_8 *state that using aluminium will not result in a fractured component, and if the aluminium component is supported with a shim then it will be balanced.*

We model options as ASPIC+ knowledge bases and use the symbol $\mathscr{O}$ for the set of all options. Each of the options $O \in \mathscr{O}$, together with the knowledge base of the decision frame, forms an argumentation system. This argumentation system is used to derive arguments about the goals achieved by choosing this option.

Decision frames also include a set of consequences, C. It contains all consequences that choosing an option could potentially have, both good and bad. We require C to be a lattice[1] over a partial order $\leq_C$, in order to be able to compare decisions by their possible consequences.

Definition 27 (Decision frame)**.** *A* decision frame *is a tuple* (K, C, R) *where*

1. K *is an* ASPIC+ *knowledge base,*

2. $C \subseteq \mathscr{L}$ *is a finite set of* ***consequences*** *equipped with a lattice* $(C, \sqcap, \sqcup)$ *over a partial order* $\leq_C$ *and*

[1] See Definition 18 on page 37

3. $R \subseteq C$ *is a set of requirements.*

The set of options itself is not part of a decision frame. This is because we view decision frames as functions from the set of all possible options $\mathscr{O}$ to the powerset of all arguments, $\mathscr{P}(\mathscr{A})$. By adopting this definition we address the problem of "local optimum" (page 51), which is caused by the need to supply a predetermined set of options in the traditional models of decision making.

The set of all decision frames is $\mathscr{D}$. We will make use of two utility functions, args_D and $\mathsf{consequences}_D$ defined as follows.

Definition 28 (Utility Functions)**.** *Let $D = (K, C, R)$ be a decision frame. The functions $\mathsf{args}_D : \mathscr{O} \to \mathscr{P}(\mathscr{A})$ and $\mathsf{consequences}_D : \mathscr{P}(\mathscr{A}) \to C$ are defined as follows:*

$$\mathsf{args}_D(O) = \mathsf{args}(K \cup O)$$
$$\mathsf{consequences}_D(A) = \{c \in C \mid \exists a \in A.\mathsf{conclusion}(a) = c\}$$

In the remainder of this chapter we will use the term *decision frame* to refer specifically to Definition 27, and the term *argument-based decision framework*, short ADF, to refer to our approach in general.

Example 13. *Continuing with Example 12 on page 53, we can define a decision frame $D_M = (K, C, R)$, with the components defined below, as well as two options, $O_1, O_2 \in \mathscr{O}$, given by Table 3.4. The variable t stands for the structure (a t-connector) that is being designed. For now we set $C = R$, so we do not pay any special attention to the set of consequences beyond what is needed to capture the requirements. Consequences will play a role in Section 3.3.4 on decision making with uncertainty.*

For option O_1, we get the argumentation system $(A_1, \mathsf{attacks}(A_1))$ with $A_1 = \{a_1, \ldots, a_9\}$. The arguments look as follows:

$$a_1 = [\Rightarrow \neg\mathtt{shim}(\mathtt{t})] \qquad a_6 = [a_5; r_1; \mathtt{fractured}(\mathtt{t})]$$
$$a_2 = [a_1; r_3; \mathtt{balanced}(\mathtt{t})] \qquad a_7 = [a_6; r_2; \mathtt{damaged}(\mathtt{t})]$$
$$a_3 = [a_1; r_4; \neg\mathtt{fractured}(\mathtt{t})] \qquad a_8 = [\Rightarrow \mathtt{varnish}(\mathtt{t}, 2)]$$
$$a_4 = [a_2, a_2; r_5; \neg\mathtt{damaged}(\mathtt{t})] \qquad a_9 = [a_8; r_6; \mathtt{coRes}(\mathtt{t}, \mathtt{high})]$$
$$a_5 = [\Rightarrow \mathtt{mat}(\mathtt{bb}, \mathtt{st})]$$

Table 3.4: Decision Frame for Engineering Example

Name	Definition
O_1	$\{\Rightarrow \texttt{mat(bb,st)}, \Rightarrow \neg\texttt{shim(t)}, \Rightarrow \texttt{varnish(T,2)}\}$
O_2	$\{\Rightarrow \texttt{mat(bb,al)}, \Rightarrow \texttt{shim(t)}, \Rightarrow \texttt{varnish(T,2)}\}$
K	$\{r_1, \ldots, r_6\}$, as in Table 3.3
C	$\{\neg\texttt{damaged(t)}, \texttt{coRes(T,high)}\}$
R	$= C$

There are two pairs of mutually rebutting arguments: a_3 and a_6, and a_4 and a_7. a_3 is also an undercutter of a_7, and a_6 is an undercutter of a_4. The grounded extension consists of $\{a_1, a_2, a_8, a_9\}$ and the argumentation graph for option O_1 is shown in Figure 3.3 on page 57 (conflicting arguments only).

To see why the set of consequences C is required to be a lattice, consider the case where we argue about the outcome of a decision on two distinct, qualitative scales - for example, form and function. Each of these can be good, fair or bad. This results in six values $V = \{good\_form, good\_function, fair\_form, fair\_function, bad\_form, bad\_function\}$ with a partial order $\leq_V$. Each of the scales on its own is totally ordered, but $\leq_V$ does not relate elements from different scales. As a result, neither $(fair\_function, good\_form) \in \leq_V$ nor $(good\_form, fair\_function) \in \leq_V$.

The lattice structure comes into play when we summarise the set of arguments for an option. For example, assume that an option gives rise to three sceptically acceptable arguments a_1, a_2, a_3 with conclusions $good\_form$, $fair\_form$ and $fair\_function$ respectively. In order to apply the *maxmin* rule we need to identify the maximum (best) among the conclusions of a_1 to a_3. Clearly, $good\_form \geq fair\_form$, because both are on the same scale, but the better one of $fair\_function$ and $good\_form$ can only be chosen with additional information. This is a common problem in qualitative decision-making (cf. Dubois [54]). The purpose of the lattice structure with its $\bigsqcap, \bigsqcup$ operators is to make any two elements of C comparable in some sense (by choosing an element of C that is smaller than or greater than both), while still retaining the requirement for $\leq$ to be a partial order only, and not a full order.

A possible solution for our example is to extend the set V with a common scale, say $C = V \cup \{overall\_good, overall\_fair, overall\_bad\}$. We can then – within the boundaries required for C to be a lattice – define $\sqcap$ according to our preferences e.g. by

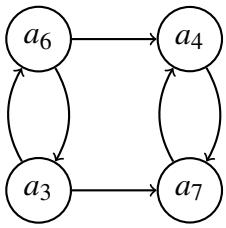


Figure 3.3: Argument graph for Example 13.

stipulating $fair\_function \sqcap good\_form = overall\_fair$.

In engineering design we frequently have to handle different ordinal scales when discussing qualitative criteria such as fire safety ratings or usability.

Having established the general concept of ADF, we will evaluate our approach in relation to MCDM and DMU in the following two sections.

3.3.3 Multi-Criteria Decision Making

We will now translate some concepts that are specific to multi-criteria decision making (Section 3.2.1) to our model. A common notion in multi criteria decision making is that of *dominated options*. Informally, an option O is dominated if there is another option O' that satisfies the same criteria at least as well as O. It is not reasonable to choose a dominated option, because there is a better option that can be chosen without any disadvantages. In order to identify the goals that an option satisfies, we define a *satisfaction function* sat. It returns the goals that can be shown by arguments generated with this option.

Definition 29 (Satisfaction Function)**.** *Let $D = (K, C, R)$ be a decision frame.* $\mathsf{sat}_D : \mathscr{O} \to \mathscr{P}(R)$ *is defined as* $\mathsf{sat}_D(O) = \{r \in R \mid \exists a \in \mathsf{args}_D(O)$ *such that a is in the grounded extension of the graph* $(\mathsf{args}_D(O), \mathsf{attacks}(\mathsf{args}_D(O)))$ *and* $\mathsf{conclusion}(a) = r\}$.

Example 14. *The grounded extension for option O_1 is $\{a_1, a_2, a_8, a_9\}$. Since only the conclusion of a_9 is a goal, we get* $\mathsf{sat}_D(O_1) = \{\mathtt{coRes(t,high)}\}$.

It is important to note that we restrict the satisfaction function to arguments in the grounded extension. If, for example, admissible or stable semantics had been used instead, we would face the problem of having to choose one out of a set of extensions. Each of the extensions could contain arguments in favour of the option, but they cannot

all be accepted at the same time, and thus should not be returned by sat. Further information would be required to resolve this conflict. In Section 3.3.4, we will use preferred extensions to represent uncertainty about the consequences of an option.

Definition 30 (Dominated Option)**.** *Let $D = (K,C,R)$ be a decision frame. An option $O \in \mathscr{O}$ is* dominated *by an option $O' \in \mathscr{O}$ with $O \neq O'$ such that $sat_D(O) \subseteq sat_D(O')$. O is* strictly dominated *if $sat_D(O) \subset sat_D(O')$.*

Example 15. *In Example 13, O_1 is strictly dominated by O_2, because $\mathsf{sat}_{D_M}(O_2) = \{\texttt{coRes(t,high)}, \neg\texttt{damaged(t)}\}$, so $\mathsf{sat}_{D_M}(O_1) \subset \mathsf{sat}_{D_M}(O_2)$.*

3.3.3.1 Comparison with Current Approach

To conclude the comparison with MCDM, we show how our system formally corresponds with the class of multi criteria decision making problems characterised by Definition 23 on page 45. Our proof of the correspondence consists of two results: That every decision frame can be expressed as an MCDM system (assuming a fixed set of options), and conversely that every MCDM system can be expressed as a decision frame. The first result can be found in Amgoud and Prade, 2008 [33] and we repeat it here for completeness. In order to achieve the second result, we will construct a function that maps multi criteria decision problems to decision frames. We are going to model MCDM options as options in a decision frame, each containing a single element. The values of the criteria functions are also going to be represented as literals. We then express the results of the aggregation function *agg* as goals in the decision frame, and finally we will create a rule for each decision that leads to the desired goals. We will use the function $\langle\cdot\rangle$ whenever a value (for example a number) is meant to be read as a symbol in $\mathscr{L}$, rather than as the value itself, in the same sense that $\langle\cdot\rangle$ is used to refer to the names of ASPIC+ rules in the object language.

Proposition 5 (Equivalent to property 7 in [28])**.** *For every argumentation decision problem $D = (K,C,R)$ and set of options $\{O_1,\ldots,O_n\} \subseteq \mathscr{O}$, there exists a multi criteria decision problem $P' = (O,Cr,agg)$ such that for all O_i, O_j with $i,j \leq n$:*

$$\mathsf{sat}_D(O_j) \subseteq \mathsf{sat}_D(O_i) \Leftrightarrow agg(V_{P'})_j \leq agg(V_{P'})_i$$

This shows that we can construct a mapping from ADF to MCD which preserves the preference relation over decisions. Next, we will show that a similar mapping can be constructed in the other direction, from MCD to ADF, again preserving the preference relation.

Definition 31 (Mapping from MCD to ADF)**.** *Let $P = (O, Cr, agg)$ be a multicriteria decision problem. We construct a decision frame $P' = (K, C, R)$ and a set of options $O_P \subseteq \mathscr{O}$ as follows:*

1. $O_P = \{\{\Rightarrow \langle d_i \rangle\} \mid d_i \in O\}$
2. $K = R_1 \cup R_2 \cup R_3$ *and*
   (a) $R_1 = \{\langle d_i \rangle \Rightarrow \langle c_j(v_{i,j}) \rangle \mid v_{i,j} \in V_P \text{ and } c_j \in Cr\}$
   (b) $R_2 = \{\langle v_{i,1} \rangle, ..., \langle v_{i,k-1} \rangle, \langle v_{i,k} \rangle \Rightarrow \langle agg(V_P)_i \rangle \mid \{\langle d_i \rangle\} \in O_P \text{ and } k = |Cr|\}$
   (c) $R_3 = \{\langle agg(V_P)_i \rangle \Rightarrow \langle agg(V_P)_j \rangle \mid agg(V_P)_i \geq agg(V_P)_j\}$
3. $R = \{\langle agg(V_P)_i \rangle \mid d_i \in O\}$
4. $C = R$

The mapping in Definition 31 has four components - a set of options, O_P and three constituent parts of the decision frame (K, C, R). O_P contains a one-element knowledge base $\{\Rightarrow \langle d_i \rangle\}$ for each original option in O. The literal $\langle d_i \rangle$ is to be read as "d_i is selected". K, the knowledge base of the decision frame, contains three types of rules, R_1, R_2 and R_3. Rules in R_1 map decisions (d_i) to their score on each criterion in Cr. R_1 therefore contains $|O| * |Cr|$ rules. Rules in R_2 connect the individual criteria of options in O_P to their aggregated score in the multi-criteria decision making problem P. R_2 contains $|O|$ rules. The set R_3 contains perhaps the most interesting rules. To understand the construction, consider the definition of C first: $C = R = \{\langle agg(V_P)_i \rangle \mid d_i \in D\}$, so there is one element in C for each possible aggregated score that an option might have. To ensure that options which rank higher under *agg* also rank higher under sat, we need to translate "high numerical scores" (*agg*) to "large sets of acceptable arguments" (sat). This is the purpose of the rules in R_3. The effect of the rules is that every time we have an acceptable argument for a high aggregated score in P, we

also have arguments for all lower scores. As a result, the set of arguments for a high-ranking option includes more arguments with conclusions in C, and hence the option ranks higher under $\subseteq$ and sat. The number of rules in R_3 is equal to the number of different aggregated scores (and so smaller than or equal to the number of options).

We will now give a lemma (Propage 6) before our main result, Theorem 1, in which we show that every MCDM problem can be represented as a decision frame with the same results, i.e. with the same ranking of decisions. This demonstrates that ADF is at least as expressive as MCDM – and it adds benefits such as reusability, accountability, inference of decisions and the ability to compare multiple possible worlds, as will be explored below.

Proposition 6. *Let $P = (O, Cr, agg)$ be a multi criteria decision problem and $P' = (K, C, R)$ and O_P as constructed according to Definition 31. For every option $O = \{\Rightarrow \langle d_i \rangle\} \in O_P$ and every criterion $c \in Cr$, there exists an argument $[[\Rightarrow \langle d_i \rangle], \langle d_i \rangle \Rightarrow \langle c(d_i) \rangle; \langle c(d_i) \rangle]$ in the grounded extension of the argument graph $(\mathsf{args}_P(O), \mathsf{attacks}(\mathsf{args}_P(O)))$.*

Proof. Let $O = \{\Rightarrow \langle d_i \rangle\} \in O_P$ for a $d_i \in O$ and let $c_j \in Cr$ be a criterion. By Definition 31 Cond. 2a, there is a rule $r = \langle d_i \rangle \Rightarrow \langle c_j(v_{i,j}) \rangle$ in K, so $\mathsf{args}_{P'}(O)$ contains an argument $a = [[\Rightarrow \langle d_i \rangle], r, \langle c_j(v_{i,j}) \rangle]$. Since the rules in K do not have any conflicting conclusions, and the literals in O are conflict-free too, the set of arguments $\mathsf{args}_{P'}(O)$ is conflict free. Therefore the grounded extension of the argument graph of $\mathsf{args}_{P'}(O)$ contains all arguments, including a. □

If a decision frame is generated from a multi criteria decision making system, then its knowledge base is very simplistic, because it does not contain any domain knowledge. This results from the fact that the domain knowledge which was applied to assign the criteria values for each decision is not represented in the model, and therefore cannot be included in the decision frame.

Theorem 1. *For every multi criteria decision problem $P = (O, Cr, agg)$, there exists a decision frame $P' = (K, C, R)$ and a set of options $O_P \subseteq \mathscr{O}$ such that for all $O_i, O_j \in O_P$:*

$$agg(V_P)_j \leq agg(V_P)_i \Leftrightarrow \mathsf{sat}_{P'}(O_j) \subseteq \mathsf{sat}_{P'}(O_i)$$

The proof relies on the fact that in the mapping constructed according to Definition 31, if an option $\{\Rightarrow \langle d_k \rangle\} \in O_P$ subsumes all options $\{\Rightarrow \langle d_i \rangle\} \in O_P$ that are worse than d_k, that is if $agg(V_P)_i \leq agg(V_P)_k$, then the corresponding option d_k satisfies all goals that are satisfied by d_k.

Proof. Let $P = (O, Cr, agg)$ be a multi criteria decision problem and let $P' = (K, C, R)$ and O_P the decision frame and set of options as constructed according to Definition 31.

($\Rightarrow$) Let $d_i, d_j \in O$ such that $agg(V_P)_j \leq agg(V_P)_i$. We are going to show that every element $v \in \mathsf{sat}_{P'}(D_j)$ is also in $\mathsf{sat}_{P'}(D_i)$. Let $v \in \mathsf{sat}_{P'}(D_j)$. Then $\langle v \rangle \in R$. By Cond. 3 of Definition 31, $v \in agg(V_P)$, and $v \leq agg(V_P)_j$ (Cond. 2c of Definition 31). Let $v' = agg(V_P)_i$. By Cond. 3 of Definition 31, $v' \in \mathscr{G}$, and by Cond. 2a and 2b of Definition 31 there is an argument $[[\Rightarrow \langle d_i \rangle], \langle d_i \rangle \Rightarrow \langle v' \rangle; \langle v' \rangle]$ in the grounded extension of $(\mathsf{args}_{P'}(\{\Rightarrow \langle d_i \rangle\}), \mathsf{attacks}(\mathsf{args}_{P'}(\{\Rightarrow \langle d_i \rangle\})))$ By Cond. 2c of Definition 31, $\langle v' \rangle \Rightarrow \langle v \rangle \in K$, so there is an argument $[[\Rightarrow \langle d_i \rangle]; \langle d_i \rangle \Rightarrow \langle v \rangle; \langle v \rangle]$ in the grounded extension of $(\mathsf{args}_{P'}(\{\Rightarrow \langle d_i \rangle\}), \mathsf{attacks}(\mathsf{args}_{P'}(\{\Rightarrow \langle d_i \rangle\})))$. Therefore, $\langle v \rangle \in sat_{P'}(D_i)$.

($\Leftarrow$) Let O_j, $O_i \in O_P$ such that $\mathsf{sat}_{P'}(O_j) \subseteq \mathsf{sat}_{P'}(O_i)$. By Definition 29 Cond. 3, for all requirements $r \in R$, $r = \langle agg(V_P)_i \rangle$ for some $d_i \in D$ – that is, every r corresponds to the aggregated score of some option d_i. Let $r_j = \langle agg(V_P)_k \rangle$ be the highest-ranking requirement in $\mathsf{sat}_{P'}(O_j)$, and $r_i = \langle agg(V_P)_l \rangle$ be the highest-ranking requirement in $\mathsf{sat}_{P'}(O_i)$. By Definition 29 Cond. 2b and 2c, we know that in fact $k = j$ and $l = i$, because the highest ranking for which an argument exists in the argument graph for an option $\{\Rightarrow \langle d_m \rangle\} \in O_P$ is exactly $agg(V_P)_m$. So $r_j = \langle agg(V_P)_j \rangle$ and $r_i = \langle agg(V_P)_i \rangle$ and it remains to show that $agg(V_P)_j \leq agg(V_P)_i$. We do this by contradiction: Assume $agg(V_P)_j > agg(V_P)_i$. Then, by Definition 29 Cond. 2c, there is *no* argument $a \in \mathsf{args}_{P'}(O_i)$ with $\mathsf{conclusion}(a) = \langle agg(V_P)_j \rangle$, so by Definition 29 $\mathsf{sat}_{P'}(O_j) \not\subseteq \mathsf{sat}_{P'}(O_i)$. This contradicts the assumption that $\mathsf{sat}_{P'}(O_j) \subseteq \mathsf{sat}_{P'}(O_i)$. □

In the definition of sat (Definition 29 on page 57) we limited ourselves to arguments that are acceptable under grounded semantics. The reason for this restriction is that we have reserved preferred semantics for decision making with uncertainty (as opposed to multi-criteria decision making). This allows us to treat both kinds of decision making in the framework of arguments about the consequences of options.

3.3.4 Decision Making With Uncertainty

Decision making with uncertainty is characterised by having only uncertain knowledge about the state of the world, and considering multiple possible worlds. Options can have different outcomes and utilities depending on the possible world. Decision making with uncertainty as a branch of decision theory has been pioneered by Savage [10], but unlike multi-criteria decision making it has not been investigated widely in the context of formal argumentation.

Decision Rules The term "decision rule" denotes the method used to derive a ranking of options from a ranking of outcomes and the knowledge of how options relate to outcomes.

For example, the literature on decision making with uncertainty in general (with or without argumentation) distinguishes *optimistic* from *pessimistic* decision rules. Another characteristic of decision rules is their decisiveness, which has played a role in analyses of the "drowning effect", i.e. of some options being ranked the same even though their different outcomes suggest that one should be preferred over the other.

Decision rules are important for evaluating formal decision models, because they are the means by which decision models are interpreted. In our application domain of engineering design, decision rules serve two purposes. First, they can be used to recommend an option based on a set of arguments for and against each of a number of options. In this regard they are similar to the notion of dominated options and the sat function we introduced for multi-criteria decision making. Second, they can be used to analyse decisions after the fact. When a decision has been made, we can characterise that decision as optimistic, pessimistic, etc. by checking which of the decision rules the chosen option corresponds to. The second use case is perhaps more relevant to our application because – as set out in the introduction – our primary aim is to document and analyse decision processes, rather than to automate the actual decision making.

In our approach, decision rules are maps from an option and a knowledge base to a sequence (word in a formal language) of consequences, an ordered set of literals. The exact meaning of words in this language is not specified and different rules have different languages. We use the languages generated by decision rules as proxies to compare the rules themselves. Apart from the general framework of decision rules and their languages we introduce four concrete decision rules, maxmin, lexmaxmin,

smaxmin and maxmax to demonstrate the concepts.

Uncertainty The interpretation of attacks in our decision making framework is novel: Symmetric attacks in an option's set of arguments stand for multiple possible worlds, because possible worlds are exactly the preferred extensions of an argument graph and preferred extensions arise from symmetric attacks. This interpretation also gives an intuitive explanation of argument graphs whose grounded and preferred extensions conincide: In this case, there is only one possible world and we obtain a model for decision making without uncertainty.

Example 16. *We will now extend the running example (started in Example 11 on page 52) with uncertainty about consequences. We still want to decide which bolts to use, whether or not to use a shim, and how many layers of varnish to apply. The requirements remain the same (structural integrity and corrosion resistance), and so do the options. Unlike before, we are not certain about the consequences of options and consider some alternative outcomes. We will again describe the arguments for option* O_1 *(use steel and titanium bolts, do not use a shim, and apply two layers of varnish).*

- a_1 *Not using a shim in an undamaged structure means that the structure remains balanced.*
- a_2 *Steel/titanium bolts cause microscopic fractures in* s_1 *and* s_2*, resulting in damage to the structure.*
- a_3 *Steel/titanium bolts* do not *cause microscopic fractures in* s_1 *and* s_2*, so the structure will not be damaged.*
- a_4 *Steel/titanium bolts are too strong for the material that* s_1 *is made of, so there will be microscopic fractures.*
- a_5 *Steel/titanium bolts are too strong for the material that* s_1 *is made of, but there* will not be *any microscopic fractures.*
- a_6 *The structure has a high corrosion resistance, because two layers of varnish are used.*

The core issue can be seen in arguments a_4 *and* a_5*: Will there be microscopic fractures as a result of using steel/titanium bolts?* a_4 *and* a_5 *agree that steel/titanium bolts are*

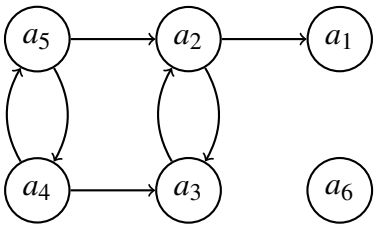


Figure 3.4: Argument graph for Example 16.

stronger than necessary, but they disagree about the consequences: a_4 argues that there will be microscopic fractures, and a_5 argues the opposite. Based on that, arguments a_2 and a_3 are also conflicting, a_2 being based on the assumption that there are some fractures, and a_3 on the assumption that there are none.

The abstract argument graph for this discussion is shown in Figure 3.4. Besides the mutual attacks between a_4 and a_5, and a_2 and a_3 respectively, there are three asymmetric attacks. a_5 attack a_2 because a_2 is based on the assumption that there are fractures whereas a_5 argues that there are none. Similarly, a_4 attacks a_3 because the latter assumes no fractures whereas the former argues for fractures. And finally, a_2 attacks a_1 because the latter assumes that the structure is not damaged.

There are two preferred extensions: $\{a_5, a_3, a_1, a_6\}$ and $\{a_4, a_2, a_6\}$. The first one stands for the case that the bolts do not result in microscopic fractures, and the second one stands for the opposite case. A possible solution to this dilemma would be acquiring more information (for example through experiments – see Chapter 5, or using probabilistic argumentation – see Section 3.5.2), but for now we have to assume that both scenarios are possible.

Example 16 demonstrated how preferred extensions are related to possible worlds. The following definition makes the idea precise:

Definition 32 (Possible World)**.** *Let $D = (K, C, R)$ be a decision frame and let $O \in \mathcal{O}$ be an option. Let $G = (\mathsf{args}_D(O), \mathsf{attacks}(\mathsf{args}_D(O)))$ be the argument graph for O. The set of **possible worlds** of O in D is given by*

$$\mathsf{possibleWorlds}(D, O) = \{E \in \Sigma_{\mathsf{pr}}(G) \mid \mathsf{consequences}_D(E) \neq \emptyset\}$$

Example 17. *Rules for the arguments of Example 16 are shown in Table 3.5 (*`mat` *stands for material). Together with $O_1 = \{\Rightarrow \mathtt{mat(bb, st)}, \Rightarrow \neg\mathtt{shim(t)}, \Rightarrow \mathtt{varnish(t, 2)}\}$,*

Table 3.5: Rules for Example 17

Name	Rule
r_1	$\neg\texttt{shim}(\texttt{t}), \neg\texttt{damaged}(\texttt{t}) \Rightarrow \texttt{balanced}(\texttt{t})$
r_2	$\texttt{microscopic\_fractures} \Rightarrow \texttt{damaged}(\texttt{t})$
r_3	$\neg\texttt{microscopic\_fractures} \Rightarrow \neg\texttt{damaged}(\texttt{t})$
r_4	$\texttt{bolts\_strong} \Rightarrow \texttt{microscopic\_fractures}$
r_5	$\texttt{bolts\_strong} \Rightarrow \neg\texttt{microscopic\_fractures}$
r_6	$\texttt{mat}(\texttt{bb},\texttt{st}) \Rightarrow \texttt{bolts\_strong}$
r_7	$\texttt{varnish}(\texttt{t},2) \Rightarrow \texttt{coRes}(\texttt{t},\texttt{high})$

they result in the same argument graph as shown in Figure 3.4, with additional arguments for the antecedent-free rules in O_1:

$$a_7 = [\Rightarrow \texttt{mat}(\texttt{bb},\texttt{st})]$$
$$a_8 = [\Rightarrow \neg\texttt{shim}(\texttt{t})]$$
$$a_9 = [\Rightarrow \texttt{varnish}(\texttt{t},2)]$$

The graph for $\mathsf{args}_D(O_1)$ *has two preferred extensions, E_1 and E_2:*

$$E_1 = \{a_5, a_3, a_1, a_6, a_7, a_8, a_9\}$$
$$E_2 = \{a_4, a_2, a_6, a_7, a_8, a_9\}$$

So we get two possible worlds: $\mathsf{possibleWorlds}(D, O_1) = \{E_1, E_2\}$.

If we have the set of possible worlds for an option, we want to compare them to see how good or bad the option looks in each of them. The comparison of possible worlds should therefore involve the set of consequences C. This concept is similar to the sat function in multi-criteria decision making, where we compared options by the requirements they fulfil (Definition 29 on page 57). Possible worlds can be analysed in several ways. For example, we could find the worst (least desirable) consequence in each world. This gives us a set of worst-case scenarios, which is useful for a pessimistic evaluation of the consequences of an option. On the other end of this spectrum, we could compare possible worlds by taking only their best consequences. Either way, by comparing possible worlds we distil the set of all arguments for an option down to a set of consequences representing the different potential outcomes of this option. In

a second step (Section 3.3.5), we will use this representative set of consequences to compare options themselves.

To start with, we compare (rather optimistically – see Section 3.3.5.2) the upper bound of the consequences of possible worlds. The best consequence of a possible world E is the upper bound (under $\sqcup$) of all possible consequences of that world.

Definition 33. *Let E be a set of arguments.*

$$\mathsf{bestConsequence}(E) = \bigsqcup \mathsf{consequences}(E)$$

Example 18. *Before we can compute the best consequences for the previous example, we need to establish an ordering of all consequences. As required by Definition 27, the set of all consequences contains the requirements $R = \{\neg\texttt{damaged(t)}, \texttt{coRes(t,high)}\}$ as before. In this example we also consider other consequences that are not requirements, including bad consequences that should be avoided: $C' = R \cup T$ with $T = \{\texttt{balanced(t)}, \texttt{damaged(t)}, \texttt{weight(t,high)}\}$. The set C' is ordered as follows: $\neg\texttt{damaged(t)} > \texttt{coRes(t,high)} > \texttt{balanced(t)} > \texttt{weight(t,high)} > \texttt{damaged(t)}$. Since we have a total order of requirements we get $\sqcap = \min$ and $\sqcup = \max$ for the lattice $(C', \sqcap, \sqcup)$. Now we can compute the best consequences for both possible worlds:*

$$\mathsf{bestConsequence}(\mathsf{E}_1) = \max\{\texttt{balanced}, \neg\texttt{damaged(t)}, \texttt{coRes(t,high)}\} = \neg\texttt{damaged(t)}$$
$$\mathsf{bestConsequence}(\mathsf{E}_2) = \max\{\texttt{coRes(t,high)}, \texttt{damaged(t)}\} = \texttt{coRes(t,high)}$$

The first possible world, E_1 has a better best consequence, namely $\neg\texttt{damaged(t)}$.

Every option has a set of possible worlds and one of those possible worlds is the least desirable one, that is the worst of the possible worlds for this option. The following definition of worstCase returns the best consequence that can be achieved should the worst possible world come true. It can also be viewed as the minimum outcome of an option.

Definition 34. *Let $\mathscr{E}$ be a set of sets of arguments. The **worst case outcome** of $\mathscr{E}$ is given by*

$$\mathsf{worstCase}(\mathscr{E}) = \bigsqcap \{c \in C \mid \exists E \in \mathscr{E} \text{ such that } c = \mathsf{bestConsequence}(E)\}$$

As pessimistic decision makers, we compare two options by comparing their outcomes in the worst case, in order to maximise the minimum result of our decision. This decision rule is known as maxmin.

Definition 35. *Let $D = (K, C, R)$ be a decision frame and let O be an option. The* ***minimum result*** *of O in D, short* $\mathsf{maxmin}(D, O)$ *is given by*

$$\mathsf{maxmin}(D, O) = \mathsf{worstCase}(\mathsf{possibleWorlds}(D, O))$$

Example 19. *For O_1 in Example 18 we get* $\mathsf{maxmin}(D, O_1) = \min\{\neg\mathtt{damaged}(\mathtt{t}), \mathtt{coRes}(\mathtt{t}, \mathtt{high})\} = \mathtt{coRes}(\mathtt{t}, \mathtt{high})$.

3.3.4.1 Comparison with Current Approach

Just as we compared ADF with the classical (decision theory) approach to MCDM, we can compare it with DMU. We do this by giving another correspondence result: Given a DMU problem with a set of options ordered by the *maxmin* rule (see Definition 26 on page 49), we can produce a decision frame in which the corresponding options have the same ordering under the maxmin decision rule. The result is specific to the maxmin rule because the underlying information about utilities and probabilities in the DMU problem are lost when translating it to a decision frame – similar to Propage 1, which is based on a mapping that incorporates the ordering of options under one specific aggregation *agg*.

Proposition 7. *Let $P = (S, X, F)$ be a decision making under uncertainty problem (Definition 25) with $F = \{f_1, \ldots, f_n\}$. There exists a decision frame D and a set of options $\{O_1, \ldots, O_n\}$ such that for every $f_i, f_j \in F$:*

$$f_i \leq_{maxmin_P} f_j \Leftrightarrow O_i \leq_{\mathsf{maxmin}_D} O_j$$

For the proof we will use angled braces $\langle\cdot\rangle$ to denote $\cdot$ as a symbol in the logical language $\mathscr{L}$ – in the same way as in Section 3.3.3.1 when we compared ADF with MCDM. For example, $\langle 1.3\rangle$ maps the number 1.3 to the three-digit symbol 1.3. This operator exists only to improve readability.

The proof itself is analogous to that of Theorem 1. For every action f_i we construct a single defeasible rule $\langle f_i\rangle \Rightarrow \langle maxmin_P(f_i)\rangle$. The set of consequences C is exactly the

set containing $\langle maxmin_P(f_i)\rangle$ for every $f_i \in F$. C is totally ordered (by $\leq$ on the range of $maxmin_P$), and the lattice is given by $\sqcap = \min$ and $\sqcup = \max$.

Proof. Let $P = (S,X,F)$ be as required. Let $D = (K,C,R)$ be a decision frame with

1. $K = \{\langle f_i\rangle \Rightarrow \langle maxmin_P(f_i)\rangle \mid f_i \in F\}$
2. $C = \{\langle maxmin_P(f_i)\rangle \mid f_i \in F\}$
3. $R = \emptyset$

Further, let $O = \{\{\Rightarrow \langle f_i\rangle\}, \ldots, \{\Rightarrow \langle f_n\rangle\}\} \subseteq \mathscr{O}$. We show that D meets the condition of the proof for the set of options O.

($\Leftarrow$) Let $O_i, O_j \in O$ with $O_i = \{\Rightarrow \langle f_i\rangle\}$ and $O_j = \{\Rightarrow \langle f_j\rangle\}$, and with $O_i \leq_{\mathsf{maxmin}_D} O_j$. Now we need to show that $f_i \leq_{maxmin_P} f_j$. By construction, $\mathsf{args}_D(O_i)$ and $\mathsf{args}_D(O_j)$ are conflict-free and contain exactly two arguments each, namely $\{[\Rightarrow \langle f_i\rangle], [\langle f_i\rangle \Rightarrow \langle maxmin_P(f_i)\rangle]\} = \mathsf{args}_D(O_i)$ and $\{[\Rightarrow \langle f_j\rangle], [\langle f_j\rangle \Rightarrow \langle maxmin_P(f_j)\rangle]\} = \mathsf{args}_D(O_j)$. Since $O_i \leq_{\mathsf{maxmin}_D} O_j$, $f_i \leq_{maxmin_P} f_j$ (by Definition of $\leq_C$).

($\Rightarrow$) Let $f_i, f_j \in F$ with $f_i \leq_{maxmin_P} f_j$. Then there exist $\{\Rightarrow \langle f_i\rangle\}, \{\Rightarrow \langle f_j\rangle\} \in O$ and, by construction of K, $\mathsf{args}_D(\{\Rightarrow \langle f_i\rangle\}) = \{[\Rightarrow \langle f_i\rangle], [\langle f_i\rangle \Rightarrow \langle maxmin_P(f_i)\rangle]\}$ and $\mathsf{args}_D(\{\Rightarrow \langle f_j\rangle\}) = \{[\Rightarrow \langle f_j\rangle], [\langle f_j\rangle \Rightarrow \langle maxmin_P(f_j)\rangle]\}$. Both are conflict-free, so we get $\mathsf{maxmin}(\{\Rightarrow \langle f_i\rangle\}) = \{\langle maxmin_P(f_i)\rangle\}$ and $\mathsf{maxmin}(\{\Rightarrow \langle f_j\rangle\}) = \{\langle maxmin_P(f_j)\rangle\}$, and by construction of C, $\mathsf{maxmin}(\{\Rightarrow \langle f_i\rangle\}) \leq \mathsf{maxmin}(\{\Rightarrow \langle f_j\rangle\})$. □

Example 20. *To demonstrate the encoding, consider the DMU problem $P = (S,X,F)$:*

$$S = \{Stagnation, Increase, Decrease\}$$
$$X = \{1,2,3,4\}$$
$$F = \{f_{al}, f_{pl}, f_{st}, f_{co}\} \text{ with}$$
$$f_{al} = Stagnation \mapsto 3, Increase \mapsto 1, Decrease \mapsto 4$$
$$f_{pl} = Stagnation \mapsto 3, Increase \mapsto 3, Decrease \mapsto 1$$
$$f_{st} = Stagnation \mapsto 3, Increase \mapsto 1, Decrease \mapsto 3$$
$$f_{co} = Stagnation \mapsto 2, Increase \mapsto 2, Decrease \mapsto 2$$

Since $X \subseteq \mathbb{R}$, the utility measure u is simply the identity function $u(x) = x$.

To encode P in a decision frame $D = (K, C, R)$ as in the proof of Propage 7, we first need to compute the result of $maxmin_P$ for each of the actions f_{al}, f_{pl}, f_{st} and f_{co}.

$$maxmin_P(f_{al}) = 1 \qquad maxmin_P(f_{pl}) = 1$$
$$maxmin_P(f_{st}) = 1 \qquad maxmin_P(f_{co}) = 2$$

Therefore, K contains the following rules: $K = \{\ \langle f_{al}\rangle \Rightarrow \langle 1\rangle,\ \langle f_{pl}\rangle \Rightarrow \langle 1\rangle,\ \langle f_{st}\rangle \Rightarrow \langle 1\rangle,\ \langle f_{co}\rangle \Rightarrow \langle 1\rangle\ \}$. The set of criteria is $C = \{\langle 1\rangle, \langle 2\rangle\}$.

O, the set of options, contains one entry for each action: $O = \{\{\Rightarrow \langle f_{al}\rangle\}, \ldots\}$. For each option $o \in O$, the set of arguments $\mathsf{args}_D(o)$ contains two arguments, one for the option itself and one with a rule application: $\mathsf{args}_D(\{\Rightarrow \langle f_{al}\rangle\}) = \{[\Rightarrow \langle f_{al}\rangle], [[\Rightarrow \langle f_{al}\rangle]; \langle f_{al}\rangle \Rightarrow \langle 1\rangle; \langle 1\rangle]\}$. The claim of the rule application is a consequence in C. The ordering of C is determined by the ordering of the range of $maxmin_P$. We therefore get

$$\{\Rightarrow \langle f_{al}\rangle\} \leq_{\mathsf{maxmin}_D} \{\Rightarrow \langle f_{co}\rangle\}$$

and so forth.

Proposition 7 is the DMU equivalent of Propage 1 on page 60. This is important, because the two results together show that our model of decision frames subsumes both multi-criteria decision making and decision making with uncertainty.

3.3.5 Decision Rules

Decision rules are useful both for selecting options and for analysing decisions after the fact. In our use case – documenting engineering decisions – decision making itself is a human task, so the second use case is much more relevant: Given a set of decisions that make up a design, we can use decision rules to check how many decisions were made optimistically, pessimistically etc.

In Propage 7, we saw how the maxmin function can be used to establish a ranking of options. This was the first formal example of a decision rule. In this section we will give a precise definition of the term and discuss some additional rules including maxmin.We show that maxmin suffers from the so-called "drowning effect", which means that it may fail to distinguish options even though their consequences are equal

only in some of the possible futures. We then define a second decision rule, lexmaxmin, that solves this problem using a lexicographic criterion. We show that, while not suffering from the drowning effect, lexmaxmin still fails to distinguish some options, and propose a third rule that exploits some argumentation-specific properties to be even more decisive.

Decision rules are mappings from decision frames and options to words over the set of all consequences. We will denote the set of all consequences with $\mathscr{C}$ and its partial order with $\leq_{\mathscr{C}}$. Consequently, the set of words over $\mathscr{C}$ will be called $\mathscr{C}^*$ (see Definition 20 on page 38).

Definition 36. *A **decision rule** Δ is a function* $\Delta : \mathscr{D} \times \mathscr{O} \to \mathscr{C}^*$

The function maxmin is already a decision rule, since it assigns a consequence to each option (and thus produces words of length one). We write Δmaxmin to make clear that we are talking about the decision rule: $\Delta\mathsf{maxmin}(D,O) = [\mathsf{maxmin}(D,O)]$. If $\mathsf{maxmin}(D,O)$ is undefined then $\Delta\mathsf{maxmin} = \varepsilon$.

Example 21. *Recall the set of consequences* $C' = \{$ $\neg$damaged(t), coRes(t,high), balanced(t), weight(t,high), damaged(t) $\}$ *from Example 18 above. A decision rule is any function that maps options to words over R. For example, it could return* [coRes(t,high) balanced(t)] *for an option where the most likely outcome is* coRes(t,high) *and the second most-likely outcome is* balanced(t).

To compare two options with a decision rule, one compares the words generated for the two options using the lexicographic ordering of words over $\mathscr{C}$ with the underlying partial order $\leq_{\mathscr{C}}$ (see page 38).

Definition 37. *Let Δ be a decision rule. For every decision frame D, Δ **induces a partial order** of $\mathscr{O}$ on D as follows:*

$$O \leq^D_\Delta O' \text{ if and only if } \Delta(D,O) \preceq \Delta(D,O')$$

where $\preceq$ is the lexicographic ordering of words in $\mathscr{C}^*$. If $O \leq^D_\Delta O'$ and $O' \leq^D_\Delta O$ then we write $O =^D_\Delta O'$.

Example 22. *Before we can compare O_1 and O_2 in our running example (Example 17), we need to add some rules about the consequences of O_2. The rules are listed in Table 3.6. In summary, the rules say that aluminium increases the weight of the component, and aluminium bolts will definitely not damage the structure.*

For $O_2 = \{\Rightarrow \texttt{mat(bb,al)}, \Rightarrow \texttt{shim(t)}, \Rightarrow \texttt{varnish(t,1)}\}$ *we get the following arguments:* $\mathsf{args}_D(O_2) = \{a_1, \ldots, a_6\}$ *with*

$$a_1 = [\Rightarrow \texttt{mat(bb,al)}] \qquad a_2 = [\Rightarrow \texttt{shim(t)}]$$
$$a_3 = [\Rightarrow \texttt{varnish(t,1)}] \qquad a_4 = [a_1; r_8; \texttt{weight(t,high)}]$$
$$a_5 = [a_2; r_9; \neg\texttt{balanced}] \qquad a_6 = [a_1; r_9; \neg\texttt{damaged(t)}]$$

As a result, there is only a single preferred extension in $\mathsf{args}_D(O_2)$*, containing all arguments a_1 to a_6. This means there is no uncertainty about the consequences of choosing O_2. We then get* $\Delta^D_{\mathsf{maxmin}}(O_2) = \neg\texttt{damaged(t)}$*, because* $\neg\texttt{damaged(t)}$ *is the best outcome we can expect to achieve by choosing O_2.*

Using the Δ_{maxmin} *decision rule,* $O_1 \leq^D_{\Delta_{\mathsf{maxmin}}} O_2$ *and* $O_2 \not\leq^D_{\Delta_{\mathsf{maxmin}}} O_1$*, because the best consequence we are certain to get with O_2 is* $\neg\texttt{damaged(t)}$*, which is better than* $\texttt{coRes(t,high)}$.

In Definition 36 and 37, the two tasks of evaluating an individual option and comparing two options are separated, but in existing work on decision making with uncertainty, both tasks are performed in a single stepage The name maxmin hints at the two steps: First find the minimum expected result of an option and then maximise this value across all options. In our approach, minimisation is performed by the decision rule Δmaxmin and maximisation happens in the partial order established by Definition 37.

The codomain of every decision rule Δ is a language called $L(\Delta)$. The key idea

Table 3.6: Additional rules for Example 22

Name	Rule
r_8	$\texttt{mat(bb,al)} \Rightarrow \texttt{weight(t,high)}$
r_9	$\texttt{shim(t)} \Rightarrow \neg\texttt{balanced}$
r_{10}	$\texttt{mat(bb,al)} \Rightarrow \neg\texttt{damaged(t)}$

in our approach is to compare decision rules by comparing their languages, in order to abstract away from individual decision frames and options.

Definition 38 (Language of a decision rule)**.** *Let* Δ *be a decision rule. The* ***language of*** Δ*,* $L(\Delta)$ *is the codomain of* Δ*:*

$$L(\Delta) = \{w \in \mathscr{C}^* \mid \exists D \in \mathscr{D}.\exists O \in \mathscr{O}.w = \Delta(D,O)\}$$

The language of a decision rule Δ therefore contains all words w over $\mathscr{C}$ for which there is a decision frame D and an option O such that Δ applied to D and O produces the word w.

Proposition 8. *For every* $w \in L(\Delta\mathsf{maxmin})$, $|w| \leq 1$

Proof. Since maxmin (Definition 35) returns exactly one consequence, the words in $L(\Delta\mathsf{maxmin})$ are of length one, or 0 when maxmin is undefined. □

We represent decision rules by their languages, the sets of words over consequencs they can produce. In the next sections, we will look at two characteristics of decision rules: Decisiveness and optimism. We give formal definitions of these concepts, evaluate Δmaxmin on them and propose improved rules that are more decisive and more optimistic.

3.3.5.1 Decisiveness

Multiple options may be mapped to the same word and considered equal under $\leq_{\Delta\mathsf{maxmin}}$, even though their outcomes in possible worlds other than the worst one are actually quite different. This phenomenon, known as the *drowning effect*, is a symptom of a lack of decisiveness of decision rules. The drowning effect occurs when two options are ranked equally even though they should not be, because they lead to different outcomes. Before we can define it formally, we need need to specify when two options *should be* ranked equally. This is the case if they lead to the same possible consequences, denoted by the equivalence relation $\cong_D$:

Definition 39. *Let* $D = (K,C)$ *be a decision frame and let* O, O' *be two options with*

$\mathscr{E} = \mathsf{possibleWorlds}(D, O)$ *and* $\mathscr{E}' = \mathsf{possibleWorlds}(D, O')$

$$O \cong_D O' \textit{ iff } \bigcup_{E \in \mathscr{E}} \mathsf{bestConsequence}(E) = \bigcup_{E' \in \mathscr{E}'} \mathsf{bestConsequence}(E')$$

If a decision rule Δ has the drowning effect, then it will rank two options O, O' as equal ($O \leq^D_\Delta O'$ and $O' \leq^D_\Delta O$) even though they are not equal under $\cong_D$.

Definition 40. *A decision rule* Δ ***has the drowning effect*** *if and only if there exists a decision frame* D *and two options* $O, O' \in \mathscr{O}$ *such that* $\Delta(D, O) = \Delta(D, O')$ *and* $O \ncong_D O'$.

Example 23. *To demonstrate the drowning effect on* maxmin*, let us introduce a third option* O_3 *that will be ranked equal to* O_2*, even though we can expect a slightly better outcome from it.* $O_3 = \{ \Rightarrow \mathtt{mat(bb,al)}, \Rightarrow \mathtt{shim(t)}, \Rightarrow \mathtt{varnish(t,2)} \}$*. The only difference between* O_2 *and* O_3 *is that we apply two layers of varnish (*$\mathtt{varnish(t,2)}$*) instead of one. This allows us to apply rule* $r_7 = \mathtt{varnish(t,2)} \Rightarrow \mathtt{coRes(t,high)}$ *(see Table 3.5 on page 65), resulting in the following arguments for* O_3*.* $\mathsf{args}_D(O_3) =$

$$a_1 = [\Rightarrow \mathtt{mat(bb,al)}] \qquad a_2 = [\Rightarrow \mathtt{shim(t)}]$$
$$a_7 = [\Rightarrow \mathtt{varnish(t,2)}] \qquad a_4 = [a_1; r_8; \mathtt{weight(t,high)}]$$
$$a_5 = [a_2; r_9; \neg\mathtt{balanced}] \qquad a_6 = [a_1; r_9; \neg\mathtt{damaged(t)}]$$
$$a_8 = [a_3; r_7; \mathtt{coRes(t,high)}]$$

Let us now apply Definition 39 to see whether $O_2 \cong_D O_3$*. Since both* $\mathsf{args}_D(O_2)$ *and* $\mathsf{args}_D(O_3)$ *are conflict-free,* $\mathscr{E} = \mathsf{possibleWorlds}(D, O_2) = \{\mathsf{args}_D(O_2)\}$ *and* $\mathscr{E}' = \mathsf{possibleWorlds}(D, O_3) = \{\mathsf{args}_D(O_3)\}$*. Taking the best consequences of the possible worlds of* O_2 *and* O_3*, we get*

$$\bigcup_{E \in \mathscr{E}} \mathsf{bestConsequence}(E) = \{\neg\mathtt{damaged}, \mathtt{weight(t,high)}\}$$
$$\bigcup_{E \in \mathscr{E}'} \mathsf{bestConsequence}(E) = \{\neg\mathtt{damaged}, \mathtt{weight(t,high)}, \mathtt{coRes(t,high)}\}$$

So because there is an argument for $\mathtt{coRes(t,high)}$ *in at least one possible world of* O_3*, but not of* O_2*,* O_2 *is not D-equivalent to* O_3 *(*$O_2 \ncong_D O_3$*).*

Proposition 9. Δmaxmin *has the drowning effect*

Proof. In Example 23, $O_2 \not\geq_D O_3$, but $\mathsf{maxmin}(D, O_2) = [\neg\mathtt{damaged}] = \mathsf{maxmin}(D, O_3)$, so $\Delta\mathsf{maxmin}$ has the drowning effect. □

The decision rule *maxmin* can be repaired by considering not just the single best minimum outcome, but also the second-best outcome, the third-best outcome and so forth (see [53]). This is achieved through repeatedly evaluating the set of possible worlds, removing the one with the "minimal outcome" each time. We are now going to extend $\Delta\mathsf{maxmin}$, the argumentation-specific version of the rule, in similar fashion.

In order to be able to pick the possible world with the minimal outcome, we require the lattice operation $\sqcup$ on C to have the property $a \sqcup b \in \{a, b\}$ - in other words, is must be join-irreducible (cf. page 37). Also please note that $\circ$ denotes concatenation of words (as defined on page 38).

Definition 41. *Let D be a decision frame and let O be an option.* $\Delta\mathsf{lexmaxmin}(D, O) = \mathsf{lmaxmin}(\mathsf{possibleWorlds}(D, O))$ *where*

$$\mathsf{lmaxmin}(\mathscr{E}) = \begin{cases} \varepsilon & \text{if } \mathscr{E} = \emptyset \\ [c] \circ \mathsf{lmaxmin}(\mathscr{E}') & \text{otherwise, with} \\ & c = \mathsf{worstCase}(\mathscr{E}) \text{ and} \\ & \mathscr{E}' = \mathscr{E} \setminus \{E \in \mathscr{E} \mid c = \bigsqcup(\mathsf{consequences}(E))\} \end{cases}$$

The behaviour of $\Delta\mathsf{lexmaxmin}$ for option O_1 in Example 24 is an example of the property that each one of the different "best consequences" of an option O appears exactly once in $\Delta\mathsf{lexmaxmin}(D, O)$, for any decision frame D:

Proposition 10. *For every decision frame D and every option O, if $\mathscr{E} = \mathsf{possibleWorlds}(D, O)$ then*

$$\Delta\mathsf{lexmaxmin}(D, O) \text{ is a permutation of } \bigcup_{E \in \mathscr{E}} \mathsf{bestConsequence}(E)$$

Proof. (Sketch) Let D be a decision frame and let O be an option. Let $\mathscr{E} = \mathsf{possibleWorlds}(D, O)$ and let $w = [c_1 \ldots c_n] = \Delta\mathsf{lexmaxmin}(D, O)$. ($\subseteq$:) By Definition 41, $w = \mathsf{lmaxmin}(\mathscr{E})$. Let $c_i \in \mathscr{C}$ with $1 \leq i \leq n$. By Definition 41, there exists a $\mathscr{E}' \subseteq \mathscr{E}$ such that $c_i = \mathsf{worstCase}(\mathscr{E}')$, so by Definition 34, there exists an $E \in \mathscr{E}'$ such that $c = \mathsf{bestConsequence}(E)$, so $c_i \in \bigcup_{E \in \mathscr{E}} \mathsf{bestConsequence}(E)$. ($\supseteq$:) Let

Example 24. *For option O_1 from Example 18, we get*

$$\begin{aligned}
\Delta\mathsf{lexmaxmin}(D,O_1) &= \mathsf{lmaxmin}(\mathsf{possibleWorlds}(D,O_1)) \\
&\quad \textit{(see Example 17)} \\
&= \mathsf{lmaxmin}(\{E_1,E_2\}) \\
&\quad \textit{(by Definition 41)} \\
&= [\mathsf{worstCase}(\{E_1,E_2\})] \circ \mathsf{lmaxmin}(\mathscr{E}') \\
&\quad \textit{(by Definition 34 and Example 18)} \\
&= [\mathtt{coRes}(\mathtt{t},\mathtt{high})] \circ \mathsf{lmaxmin}(\mathscr{E}') \\
&\quad \textit{(by Definition 41)} \\
&= [\mathtt{coRes}(\mathtt{t},\mathtt{high})] \circ \mathsf{lmaxmin}(\{E_1\}) \\
&\quad \textit{(by Definition 41)} \\
&= [\mathtt{coRes}(\mathtt{t},\mathtt{high})] \circ [\mathsf{worstCase}(\{E_1\})] \circ \mathsf{lmaxmin}(\mathscr{E}'') \\
&\quad \textit{(by Definition 34 and Example 18)} \\
&= [\mathtt{coRes}(\mathtt{t},\mathtt{high})] \circ [\neg\mathtt{damaged}] \circ \mathsf{lmaxmin}(\mathscr{E}'') \\
&\quad \textit{(by Definition 41)} \\
&= [\mathtt{coRes}(\mathtt{t},\mathtt{high})] \circ [\neg\mathtt{damaged}] \circ \mathsf{lmaxmin}(\emptyset) \\
&= [\mathtt{coRes}(\mathtt{t},\mathtt{high})] \circ [\neg\mathtt{damaged}] \circ \varepsilon \\
&\quad \textit{(Simplify)} \\
&= [\mathtt{coRes}(\mathtt{t},\mathtt{high})\neg\mathtt{damaged}]
\end{aligned}$$

showing that the guaranteed minimal outcome we can achieve by choosing O_1 is $\mathtt{coRes}(\mathtt{t},\mathtt{high})$*, and the "almost guaranteed" (in all but the worst possible worlds) minimal outcome is* $\neg\mathtt{damaged}$.

$c \in \bigcup_{E\in\mathscr{E}} \mathsf{bestConsequence}(E)$. To show that there is an $i \leq n$ such that $c = c_i$, consider that there exists an $E \in \mathscr{E}$ such that $c = \mathsf{bestConsequence}(E)$. Definition 41 ensures that for every $E \in \mathscr{E}$, $\mathsf{bestConsequence}(E)$ appears exactly once in $\Delta\mathsf{lexmaxmin}(\mathscr{E})$, so the c_i exists and w is a permutation. □

Lemma 1. $\Delta\mathsf{lexmaxmin}(D,O) = \Delta\mathsf{lexmaxmin}(D,O')$ *if and only if* $O \cong_D O'$

Proof. $\Rightarrow$: Let options O,O' and a decision frame D s.t. $\Delta\mathsf{lexmaxmin}(D,O) = \Delta\mathsf{lexmaxmin}(D,O')$. Then by Propage 10 and Definition 39, $O \cong_D O'$. $\Leftarrow$: Assume options O,O' and a decision frame D such that $O \cong_D O'$. Let $w = \Delta\mathsf{lexmaxmin}(D,O)$ and let $w' = \Delta\mathsf{lexmaxmin}(D,O')$. Definition 41 Case 2 implies that the letters in any word in $L(\Delta\mathsf{lexmaxmin})$ are in strictly increasing order (over $\leq_{\mathscr{C}}$), and since w and w' contain the same letters (Propage 10 and Definition 39) $w = w'$.

□

For the next result, recall that if S is a set and $\sim$ is an equivalence relation over S then $S/\sim$ is the set of equivalence classes of S (see page 37).

Theorem 2. *For every decision frame D, $\leq^D_{\Delta\mathsf{lexmaxmin}}$ is a total order of $\mathscr{O}/\cong_D$*

Proof. (Sketch) $\leq^D_{\Delta\mathsf{lexmaxmin}}$ is a total order if it is reflexive, antisymmetric, transitive and defined on every pair $O, O' \in \mathscr{O}$. Reflexivity and antisymmetry follow from Lemma 1. To show transitivity, let D be a decision frame and consider three options O_1, O_2 and O_3 such that $O_1 \leq^D_{\Delta\mathsf{lexmaxmin}} O_2$ and $O_2 \leq^D_{\Delta\mathsf{lexmaxmin}} O_3$. Then, by Definition 37, $\Delta\mathsf{lexmaxmin}(D, O_1) \preceq \Delta\mathsf{lexmaxmin}(D, O_2)$ and $\Delta\mathsf{lexmaxmin}(D, O_2) \preceq \Delta\mathsf{lexmaxmin}(D, O_3)$. By transitivity of $\preceq$, $\Delta\mathsf{lexmaxmin}(D, O_1) \preceq \Delta\mathsf{lexmaxmin}(D, O_3)$ and therefore $O_1 \leq^D_{\Delta\mathsf{lexmaxmin}} O_3$. □

Corollary 1. Δlexmaxmin *does not have the drowning effect*

Another way of comparing the decision functions Δmaxmin and Δlexmaxmin is by establishing a relationship between $L(\Delta\mathsf{maxmin})$ and $L(\Delta\mathsf{lexmaxmin})$, namely that Δmaxmin is a prefix of Δlexmaxmin:

Proposition 11. *For every decision frame D and every option O,*

$$\Delta\mathsf{maxmin}(D, O) \sqsubseteq \Delta\mathsf{lexmaxmin}(D, O)$$

Proof. Let D be a decision frame, let O be an option and let $w = [c_1] \circ w' = \Delta\mathsf{lexmaxmin}(D, O)$. Then, by Definition 41, $c_1 = \mathsf{worstCase}(\mathsf{possibleWorlds}(D, O))$. Since $\mathsf{worstCase}(\mathsf{possibleWorlds}(D, O)) = \Delta\mathsf{maxmin}(D, O)$ (by Definition 35) the claim holds. □

The preceding discussion of the drowning effect in Δmaxmin leads to the general question of how decisive a decision rule is. While we cannot assign a "degree of decisiveness", we can say which of two decision rules is more decisive than the other one. The more decisive a rule is, the fewer pairs of options are ranked equally by it.

Definition 42. *Let Δ, Δ' be two decision rules. Δ is **more decisive than** Δ' iff for every decision frame D and for every pair of options O, O': If $O =^D_\Delta O'$ then $O =^D_{\Delta'} O'$*

If Δ is more decisive than Δ' and Δ' is not more decisive than Δ, then we say Δ is *strictly more decisive* than Δ'.

Theorem 3. Δlexmaxmin *is strictly more decisive than* Δmaxmin

Proof. We first show that Δlexmaxmin is more decisive than Δmaxmin, and then that Δmaxmin is not more decisive than Δlexmaxmin. (1) Let D be a decision frame and let O, O' be options such that $O =^{D}_{\Delta\mathsf{lexmaxmin}} O'$. Let $w = \Delta\mathsf{lexmaxmin}(D,O) = \Delta\mathsf{lexmaxmin}(D,O')$. If $w = \varepsilon$ then $\Delta\mathsf{maxmin}(D,O) = \Delta\mathsf{maxmin}(D,O') = \varepsilon$ so $O =^{D}_{\Delta\mathsf{maxmin}} O'$. If $w \neq \varepsilon$ then $w = [c] \circ w'$ for a $c \in \mathscr{C}$. Consider $v = \Delta\mathsf{maxmin}(D,O)$ and $v' = \Delta\mathsf{maxmin}(D,O')$. By Propage 8, $|v| = |v'| = 1$ and since both v and v' are prefixes of w (by Propage 11), $v = v' = [c]$, so $O =^{D}_{\Delta\mathsf{maxmin}} O'$. (2) Refer to the examples for a counterexample □

The definitions and results presented so far clarified the meaning of decisiveness of decision rules, but the actual rules discussed were not novel (except for their translation to our argumentation-based model of decision making with uncertainty). With Δsmaxmin we will now study a novel decision rule that considers information which is not available in the non-argumentative appproach.

Argument strength is a notion that is, by definition, only available in settings where there are arguments. Several measures of argument strength have been developed in the literature, based for example on abstract argumentation [55] or on additional information about arguments such as preferences [56, 57]. The idea of argument strength is to measure the "convincingness" of an argument numerically. At this point we will not review all measures of argument strength, nor pick a specific one. Instead we are going to use the number of arguments in favour of a claim as a proxy for the claim's strength. Counting arguments is generally dubious - we only use it here to give an example of an argumentation-specific decision rule, not as a representative example of argument strength.

With this in mind we simply set "strength" to be equal to the number of accepted arguments in favour of it: $\mathsf{strength}(c,E) = |\{a \in E \mid \mathsf{claim}(a) = c\}|$.

First note that in $L(\Delta\mathsf{lexmaxmin})$ and in $L(\Delta\mathsf{maxmin})$, every word contains a consequence at most once, and the letters of each word are ordered by preference ($\leq_{\mathscr{C}}$) in descending order. The idea for expressing argument strength in decision functions is to repeat claims backed by stronger arguments proportionately to their strength. Due to the lexicographic ordering used in decision rules (Definition 37), words with repeated

letters will be preferred over their single-letter counterparts.

This idea is the basis of the final decision rule Δsmaxmin (s for strength).

Definition 43. *Let D be a decision frame and let O be an option.*

$\Delta\mathsf{smaxmin}(D,O) = \mathsf{smaxmin}'(\mathsf{possibleWorlds}(D,O))$ *where*

$$\mathsf{smaxmin}'(\mathscr{E}) = \begin{cases} \varepsilon & \text{if } \mathscr{E} = \emptyset \\ [c]^n \circ \mathsf{smaxmin}'(\mathscr{E}') & \text{otherwise, with} \\ & c = \mathsf{worstCase}(\mathscr{E}) \text{ and} \\ & \mathscr{E}' = \mathscr{E} \setminus \{E \in \mathscr{E} \mid c = \mathsf{bestConsequence}(E)\} \\ & n = \sum_{E \in \mathscr{E}'} \mathsf{strength}(c,E) \end{cases}$$

Example 25. *With option $O_1 = \{\Rightarrow \texttt{mat(bb,st)}, \Rightarrow \neg\texttt{shim(t)}, \Rightarrow \texttt{varnish(t,2)}\}$ in our running example, we get $\Delta\mathsf{smaxmin}(D,O_1) = [\texttt{coRes(t,high)}\texttt{coRes(t,high)}\neg\texttt{damaged}]$, because the argument for* coRes(t,high) *is part of both possible worlds, E_1 and E_2. The argument for* ¬damaged *is only part of one possible world.*

Note that our Definition 43 applies the measure of argument strength to each extension $E \in \mathscr{E}'$. With our definition of strength, this amounts to counting the number of arguments for c across all extensions (so the same argument may be counted multiple times if it is in more than one extension).

Proposition 12. Δsmaxmin *does not have the drowning effect*

Proof. (Sketch) Proof analogous to proof of Theorem 2 and Corollary 1. □

Proposition 13. Δsmaxmin *is more decisive than* Δlexmaxmin.

Proof. (Sketch) Every claim in $\bigcup_{E \in \mathsf{possibleWorlds}(D,O)} \mathsf{bestConsequence}(E)$ has a strength of at least 1, so lexmaxmin can be viewed as a special case of maxmin where the strength of each claim is exactly 1. So whenever two options result in equal words with smaxmin, they also result in equal words in lexmaxmin. □

We conclude this section with two general results on the connection between injectivity of a decision rule Δ and its decisiveness. By injectivity of a decision rule we specifically mean its injectivity *for the same decision frame*: A decision rule Δ is

option-injective if for every decision frame D, the function $\Delta_D(x) = \Delta(D,x)$ is injective (Δ_D is Δ with the first argument fixed).

Proposition 14. *If a decision rule Δ is option-injective then it does not have the drowning effect.*

Proof. Assume a decision rule Δ option-injective and assume Δ has the drowning effect. By option-injectivity of Δ, $O \cong_D O'$ if and only if $O = O'$, so if $O \ncong_D O'$ for a decision frame D then $O \neq O'$ so $\Delta(D,O) \neq \Delta(D,O')$. This contradicts the assumption that f has the drowning effect (Definition 40). □

As one might expect, option-injectivity of decision rules acts as an upper bound on decisiveness:

Theorem 4. *If two decision rules Δ, Δ' are option-injective then both are equally decisive.*

Proof. Assume two decision rules Δ, Δ' option-injective such that Δ is strictly more decisive than Δ'. Then there exists a decision frame D and two options O, O' such that (1) $O =^D_{\Delta'} O'$ and (2) $O \neq^D_{\Delta} O'$. By option-injectivity of Δ, $O \neq O'$, but then by injectivty of Δ', $O \neq^D_{\Delta'} O'$, which contradicts the assumption that Δ is strictly more decisive than Δ'. □

In this chapter we only used a simple measure of the strength of a claim – namely the number of arguments in favour of it – in order to demonstrate how strength can be reflected in the language of decision rules. However, any of the more sophisticated measures proposed in the literature could be used in a similar way. Additionally, proposals for probabilistic argumentation [58, 59, 60, 61, 62, 63] or argumentation with belief values [64] can be harnessed to obtain quantitative decision making with uncertainty in an argumentation-theoretic setting.

The Δsmaxmin rule (Definition 43) is also an example of how argumentation-specific properties may be combined with existing decision rules for decision making with uncertainty. This illustrates how decision rules such as Δmaxmin and Δlexmaxmin, which treat the argumentation system as a black box and only use its preferred extensions, do not achieve the same decisiveness as, for example, Δsmaxmin. The language-

based approach we advocate in this thesis enables decision makers to compare both kinds of rules systematically.

3.3.5.2 Optimism

The Δmaxmin decision rule and its descendants Δlexmaxmin and Δsmaxmin are rather pessimistic since they pick the consequences that will be achieved with certainty (in all possible worlds) rather than those that may only be achieved in one or two cases. A more optimistic decision maker would compare options by their best consequences in any possible world. For decision rules this means a rule Δ is more optimistic than a rule Δ' if it selects better consequences for the same option. Formally:

Definition 44. *Let Δ, Δ' be two decision rules. Δ is **more optimistic** than Δ' if for every decision frame D and every option O: $\Delta'(D,O) \preceq \Delta(D,O)$*

The optimistic decision rule Δmaxmax can now be defined as

Definition 45. *Let D be a decision frame and let O be an option.*

$$\Delta\mathsf{maxmax}(D,O) = \sqcup\{c \in \mathscr{C} \mid \exists E \in \mathsf{possibleWorlds}(D,O) \text{ such that } c = \mathsf{bestConsequence}(E)\}$$

Proposition 15. Δmaxmax *is more optimistic than* Δmaxmin.

Proof. (Sketch) Assume that the proposition is false. Then there exist D, O such that $\Delta\mathsf{maxmax}(D,O) \prec \Delta\mathsf{maxmin}(D,O)$. Let $[c] = \Delta\mathsf{maxmin}(D,O)$ and let $[c'] = \Delta\mathsf{maxmax}(D,O)$. Then, by Definition 35, w is the worst outcome of all possible worlds and by Definition 45, c' is the best outcome. Since the best outcome is always greater than or equal to the worst outcome, $[c] \preceq [c']$, so the assumption is false. □

3.3.5.3 Summary

Let us briefly summarise our tour of decision making with uncertainty in ADF, and specifically the topic of decision rules. Starting with the idea that uncertainty is manifested in preferred extensions of an argument graph (Section 3.3.4 on page 63), we translated the well-known *maxmin* decision rule into our ADF framework, demonstrating that every decision-making with uncertainty problem formulated in the tradi-

tional model can be expressed as an ADF in a way that *maxmin* gives the same results (Propage 7).

Decision rules are important in the analysis of engineering design decisions because they allow us to classify past decisions by their optimism or other characteristics. In support of this application we developed a theory of decision rules. Our theory was built on a novel, language-based understanding of rules (see Definition 36 on page 70). We translated the two well-known rules *maxmin* and *lexmaxmin* into our argumentation-theoretic setting, and proposed a new rule $\Delta\mathsf{smaxmin}$ based on argument strength. The decisiveness of decision rules is crucial for our use case, and we showed that the three rules we studied have different degrees of decisiveness.

In the preceding paragraphs we developed a useful toolkit for analysing decisions. This includes the identification of recommended options according to some formal principles, and the classification of past decisions by the same principles. Using our work, an engineering design manager will thus be able to say for example "two-thirds of decisions in the last quarter were made rather optimistically" – a valuable piece of information.

3.4 Accepting a Decision

The previous sections equipped us with the ability to weigh the different options of a decision against each other in the presence of multiple decision criteria (Section 3.3.3) and of multiple possible worlds (Section 3.3.4). We are now going to add the ability to actually make a decision, which is also quite important. By making a decision we mean adjusting the knowledge base to reflect the fact that an option was chosen. Afterwards, the knowledge representing the chosen option should be part of the knowledge base, in a way that all arguments pro the option are sceptically acceptable.

3.4.1 On the Deactivation of Rules in ASPIC+

Before we get to our actual goal (accepting a decision), we need to define a general operation on ASPIC+ knowledge bases that will be required later: Enforcing a set of arguments, which in turn relies on the operation of deactivating a defeasible rule r. To enforce a set of arguments means to make it part of the grounded extension of an argument graph, and to deactivate a rule means to make it unavailable for reasoning in

accepted arguments (any arguments that use r defeated by an attacker whose conclusion is $\neg\langle r\rangle$). Both operations will be defined only in terms of additional rules that are added to the knowledge base. They do not require the deletion of any knowledge. Enforcement has been studied for abstract argument graphs [65, 66, 67, 68]. However, no prior work exists on enforcement in the context of instantiated arguments, such as with ASPIC+.

As an example, if a rule r_1 expresses the assumption that all birds can fly ($r_1 = \texttt{bird}(X) \Rightarrow \texttt{fly}(X)$) then r_1 could be deactivated by adding the literal $l = \neg\langle r_1\rangle$ to the knowledge base. The effect of l is that every argument which uses r_1 is attacked asymmetrically by the argument $[\neg\langle r_1\rangle]$. Of course, r_1 could also be attacked in a specific context only: $r_2 = \texttt{penguin}(X) \Rightarrow \neg\langle r_1\rangle$.[2] In this section we are interested in a generic approach to deactivation, such as l. In contrast, rules such as r_2 require domain-specific knowledge and can therefore not be the result of a generic deactivation that can be applied to any possible rule.

Deactivation of rules and enforcement of arguments are highly relevant for engineering design. Since design processes are iterative, it is common to revisit previous decisions when assumptions or external requirements have changed. In such cases it is important to keep a record of the reasoning behind the original (changed) decision, in order to prevent repeating the same mistakes, and for auditing purposes. Therefore, the processes analysed in the next chapter have the property that while their underlying knowledge bases grow monotonically, their sets of acceptable arguments do not. This non-monotonicity requires us to override – that is, to make unavailable – rules without actually removing them from the knowledge base.

Our proposal for deactivation and enforcement consists of two functions, which we will define in this section: enforce and deactivate. enforce relies on the more fundamental operation, deactivate. If a rule r is deactivated then any argument that uses it is attacked asymmetrically (by an argument with conclusion $\neg\langle r\rangle$) and thus excluded from any preferred extension.

In order to deactivate a rule r in ASPIC+, the deactivate operation defined below (Definition 47) adds a rule $r' = \Rightarrow \neg\langle r\rangle$ with empty antecedent. Because r' is itself a

[2] With either of l and r_2, the argument using r_1 is attacked by the grounded extension of the graph, so $\neg\texttt{fly}(X)$ is sceptically acceptable. This is different from the classical penguin-fly example, where the two conclusions $\texttt{fly}(X)$ and $\neg\texttt{fly}(X)$ each have a credulously acceptable argument.

rule, it can be deactivated (and r re-activated) with another rule $r'' = \Rightarrow \neg\langle r' \rangle$, resulting in a chain of arguments in the corresponding argument graph. The chain acts as a record of repeated de- and re-activations of r.

In other words, to enable a rule r, we need to "disable" all arguments whose conclusion is $\neg\langle r \rangle$. By this definition, to activate a rule r is to un-deactivate its name $\langle r \rangle$. Note that this does not guarantee the availability of r for arguments, it only means if an argument uses r then it will not be defeated by default.

In this section we will find a correct definition of deactivate in several iterations, by considering various candidates.

3.4.1.1 Naive Approach

The simplest definition of $\mathsf{deactivate}(r, KB)$ would be to add the literal $\neg\langle r \rangle$ to the rules in KB:

$$\mathsf{deactivate1}(r, KB) = KB \cup (\{\neg\langle r \rangle\}, \emptyset)$$

This will allow us to generate an argument for $\neg\langle r \rangle$ which asymmetrically attacks any argument that uses r, effectively deactivating r. However, there are two disadvantages to this approach:

1. deactivate1 cannot be reversed easily (through adding more knowledge). We could add another literal $\langle r \rangle$ to the knowledge base, but that would lead to a symmetric attack between it and the argument for $\neg\langle r \rangle$ (assuming the contrariness function behaves like classical negation), so any argument that uses r would be acceptable in some preferred extensions and rejected in others, so its acceptability would not be exactly the same as before applying deactivate.

2. deactivate1 implictly assumes that there are no arguments with conclusions $\langle r \rangle$ or $\neg\langle r \rangle$ in KB (because if there were, either the new argument would be defeated, or it would be engaged in a symmetric attack, so it would potentially be only credulously acceptable). Ideally the approach should work without any assumptions about KB.

Example 26. *To demonstrate why* deactivate1 *is not an adequate definition, consider the* ASPIC+ *knowledge base* $KB_1 = (\mathscr{R}_1, \mathscr{K}_1)$ *with* $\mathscr{R}_1 = \{r_1\}$, $r_1 = \mathtt{a} \Rightarrow \mathtt{b}$ *and* $\mathscr{K}_1 = \{\mathtt{a}, \langle \mathtt{a} \Rightarrow \mathtt{b} \rangle\}$. $KB_1' = (\mathscr{R}_1', \mathscr{K}_1') = \mathsf{deactivate1}(KB_1, r_1)$ *with* $\mathscr{K}_1' = \mathscr{K}_1 \cup \{\neg\langle r_1 \rangle\}$ *and*

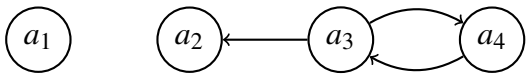


Figure 3.5: Argument graph for Example 26

$\mathscr{R}'_1 = \mathscr{R}_1$. *The argument graph of* KB'_1 *contains four arguments:*

$$a_1 = [\mathsf{a}]$$
$$a_2 = [a_1; \mathsf{a} \Rightarrow \mathsf{b}; \mathsf{b}]$$
$$a_3 = [\neg\langle r_1 \rangle]$$
$$a_4 = [\langle r_1 \rangle]$$

As shown in Figure 3.5, a_3 *attacks* a_2 *and* a_4. a_4 *attacks* a_3. *Therefore there are two preferred extensions,* $\{a_1, a_2, a_4\}$ *and* $\{a_1, a_3\}$. *So the rule* r_1 *is still used in a credulously acceptable argument (*a_2*) and* deactivate1 *did not work as intended.*

3.4.1.2 Improvement

The first problem of deactivate1 is irreversibility. More precisely, the problem is that the reversal of deactivate1 would produce a symmetric attack, since arguments can be attacked asymmetrically only on their rules and the argument for $\neg\langle r \rangle$ does not use any rules, since $\neg\langle r \rangle$ is a fact in KB'. This means that any definition of deactivate should only add rules to the knowlege base, not facts. This is what deactivate2 does:

$$\mathsf{deactivate2}(r, KB) = KB \cup \{\Rightarrow \neg\langle r \rangle\}$$

In order to reverse the effect of deactivate2, we could add another rule $\Rightarrow \neg\langle \Rightarrow \neg\langle r \rangle\rangle$, which creates a defender of all arguments that use r. This solves the first issue, but the second one still remains, in addition to a new problem:

1. deactivate2 can be reversed by adding a rule $\Rightarrow \neg\langle \Rightarrow \neg\langle r \rangle\rangle$ as described, but what if r is to be de-activated again after that? Simply applying deactivate2 again is not enough, since the rule it creates is already part of the knowledge base and it therefore has no effect at all.

2. The second problem remains: deactivate2 implictly assumes that neither $\Rightarrow \neg\langle r \rangle$ nor $\Rightarrow \neg\langle \Rightarrow \neg\langle r \rangle\rangle$ can be derived from KB.

We are getting closer to a solution, since problem 1 is now a special case of problem 2.

3.4.1.3 Final Definition

The discussion of deactivate2 indicated that the real definition of deactivate has to add a new rule whose content depends on the argument graph of KB. This leads to the following approach for deactivating a rule:

1. Let $r' = \Rightarrow \neg\langle r\rangle$[3]
2. Add r' to the knowledge base
3. For each credulously acceptable attacker $a \in A$ of the argument $[;r';\neg\langle r\rangle]$:
   (a) Deactivate $\mathsf{topRule}(a)$
4. Repeat until no more rules are added.

Definition 47 is a straightforward translation of this recipe, except for step 4, which is realised by recursion. deactivate delegates all work to $\mathsf{deactivate}'$, which we will define first:

Definition 46. *Let r be a rule and let A be a set of* ASPIC+*-arguments.*

$$\mathsf{deactivate}'(r,A) = \{r'\} \cup \bigcup_{a \in \mathsf{attackers}(b,A)} \mathsf{deactivate}'(\mathsf{topRule}(a), A \setminus \{a\})$$

with $r' \Longrightarrow \neg\langle r\rangle$, and $b = [;r';\neg\langle r\rangle]$.

In Definition 46, the variables have the following intuitive meaning: r is the rule to be deactivated by introducing a new argument b, A is the set of arguments with potential attackers of b, r' is the deactivating rule that enables b, and A' contains all arguments in A except the "offending" one, a (this guarantees termination as the recursive call is made with a strictly smaller set than A). An example of $\mathsf{deactivate}'$ can be found on page 87 (Example 29).

You may have noticed that the function topRule, which we used in the definition of $\mathsf{deactivate}'$, is only defined for arguments that have a top rule, and undefined for literal arguments such as $[\mathsf{a}]$ (cf. its definition on page 30). This is the reason why we only

[3] r and r' stand for rules, not for rule names. If we want to refer to them in definitions we need to use the naming function $\langle\cdot\rangle$ which maps rules to elements of the language $\mathscr{L}$.

consider knowledge bases without literals in this thesis (see discussion on page 32). All literals are represented by defeasible rules of the form $\Rightarrow \mathtt{a}$, and instead of atomic arguments $[\mathtt{a}]$ we have arguments $[\Rightarrow \mathtt{a}]$.

With the previous definitions in place, deactivate itself is relatively simple:

Definition 47 (Deactivate)**.** *Let r be a rule and let KB be an* ASPIC+ *knowledge base.*

$$\mathsf{deactivate}(r, KB) = KB \cup \mathsf{deactivate}'(r, \mathsf{arguments}(KB))$$

By this definition of deactivate (Definition 47), all acceptable attackers of r are themselves attacked (through deactivation of their top rule). The recursion ensures that all leaves of the dialectical tree for each "user" of r are attacked.

Example 27. *We begin with a simple example. Consider the argumentation system $KB_1 = \{r_1, \Rightarrow \mathtt{a}, \Rightarrow \mathtt{b}, \Rightarrow \mathtt{c}\}$ where there is only one rule with non-empty antecedent, $r_1 = \mathtt{a} \Rightarrow \mathtt{d}$. Then $KB_2 = \mathsf{deactivate}(r_1, KB_1) = KB_1 \cup \{r'\}$ where $r' = \Longrightarrow \neg\langle r_1 \rangle$. The argument graph of KB_2 contains five arguments, $a_1 = [\Rightarrow \mathtt{a}]$, $a_2 = [\Rightarrow \mathtt{b}]$, $a_3 = [\Rightarrow \mathtt{c}]$, $a_4 = [a_1; r_1 \Rightarrow \mathtt{d}]$ and $a_5 = [; \Rightarrow \neg\langle r_1 \rangle; \neg\langle r_1 \rangle]$. a_5 attacks a_4 asymmetrically so the only preferred extension is $\{a_1, a_2, a_3, a_5\}$ which coincides with the grounded extension.*

Example 28. *An example with multiple preferred extensions, $KB_2 = \{r_1, r_2, r_3, \Rightarrow \mathtt{a}\}$ with $r_1 = \mathtt{a} \Rightarrow \mathtt{b}$, $r_2 = \mathtt{a} \Rightarrow \neg\mathtt{b}$ and $r_3 = \mathtt{b} \Rightarrow \mathtt{c}$.*

The arguments for KB_2 are

$$a = [\Rightarrow \mathtt{a}]$$
$$b = [a; \mathtt{a} \Rightarrow \mathtt{b}; \mathtt{b}]$$
$$c = [a; \mathtt{a} \Rightarrow \neg\mathtt{b}; \neg\mathtt{b}]$$
$$d = [b; \mathtt{b} \Rightarrow \mathtt{d}; \mathtt{b}]$$

with attacks as in Figure 3.6 (left) on p 87. Dashed lines indicate sub-argument relations (transitive sub-arguments may be inferred, for example a is a sub-argument of d). The graph on the left is of KB_2 and the graph on the right is of $KB_2' = \mathsf{deactivate}(KB_2, r_2)$.

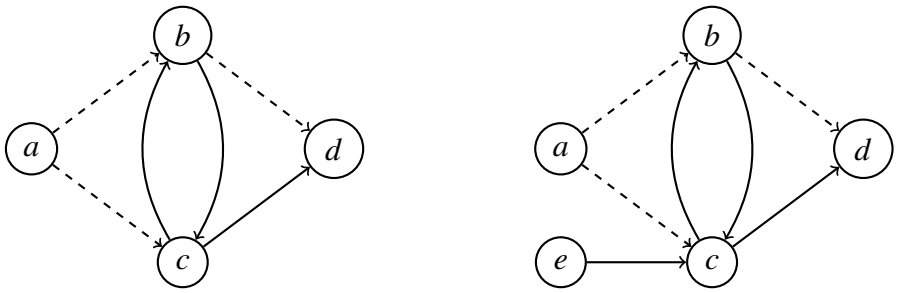


Figure 3.6: Argument graphs for Example 28.

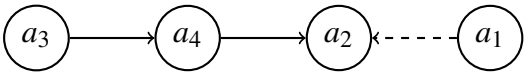


Figure 3.7: Argument graph for Example 29.

There are two preferred extensions: $E_1 = \{a,b,d\}$ *and* $E_2 = \{a,c\}$*. In* E_2*, rule* r_2 *holds and in* E_1 *it does not hold.*

After deactivating rule r_2 *with* $KB_2' = \mathsf{deactivate}(r_2, KB_2)$*, we get the argument graph shown in Figure 3.6 on the right. It contains all of* KB_2*'s arguments plus a new one, e with*

$$e = [;\Rightarrow \neg\langle r_2\rangle; \neg\langle r_2\rangle]$$

Argument e attacks c and thus makes rule r_2 *unusable in any accepted arguments. The preferred extension of* KB_2'*'s argument graph now conincides with its grounded extension and contains the arguments* a,b,d *and e.*

Example 29. *This example demonstrates how* deactivate *traverses all counter-arguments, counter-counter arguments and so forth for each user of* r_1 *(each argument that uses* r_1*). The original knowledge base (before applying* deactivate*) contained a defeasible rule with empty antecedent,* $\Rightarrow \mathsf{a}$*, and two rules with non-empty antecedents:*

$$r_1 = \mathsf{a} \Rightarrow \mathsf{b}$$
$$r_2 = \Rightarrow \neg\langle \Rightarrow \neg\langle r_1\rangle\rangle$$

If we compute $\mathsf{deactivate}(r_1, KB)$*, then a new rule* $r_3 = \Rightarrow \neg\langle r_1\rangle$ *will be added, disabling* r_1*. However, rule* r_2 *(equal to* $\Rightarrow \neg\langle r_3\rangle$*) which is already in the knowledge base now results in an argument that attacks all arguments which use* r_3*. This is*

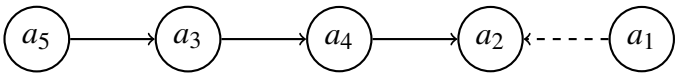


Figure 3.8: Argument graph for Example 29, part II.

why the definition of deactivate *contains a recursive clause. Figure 3.7 shows the resulting graph, after applying* deactivate(r_1, KB). *The arguments are* $a_1 = [\Rightarrow \mathtt{a}]$, $a_2 = [a_1; r_1; \mathtt{b}]$, $a_3 = [; \mathtt{r_2}; \neg\langle r_3\rangle]$ *and* $a_4 = [; r_3; \neg\langle r_1\rangle]$. *Argument* a_4 *was introduced by the* deactivate2 *operation, but it is defeated by* a_3. *This is the reason why* deactivate *needs to examine the resulting argument graph again.*

Therefore, deactivate(r_1, KB) *also contains the knowledge added by* deactivate(r_2, KB). *The final result of* deactivate(r_1, KB) *is the following set of arguments:*

$$a_1 = [\Rightarrow \mathtt{a}]$$
$$a_2 = [a_1; r_1; \mathtt{b}]$$
$$a_3 = [; \Rightarrow \neg\langle\Rightarrow \neg\langle r_1\rangle\rangle; \neg\langle\Rightarrow \neg\langle r_1\rangle\rangle] = [; r_2; \neg\langle r_3\rangle]$$
$$a_4 = [; \Rightarrow \neg\langle r_1\rangle; \neg\langle r_1\rangle] = [; r_3; \neg\langle r_1\rangle]$$
$$a_5 = [; \Rightarrow \neg\langle\Rightarrow \neg\langle\Rightarrow \neg\langle r_1\rangle\rangle\rangle; \neg\langle\Rightarrow \neg\langle\Rightarrow \neg\langle r_1\rangle\rangle\rangle]$$

And attacks $a_5 \to a_3 \to a_4 \to a_2$. *The grounded extension consists of* a_5, a_4 *and* a_1. *The graph is shown in Figure 3.8.*

We conclude this example with a detailed description of how deactivate(r_1, KB) *is computed. Let* $R' =$ deactivate$'(r_1,$ arguments$(KB))$, *so* deactivate$(r_1, KB) = KB \cup KB'$. *Figure 3.9 on page 89 shows the computation of* deactivate$'$ *step by step.*

3.4.2 Results on Deactivating Rules

3.4.2.1 deactivate Is Well-Defined

The first question we might like to answer about deactivate is whether it produces finite output for finite input. Proposition 16 shows that the number of rules added to the knowledge base by deactivate$'$ is indeed bounded, and that deactivate$'$ terminates whenever it is applied to a finite set of arguments.

Proposition 16. *For all sets of* ASPIC+*-arguments A and rules r, if A is finite, and* topRule(a) *is defined for all* $a \in A$, *then* deactivate$'(r, A)$ *is defined and finite.*

$$
\begin{aligned}
KB' &= \mathsf{deactivate}'(r_1, \mathsf{arguments}(KB)) \\
&\qquad \text{(Simplify)} \\
&= \mathsf{deactivate}'(r_1, \{a_1, a_2, a_3\}) \\
&\qquad \text{(Apply Definition 46; Let } r_3 \Longrightarrow \neg\langle r_1 \rangle \text{)} \\
&= \{r_3\} \cup \bigcup_{a \in \mathsf{attackers}([;r_3;\neg\langle r_1\rangle],\{a_1,a_2,a_3\})} \mathsf{deactivate}'(\mathsf{topRule}(a), \{a_1, a_2, a_3\} \setminus \{a\}) \\
&\qquad (\mathsf{attackers}([;r_3;\neg\langle r_1\rangle],\{a_1,a_2,a_3\}) = \{a_3\}) \\
&= \{r_3\} \cup \bigcup_{a \in \{a_3\}} \mathsf{deactivate}'(\mathsf{topRule}(a), \{a_1, a_2, a_3\} \setminus \{a\}) \\
&\qquad \text{(Simplify)} \\
&= \{r_3\} \cup \mathsf{deactivate}'(\mathsf{topRule}(a_3), \{a_1, a_2\}) \\
&\qquad \text{(Simplify)} \\
&= \{r_3\} \cup \mathsf{deactivate}'(r_2, \{a_1, a_2\}) \\
&\qquad \text{(Apply Definition 46; Let } r_4 \Longrightarrow \neg\langle r_2 \rangle) \\
&= \{r_3\} \cup \{r_4\} \cup \bigcup_{a \in \mathsf{attackers}([;r_4;\neg\langle r_2\rangle],\{a_1,a_2\})} \mathsf{deactivate}'(\mathsf{topRule}(a), \{a_1, a_2\} \setminus \{a\}) \\
&\qquad (\mathsf{attackers}([;r_4;\neg\langle r_2\rangle],\{a_1,a_2\}) = \emptyset) \\
&= \{r_3\} \cup \{r_4\}
\end{aligned}
$$

Figure 3.9: Application of deactivate′ for Example 29

Proof. In Definition 46, the recursive call to deactivate′ is with a set strictly smaller than the original set A (namely the set $A \setminus \{a\}$), so deactivate′ terminates. In each recursive step, exactly one rule is added (r'). □

3.4.2.2 The deactivate Operation Works as Intended

The key to the following results is that deactivated rules are attacked asymmetrically, by an argument with a premise-less defeasible rule. We first show that deactivate behaves as expected, that is, after applying $\mathsf{deactivate}(r, KB)$, there are no (credulously or sceptically) acceptable arguments a with $r \in \mathsf{rules}(a)$. For that we require a result about deactivate′ (Definition 46):

Proposition 17. *Let r be a rule and let A be a set of* ASPIC+ *arguments such that* $\mathsf{topRule}(a)$ *is defined for all $a \in A$. Let $KB = \mathsf{deactivate}'(r, A)$ and let $A' = \mathsf{arguments}(KB)$. For every argument $a \in A$, if $\mathsf{topRule}(a) = r$ then there exists an argument $b \in A'$ such that*

1. *b attacks a and*

2. *b is sceptically acceptable in* $\mathsf{argGraph}(A \cup A')$

Proof. We prove the claim by induction over the number of elements in A, the set of arguments. In case $|A| = 0$, $A = \emptyset$ so there are no arguments $a \in A$ with $\mathsf{topRule}(A) = r$. The actual induction proof starts with $|A| = 1$. Formally: Let A be a set of arguments. For every $n \geq 1$ and for every rule r: Let $A' = \mathsf{arguments}(\mathsf{deactivate}'(r, A))$. If $|A| = n$ then for every argument $a \in A$, if $\mathsf{topRule}(a) = r$, then there exists an argument $b' \in A'$ such that b' attacks a and b' is sceptically acceptable in $\mathsf{argGraph}(A \cup A')$.

Base case ($n = 1$) If $|A| = 1$ then A has exactly one argument, $A = \{a\}$. Let $s = \mathsf{topRule}(a)$. If $s \neq r$ then the claim holds trivially. Let $r' ==\Rightarrow \neg\langle r \rangle$, let $b = [; r'; \neg\langle r \rangle]$. By Definition 46, $b \in A'$. If $s = r$ then b attacks a, and a does not attack b, so the claim holds.

Induction step ($n = k + 1$) Let $a \in A$ such that $\mathsf{topRule}(a) = r$. Let $r' ==\Rightarrow \neg\langle r \rangle$, let $b = [; r'; \neg\langle r \rangle]$. By Definition 46, $b \in A'$, and b attacks a. It remains to show that b is sceptically acceptable in $\mathsf{argGraph}(A \cup A')$. Either there exists an argument $a' \in A$ such that a' attacks b or not. If no such a' exists, then b is sceptically acceptable and the claim holds. So we assume there is an $a' \in A$ such that a' attacks b. Then $a' \in \mathsf{attackers}(b, A)$, so by Definition 46, $\mathsf{deactivate}'(\mathsf{topRule}(a'), A \setminus \{a'\}) \subseteq \mathsf{deactivate}'(r, A)$. Let $A'' = \mathsf{arguments}(\mathsf{deactivate}'(\mathsf{topRule}(a'), A \setminus \{a'\}))$. By the induction hypothesis we know that there exists an argument $b' \in A''$ such that b' is sceptically acceptable in $\mathsf{argGraph}(A \cup A'')$. $b' \in A'$ by Definition 46. We now show that b' is sceptically acceptable in $\mathsf{argGraph}(A \cup A')$, which in turn means that b is sceptically acceptable in $\mathsf{argGraph}(A \cup A')$, since b' attacks a'. By Definition 46, $A' = A'' \cup \{a'\}$, so $A \cup A' = A \cup A'' \cup \{a'\}$ – in other words, the only difference between $\mathsf{argGraph}(A \cup A')$ and $\mathsf{argGraph}(A \cup A'')$ is the single argument a'. By Definition 46, $b' = [; \Rightarrow \neg\langle \mathsf{topRule}(a') \rangle; \neg\langle \mathsf{topRule}(a') \rangle]$, and because $\langle \cdot \rangle$ is injective (by Cond. 2 on page 33), a' does not attack b', so b' is sceptically acceptable in $\mathsf{argGraph}(A \cup A')$, so b is also sceptically acceptable in $\mathsf{argGraph}(A \cup A')$. □

The correctness proof of deactivate now follows directly from the correctness of deactivate′, established in Propage 17.

Proposition 18. *Let r be a rule and let KB be an* ASPIC+ *knowledge base . Let $E \in \mathscr{E} = \Sigma_{\mathsf{pr}}(\mathsf{argGraph}(\mathsf{deactivate}(r, KB)))$. Then, for all arguments $a \in E : \langle r \rangle \notin \mathsf{rules}(a)$*

Proof. By Definition 47, $\mathsf{deactivate}(r, KB) = KB \cup \mathsf{deactivate}'(r, \mathsf{arguments}(KB))$, so the claim follows from Propage 17. □

Deactivating a rule that is already deactivated has no effect (that is, deactivate is idempotent).

Proposition 19. *For all knowledge bases $KB = (\mathscr{R}, \mathscr{K})$ with $\mathscr{K} = \emptyset$ and for all rules r, the following equation holds:*

$$\mathsf{deactivate}(r, KB) = \mathsf{deactivate}(r, \mathsf{deactivate}(r, KB))$$

The proof of Propage 19 can be found on the next page (page 92).

3.4.2.3 The deactivate Operation is Reversible

Further, every deactivation can be reversed. By reverse we mean that its effects can be undone without deleting any knowledge – simply by adding knowledge, just like the deactivate operation itself. Before we define reactivate in Section 3.4.3, we will give one more result to show that an operation with the semantics of reactivate can be defined.

Consider this situation: An initial knowledge base KB_1 with a rule r, a second knowledge base $KB_2 = \mathsf{deactivate}(r, KB_1)$ and a third knowledge base KB_3 such that KB_3 contains KB_2 and $KB_3 = \mathsf{reactivate}(r, KB_2)$ – that is, r is reactivated in KB_3.

Any argument that used r in $G_1 = \mathsf{argGraph}(KB_1)$ should have the same status (acceptability) in $G_3 = \mathsf{argGraph}(KB_3)$. However, $G_1 \neq G_3$, because deactivating and reactivating r left a trace in form of arguments and attacks. The preferred extensions that existed in G_1 will become subsets of preferred extensions in G_3. Conversely, every preferred extension in G_3 subsumes a preferred extension from G_1.

We call this relation of subsets a linear extension (see Definition 22 on page 39). The following result shows that a linear extension of the preferred extensions of G_1 can always be obtained by adding knowledge to KB_2. In other words, the status-changing effect of deactivate, which resulted in KB_2, can be reversed without losing the additional knowledge generated by deactivate in KB_2.

Proof. Let KB be a knowledge base and let r be a rule.

$= \mathsf{deactivate}(r, \mathsf{deactivate}(r, KB))$

(Definition 47, resolving the inner term $\mathsf{deactivate}(r, KB)$)

$= \mathsf{deactivate}(r, KB \cup \mathsf{deactivate}'(r, \mathsf{arguments}(KB)))$

(Definition 46; Let $r' =\!\Rightarrow \neg\langle r\rangle$; Let $b = [; r'; \neg\langle r\rangle]$; Let $A = \mathsf{arguments}(KB)$)

$= \mathsf{deactivate}(r, KB \cup \{r'\} \cup \bigcup_{a \in \mathsf{attackers}(b,A)} \mathsf{deactivate}'(\mathsf{topRule}(a), A \setminus \{a\}))$

(Let $KB' = \bigcup_{a \in \mathsf{attackers}(b,A)} \mathsf{deactivate}'(\mathsf{topRule}(a), A \setminus \{a\})$)

$= \mathsf{deactivate}(r, KB \cup \{r'\} \cup KB')$

(Definition 47)

$= KB \cup \{r'\} \cup KB' \cup \mathsf{deactivate}'(r, \mathsf{arguments}(KB \cup \{r'\} \cup KB'))$

(Let $A' = \mathsf{arguments}(KB \cup \{r'\} \cup KB')$)

$= KB \cup \{r'\} \cup KB' \cup \mathsf{deactivate}'(r, A')$

(Definition 46)

$= KB \cup \{r'\} \cup KB' \cup \{r'\} \cup \bigcup_{a \in \mathsf{attackers}(b,A')} \mathsf{deactivate}'(\mathsf{topRule}(a), A' \setminus \{a\})$

(Definition 46; $\mathsf{attackers}(b, A') = \mathsf{attackers}(b, A)$,
because $\mathsf{attackers}(b, \mathsf{arguments}(\{r'\} \cup KB')) = \emptyset$ and by Propage 3)

$= KB \cup \{r'\} \cup KB' \cup \{r'\} \cup \bigcup_{a \in \mathsf{attackers}(b,A)} \mathsf{deactivate}'(\mathsf{topRule}(a), A \setminus \{a\})$

(Definition of KB')

$= KB \cup \{r'\} \cup KB' \cup \{r'\} \cup KB'$

(Simplification)

$= KB \cup \{r'\} \cup KB'$

(Definition of KB')

$= KB \cup \{r'\} \cup \bigcup_{a \in \mathsf{attackers}(b,A)} \mathsf{deactivate}'(\mathsf{topRule}(a), A \setminus \{a\})$

(Definition 46)

$= KB \cup \mathsf{deactivate}'(r, A)$

(Definition 47)

$= \mathsf{deactivate}(r, KB)$

□

Figure 3.10: Proof of Propage 19

Proposition 20. *Let r be a rule and let KB be an* ASPIC+ *knowledge base. Let $KB' = \mathsf{deactivate}(r, KB)$. Then there exists a knowledge base KB'' such that $KB' \subseteq KB''$ and $\Sigma_{\mathsf{pr}}(\mathsf{argGraph}(KB''))$ is a linear extension of $\Sigma_{\mathsf{pr}}(\mathsf{argGraph}(KB))$*

Proof. (Sketch) The reversal operation is similar to the original deactivate, except that this time the rules r' are the ones that are deactivated. □

The definition of deactivate (Definition 47) has the advantage that it can easily be reversed by adding even more formulae to the knowledge base. Instead of deactivating the users and defenders of r, we simply have to deactivate the rule that deactivates r.

Definition 48. *Let r be a rule and let KB be an* ASPIC+ *knowledge base.*

$$\mathsf{reactivate}(r, KB) = \mathsf{deactivate}(\Rightarrow \neg\langle r\rangle, KB)$$

Example 30. *Recall the knowledge base KB' from Example 28 on page 86: $KB' = (\{\Rightarrow \mathsf{a}, r_1, r_2, r_3, r_4\}, \emptyset)$ with*

$$r_1 = \mathsf{a} \Rightarrow \mathsf{b} \qquad r_2 = \mathsf{a} \Rightarrow \neg\mathsf{b}$$
$$r_3 = \mathsf{b} \Rightarrow \mathsf{c} \qquad r_4 = \Rightarrow \neg\langle r_2\rangle$$

Rule r_4 is the result of deactivating r_2. We can reactivate r_2 as follows:

$$\begin{aligned} KB'' = \mathsf{reactivate}(r_2, KB') &= \mathsf{deactivate}(\Rightarrow \neg\langle r_2\rangle, KB') \\ &\quad (\Rightarrow \neg\langle r_2\rangle = r_4) \\ &= \mathsf{deactivate}(r_4, KB') \\ &\quad (\textit{Apply Definition 47}) \\ &= KB' \cup \{\Rightarrow \neg\langle r_4\rangle\} \end{aligned}$$

The resulting argument graph is shown in Figure 3.11. There are two preferred extensions, $E_1 = \{f, c, a\}$ and $E_2 = \{f, b, d, a\}$. In the original graph based on KB, before r_2 was deactivated and reactivated, there were two preferred extensions: $\{a, c\}$ (subset of E_1) and $\{a, b, d\}$ (subset of E_2). Since every preferred extension in the original graph is subsumed by a preferred extension in KB'', and likewise every preferred

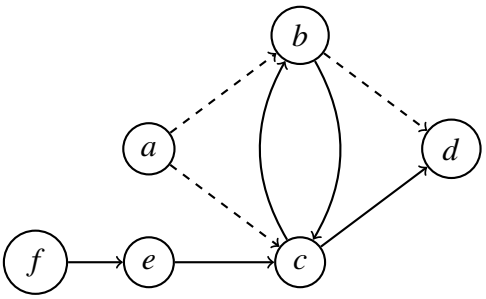


Figure 3.11: Argument graph for reactivation, see Example 30

extension in KB'' subsumes a preferred extension of the original graph, it holds that $\Sigma_{\mathsf{pr}}(\mathsf{argGraph}(KB''))$ *is a linear extension of* $\Sigma_{\mathsf{pr}}(\mathsf{argGraph}(KB))$*, as per Propage 20.*

The argument of deactivate in Definition 48 is exactly the rule that was added by deactivate before. An application of $\mathsf{reactivate}(r, KB)$ will result in (at least) one new rule

$$r' = \ \Rightarrow \neg\langle\Rightarrow \neg\langle r\rangle\rangle$$

The new rule r' is interesting, because it looks like a double negation of (the applicability of) r. Its meaning however is different from double negation in classical logic, because from the inclusion of r' in a knowledge base we cannot infer that r is available (i.e. $r \in \mathscr{R}$).

3.4.3 Enforcing a Point of View

Before we end this section we will enhance the definition of reactivate (Definition 48) to define an enforce operation - a function that takes a set of arguments A and disables their attackers, so that A becomes a subset of the grounded extension (if there are no attackers of A, it is already a subset of the grounded extension).

Definition 49 (Enforce)**.** *Let KB be an* ASPIC+ *knowledge base, let* $A = \mathsf{args}(KB)$ *and let* $B \subseteq A$*.*

$$\mathsf{enforce}(B, KB) = KB \cup \bigsqcup_{a \in \mathsf{attackers}(B,A)} \mathsf{deactivate}(\mathsf{topRule}(a), KB)$$

Proposition 21. *For all* ASPIC+ *knowledge bases KB and for all* $A \subseteq \mathsf{args}(KB)$*,* $KB \sqsubseteq \mathsf{enforce}(A, KB)$

Proof. (Sketch) deactivate (Definition 47) only uses the set union operation $\cup$ and

enforce (Definition 49) is the union of a repeated application of deactivate, so the original knowledge base KB is always part of the result $\mathsf{enforce}(A, KB)$. □

Furthermore, if the set of arguments A is conflict-free, then A will be a subset of the grounded extension after applying $\mathsf{enforce}(A, KB)$. This is the defining property of enforce.

Proposition 22. *For all* ASPIC+ *knowledge bases KB and for all* $B \subseteq \mathsf{args}(KB)$*, if* $\mathsf{attacks}(B) = \emptyset$ *then* $B \subseteq \Sigma_{\mathsf{gr}}(\mathsf{argGraph}(\mathsf{enforce}(B, KB)))$

Proof. Let KB be an ASPIC+ knowledge base, let $(A, Att) = \mathsf{argGraph}(KB)$ and let $B \subseteq \mathsf{arguments}(KB)$ such that B is conflict free, that is $B \times B \cap Att = \emptyset$. Let $KB' = \mathsf{enforce}(B, KB)$ and let $(A', Att') = \mathsf{argGraph}(KB')$. Let $\{E\} = \Sigma_{\mathsf{gr}}(A', Att')$.

Proof by contradiction. Assume there is an argument $a \in B$ such that $a \notin E$. Then there exists an argument $b \in A'$ such that $(b, a) \in \mathscr{R}'$ such that there is no $c \in E$ with $(c, b) \in \mathscr{R}'$ (i.e. b is not attacked by the grounded extension). Since $b \in \mathsf{attackers}(B)$ and $\mathscr{K} = \emptyset$, we know $b = [\ldots; r; c]$ for some rule r.

Then $\mathsf{deactivate}(\mathsf{topRule}(b), KB) = \mathsf{deactivate}(r, KB) \subseteq KB'$. By Propage 17 there is an argument $c \in \mathsf{arguments}(KB')$ such that c attacks b and c is sceptically acceptable. This contradicts the assumption that b is not attacked by an argument in the grounded extension, so the claim holds. □

The function $\mathsf{enforce}(A, KB)$ is a convenient way of "promoting" a conflict-free set of arguments to be sceptically acceptable by adding rules to an ASPIC+ knowledge base.

3.4.4 Accepting a Decision

In this section we demonstrate how enforce can be used to "make a decision" – more specifically, to update a knowledge base after a decision has been made by the design team. This is achieved by making the selected option and its consequences part of the grounded extension.

Accepting a decision involves two steps. First, after choosing an option O, we need to determine which of the possible worlds in $\mathsf{possibleWorlds}(D, O)$ best models the actual circumstances. This means we have to select a preferred extension $E \in \Sigma_{\mathsf{pr}}(\mathsf{args}_D(O), \mathsf{attacks}(\mathsf{args}_D(O)))$. Then, we use the enforce operation (Definition 49)

to extend D's knowledge base so that it contains the knowledge to make all arguments in E sceptically acceptable.

Definition 50 (Making a Decision)**.** *Let $D = (K, C, R)$ be a decision frame, let $O \in \mathcal{O}$ be an option, and let $E \in \Sigma_{\mathsf{pr}}(\mathsf{args}_D(O), \mathsf{attacks}(\mathsf{args}_D(O)))$.*

$$\mathsf{decide}_D(O, E) = \mathsf{enforce}(E, K \cup O)$$

Definition 50 produces a new knowledge base K'. Because of the way deactivate is defined in Definition 47, $K \subseteq K'$, and because of Propage 22, the chosen set of arguments E is part of the grounded extension of the result.

Proposition 23. *Let $D = (K, C, R)$ be a decision frame, let $O \in \mathcal{O}$ be an option, and let $E \in \Sigma_{\mathsf{pr}}(\mathsf{args}_D(O), \mathsf{attacks}(\mathsf{args}_D(O)))$. Let $KB' = \mathsf{enforce}(E, K \cup O)$ and let $\{E'\} = \Sigma_{\mathsf{gr}}(\mathsf{argGraph}(KB'))$. Then $E \subseteq E'$.*

Proof. Since E is a preferred extension, E is conflict-free, so by Propage 22, E is a subset of the grounded extension of KB', so the claim holds. □

Example 31. *Assume that we decide to go with option O_1 from Example 17 - that is, with $O_1 = \{\mathtt{mat(bb, s-t)}, \neg\mathtt{shim(t)}, \mathtt{varnish(t, 2)}\}$. We also learned that out of the two possible worlds E_1 and E_2, the first one was correct - so the oversized bolts do not damage the structure (through microscopic fractures). We want to adjust the knowledge base C to reflect our decision, and the consequences it had. Recall that the arguments are $E_1 = \{a_1, a_3, a_5, a_6, a_7, a_8, a_9\}$ as shown in Table 3.7 below. We therefore compute*

$$\begin{aligned}\mathsf{decide}_D(O_1, E_1) &= \mathsf{enforce}(E_1, K \cup O_1) \\ &= K \cup \{\Rightarrow \neg\langle r_2\rangle, \Rightarrow \neg\langle r_4\rangle\}\end{aligned}$$

The two added rules $\Rightarrow \neg\langle r_2\rangle$ and $\Rightarrow \neg\langle r_4\rangle$ effectively deactivate the second possible world E_2. We can now use this knowledge base in any subsequent decisions, allowing us to re-use both the justifications for choosing O_1 and the knowledge that of all possible worlds it was E_1 which was correct.

The example was based on the assumption that we learned which of the possible worlds was the real one. This allowed us to choose one of the preferred extensions and

enforce it, as in the definition of decide*. However, even if we had had this information, we could have used* decide *to enforce the arguments of the grounded extension to the knowledge base.*

3.5 Related Work

Decision making with argumentation has been the focus of much recent work (see Carstens *et al.* [69] for a survey). The existing work on argument-based decision making primarily covers multi-criteria decision analysis. The literature can be divided into two parts: Some approaches [70, 23, 31] start with an existing (multi criteria) decision making formalism and translate it into an argumentation system. The second category is work that extends an argumentation system with notions of options and arguments for and against those options (e.g. [33, 28, 29]), often involving a representation result with existing decision making formalisms.

Our work falls into the second category. While we base our model on an analysis of MCDM and DM, the data (input) of a concrete decision problem is not required to be formulated as an MCDM or DM problem. We also have two representation results, one for the DM case (Propage 7) and one for the MCDM case (Theorem 1), but in contrast with the existing work it contains a novel decision rule based on argumentation-specific properties (smaxmin), making it strictly more powerful than approaches that achieve exactly the same results as the non-argumentative systems they are based on. In addition, we address additional requirements related to decision processes: Re-usability of formal models, and the ability to automatically determine the consequences a decision has on subsequent decisions. This is the content of the next chapter. In the following we will review some existing work in detail.

Table 3.7: Arguments to be enforced in Example 31

Name	Content
a_1	$[a_8, a_3; r_1; \texttt{balanced(t)}]$
a_3	$[a_5; r_3; \neg\texttt{damaged(t)}]$
a_5	$[; r_5; \neg\texttt{microscopic\_fractures}]$
a_6	$[a_7; r_6; \texttt{bolts\_strong}]$
a_7	$[\texttt{mat(bb, s - t)}]$
a_8	$[\neg\texttt{shim(t)}]$
a_9	$[\texttt{varnish(t, 2)}]$

3.5.1 Multi-Criteria Decision Making

3.5.1.1 Decision Making With ABA

The ABA argumentation system has been used as the foundation for argument-based decision making in several papers by Matt *et al.* [31] as well as Fan, Toni *et al.* [24, 70, 30].

In [70], decisions are modeled using structured arguments in the ABA system [47]. Here the argumentation formalism is used only as an interpretation of the actual decision model, which consists of a set of decisions, a set of goals and two tables relating decisions to goals. The proposal takes up ideas from earlier work on decision making with ABA [31]. It is extended to cover preferences over sets of goals (as opposed to just individual goals) and applied to an extensive case study from the medical domain in [24]. In [30], a similar ABA-based decision framework is used for collaborative decision making with two agents, each with a set of goals and decisions.

The paper [31] uses ABA for multi-criteria decision making. Three decision rules are discussed: (Weak) dominance, degree of admissibility, and relative value of a decision. Dominance is the usual weak dominance principle where option A dominates option B if every one of the goals provided by B is also provided by A. A multi-criteria decision making system is given and translated to ABA in a way that dominating decisions result in admissible arguments.

Degree of admissibility takes into account how many of an argument's incoming attacks are symmetric. The ratio of symmetric attacks within all attacks is called degree of admissibility. Since dominant decisions are admissible and admissible decisions always have the highest degree of admissibilty (namely 1), this second decision rule increases the decisiveness for non-dominant options.

The last decision rule, relative value, combines degrees of admissibility with numerical weights of goals. Such weights can be used to distinguish goals and play a role similar to the ranking of consequences $\leq_C$ in our framework.

The argumentation formalism of [31] is ABA and the decision problems it solves are for MDCM only. These differences notwithstanding, the second decision rule it introduces (degree of admissibility) seems to be promising for our approach too, as it potentially allows for a greater decisiveness than the traditional dominance principle in

multi-criteria decision making. The extension of our framework to handle multi-criteria decision making alongside decision making with uncertainty is a subject of future work.

Comparing our system with the more recent work by Fan *et al.* [70], their decision rules are defined on the table-based decision model only,[4] which means that the expressiveness gained from using an argumentation formalism is not put to use in the decision making. Therefore the value added by the argumentation system is that decisions are explained by arguments. However, those arguments always have the same structure because of the translation into ABA. The use of argumentation does not enable any decision rules in addition to the ones defined on the tabular model.

3.5.1.2 Other Approaches

Dimopoulos [29] covers multi-criteria decision making with argumentation, introducing a new decision rule called *regime*. Under this rule, an option A_1 is preferred over an option A_2 if the criteria in which A_1 outranks A_2 are more important than those in which A_2 outranks A_1. The regime rule is applied to decision making with abstract arguments.

Applying the regime rule to our decision making framework is left for future work. At first sight, the rule seems more interesting than, for example, strict and weak domination, because it allows to make more comparisons (i.e. it is more decisive).

An interesting approach to argumentation in decision making is presented by Visser [23]. An existing formalism for qualitative multi-criteria decision making, QPS (Qualitative Preference Systems) is translated to an argumentation language. The aim of a QPS is to determine preferences between outcomes, using a combination of primitive criteria and compound criteria. The latter are divided into cardinality criteria (preferring an outcome if it meets a larger number of criteria) and lexicographic criteria (preferring an outcome if it meets a single more important criterion).

The argumentation language in which QPS are modeled has a fixed set of primitives and inference rules. The primitives can be used to describe the QPS (for example, to say that $\mathtt{c}$ is a cardinality criterion with subcriteria $\mathtt{c}_1, \mathtt{c}_2$). The inference rules codify how preferences are derived in QPS. Arguments about preferences are built from an object language and inference rules and evaluated using dialectical trees. The correspondence between preferences in QPS and preferences in the argumentation system

[4] In fact the translation of the table-based model into ABA depends on the choice of decision rule

is proved.

The second part of the paper [23] concerns reasoning with background knowledge. It adds domain-specific rules to the object language and a defeasible modus ponens inference rule to the meta language.

To compare the work of [23] with our own work, we first discuss the differences in the argumentation formalisms. Our proposal uses ASPIC+ whereas [23] uses a custom language for expressing preferences. The two works also have different semantics as we use Dung's argument graphs whereas [23] uses dialectical trees.

Further differences between the two proposals can be found on a more conceptual level. [23] is a multi-criteria approach whose input is a set of preferences over outcomes and a hierarchy of criteria, from which a larger set of preferences over outcomes is derived. We on the other hand propose a system for decision making with uncertainty that takes as input an ordered set of consequences and some background knowledge describing how those consequences may come about, and produces as output a ranking of options. The focus of [23] is on how to produce the ordered set of consequences, which we assume to exist already.

The approach by Baroni *et al.* [71, 72] is closely related to our proposal because it covers the same domain, namely engineering debates. In [72] a new argumentation system is developed which combines bipolar argumentation with the IBIS framework for design documentation. Their method assigns a numeric score to each option by using a scoring function, a concept similar to the aggregate function we saw in multi-criteria decision making. The aggregate function proposed in [72] can potentially be applied to our own work as well. Conversely, our approach extends to decision making with uncertainty (about the state of the world), which is not covered by theirs.

Evripidou *et al.* [73] describe a system for collaborative decision support in design. The system combines argumentation theory with case based reasoning and ontology techniques to provide a comprehensive platform for analysing debates of individual issues and retrieving relevant information from past decisions.

3.5.2 Decision Making With Uncertainty

Decision making with uncertainty in the context of abstract argumentation has been studied in several papers by Amgoud and Prade [33] and Amgoud *et al.* [28].

Besides the fact that we require a logic-based argumentation framework – in order to be able to make inferences (see page 42 in Chapter 2) – there are several differences in the interpretation of possible worlds and how they relate to the acceptability of arguments, which we are going to explain in detail.

In [28], abstract argument graphs are used for a decision making system with uncertainty. In their system, there are two distinct sets of arguments: practical arguments and epistemic arguments. Epistemic arguments represent the possibly inconsistent knowledge from which practical arguments for options are made upage All practical arguments are in a positive relation to options, that is, they are arguments pro but never arguments con. Practical arguments cannot support more than one option, which is reflected in the attacks relation between practical arguments: If two practical arguments support two different options then they attack each other. This leads to an argumentation system that (in its practical arguments) has symmetric attacks only.

Every option is assigned a status of sceptical, universal, argued, credulous, rejected or non-supported. This status is determined by the status of practical arguments in favour of that option. The status of practical arguments in turn is determined by the number and kind of extensions an argument belongs to. For example, a sceptically accepted argument belongs to every preferred extension. This implies that a system with symmetric attacks only does not have any sceptically accepted arguments (unless the attacks relation is empty).

Attacks are turned into defeats using a preference relation over arguments, which results in the resolution of symmetric attacks unless the two arguments involved cannot be compared. The preference relation has great impact on the extensions of the argument graph and thus on the ranking of options. In spite of its importance, the preference relation is not formally specified. In the examples, preferences between arguments are derived from (a) preferences of goals and (b) likelihood of states of the world, but this is done *ad hoc* (as evidenced by the fact that neither states of the world nor goals play a role in the formal definitions).

The approach [33] deals with uncertainty and preferred extensions. It does so by using preferred semantics (unless stable extensions exist) and defines arguments to be acceptable if they are in every extension. However, this is problematic if a mutual attack cannot be resolved. Consider the classical “umbrella” example: It may rain or it

may not rain with equal likelihood and we have to decide whether to take an umbrella. Apart from the two goals "staying dry" and "not carrying extra weight" we want to look respectable because we are going to a funeral. Unfortunately we only have a colourful children's umbrella. Now there are two arguments for leaving the umbrella and only one for taking it, so intuitively we would decide against taking the umbrella. If we modeled the decision using the system of [33], both options would have the same number of arguments pro: zero. This is because neither argument pro is in every preferred extension.

Of course one can usually determine whether it is more likely to rain or not, but in complex decision problems the probabilities or even relative likelihoods are often impossible to assign. In that case it is common to assume equal probabilities for all outcomes, a principle known as "tallying" (which we applied in Section 3.3.5 on decision rules).

In order to represent multiple possible worlds in the system of [33], they would have to be made part of the options. This has two disadvantages: There is a growth of the number of options (e.g. two possible actions to choose from and two possible states of the world result in four options) and, since arguments are for or against options, they can now be for or against states of the world, conflating the original problem of decision making with the tangential problem of determining the state of the world. In contrast, our system uses preferred extensions to separate states of the world from options.

By delegating the definition of this preference relation, the system of [28] therefore fails to answer the question of how goals, knowledge and possible worlds influence the ranking of options.

In summary, the proposal [28] differs from our approach in the following ways. (1) We instantiate arguments with knowledge whereas they use abstract arguments. (2) We explicitly consider consequences (goals) and have a formal notion of states of the world. (3) The ranking of options in our system is ultimately determined by the knowledge available and by the ranking of consequences (goals), whereas in [28] it is determined by a preference relation over arguments that has to be supplied as the input.

3.5.3 Qualitative Decision Theory

Qualitative decision theory is not concerned with formal methods of argumentation, but we review it here briefly because it highlights some of the limitiations of the non-numeric approach to decision making under uncertainty. In particular we review the work of Didier Dubois *et al.* on qualitative decision theory ([74, 54, 75]).

In Dubois *et al.* 2002 [54], a purely symbolic approach to decision making under uncertainty is investigated. Symbolic in this context means that neither the probabilities of possible worlds, nor the decision maker's preferences are quantified (our approach falls into the same category). The article compares the symbolic approach with the probabilistic approach to decision making under uncertainty, and comes to the rather negative conclusion that "purely symbolic approaches to both rational and practically useful criteria for decision making under uncertainty have serious limitations in terms of expressive power" [54, page 483], because the necessary rationality conditions impose great limitations on the likelihood ordering of possible worlds.

In a subsequent journal article [75] Dubois *et al.* extend the qualitative decision making framework introduced in [54] to a full-fledged axiomatic approach to qualitative decision theory with preference relations and comparative uncertainty.

It is clear from [54, 75] that a purely symbolic "version" of Savage's [10] decision theory is either very limited in the models it admits, or produces undesirable output (i.e. with an intransitive ordering of actions). The natural question to ask now is whether our own decision making framework is open to similar criticism. We can answer it by comparing our approach to qualitative decision theory in Dubois' sense in more detail. The decision framework in [54] tries to stay as close as possible to Savage's original definitions by replacing the requirement for quantified probabilities and preferences with a (weaker) requirement for orderings on both dimensions. As a result, there is a two-tiered model of probabilities, where possible worlds are sets of atomic *events*, and it is the ordering of events that determines the ordering of possible worlds (possibly resulting in cycles such as $a > b > c > a$). In our system we adopt a different notion of possible world: It is the set of preferred extensions of an argument graph G (see Definition 32). Possible worlds in our work are therefore an epistemic concept, based purely on the acceptability of sets of arguments. Further, we do not assume an ordering of possible worlds - the only place where we make assumptions about the elements of

a possible world is in Section 3.3.5.1 where the number of possible worlds in which an argument is acceptable is taken as a measure of the strength of that argument. By removing the need to partially order events (as the constituent parts of possible worlds in Dubois' sense) we avoid the problem of circularities in the ordering of actions, because we do not need to combine the two scales, events and preferences. In probabilistic decision theory, this "combination" of the two scales is achieved through multiplication, resulting in expected utilities. In qualitative decision theory, the "combination" is exactly what causes the inconsistencies mentioned above.

3.5.4 Enforcement

The issue of enforcement has been addressed both on the level of abstract argumentation (e.g. [65]) and in terms of concrete argumentation systems [76].

Čyras and Toni [76] address enforcement indirectly, as part of an investigation of non-monotonic inference properties in ABA. They define two properties – cumulative transitivity and cautious monotonicity – and study them under different semantics in ABA. The paper then deals with confirmation of assumptions, which means deleting an assumption ψ from the set of assumptions and adding a defeasible rule with conclusion ψ instead. The difference to our work is that we are interested in purely additive information changes, i.e. achieving status change without the deletion of arguments. On the other hand, we only consider two semantics (grounded and preferred), and did not study cumulative transitivity and cautious monotonicity in our setting.

3.5.4.1 Argumentation-Based Belief Revision

Enforcement is also related to belief revision. Within argumentation, belief revision has been addressed for abstract argumentation [67, 77, 78], agent-based argumentation [79, 80], probabilistic argumentation with DeLP [81, 82] and possibilistic argumentation with fuzzy labels [83].

Belief revision in abstract argumentation is achieved by modifying the set of arguments or the set of attacks of an abstract argument graph. Since the arguments in abstract graphs are devoid of content and their only interaction is captured in the attacks relation, one can add arguments and attacks as required, and the challenge for belief revision approaches is to find the best (usually minimal in some sense) of all possible changes that achieve the desired goal.

In our approach, we work with structured arguments in the ASPIC+ framework. The graph of arguments and attacks is a product of the knowledge base, and so any changes to it have to be expressed as changes to the knowledge base. This is why we imposed a number of constraints on the ASPIC+ systems used in this thesis (see Section 2.3.2 on page 32). Our contribution to the field is therefore to establish the parameters that allow enforcement in ASPIC+.

3.5.4.2 Dynamics of Argumentation Systems

Our development of enforce and related operations was driven by the requirement to change the status of a set of arguments from credulously accptable to sceptically acceptable. The change of status of an argument after updating an argumentation system has been researched under the term argument dynamics [84], or dynamic argumentation frameworks [85]. Similar to the work on belief revision discussed above (a lot of builds upon argument dynamics), these approaches focus abstract argumentation or DeLP [86, 87]. While our work is aimed at (a subset of) the ASPIC+ system, it would be interesting to compare it with prior work on DeLP and ultimately position it in the framework of belief revision, similar to the approach taken in [81].

3.6 Conclusion

We will now discuss how ADF meets the five requirements that were set in section 3.2.4, and then review how the developments of this chapter prepare us to analyse decision processes, sequences of decisions, in the next chapter.

3.6.1 Discussion

In Section 3.2.4 we laid out five shortcomings of the traditional, non-argumentation based approach to decision making in light of knowledge reusability. Our decision framework ADF addresses these five points:

1, Opaque resoning process Preferred decisions in ADF are backed by arguments. Those arguments are part of grounded extensions of an argument graph generated from a knowledge base and are therefore based on a formal model of the domain. This model is part of the ADF, which means outside of that knowledge base, no additional information is needed to reproduce the reasoning process.

2, Local optimum With the notion of recommended decisions in ADF, a decision maker does not have to enumerate the possible decisions manually. Rather, they are the decisions are defined by the knowledge base. The problem of identifying possible decisions has thus been replaced by the problem of modeling the domain.

3, Proprietary documentation formats Since ADF relies on a formal language with an inference mechanism, it may easily be transformed to any logic-based model of knowledge, for example an ontology. Ontologies, which are based on description logics, are a widely used method of recording domain knowledge. Conversely, domain knowledge from an ontology can be transformed into ADF rules. Williams and Hunter [25] describe how knowledge from an ontology can be used in the argumentation process.

4, Manual analysis With the domain knowledge represented as rules used in argumentation, automated decision analysis is possible as discussed in Sections 3.3.3 and 3.3.4.

5, Costly retrieval of documentation For ADF systems, the problem of re-evaluating previous decisions depends mainly on the data structure chosen for the knowledge base, because it is sufficient to retrieve the relevant rules and assumptions from previous decisions. It is not necessary to perform the entire analysis again.

In Section 3.2.3, we presented decision reusability and claims management as two use cases for documentation in the aerospace industry. Both require a formal, structured documentation of decisions. If an argumentation based system is used for decision making, then the documentation could be generated by the same system with no additional cost, since the arguments used by the system to evaluate decisions are, at the same time, formal justifications of decisions. Because they are structured (as determined by the contents of the knowledge base), they can easily be converted into a structured format for documentation, for example an ontology.

In this chapter we developed an argumentation-based model of decision making that combines decision making with uncertainty with multi-criteria decision making. We now have the building blocks needed to model sequences of decisions: We can evaluate options according to how well they meet our criteria under scenarios, and we can adjust our knowledge base after making a decision.

3.6.2 Future Work

The interplay between the logic based on which arguments are formed, and the acceptability of arguments – as determined by argument graph semantics – looks to be a promising area for future work in argumentation in general [88], and this is also the case for our application of the theory in the domain of decision making.

We see opportunities for future work mainly in the area of decision making with uncertainty (Section 3.3.4). First, we want to extend our notion of possible worlds (Definition 32) with a degree of probability - whether qualitative as in the work of Dubois [75], or quantitative, using classical probability theory or one of the weaker alternatives proposed in the AI literature. Since possible worlds in our understanding are the same as preferred extensions and thus sets of arguments, assigning probabilities to them may create an interesting dynamic with the underlying knowledge base. This work can build on recent proposals for probabilistic argumentation (e.g. [60, 61]).

A second and related concern is the relationship between probabilities of possible worlds, and the utilities of goals. In the purely symbolic setting of [54], this relationship was the source of some problems related to the consistency of the system's output. It would be interesting to investigate if the fact that our decision problems are generated from a knowledge base allows us to solve some of those problems, based on the fact that arguments (and extensions) are internally consistent.

In the area of multi-criteria decision making (Section 3.3.3), we want to improve the definition of $\mathtt{sat}_D$ (Definition 29), the function determining how many criteria are met. In this thesis, sat simply returns a set of criteria that are achieved by a given option, and does not distinguish the criteria any more. Therefore, the notion of dominated options (Definition 30) can only be applied if an option O satisfies *all* of another option O's criteria, which seems to be relatively rare in practice. We want to consider additional means of summarising the outcome of an option, similar to the way an aggregation function *agg* can be defined in the traditional approach to MCDM (Definition 23).

In the area of enforcement and deactivation of rules, we are planning to compare our proposal with the work on enforcement in abstract argumentation [65, 66, 67, 68], and to position them within the wider field of belief revision.

Chapter 4

Argument-based Decision Process

4.1 Introduction

In Chapter 3 we developed an argumentation-based model for decision making, taking into account the need to balance several criteria as well as uncertainty about the outcome of options. In our framework we also have the ability to adjust a knowledge base once a decision has been made, so that knowledge which used to be only one of several options becomes part of the "safe" core of the knowledge base (the grounded extension).

Let us now take a step back and take another look at our goals set out in the introduction. We want to describe not just individual design decisions, but entire processes consisting of many decisions.

A lot of uncertainty exists in design processes [6], not only on the level of individual decisions (which we considered in Chapter 3), but also on the process level – for example, it is not clear what requirements will exist at the end of the process, because they change frequently. For example, the design of a new car should combine new technological advancements (efficient engines, computer-assisted driving and so on) with the aesthetic language that has already been established as part of the car's brand. It also has to meet a target price and profit margin. It is common for decisions to be reconsidered at later stages, when more knowledge has become available. In the car example, new knowledge could take the form of changed commodity prices or tax breaks in some countries.

In this chapter we will extend our decision model by looking at decisions in context, taking decision frames (see Chapter 3) as the building blocks of decision pro-

cesses. We view decision processes as sequences of decisions that are related by a common goal, or set of requirements, that they work towards fulfilling. When recording the results of decisions (which options were chosen) we therefore record not only the actual choices that were made, but also the justifications for making them, in the form of arguments. In argumentation-theoretic terms, we record the preferred extension of a decision frame. As a consequence, every decision outcome on its own is conflict-free. By combining the knowledge from two decision outcomes we may get conflicts that tell us if the outcomes are incompatible.

The main contribution of this chapter is an argumentation-based model of decision sequences based on a simple notion of decision outcome and designed to be compatible with argument extraction techniques from natural language processing (see. Section 4.5.1 for details). The chapter is organised as follows. We begin by introducing a use case that serves as a guide through the definitions and results. In Section 4.3, we develop the model of decision processes and their properties. In Section 4.4 we analyse the impact of changing a decision, by choosing a different option. Section 4.5 describes a graph-based interpretation of decision sequences, which is useful for displaying them graphically. We conclude with a short review of related work and a discussion (Section 4.6).

4.2 Use Case

In this section we describe the use case that motivated our work on decision processes, expanding on the design problem introduced in Chapter 3, page 52. While the details of the design problem are fictional, its general structure was developed as part of my work on project DEEPFLOW. During this project, my colleagues at SAP and I created the use case with the help of aerospace engineers at Bombardier Belfast and several resources on engineering design [2, 5, 1]. As a consequence, the structures of this use case and the issues it highlights are typical of large engineering projects, even though its scope is small.

The task we are modeling is to design a component that is used in the wing of an airplane. The design process consists of two stages, initial design and detailed design. The choices made in the initial design stage influence the options available in the detailed design stage.

In the examples, the detailed design stage will have constraints that are unsolvable, so the preliminary design has to be changed. This illustrates how our model reflects the non-linear nature of design processes.

4.2.1 Overview

Two metal components are to be joined by a connecting piece, forming a T-shape. The connector itself is L-shaped and will be used on either side of the T (see Figure 3.1 on page 52). It has two or more holes for attaching it to the two components with bolts or screws. We have to choose a material for the connector and decide how to attach it to the components (size and number of holes, screws or bolts, whether to insert a protective layer between the connector and each component).

Our design has to meet some requirements. In order to function in adverse environments, the connector's mechanical properties (tensile strength etc.) should change as little as possible even in low temperatures. It should withstand corrosion and strong mechanical pressure exercised by either of the two components. In addition to these external (imposed by the customer) requirements, there is an internal requirement that the connector can be produced with the existing machinery in our factory. Finally there are three dimensions in which the design should be optimised: Cost (low), production time (short) and weight (low).

4.2.2 Stage 1: Preliminary Design

In the first stage, the material of the connecting piece has to be chosen. The dimensions of the connector are determined by the two components it connects and cannot be changed. Components like the one we are designing are usually made from a material such as aluminium, aluminium-based alloys, or composites. Each material is represented by a literal (`al`, `alloys`, `composites`).

The requirements are expressed as a set of literals, too. The component should be `strong` and `corrosion_resistant`.

We use defeasible rules to model our knowledge about the pros and cons of each option. Aluminium is cheap and strong, but it results in a high weight. Alloys are not as cheap as aluminium, but they have good corrosion resistance and alloy components can be integrated easily. Finally, composites are expensive (because they are relatively new), but their weight is low.

Having weighed the options against each other we choose `al` as the material. In this case study, however, our primary concern are not the individual decisions - the framework for modeling them is the subject of the previous chapter (Chapter 3). The aim in this chapter is to analyse the interplay of several decisions that were made in sequence. For this reason, the knowledge base in stage 1 is relatively simple, and detailed models will not even be given for subsequent stages. Instead, we only look at the outcome of each stage, and the knowledge that was used to justify it.

Stage 1 is modeled in Example 32 on page 114 and its outcome in Example 33 on page 117.

4.2.3 Stage 2: Detailed Design

In a real-world design environment, the stages of the design process would be carried out by separate teams, each with its own area of expertise. For example, what has been described as stage one in this case study would be the work of a structural engineering team and stage two would be handled by specialists in electromagnetic behaviour of the components. This enables companies to divide the work by domain rather than by component, which would require every team to have expertise in all domains.

The second stage of the process continues with the design chosen in stage one and implements it in greater detail. In this example, the outcome of the first stage was to choose alloys as the material for the connector. The next problem is how to attach the connector to the two components. There are two options, `bolts` or `screws`. If bolts are used, then we also need to place a shim between the L-shape and one of the components (as shown in Figure 3.1 on page 52), because there bolts alone would not be strong enough. As a result, there are two options $\mathscr{O}_2 = \{O_4, O_5\}$ with $O_4 = \{\texttt{bolts}, \texttt{shim}\}$ and $O_5 = \{\texttt{screws}, \texttt{no\_shim}\}$.

When connecting two components with different electromagnetic properties, there is a risk of sparks (and therefore fire) in the event of a lightning strike, caused by the transition of an electric charge from one component to another. We therefore try to achieve good electromagnetic capabilities (EMC) in order to minimise the risk of sparks: $\mathscr{R}_2 = \{\texttt{good\_EMC}\}$.

In this stage, we do not use the same knowledge base as in the previous stage, since the work is carried out by a different group of engineers. We focus on the

electromagnetic capabilities of different materials. $\mathscr{K}_2 = \{\texttt{screws} \Rightarrow \texttt{two\_holes}, \texttt{alloys} \Rightarrow \texttt{one\_hole}\}$. The use of bolts makes aluminium more susceptible to corrosion:

$$\texttt{bolts} \Rightarrow \neg\langle \texttt{al} \Rightarrow \texttt{non\_corrosive}\rangle \tag{4.1}$$

Additionally, we define different numbers of holes to be contradictory: $\overline{\texttt{one\_hole}} = \{\texttt{two\_holes}, \texttt{three\_holes}, \ldots\}$ and so forth.

When the outcome of this (the second) stage is combined with that of the first stage, some of the earlier arguments are no longer acceptable because of the knowledge that was added here. Example 34 on page 118 contains the outcome of this stage, including the added knowledge.

4.2.4 Stage 3: Preliminary Design Revisited

The outcome of the previous decision stage was that some of the original arguments in favour of choosing `al` are no longer acceptable, because they were attacked by arguments from knowledge related to the choice of `bolts` for fastening the two components. In Stage 3, we reject `bolts` because of those consequences, and choose `screws` instead. With this choice there are no corrosion problems, and the arguments in favour of `screws` are compatible with the earlier ones for choosing aluminium. The transition from stage one to three – hypothetically, assuming we skipped stage two – is an example of a *monotonic* decision sequence, which means that the design has been further specialised (by adding more formulae to the chosen option) and that there are no conflicts in the arguments supporting it. Monotonic decision sequences are discussed in Section 4.4.4.1.

4.2.5 Stage 4: Another Process Running in Parallel

While the materials and the mode of attachment are being finalised in stages 1 and 3, a different engineer designs the shape and dimensions of the components. This task could be an entire design process in itself, but for the purposes of this use case we only consider the final outcome represented by a single formula: $\mathsf{length}(s_2) = 25cm$. The length of s_2 does not affect the previous decisions in any way. Example 34 on page 118 demonstrates how our system handles parallel sub-processes.

4.2.6 Stage 5: Joining the Two Processes

At this stage the two parallel decision processes (structural design of the components in stages 1 to 3 and their protective coating in stage 4) are combined into one final design. It has now become apparent that screws, while still an acceptable option, do not achieve the property of good_EMC. Therefore, the outcome of joining the two decision sequences is $\{\texttt{length}(s_2, 25cm), \texttt{al}, \texttt{no\_coating}, \texttt{screws}\}$. This makes stage 5 a monotonic continuation of stage 4 but a weakening of stage 3 (since good_EMC is not achieved anymore). See Section 4.4.4.2 for a formal definition and example.

4.2.7 Conclusions Drawn from Use Case

The use case highlights several characteristics of decisions in the engineering design processes and their documentation. First, it is possible for decisions to be reversed if they turn out to have undesirable consequences later on in the process. This was the case for decision 2, which was reverted in decision 3. If a decision is reversed then the old design documents still exist, but they are superseded by documents for the new decision. A formal model of decision processes must be able to deal with the changing status of decisions (and their supporting arguments) over time.

Secondly, by using design documentation to justify an option, we create an implicit constraint on future decisions, namely that their justifications do not conflict with the justifications of the current decisions. Decision 2 violated this constraint, because rule 4.1 removed the support for an argument in favour of the outcome of decision one, weakening the justification of decision 1.

Thirdly, decision processes are not always linear, but may comprise several sub-processes which are executed in parallel (e.g. by different teams or contractors) and eventually merged into the overall design. The merging of sub-processes was shown in decision 5, which integrated decisions 4 and 3. This lead to a revision of decision 3, where the argument for good_EMC was no longer acceptable.

4.3 A Model of Decision Processes

In this section we consolidate the notion of decision outcomes (which are always conflict-free) with our assumption that knowledge in decision processes grows monotonically (including conflicts, which are resolved by rejecting some assumptions).

An additional benefit of the technique developed here is that it produces diagrams which can be seen as visualisations of decision processes.

When modeling decision processes we have to be careful not to conflate two distinct concepts: A specific "path" or sequence of decisions taken (a description of how the design arrived at a certain point in the design space) and all possible paths or sequences of decisions, i.e. the entire design space. Since we want to analyse past decisions, our model will focus on individual sequences of decisions. That way we do not have to formalise the entire design space, but only a trace of the options that were actually chosen and the reasoning behind decisions.

Decision sequences consist of stages. Each stage consists of a decision frame (K,C,R) (as defined in Chapter 3, page 54) and the set of options available at that stage. In the previous chapter, options in a decision frame were defined to be ASPIC+ knowledge bases.

Definition 51 (Decision Stage)**.** *A **decision stage** is a tuple $S = (K,C,\mathscr{O},R)$ where*

1. *(K,C,R) is a decision frame (Definition 27, page 54)*

2. *$\mathscr{O}$ is a set of options*

As the decision process unfolds, the wing component in our example and the "product" in the general case are being specified in more and more detail. In our model, the product at a specific stage is captured by the union of all options that have been taken until that stage.

If the process is straightforward without any errors or backtracking then the knowledge that makes up the product increases monotonically from start to finish. However, the decision processes we are interested in are iterative and therefore may involve many changes to the design, so the knowledge in their corresponding decision sequences does not increase monotonically.

Example 32. *The first decision stage (see Section 4.2.2) is defined as $S_1 = (K_1,C_1,\mathscr{O}_1,R_1)$ with the contents given below. The set of consequences C_1 contains values such as* `low_weight`*. In S_1, there is no inconsistency because none of the options O_1 to O_3 contain knowledge that results in conflicting arguments. In the terminology*

established in Chapter 3, we have a multi-criteria decision problem but not a decision problem with uncertainty.

$$
\begin{aligned}
K_1 = \{&\texttt{al} \Rightarrow \texttt{high\_weight},\\
&\texttt{composites} \Rightarrow \texttt{low\_weight},\\
&\texttt{al} \Rightarrow \texttt{strong},\\
&\texttt{composites} \Rightarrow \texttt{high\_cost},\\
&\texttt{composites} \Rightarrow \texttt{corrosion\_resistant},\\
&\texttt{al} \Rightarrow \texttt{non\_corrosive},\\
&\texttt{non\_corrosive} \Rightarrow \texttt{corrosion\_resistant},\\
&\texttt{alloys} \Rightarrow \texttt{medium\_cost},\\
&\texttt{alloys} \Rightarrow \texttt{easy\_integration}\}\\
C_1 = \{&\texttt{low\_weight}, \texttt{high\_weight}, \ldots\}\\
\mathcal{O}_1 = \{&O_1, O_2, O_3\}\\
O_1 = \{&\Rightarrow \texttt{al}\}\\
O_2 = \{&\Rightarrow \texttt{alloys}\}\\
O_3 = \{&\Rightarrow \texttt{composites}\}\\
R_1 = \{&\texttt{strong}, \texttt{corrosion\_resistant}\}
\end{aligned}
$$

Comparing the arguments for each option, we can see that $O_2 = \{\Rightarrow \texttt{alloys}\}$ *yields the best results, because with it we can claim* `medium_cost` *and* `easy_integration`.

4.3.1 Outcome of Decision Stage

The purpose of this section is to find a suitable representation of the outcome of a decision stage. The representation should closely model the body of knowledge that makes up the product – that is, it should contain the options that were chosen as well as arguments for choosing them.

It should also comply with our observation that the design passes through several intermediate stages, each with its own specialist knowledge. Only parts of that specialist knowledge are passed on to the next stage. Therefore, we cannot simply define

decision outcomes to be the same as decision stages (that is, as Definition 51).

We will include only a subset of the knowledge of a decision stage in its decision outcome. If an option O was chosen at a decision stage S then the outcome of S for O includes O itself plus exactly the knowledge (set of defeasible rules) that is necessary to create the arguments in the preferred extension that contains arguments in favour of O. Formally:

Definition 52 (Outcome of a Decision Stage)**.** *Let $S = (K, C, \mathscr{O}, R)$ be a decision stage and let $O \in \mathscr{O}$. The* outcome *of S for O is an* ASPIC+ *knowledge base Res where $Res \subseteq K \cup O$ such that*

1. *There exists a preferred extension $E \in \Sigma_{\mathsf{pr}}(\mathsf{argGraph}(K \cup O))$ such that $E = \mathsf{args}(Res)$ and*
2. *Res is the minimal knowledge base (with respect to $\subseteq$) fulfilling Cond. 1*

To distinguish decision outcomes from ASPIC+ knowledge bases in general, we will refer to them using variations of the variable Res (instead of KB). The notation $(Res_1, \ldots, Res_n)$ will be used for sequences of decision outcomes, where each Res_i is a decision outcome. .

The outcome Res of a decision stage is *non-empty* if $\mathsf{args}(Res) \neq \emptyset$. Note that every outcome of a decision stage is an ASPIC+ knowledge base, so the argument graph $\mathsf{argGraph}(Res)$ can be computed. Its attacks relation will always be empty, as shown next.

Proposition 24. *If Res is the outcome of a decision stage and $(A, Att) = \mathsf{argGraph}(Res)$ then $Att = \emptyset$.*

Proof. By contradiction. Assume that the attacks relation Att is not empty, so there is an $(a_1, a_2) \in Att$. Then there are two arguments $a_1, a_2 \in A$ such that a_1 attacks a_2, so A attacks itself. This contradicts Definition 52 Cond. 1, by which A is a preferred extension of some argument graph and therefore a conflict-free set. □

According to Definition 52, every outcome Res of a decision stage $(K, C, \mathscr{O}, R)$ subsumes one of the options $O \in \mathscr{O}$, that is $O \subseteq Res$. We will use the function $\mathsf{option}(Res)$ to refer to this option O specifically.

Example 33. *In decision stage S_1 (Example 32), we get the following decision result for $O_1 = \{\Rightarrow \texttt{al}\}$, aluminium:*

$$Res_1 = \{\Rightarrow \texttt{al}, r_1, r_2, r_3\} \textit{ with}$$
$$r_1 = \texttt{al} \Rightarrow \texttt{strong}$$
$$r_2 = \texttt{al} \Rightarrow \texttt{non\_corrosive}$$
$$r_3 = \texttt{al} \Rightarrow \texttt{high\_weight}$$

This corresponds to the outcome of stage 1 described in Section 4.2.2 on page 110.

An important property of two decision outcomes Res_1, Res_2 is that any inconsistency (attack) in the argument graph of $Res_1 \cup Res_2$ originates from an argument in Res_1 (respage Res_2) and targets an argument in Res_2 (respage Res_1). In other words, taking the union of two decision outcomes does not introduce "internal conflicts" in either of them, and all conflicts involve one argument from each of Res_1 and Res_2.

Proposition 25. *Let Res_1, Res_2 be two outcomes of decision stages and let $(A, Att) = \mathsf{argGraph}(Res_1 \cup Res_2)$. For every $(a,b) \in Att$, $a \in \mathsf{args}(Res_1)$ and $b \in \mathsf{args}(Res_2)$ or vice versa.*

Proof. By contradiction. Let Res_1, Res_2 be two outcomes of decision stages and let $(A, Att) = \mathsf{argGraph}(Res_1 \cup Res_2)$. Let $(a,b) \in Att$ and assume $a, b \in \mathsf{args}(Res_1)$ (w.l.o.g.) This directly contradicts Propage 24. □

Proposition 26. *Let Res_1, Res_2 be two outcomes of decision stages and let $G = (A, Att) = \mathsf{argGraph}(Res_1 \cup Res_2)$. Then,*

$$\mathsf{args}(Res_1) \cap \mathsf{args}(Res_2) \subseteq \Sigma_{\mathsf{gr}}(G)$$

Proof. Let Res_1, Res_2 and G be as required. Let $A_1 = \mathsf{args}(Res_1)$ and let $A_2 = \mathsf{args}(Res_2)$. Let $a \in A_1 \cap A_2$. Either a has an attacking argument b with $(b,a) \in Att$ or not. Assume there is a $b \in A$ such that $(b,a) \in Att$. Since $a \in A_1$, $b \in A_2$ (by Propage 25). But $a \in A_1 \cap A_2$, so $a \in A_2$ and $b \in A_2$, contradicting Definition 52 Cond. 1 which implies that both A_1 and A_2 are conflict-free. □

Decision outcomes (Definition 52) contain exactly the knowledge that is required to justify a decision, that is, to produce all the arguments in favour of the option that was chosen. Decision outcomes on their own are conflict-free, but the union of two decision outcomes may be inconsistent. In the remainder of this section we will study decision outcomes as building blocks of *decision sequences*, that is, of the many decisions that together form a design.

4.3.2 Embedding Decisions Results

Decision outcomes do not exist in a vacuum. They are part of a design process that is ultimately driven by a goal, namely to meet a set of requirements within the constraints of time and budget. As a result, the decisions made at each stage (and their supporting arguments) have an effect on subsequent decisions.

Our definition of decision outcomes (Definition 52) can express such correlations only in a limited way: Because each individual decision outcome is conflict-free, arguments may be re-used in later stages, but only in the same polarity (that is, the system cannot express the phenomenon that an argument in favour of one decision outcome can also act as an argument against another decision outcome). Nonetheless, Definition 52 is a suitable representation of design documents, as seen in Section 4.5.1. The decision processes introduced in this section should therefore seen as a second layer on top of the individual decision stage. The purpose of this layer is to draw connections between decisions, and the purpose of the lower (decision making) layer is to assess individual decisions.

4.3.2.1 Embeddings by Example

Let us look at a bigger example to illustrate the points made above.

Example 34. *The decision outcome Res_2 documents a decision that was made after Res_1.*

$$Res_2 = \{\Rightarrow \texttt{al}, \Rightarrow \texttt{bolts}, \Rightarrow \texttt{shim}, r_1, r_4, r_5\}$$

$$r_4 = \texttt{bolts} \Rightarrow \neg\langle \texttt{al} \Rightarrow \texttt{non\_corrosive}\rangle$$

$$r_5 = \texttt{shim}, \texttt{bolts} \Rightarrow \texttt{good\_EMC}$$

We can see that $\Rightarrow \texttt{al} \in Res_2$, so the decision for aluminium made earlier is still re-

spected in the design, but rule r_2 is missing from Res_2. This indicates that, with the additional knowledge available at stage 2, arguments based on r_2 are defeated. The new rule r_4 in fact deactivates rule r_2 so arguments that use it are not (sceptically or credulously) acceptable.

Suppose the result Res_1 from Example 33 is not followed by Res_2, but by Res_3 (below) instead, after the choice of adding `bolts`, `shim` *was rejected because it resulted in attacks on arguments based on rule r_2. In Res_3, we choose not to apply a coating because aluminium is corrosion-resistant without coating. Also we use screws instead of bolts for attaching the structure.*

$$Res_3 = \{\Rightarrow \texttt{al}, \Rightarrow \texttt{no\_coating}, \Rightarrow \texttt{screws}, r_1, r_2, r_3, r_6, r_7\} \text{ with}$$
$$r_6 = \texttt{al}, \texttt{no\_coating} \Rightarrow \texttt{corrosion\_resistant}$$
$$r_7 = \texttt{al}, \texttt{screws} \Rightarrow \texttt{good\_EMC}$$

Independently of decisions one to three (with results Res_1 to Res_3), the length of the component s_2 (cf. Figure 3.1 on page 52) was decided:

$$Res_4 = \{\Rightarrow \texttt{length}(s_2, 25cm), r_8\}$$
$$r_8 = \texttt{length}(s_2, 25cm) \Rightarrow \texttt{weight}(s_2, 450g)$$

And, lastly, the two separate decision sequences (Res_1, Res_2, Res_3 and Res_4) are combined in the final decision Res_5.

$$Res_5 = \{\Rightarrow \texttt{length}(s_2, 25cm), \Rightarrow \texttt{al}, \Rightarrow \texttt{no\_coating}, \Rightarrow \texttt{screws}, r_1, r_2, r_3, r_6, r_9\}$$
$$r_9 = \texttt{length}(s_2, \leq 30cm) \Rightarrow \neg\texttt{good\_EMC}$$

Note that the conclusion of r_9 is $\neg$`good_EMC`*, which directly contradicts the claim of a_{12} (*`good_EMC`*). Therefore a_{12} and a_{15} attack each other, so they cannot be in the same conflict-free set, so at most one of the two arguments can be part of a decision outcome. For the outcome Res_5 in particular, a_{15} is included so a_{12} is excluded.*

To summarise the decision process Res_1 to Res_5, Table 4.1 lists all arguments

Table 4.1: Arguments in the example decision process. The column "Decision Results" gives the decision results in which an argument is included.

Name	Argument	Decision Results
a_1	$[\Rightarrow \texttt{al};\texttt{al}]$	$Res_1, Res_2, Res_3, Res_5$
a_2	$[a_1;\texttt{al} \Rightarrow \texttt{strong};\texttt{strong}]$	Res_1, Res_3, Res_5
a_3	$[a_1;\texttt{al} \Rightarrow \texttt{non\_corr.};\texttt{non\_corr.}]$	Res_1, Res_3, Res_5
a_4	$[a_1;\texttt{al} \Rightarrow \texttt{high\_weight};\texttt{high\_weight}]$	Res_1, Res_3, Res_5
a_5	$[\Rightarrow \texttt{bolts};\texttt{bolts}]$	Res_2
a_6	$[a_5;\texttt{bolts} \Rightarrow \neg\langle r_2\rangle;\neg\langle r_2\rangle]$	Res_2
a_7	$[\Rightarrow \texttt{shim};\texttt{shim}]$	Res_2
a_8	$[a_5, a_7;\texttt{shim},\texttt{bolts} \Rightarrow \texttt{good\_EMC};\texttt{good\_EMC}]$	Res_2
a_9	$[\Rightarrow \texttt{no\_coating};\texttt{no\_coating}]$	Res_3, Res_5
a_{10}	$[a_1, a_9; r_6;\texttt{corrosion\_resistant}]$	Res_3, Res_5
a_{11}	$[\Rightarrow \texttt{screws};\texttt{screws}]$	Res_3, Res_5
a_{12}	$[a_1, a_{11}; r_7;\texttt{good\_EMC}]$	Res_3
a_{13}	$[\Rightarrow \texttt{length}(s_2, 25cm);\texttt{length}(s_2, 25cm)]$	Res_4, Res_5
a_{14}	$[a_{13}; r_8;\texttt{weight}(s_2, 450g)]$	Res_4
a_{15}	$[a_{13}; r_9;\neg\texttt{good\_EMC}]$	Res_5

produced at any stage. The column "Decision Results" gives the decision results in which an argument is included (that is, the decision results from which the argument can be formed). Many arguments appear in more than one decision result.

The knowledge on which decisions are based often increases monotonically during the design process. In Example 34, options and rules of Res_1 are a subset of those of Res_3 and therefore $\mathsf{args}(Res_1) \subseteq \mathsf{args}(Res_3)$. This is not the case for all decision results as earlier arguments may disappear. For example $a_{12} \in \mathsf{args}(Res_3)$, but $a_{12} \notin \mathsf{args}(Res_5)$, even though the knowledge base representing the actual design – as determined by the option function – increased monotonically, $\mathsf{option}(Res_3) \subseteq \mathsf{option}(Res_5)$.

How can the apparent disappearance of knowledge conform with our claim that knowledge increases monotonically in decision processes? The answer to this question lies in our definition of decision outcomes in Section 4.3.1. There, we defined decision outcomes as the logical equivalent of design documents, which means that they only contain enough knowledge to justify (make arguments for) the current decision, not all previous decisions. However, even though some knowledge is not written down in a design document, it is not lost to the designers. In our formal model we are free to include additional knowledge. This inclusion of background knowledge is the idea behind the definition of embeddings (Definition 54).

4.3.2.2 Properties of Embeddings

In the rest of this section we will state the conditions we would like to hold for a embeddings, and in the next section (Section 4.3.3) we will address the question of how to construct an embedding for a given decision sequence.

We first need to define the concept of increasing knowledge formally. A sequence of ASPIC+ knowledge bases is monotonically increasing if each element is a subset of its successor.

Definition 53 (Monotonically Increasing)**.** *A sequence* $(k_1,\ldots,k_n)$ *of* ASPIC+ *knowledge bases is* monotonically increasing *if and only if for all* $1 \leq i < n$, $k_i \subseteq k_{i+1}$.

Even if a sequence of knowledge bases is montonically increasing, the set of (sceptically or credulously) acceptable arguments at each stage may not be increasing, because new counter-arguments may be introduced.

A monotonically increasing sequence of ASPIC+ knowledge bases is an *embedding* of a sequence of decision results (of the same length) if the arguments of the i-th decision result are sceptically acceptable in the graph of the i-th knowledge base. Since this requires both sequences (the monotonically increasing sequence of knowledge bases, and the sequence of decision results) to have the same length, we represent embeddings as functions from decision results to ASPIC+ knowledge bases with the following properties:

Definition 54 (Embedding of Decision Results)**.** *Let* $S = (Res_1,\ldots,Res_n)$ *be a sequence of decision outcomes (Definition 52). An* embedding E *of* S *is a function from decision outcomes to* ASPIC+ *knowledge bases such that*

1. *For all* $i \leq n$, $Res_i \subseteq E(Res_i)$,
2. *For all* $i \leq n$, $\mathsf{args}(Res_i) \subseteq \Sigma_{\mathsf{gr}}(\mathsf{argGraph}(E(Res_i)))$ *and*
3. *For all* $i < n$, $E(Res_i) \subseteq E(Res_{i+1})$

Let us look at the conditions of Definition 54 in detail. The first one says that every decision result Res_i in S must be subsumed by its corresponding result $E(Res_i)$. Condition 2 ensures that, at every stage i in the design process, the arguments that are sceptically acceptable in Res_i are sceptically acceptable in $E(Res_i)$

also. Since every decision outcome Res_i is conflict-free, its argument graph contains no attacks, so $\Sigma_{\mathsf{gr}}(Res_i) = \mathsf{args}(Res_i)$, and the condition could alternatively be stated as "For all $i \leq n$, $\Sigma_{\mathsf{gr}}(Res_i) \subseteq \Sigma_{\mathsf{gr}}(\mathsf{argGraph}(E(Res_i)))$." Finally, Cond. 3, states that $(E(Res_1),\ldots,E(Res_n))$ is monotonically increasing, that is, no rules are lost when progressing from $E(Res_i)$ to $E(Res_{i+1})$.

A consequence of conditions 2 and 3 for our example (Example 34) is that any embedding E of (Res_3, Res_5) must contain attacks in $\mathsf{argGraph}(E(Res_5))$, even though Res_3 and Res_5 on their own are conflict-free. This is because argument a_{15} from Res_5 attacks argument a_{13} from Res_3, and by Definition 54 Cond. 3, a_{13} is also an argument of $E(Res_5)$. More generally, we find that embeddings of decision outcomes may introduce conflict even though each outcome on its own is conflict-free:

Proposition 27. *Let $S = (Res_1,\ldots,Res_i,\ldots,Res_k,\ldots,Res_n)$ be a decision sequence with $i < k \leq n$ such that the argument graph $G = (A,Att) = \mathsf{argGraph}(Res_i \cup Res_k)$ has a non-empty set of attacks, $Att \neq \emptyset$. Let E be an embedding of S. Then, $\mathsf{argGraph}(E(Res_k))$ is not conflict-free.*

Proof. Let $a \in \mathsf{args}(Res_i)$, $b \in \mathsf{args}(Res_k)$ such that $(a,b) \in Att$ (the proof for $(b,a) \in Att$ proceeds similarly). Let $G' = (A',Att') = \mathsf{argGraph}(E(Res_k))$. By Definition 54 Cond. 2, $a \in \mathsf{args}(E(Res_k))$, and by Definition 54 Cond. 1, $b \in \mathsf{args}(E(Res_k))$. Since b attacks a, $(b,a) \in Att'$ so $Att \neq \emptyset$. □

Embeddings are mappings of individual ASPIC+ knowledge bases (see Definition 2.3.2.1 on page 33), but every embedding E also defines a mapping of *sequences* of knowledge bases, obtained by applying E to each element in the sequence. We will use embeddings mostly in the latter (sequence) sense, but define them in the former (element) sense, because it simplifies proofs and it enforces the property that embeddings of sequences of decision outcomes are really embeddings of their constituent parts.

With embeddings we can describe the relationship between individual design documents that are conflict-free and the aggregated knowledge in a design process that may have conflicts. The remainder of this chapter will be spent constructing embeddings and exploring them from different angles.

4.3.3 Embed and Extract

First we will look at the problem of producing embeddings. It is not obvious that every sequence of decision outcomes has an embedding, because of conditions 2 and 3 of Definition 54, which imply that asymmetric attacks must be introduced during the course of the decision process, because the accumulated knowledge grows monotonically (Cond. 3) while the grounded extension has no such constraint (Cond. 2), so it can shrink as well as grow. Asymmetric attacks are introduced by deactivate (Definition 47), which we can use to compute a *canoncial embedding* of any decision sequence S.

Definition 55 (Embed)**.** *For any sequence of decision outcomes $S = (Res_1, \ldots, Res_n)$, the* canonical embedding *of S is defined as*

$$\begin{aligned} \mathsf{emb}_S(Res_1) &= Res_1 \\ \mathsf{emb}_S(Res_{i+1}) &= \mathsf{enforce}(\mathsf{args}(Res_{i+1}), \mathsf{emb}_S(Res_i) \cup Res_{i+1}) \end{aligned}$$

Example 35, below, demonstrates another intuitive explanation of embeddings: They reify preferences of arguments, but this preference information is based on a sequence of argument graphs and not an ordering of the arguments themselves (or the knowledge they are based on).

Example 35. *If we wanted to obtain the overall knowledge expressed in Res_3 and Res_5, we could simply take their union $Res_3 \cup Res_5$. However, when comparing the arguments in those two decision outcomes, we see that Res_3 has an argument a_{12} with conclusion* `good_EMC` *and Res_5 has an argument a_{15} with conclusion $\neg$*`good_EMC`*. These two arguments are mutually exclusive, leading to a symmetric attack in* $\mathsf{argGraph}(Res_3 \cup Res_5)$*. As a result, neither a_{12} nor a_{15} are part of $\Sigma_{\mathsf{gr}}(\mathsf{argGraph}(Res_3 \cup Res_5))$, even though it is clear from the context of our example that Res_5 supersedes Res_3. For this reason, $Res_3 \cup Res_5$ cannot be an embedding of Res_5, because a_{15}'s exclusion from the grounded extension would violate Definition 54 Cond. 3. We therefore need to encode the "preference" that a_{15} has over a_{12}.*

An embedding emb_S *of the sequence $S = (Res_3, Res_5)$ recovers this missing piece of information by creating a new rule $\Rightarrow \neg\langle r_7 \rangle$. In detail, the embedding of*

$S = (Res_3, Res_5)$ *is given by*

$$\begin{aligned}
\mathsf{emb}_S(Res_3) &= Res_3 \\
\mathsf{emb}_S(Res_5) &= \mathsf{enforce}(\mathsf{args}(Res_5), \mathsf{emb}_S(Res_3) \cup Res_5) \\
&= \mathsf{enforce}(\mathsf{args}(Res_5), Res_3 \cup Res_5) \\
&= \mathsf{enforce}(\{a_{1-4}, a_{9-11}, a_{13}, a_{15}\}, Res_3 \cup Res_5)) \\
&= \mathsf{deactivate}(\mathsf{topRule}(a_{12}), Res_3 \cup Res_5) \\
&= \mathsf{deactivate}(r_7, Res_3 \cup Res_5) \\
&= Res_3 \cup Res_5 \cup \{\Rightarrow \neg\langle r_7\rangle\}
\end{aligned}$$

The new rule $\Rightarrow \neg\langle r_7\rangle$*, added in the last line, creates an object-level representation of the fact that* r_7 *is not applicable anymore. This fact was only implicit in the simple union* $Res_3 \cup Res_5$*, leading to the presence of two preferred extensions in* $\mathsf{argGraph}(Res_3 \cup Res_5)$.

The following result shows that we are justified in calling emb_S a *canonical embedding* of S, since it works for all sequences of decision outcomes S.

Theorem 5. *If* S *is a sequence of decision outcomes then* emb_S *is an embedding of* S.

Proof. Let $S = (Res_1, \ldots, Res_n)$ be a sequence of decision outcomes. We will prove that the three conditions of Definition 54 hold.

Cond. 1 Let $i \leq n$. If $i = 1$ then $\mathsf{emb}_S(Res_i) = Res_1$, so by Definition 55, $Res_i \subseteq \mathsf{emb}_S(Res_i)$. If $i > 1$, then

$$\begin{aligned}
\mathsf{emb}_S(Res_i) &= \mathsf{enforce}(\Sigma_{\mathsf{gr}}(\mathsf{argGraph}(Res_i)), \mathsf{emb}_S(Res_{i-1}) \cup Res_i) \\
&\qquad \{\text{by Propage 21, page 94}\} \\
&\supseteq Res_i
\end{aligned}$$

Cond. 2 We can distinguish two cases, $i = 1$ and $i > 1$.

$(i = 1)$ Let $a \in \mathsf{args}(Res_1)$. $\mathsf{emb}_S(Res_i) = Res_1$ (by Definition 55), so $\mathsf{args}(Res_1) = \mathsf{args}(\mathsf{emb}_S(Res_1))$ and because Res_1 is a decision outcome, $\mathsf{argGraph}(Res_1)$ is conflict-free by Propage 24. Therefore the grounded extension of $\mathsf{argGraph}(\mathsf{emb}_S(Res_1))$ contains extactly the arguments in $\mathsf{args}(\mathsf{emb}_S(Res_1))$, so $a \in \Sigma_{\mathsf{gr}}(\mathsf{argGraph}(\mathsf{emb}_S(Res_1)))$.

($i > 1$) If $i > 1$ then by Definition 55 case 2, $\mathsf{emb}_S(Res_i) = \mathsf{enforce}(\mathsf{args}(Res_i), \mathsf{emb}_S(Res_{i-1}) \cup Res_i)$. $\mathsf{args}(Res_i)$ is conflict-free by Definition 24, so we can apply Propage 22 and obtain $\mathsf{args}(Res_i) \subseteq \Sigma_{\mathsf{gr}}(\mathsf{argGraph}(\mathsf{emb}_S(Res_i)))$.

Cond. 3 The final condition stipulates that $(\mathsf{emb}_S(Res_1), \ldots, \mathsf{emb}_S(Res_n))$ be monotonically increasing, that is, for all $k < n$, $\mathsf{emb}_S(Res_k) \subseteq \mathsf{emb}_S(Res_{k+1})$. Let $k < n$. We distinguish two cases, $k = 1$ and $k > 1$.

Case 1. If $k = 1$ then

$$\begin{aligned} \mathsf{emb}_S(Res_2) &= \mathsf{enforce}(\mathsf{args}(Res_2), \mathsf{emb}_S(Res_1) \cup Res_2) \\ &\qquad \{ \text{ by Definition 55 Cond. 1} \} \\ &= \mathsf{enforce}(\mathsf{args}(Res_2), Res_1 \cup Res_2) \end{aligned}$$

so by Propage 21, $Res_1 \subseteq \mathsf{emb}_S(Res_2)$ and therefore $\mathsf{emb}_S(Res_1) \subseteq \mathsf{emb}_S(Res_2)$.

Case 2 If $k > 1$ then

$$\mathsf{emb}_S(Res_{k+1}) = \mathsf{enforce}(\mathsf{args}(Res_{k+1}), \mathsf{emb}_S(Res_k) \cup Res_{k+1})$$

so by applying Propage 21, we directly obtain $\mathsf{emb}_S(Res_k) \subseteq \mathsf{emb}_S(Res_{k+1})$. □

We can also go the other way, from arbitrary sequences of ASPIC+ knowledge bases to sequences of (conflict-free) decision outcomes, using the extract operation defined below. It returns all the rules used by arguments in the grounded extension.

Definition 56 (Extract)**.** *For any* ASPIC+ *knowledge base KB, if* $\Sigma_{\mathsf{gr}}(\mathsf{argGraph}(KB)) = \{E\}$ *then*

$$\mathsf{extract}(KB) = \{\mathsf{rules}(a) \mid a \in E\}$$

extract produces a conflict-free ASPIC+ knowledge base which contains exactly the assumptions and rules necessary to construct all arguments in the grounded extension of $\mathsf{argGraph}(KB)$.

Example 36. *Applying* extract *to the previous example (Example 35) yields the follow-*

ing:

$$\begin{aligned}\mathsf{extract}(\mathsf{emb}_S(Res_5)) &= \mathsf{extract}(Res_3 \cup Res_5 \cup \{\Rightarrow \neg\langle r_7\rangle\}) \\ &= \{\Rightarrow \mathtt{length}(s_2, 25cm), \Rightarrow \mathtt{al}, \Rightarrow \mathtt{no\_coating}, \\ &\quad \Rightarrow \mathtt{screws}, r_1, r_2, r_3, r_6, r_9, \Rightarrow \neg\langle r_7\rangle\}\end{aligned}$$

$\mathsf{extract}(\mathsf{emb}_S(Res_5))$ *subsumes the original result* Res_5*, adding the new rule* $\Rightarrow \neg\langle r_7\rangle$*. This added knowledge represents the tacit assumptions (knowledge that was implicitly assumed) in* Res_5*.*

Example 36 demonstrated a property of extract. When we combine extract with emb_S by calculating $\mathsf{extract}(\mathsf{emb}_S(Res_i))$ for some decision sequence S and a result Res_i in S, the result contains all rules and assumptions in the original decision outcome Res_i.

Proposition 28. *For every sequence of decision outcomes* $S = (Res_1, \ldots, Res_n)$ *and every* Res_i *in* S*:*

$$Res_i \subseteq \mathsf{extract}(\mathsf{emb}_S(Res_i))$$

Proof. Let $S = (Res_1, \ldots, Res_n)$ be a sequence of decision outcomes and let $i \leq n$. Let $Res_i' = \mathsf{extract}(\mathsf{emb}_S(Res_i))$. Let $r \in Res_i$. Since Res_i is a decision outcome, it is minimal (Definition 52 Cond. 2), so there exists an argument $a \in \mathsf{args}(Res_i)$ with $r \in \mathsf{rules}(a)$. As emb_S is an embedding of S (by Theorem 5), $a \in \Sigma_{\mathsf{gr}}(\mathsf{argGraph}(\mathsf{emb}_S(Res_i)))$ (Definition 54 Cond. 2). Then $r \in \mathsf{rules}(\Sigma_{\mathsf{gr}}(\mathsf{argGraph}(\mathsf{emb}_S(Res_i))))$, so by Definition 56, $r \in Res_i'$. □

Embedding and extracting the first element of a sequence has no effect:

Proposition 29. *For every decision sequence* $S = (Res_1, \ldots, Res_n)$*,*

$$\mathsf{extract}(\mathsf{emb}_S(Res_1)) = Res_1$$

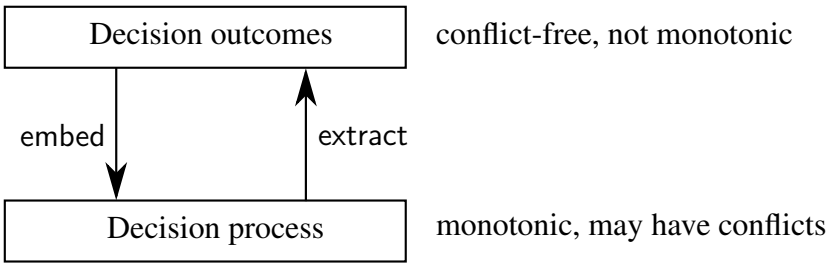


Figure 4.1: Conceptual relationship of decision processes and outcomes

Proof. Let Res_1 be a decision outcome and let $G = (A, Att) = \mathsf{argGraph}(Res_1)$.

$$
\begin{aligned}
&\mathsf{extract}(\mathsf{emb}_S(Res_1)) \\
&\qquad \{\text{By Definition 55 case 1}\} \\
&= \mathsf{extract}(Res_1) \\
&\qquad \{\text{Apply Definition 56, let} \{E\} = \Sigma_{\mathsf{gr}}(G)\} \\
&= \bigcup_{a \in E} \mathsf{assumptions}(a) \cup \bigcup_{a \in E} \mathsf{rules}(a) \\
&\qquad \{\text{By Propage 24, } Att = \emptyset, \text{ so } E = A\} \\
&= \bigcup_{a \in A} \mathsf{assumptions}(a) \cup \bigcup_{a \in A} \mathsf{rules}(a) \\
&\qquad \{a = \mathsf{args}(Res_1)\} \\
&= Res_1
\end{aligned}
$$

□

emb_S is not the only embedding of a decision sequence S, but it is the minimal one in a certain sense. The two operations emb and extract can be seen as the link between (conflict-free) decision outcomes and (conflicting) decision processes, as the diagram in Figure 4.1 on page 127 shows. The combination of embedding a decision sequence and then extracting is as a way of making tacit assumptions visible, as shown in Example 36.

4.4 Impact Analysis

Because most design processes are iterative, it is often necessary to adjust a decision that was made in the past. Such adjustments may have external reasons (for example, because requirements changed) or internal reasons (the design needed to be improved).

However, a past decision cannot be changed in isolation - subsequent decisions that depend on it also need to be adjusted. As this section will show, our argumentation-based model of decision sequences is well suited to analyse the impact of changed decisions on the process.

In our terminology, if $S = (Res_1, Res_2, \ldots, Res_n)$ is a sequence of decision outcomes and a change is made by replacing the second decision Res_2 with Res_2', then any embedding E of S has to be adjusted (in order to meet Cond. 1 of Definition 54). By comparing the "new" embedding $\mathsf{emb}_{S'}$ with the old embedding emb_S, we get an idea of the impact the change has made.

The actual comparison of the two embeddings is done element-wise, by comparing the argument graphs of $\mathsf{emb}_S(Res_i)$ and $\mathsf{emb}'_S(Res'_i)$, where Res_i and Res'_i are the ith decision outcomes of S and S' respectively.

What is meant by comparing two decision outcomes? In Chapter 3, an argumentation-based technique for comparing options for a single decision was introduced, based on the relative merits of each option (for example, how well it meets the requirements). There, we evaluated each decision on its own, without considering the potential impact on other, related decisions. In this chapter, we are interested in the *impact* that choosing one decision outcome over the other has. That is, the comparison is based on the difference between two decision outcomes Res and Res', and not on the pros and cons of each individual option (as in Chapter 3). This difference in turn is determined by how many of the arguments that are accepted in subsequent decision stages would be invalidated (and thus would have to be changed) if we changed our decision from Res to Res'. The following example clarifies exactly what the difference between decision outcomes is:

Example 37. *Consider the decision outcome Res_1 and Res_2 with*

$$Res_1 = \{\Rightarrow \mathtt{a}, \Rightarrow \mathtt{b}, \Rightarrow \mathtt{c}, \quad \mathtt{a} \Rightarrow \neg \mathtt{d}, \mathtt{b} \wedge \mathtt{c} \Rightarrow \mathtt{f}, \mathtt{a} \wedge \mathtt{b} \Rightarrow \mathtt{g}\}$$

$$Res_2 = \{\Rightarrow \mathtt{b}, \Rightarrow \mathtt{c}, \Rightarrow \mathtt{d}, \Rightarrow \neg\langle \mathtt{a} \wedge \mathtt{b} \Rightarrow \mathtt{g}\rangle \quad \mathtt{b} \wedge \mathtt{c} \Rightarrow \mathtt{f}, \mathtt{d} \Rightarrow \mathtt{e}, \}$$

Each of the outcomes Res_1 and Res_2 on its own results in a conflict-free argument graph, as required in Definition 52. However, their union $Res_1 \cup Res_2$ yields an argu-

Table 4.2: Arguments in Example 37

Name	Argument	Name	Argument
a_1	$[\Rightarrow \mathtt{a};\mathtt{a}]$	a_6	$[a_2, a_3; \mathtt{b} \wedge \mathtt{c} \Rightarrow \mathtt{f};\mathtt{f}]$
a_2	$[\Rightarrow \mathtt{b};\mathtt{b}]$	a_7	$[a_4; \mathtt{d} \Rightarrow \mathtt{e};\mathtt{e}]$
a_3	$[\Rightarrow \mathtt{c};\mathtt{c}]$	a_8	$[a_1, a_2; \mathtt{a} \wedge \mathtt{b} \Rightarrow \mathtt{g};\mathtt{g}]$
a_4	$[\Rightarrow \mathtt{d};\mathtt{d}]$	a_9	$[\neg\langle \mathtt{a} \wedge \mathtt{b} \Rightarrow \mathtt{g}\rangle]$
a_5	$[a_1; \mathtt{a} \Rightarrow \neg\mathtt{d}; \neg\mathtt{d}]$		

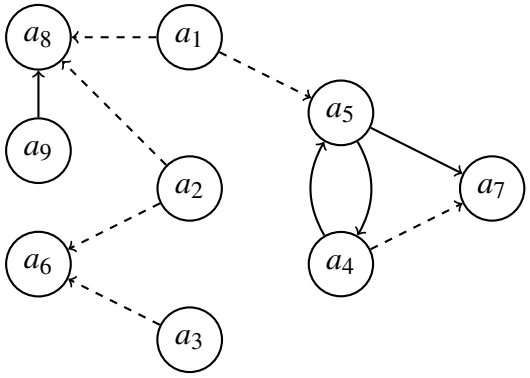


Figure 4.2: Conflicts after a decision was changed (Example 37)

ment graph with two preferred extensions, as shown in Figure 4.2, with arguments in Table 4.2.

The difference between Res_1 and Res_2 is manifested in six arguments: a_1, a_5 and a_8 are present in Res_1 and absent from Res_2, and a_4, a_7 and a_9 are present in Res_1 and absent from Res_2.

Example 37 indicates that there are two kinds of change that can be made to a decision outcome. The first kind involves adding or removing arguments that justify a decision, and it arises from a change in the decision maker's knowledge. In Example 37, this kind of change is the reason why some of r_1's arguments are missing in r_2 (e.g. argument a_4) and vice versa.

The acquisition of new knowledge is a necessary, but not a sufficient reason for actually choosing a different option. It may be that the new knowledge simply reinforces the reasons for making the original decision. We therefore have to consider a second kind of change, the act of "changing one's mind" - accepting a position that is inconsistent with the previously accepted position, thereby rejecting the old point of view in favour of the new. Since each decision outcome is in itself consistent, this second kind of change always implies the first kind (adding new arguments). This result

is formalised in Propage 34.

In the following two sections, 4.4.1 and 4.4.2 (page 137), we will analyse both kinds of change in terms of our formal model of decision processes.

4.4.1 Impact Analysis Based on Argument Strength

In the argument graph in Figure 4.2, a_8 is rejected (attacked by the grounded extension), and a_7 is only acceptable in one of the two preferred extensions, even though both arguments were undisputed as part of their respective decision outcomes (Res_2 and Res_1).

In other words, a_8 and a_7 are weaker in the graph of $Res_1 \cup Res_2$ than they were before. The fact that the acceptability of some arguments has changed (a_8 became unacceptable and a_7 became credulously acceptable) by itself does not tell us much about the impact of choosing Res_2 over Res_1, because arguments from both the old, rejected outcome (Res_1) and the new outcome (Res_2) are weakened by the change. With our model of decision processes we can quantify this change, and tell whether the overall strength of our arguments increased or decreased as a result of it.

4.4.1.1 Argument Strength

Argument strength can be interpreted in two ways (cf. Besnard and Hunter [18]). The first interpretation, which is shared among others by the ASPIC+ system (see the discussion preceding Definition 11 on page 31), treats argument strength as an external property, as meta-data that is supplied as a parameter to the argumentation system. We call this the extensional interpretation of argument strength. The second approach treats argument strength as a function of a set of arguments. That is, the strength of an argument is defined entirely by its relationship with other arguments, and can thus be computed without requiring any additional information. We call this the intensional interpretation of argument strength.

Throughout the thesis we have tried to minimise the number of parameters to our system. For example, in Chapter 3 we chose to represent different possible worlds as the preferred extensions of an argument graph, rather than an explicit list of possible worlds with a mapping to the formulae that hold in them. In the same spirit we choose the intensional interpretation of argument strength. Specifically, we are going to use the game-theoretic measure introduced by Matt and Toni [55]. In [55], the strength of an

argument is defined to be its expected pay-off in a game of argumentation strategy (in the game-theoretic sense). Their approach suits our purposes for several reasons. First, it is defined on abstract argument graphs, so we can adopt it without any additional definitions (other than the measure itself). Second, the measure is fine-grained enough to assign different strength to arguments with the same acceptability status (for example, one credulously acceptable argument might have the strength 0.45 and another one might have 0.33). This fits our requirement for the impact measure to provide more data than just acceptability. Finally, because the measure considers no information beyond arguments and their attacks, it allows us to meet our requirement to quantify the impact of changing a decision on the *justifications* of the design (as opposed to, say, the quality of the design) as discussed in the first chapter of the thesis.

We will now introduce the measure by Matt and Toni in Section 4.4.1.2 below. In the following section, we will use argument strength as a basis for measuring the impact of choosing one option instead of another one.

4.4.1.2 A Game-Theoretic Measure

This section summarises the article by Matt and Toni [55]. Please refer to the original article for a more detailed presentation. Our own contribution begins with Section 4.4.1.3 on page 133.

In the following we assume an argument graph $G = (A, Att)$ as usual. The strength of an argument $a \in A$ is the expected payoff of a game (G, a). The measure of argument strength is centered around the notion of a set of strategies that can be adopted by the opponent and proponent of a. A strategy is simply a set of arguments. The strategies available to the proponent must include a, whereas the opponent is free to choose any set of arguments.

Definition 57 (Pure Strategies [55])**.** *The set of strategies for the proponent and opponent are* $\{P \mid P \subseteq A, a \in P\}$ *and* $\{O \mid O \subseteq A\}$*, respectively.*

The proponent is trying to defend a, so their strategies should be conflict-free. The opponent's strategies on the other hand should always attack a. It should also be in the proponent's interest to attack the opponent's arguments and in turn avoid (counter) their attacks.

We will now introduce some notation that allows us to take into account the number of attacks from one set of arguments X against another one, Y: $Y_G^{\leftarrow X} = \{(x,y) \in X \times Y \mid (x,y) \in Att\}$. The acceptability of a set of arguments X with respect to Y should increase with the size of $Y_G^{\leftarrow X}$ and decrease with the size of $X_G^{\leftarrow Y}$. This can be captured by the following expression:

$$\phi(P,O) = \frac{1}{2}(1 + f(|O_F^{\leftarrow P}|) - f(|P_F^{\leftarrow O}|))$$

for a monotonic increasing function $f : \mathbb{N} \to [0,1[$ such that $f(0) = 0$ and $\lim_{n\to\infty} f(n) = 1$. For this thesis we will stay with the definition for f proposed in [55]:

$$f(n) = 1 - \frac{1}{n+1}$$

The rewards of the game (G,x) for an argument x are defined as follows.

Definition 58 (Rewards of the Game [55])**.** *If P is not conflict-free then the opponent should pay to the proponent the sum $r_G(P,O) = 0$. If P is conflict-free and O does not attack P, then the opponent should pay him the sum $r_G(P,O) = 1$. Otherwise, the opponent should pay the proponent a sum equal to $r_F(P,O) = \phi(P,O)$.*

With the strategies and rewards of the game (G,x) defined, we can now turn to its expected payoff, which is a measure of the strength of the argument x. The expected payoff is determined by the game's outcome in the long run, over a large number of repetitions. We assume that each time the game is played, proponent and opponent choose their strategies according to some probability distributions X and Y. The probability of the proponent choosing the i-th strategy (O_i) is equal to x_i, and that of the opponent choosing the jth strategy (P_j) is y_j. The proponent's expected payoff[1] is therefore

$$E = X^T RY = \sum_{j=1}^{n}\sum_{i=1}^{m} r_{i,j}x_i y_j$$

The proponent can therefore expect to get at least $\min_Y X^T RY$. The proponent can choose X so he will select X so that this minimum is as large as possible. The propo-

[1] X^T denotes the transpose of vector X and R denotes the matrix $((r_i,j))_{m\times n}$ where $r_{i,j} = r_F(P_i,O_j)$

nent's mixed strategy is therefore

$$\max_X \min_Y X^T RY$$

At the same time, the opponent chooses a strategy Y that minimises the proponent's expected payoff:

$$\min_Y \max_X X^T RY$$

By the *minmax theorem* of Neumann [89], the two quantities have the same value v:

$$\max_X \min_Y X^T RY = \min_Y \max_X X^T RY = v$$

It is v, the value of the game, that is the strength of the argument x.

Definition 59 (Argument Strength [55])**.** *The strength of an argument x in an argument graph G is noted $s_G(x)$ and defined as the value of the (G,x) game of argumentation strategy.*

The strength of x as measured by $s_G(x)$ depends only on the argument graph G. Moreover, it is determined entirely by the connected component (subgraph of G) that contains x, which can make s_G easier to compute in some situations. s_G is bounded between 0 and 1. $s_G(x) = 0$ if and only if x attacks itself, and $s_G(x) = 1$ if and only if x is not attacked by any other argument.

4.4.1.3 Impact Measure

We can use the measure s_G to assess the impact of changing a decision result Res_1 to a different result Res_2. Since the argument graph of Res_1 is conflict-free, all of its arguments have the same, maximum strength 1. The same is true for Res_2 and its arguments. Assuming $\mathsf{args}(Res_1) \neq \emptyset$, the average strength of all arguments in Res_1 is thus always

$$\frac{\sum_{a \in \mathsf{args}(Res_1)} s_{\mathsf{argGraph}(Res_1)}(a)}{|\mathsf{args}(Res_1)|} = \frac{|\mathsf{args}(Res_1)|}{|\mathsf{args}(Res_1)|} = 1$$

However, the combination of Res_1 and Res_2 may result in attacks. For example, assume that there are arguments $a \in \mathsf{args}(Res_1)$ and $b \in \mathsf{args}(Res_2)$ such that b attacks a in the combined graph $G = \mathsf{argGraph}(Res_1 \cup Res_2)$, and there are no other attacks in G. Then, $s_G(a) < 1$, but for all other arguments $a' \in \mathsf{args}(Res_1 \cup Res_2) \setminus \{a\}$, $s_G(a') = 1$. The

average strength of arguments for Res_1 in G is therefore smaller than one, and the average strength of arguments for Res_2 in G is still equal to one. We interpret this situation as having improved the justification of our design by switching from Res_1 to Res_2. If the situation was reversed and a attacked b, then we would have worsened the justification of our design, because we chose to switch from Res_1 to Res_2 in spite of the counterargument against Res_2.

Definition 60 (Average Strength)**.** *Let $G = (A, Att)$ be an argument graph and let $A' \subseteq A$ such that $A' \neq \emptyset$. The* average strength *of A' in G is defined as*

$$\mathsf{avgStrength}(A', G) = \frac{\sum_{a \in A'} s_G(a)}{|A'|}$$

Example 38. *For the decision outcomes Res_1 and Res_2 from Example 37 on page 128, we get the following arguments.*

$$\mathsf{args}(Res_1) = \{a_1, a_2, a_3, a_5, a_6, a_8\}$$
$$\mathsf{args}(Res_2) = \{a_4, a_7, a_9\}$$

As shown in Figure 4.2 on page 129, a_9 attacks a_8, a_5 and a_4 attack each other and a_5 also attacks a_7. The argument graph of $Res_1 \cup Res_2$ is therefore $G = \mathsf{argGraph}(Res_1 \cup Res_2) = (\{a_1, \ldots, a_9\}, \{(a_9, a_8), (a_5, a_4), (a_4, a_5), (a_5, a_7)\}$. The strength of the attacked arguments (a_8, a_5, a_4 and a_7) according to Definition 59 is:

$$s_G(a_4) = 0.5 \qquad s_G(a_7) = 0.41\overline{6}$$
$$s_G(a_5) = 0.5 \qquad s_G(a_8) = 0.25$$

As a result, the average strength of arguments for Res_1 is $\mathsf{avgStrength}(\mathsf{args}(Res_1), G) \approx 0.79$ and the average strength of arguments for Res_2 is $\mathsf{avgStrength}(\mathsf{args}(Res_2), G) \approx 0.96$. We conclude that in the debate over Res_1 versus Res_2, Res_2 has the stronger arguments.

We can now formalise our finding that all arguments within a single decision outcome have the same, maximum strength.

Proposition 30. *For every non-empty decision result Res,*

$$\mathsf{avgStrength}(\mathsf{args}(Res), \mathsf{argGraph}(Res)) = 1$$

Proof. Let $G = (A, Att) = \mathsf{argGraph}(Res)$. Since $Att = \emptyset$, $s_G(a) = 1$ for all $a \in A$, so the average is also 1. □

Another characteristic of avgStrength for decision outcomes is that is always positive:

Proposition 31. *Let Res_1, Res_2 be two non-empty decision results. Let $G = \mathsf{argGraph}(Res_1 \cup Res_2)$.*

Then $\mathsf{avgStrength}(\mathsf{args}(Res_1), G) > 0$ *and* $\mathsf{avgStrength}(\mathsf{args}(Res_2), G) > 0$.

Proof. Let Res_1, Res_2 be two non-empty decision results, let $G_1 = \mathsf{argGraph}(Res_1)$, $G_2 = \mathsf{argGraph}(Res_2)$ and $G = \mathsf{argGraph}(Res_1 \cup Res_2)$.

Assume $\mathsf{avgStrength}(\mathsf{args}(Res_1), G) = 0$. Then for all $a \in \mathsf{args}(Res_1)$, $s_G(a) = 0$. Since $s_G(x)$ is 0 if and only if x is self-attacking, all arguments in $\mathsf{args}(Res_1)$ are self-attacking. This contradicts Propage 24 which says that $\mathsf{argGraph}(Res_1)$ is conflict-free. The proof for Res_2 proceeds analogously. □

To determine whether choosing Res_2 over Res_1 resulted in an overall strengthening of the decision's justification we can take the difference between the average argument strength of the two.

Definition 61 (Strength-Based Difference of Decision Outcomes)**.** *Let Res_1, Res_2 be two decision outcomes, let $A_1 = \mathsf{args}(Res_1)$, $A_2 = \mathsf{args}(Res_2)$ and let $G = \mathsf{argGraph}(Res_1 \cup Res_2)$. The* strength-based difference *between Res_1 and Res_2 is*

$$\mathsf{strengthDiff}(Res_1, Res_2) = \mathsf{avgStrength}(A_2, G) - \mathsf{avgStrength}(A_1, G)$$

Example 39. *Recall that the average strength of the two results Res_1 and Res_2 in Example 38 was* $\mathsf{avgStrength}(\mathsf{args}(Res_1), G) \approx 0.79$ *and Res_2 is* $\mathsf{avgStrength}(\mathsf{args}(Res_2), G) \approx 0.96$. *Their difference is therefore*

$$\mathsf{strengthDiff}(Res_1, Res_2) \approx 0.96 - 0.79 = 0.17$$

Example 40. *Let us look at the values for* strengthDiff *for the the three decision outcomes Res_1 to Res_3. The results produce three sets of arguments, A_1 to A_3 (originally defined in the running example introduced on page 118):*

$$A_1 = \{a_1, \ldots, a_4\}$$
$$A_2 = \{a_1, a_2, a_5, \ldots, a_8\}$$
$$A_3 = \{a_1, \ldots, a_4, a_9, \ldots, a_{12}\}$$

To compute $\mathsf{strengthDiff}(Res_1, Res_2)$ *and* $\mathsf{strengthDiff}(Res_2, Res_3)$*, we need to look at the argument graphs of $Res_1 \cup Res_2$ and $Res_2 \cup Res_3$, respectively.*

For $G_I = \mathsf{argGraph}(Res_1 \cup Res_2)$, we get $G_I = (A_I, Att_I)$ with $A_I = A_1 \cup A_2$, and $Att_I = \{(a_6, a_3)\}$. For $G_{II} = \mathsf{argGraph}(Res_2 \cup Res_3)$, we get $G_{II} = (A_{II}, Att_{II})$ with $A_{II} = A_2 \cup A_3$ and $Att_{II} = \{(a_6, a_3)\}$. The differences are therefore

$$\mathsf{strengthDiff}(Res_1, Res_2) = \frac{6}{6} - \frac{3.25}{4} = 1 - 0.8125 \approx 0.19$$
$$\mathsf{strengthDiff}(Res_2, Res_3) = \frac{7.25}{8} - \frac{5}{5} = 0.90625 - 1 \approx -0.09$$
$$\mathsf{strengthDiff}(Res_1, Res_3) = \frac{8}{8} - \frac{5}{5} = 1 - 1 = 0$$

The non-zero values of $\mathsf{strengthDiff}(Res_1, Res_2)$ *and* $\mathsf{strengthDiff}(Res_2, Res_3)$ *are caused by a_6, which attacks a_4 and thus reduces its strength to* 0.25 *in the graphs of $Res_1 \cup Res_2$ and $Res_2 \cup Res_3$. When comparing Res_1 with Res_3, the average strength of their arguments does not change. This indicates that Res_3 is compatible with Res_1 in a way that will be discussed in Section 4.4.4 (page 149).*

The value of strengthDiff is zero if there are no attacks in the combined knowledge base.

Proposition 32. *If* $\mathsf{attacks}(Res_1 \cup Res_2) = \emptyset$ *then* $\mathsf{strengthDiff}(Res_1, Res_2) = 0$.

Proof. If $\mathsf{attacks}(Res_1 \cup Res_2) = \emptyset$ then for all arguments $a \in \mathsf{args}(Res_1 \cup Res_2)$, $s_G(a) = 1$, so $\mathsf{avgStrength}(\mathsf{args}(Res_2), G) - \mathsf{avgStrength}(\mathsf{args}(Res_1), G) = 1 - 1 = 0$. □

To see that Propage 32 does not hold for the other direction, consider two decision outcomes Res_1 and Res_2 with arguments $a \in \mathsf{args}(Res_1)$ and $b \in \mathsf{args}(Res_2)$ such

that in $\mathsf{argGraph}(Res_1, Res_2)$, a attacks b and vice versa. Then the average strength of $\mathsf{args}(Res_1)$ and $\mathsf{args}(Res_2)$ is less than one, but equal (and therefore their difference, as expressed in $\mathsf{strengthDiff}(Res_1, Res_2)$ is still zero).

$\mathsf{strengthDiff}(Res_1, Res_2)$ ranges from negative one to positive one. A positive sign indicates that Res_2 has stronger support since its average strength is higher, and a negative sign indicates the opposite. When comparing decision outcomes with strengthDiff we thus get an indication of the general tendency of support (whether it improved or worsened) as well as a numeric measure of the size of the change.

4.4.2 Impact Analysis Based on Knowledge Added or Removed

If we have a decision outcome Res_1 and add knowledge to it, the result will be a decision outcome Res_2 with $\mathsf{args}(Res_1) \subseteq \mathsf{args}(Res_2)$. When using abstract argument graphs (A, Att), the only statements we can make about added or removed *knowledge* (as opposed to conflict) are about the "argument" component A. In Dung's theory, the arguments in A are atomic, so any distance measure based on A must be a general distance measure for sets.

We could take into account that the argument graphs considered here are produced from decision outcomes, essentially ASPIC+ knowledge bases. However, this would create an asymmetry in our distance measures, because strengthDiff (Definition 61) does *not* take the underlying ASPIC+ knowledge base into account, and relies only on the resulting argument graph. For this reason, we will use a simple set-based distance measure. The distance measure $\mathsf{argDiff}(Res_1, Res_2)$ essentially counts how many arguments are only found in one of Res_1, Res_2.

Definition 62 (Argument-Based Difference of Decision Outcomes)**.** *Let Res_1, Res_2 be two decision outcomes. The* argument-based distance *of Res_1, Res_2 is defined as*

$$\mathsf{argDiff}(Res_1, Res_2) = |\mathsf{args}(Res_1) \Delta \mathsf{args}(Res_2)|$$

Example 41. *For the decision outcomes Res_1 and Res_2 from Example 37, we get*

$$\begin{aligned} \mathsf{argDiff}(Res_1, Res_2) &= |\mathsf{args}(Res_1) \Delta \mathsf{args}(Res_2)| \\ &= |\{a_1, a_4, a_5, a_7, a_8, a_9\}| \\ &= 6 \end{aligned}$$

The measure argDiff stays true to Dung's formalism by not assuming anything about the arguments other than the fact that they form a set. It is clear that $\mathsf{argDiff}(Res_1, Res_2)$ cannot be zero if $\mathsf{strengthDiff}(Res_1, Res_2)$ is non-zero, because each of Res_1, Res_2 is conflict-free.

Proposition 33. *For any two decision outcomes Res_1, Res_2, if* $\mathsf{strengthDiff}(Res_1, Res_2) > 0$ *then* $\mathsf{argDiff}(Res_1, Res_2) > 0$.

Proof. Let Res_1, Res_2 be two decision outcomes such that $\mathsf{strengthDiff}(Res_1, Res_2) > 0$. Let $G = (A, Att) = \mathsf{argGraph}(Res_1 \cup Res_2)$ and let $A_1 = \mathsf{args}(Res_1)$ and let $A_2 = \mathsf{args}(Res_2)$. Since $\mathsf{strengthDiff}(Res_1, Res_2) > 0$, there must be at least one attack in Att, so $Att \neq \emptyset$. Let $(a,b) \in Att$ and assume $a \in A_1$ and $b \in A_2$ (without loss of generality, by Propage 25). The argument graph $(A_1, Att_1) = \mathsf{argGraph}(Res_1)$ is conflict-free (by Propage 24), so $b \notin A_1$. Therefore, $b \in A_1 \Delta A_2$, so $A_1 \Delta A_2 \neq \emptyset$ and $\mathsf{argDiff}(Res_1, Res_2) > 0$. □

Example 42. *In terms of the running example from Section 4.3, we get the following argument differences for the decision outcomes defined in Example 34.*

$$\mathsf{argDiff}(Res_1, Res_2) = |\{a_2, a_3, a_4, a_5, a_6, a_7, a_8\}| = 7$$
$$\mathsf{argDiff}(Res_2, Res_3) = |\{a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}\}| = 11$$
$$\mathsf{argDiff}(Res_1, Res_3) = |\{a_9, a_{10}, a_{11}, a_{12}\}| = 4$$

While the strengthDiff *values for Res_1, Res_2 and Res_3 are relatively small (see page 136, Example 40), the difference in arguments between the outcomes varies considerably. The design and its justification have evolved, while conflicting justifications have been kept to a minimum, which can be interpreted as sign of a healthy decision process.*

In the following section we will take a closer look at the interplay of strengthDiff and argDiff.

4.4.2.1 Relationship between argDiff and strengthDiff

The two measures of impact correspond to the conflict-based (strengthDiff, Definition 61) and knowledge-based (argDiff, Definition 62) distance measures. The possible results of comparing decision outcomes Res and Res' with strengthDiff and argDiff, can be grouped into four categories, forming an imaginary square:

1. $\mathsf{strengthDiff}(Res, Res') = 0$ and $\mathsf{argDiff}(Res, Res') = 0$

2. $\mathsf{strengthDiff}(Res, Res') \neq 0$ and $\mathsf{argDiff}(Res, Res') = 0$

3. $\mathsf{strengthDiff}(Res, Res') = 0$ and $\mathsf{argDiff}(Res, Res') > 0$

4a. $\mathsf{strengthDiff}(Res, Res') < 0$ and $\mathsf{argDiff}(Res, Res') > 0$

4b. $\mathsf{strengthDiff}(Res, Res') > 0$ and $\mathsf{argDiff}(Res, Res') > 0$

The first case only occurs when $Res = Res'$. The second case is impossible to achieve (see Propage 34 below). This leaves us with three interesting cases: Knowledge has been changed, but there are no changes in argument strength (case 3), knowledge changed, argument strength decreased (case 4a) and knowledge changed, argument strength increased (case 4a).

Proposition 34. *For any two decision outcomes* Res, Res'*, if* $\mathsf{argDiff}(Res, Res') = 0$ *then* $\mathsf{strengthDiff}(Res, Res') = 0$.

Proof. Let $G = (A, Att) = \mathsf{argGraph}(Res \cup Res')$ and assume $\mathsf{argDiff}(Res, Res') = 0$. Then $\mathsf{args}(Res) = \mathsf{args}(Res')$, so $Res = Res'$ (by the assumption that there are no unused rules, see page 33). Since $\mathsf{args}(Res)$ is conflict-free, $\mathsf{attacks}(Res \cup Res') = \emptyset$, and we can apply Propage 32 to get $\mathsf{strengthDiff}(Res, Res') = 0$. □

We will now analyse each of the two remaining cases.

First, if $\mathsf{strengthDiff}(Res, Res') = 0$ and $\mathsf{argDiff}(Res, Res') > 0$ then either there are no conflicts in $\mathsf{argGraph}(Res \cup Res')$, or the strength of Res's arguments is exactly the same as that of the arguments of Res' (cf. the discussion of Propage 32 on page 136). In either case, the added knowledge did not result in a strengthening of the decision's justification.

If however $\mathsf{strengthDiff}(Res, Res') > 0$, then the average strength of the arguments of Res' is higher (in $Res \cup Res'$) than that of the arguments of Res. If $\mathsf{strengthDiff}(Res, Res') < 0$, then the average strength of Res is higher, and switching from Res to Res' resulted in a relative weakening of the reasons used to justify the decision.

Example 43. *For decision outcomes Res_1 and Res_3 from Example 4.3 we get* $\mathsf{strengthDiff}(Res_1, Res_3) = 0$ *and* $\mathsf{argDiff}(Res_1, Res_3) = 4$*, implying that their justifications overlap (albeit not completely) but are not inconsistent. In our running example, Res_1 is followed by Res_3, but this result shows that we could replace Res_1 with Res_3 without having to take a different stance on previously accepted arguments.*

On the other hand, if $\mathsf{strengthDiff}(r, r') > 0$, the two decision outcomes are incompatible, and replacing r with r' requires us to reject some assumptions that were made originally.

Example 44. *The two outcomes in Example 37 are not compatible, because* $\mathsf{strengthDiff}(Res_1, Res_2) = 2$*. If we replaced Res_1 with Res_2, we would have to reject arguments a_1, a_5 and a_8. This has implications for subsequent decisions that were based on Res_1: Any arguments which have a_1, a_5 or a_8 as sub-arguments will be incompatible with Res_1's replacement (Res_2) and therefore have to be adjusted.*

With strengthDiff and argDiff, we can get an idea of how different the justifications of two decisions are. They bring us closer – as alluded to in the previous two examples – to achieving our larger goal: To assess the impact of adjusting past decisions on the overall decision process, not just on the immediately affected decision outcome.

4.4.3 Impact Analysis for Decision Sequences

The two measures of impact (Definition 61 and 62) each target a pair of decision outcomes. However, when making a change to an earlier decision one is usually interested in the consequences on all subsequent decisions, not just the one immediately affected. In this section we will look at how the two impact measures can be applied to decision sequences.

Let us assume we have a decision sequence $S_1 = (Res_1, Res_2, \ldots, Res_n)$ and want to replace Res_1 with a different decision Res'_1. The result is a new decision sequence $S_2 = (Res'_1, Res_2, \ldots, Res_n)$, differing from S_1 only in its first element. We can now use strengthDiff or argDiff to compare S_1 and S_2 element-wise, by computing $\mathsf{strengthDiff}(Res_1, Res'_1)$, $\mathsf{strengthDiff}(Res_2, Res_2)$, and so on. This approach is obviously no better than simply comparing Res_1 and Res'_1 directly, because the remaining elements $Res_2, \ldots, Res_n$ are the same in both sequences. However, even though comparing Res_2 with Res_2 with Res_2 does not yield any insights, changing decision Res_1

to Res_1' potentially still affects Res_2 and other subsequent decisions. For example, if an argument a in Res_2 depends on (that is, has as a sub-argument) an argument from Res_1 that is rejected in Res_1', then a suddenly becomes inconsistent with the previous decision. The reason why this inconsistency is not flagged in our simple comparison of S_1 and S_2 is that the impact of changing Res_1 to Res_1' is not carried forward to the remaining decision outcomes. In other words, subsequent outcomes such as Res_2 may depend on Res_1 implicitly rather than explicitly.

In Section 4.3.2 we developed a toolkit for drawing out such implicit dependencies in sequences of decision outcomes: The emb operation and its counterpart extract. By applying emb and extract to the decision outcomes in S, we get a decision sequence S' in which every element is different, not just the first one. So in order to assess the impact of changing Res_1 to Res_1', we still compare the two outcomes initially (for example by computing $\mathsf{strengthDiff}(Res_1, Res_2)$, but subsequent comparisons are different. Instead of $\mathsf{strengthDiff}(Res_2, Res_2)$ we compute $\mathsf{strengthDiff}(Res_2, \mathsf{extract}(\mathsf{Emb}'_S(Res_2)))$ and so on. By using the emb operation, we propagate the change through the entire sequence S.

In order for this approach to work for changes that do not involve the very first decision outcome, we need to take the embedding of the original sequence. To get an idea of the conflicts introduced by changing S to S', with respect to the i-th outcome, we compute the strengthDiff value of the embedding (in S) of Res_i and the embedding (in S') of Res_i'.

In the following definitions, we use the term *impact measure* to denote any function from pairs of decision outcomes to real numbers, such as strengthDiff and argDiff.

Definition 63 (Impact on Decision Outcome)**.** *Let $S = (Res_1, \ldots, Res_n)$ and $S' = (Res_1', \ldots, Res_n')$ be two sequences of decision outcomes and let μ be an impact measure. The impact of changing S to S' on the i-th element is defined as*

$$\mathsf{impact}_\mu(S, S', i) = \mu(\mathsf{extract}(\mathsf{emb}_S(Res_i)), \mathsf{extract}(\mathsf{emb}_{S'}(Res_i')))$$

4.4.3.1 Example

We will dedicate the next few pages to a large example in which we illustrate the developments so far. Recall that we introduced several decision outcomes Res_1 to Res_5,

four of which are repeated here (you can find the original definition in Example 33 on page 117).

$Res_1 = O_1 \cup K_1$ with

$O_1 = \{\Rightarrow \texttt{al}\}$

$K_1 = \{r_1, r_2, r_3\}$

$r_1 = \texttt{al} \Rightarrow \texttt{strong}$

$r_2 = \texttt{al} \Rightarrow \texttt{non\_corrosive}$

$r_3 = \texttt{al} \Rightarrow \texttt{high\_weight}$

$Res_2 = O_2 \cup K_2$ with

$O_2 = \{\Rightarrow \texttt{alu}, \Rightarrow \texttt{bolts}, \Rightarrow \texttt{shim}\}$

$K_2 = \{r_1, r_4, r_5\}$

$r_4 = \texttt{bolts} \Rightarrow \neg\langle r_2 \rangle$

$r_5 = \texttt{shim}, \texttt{bolts} \Rightarrow \texttt{good\_EMC}$

$Res_3 = O_3 \cup K_3$ with

$O_3 = \{\Rightarrow \texttt{al}, \Rightarrow \texttt{no\_coating}, \Rightarrow \texttt{screws}\}$

$K_3 = \{r_1, r_2, r_3, r_6, r_7\}$

$r_6 = \texttt{al}, \texttt{no\_coating} \Rightarrow \texttt{corrosion\_resistant}$

$r_7 = \texttt{al}, \texttt{screws} \Rightarrow \texttt{good\_EMC}$

$Res_5 = O_5 \cup K_5$ with

$O_5 = \{\Rightarrow \texttt{length}(s_2, 25cm), \Rightarrow \texttt{al}, \Rightarrow \texttt{no\_coating}, \Rightarrow \texttt{screws}\}$

$K_5 = \{r_1, r_2, r_3, r_6, r_9\}$

$r_9 = \texttt{length}(s_2, \leq 30cm) \Rightarrow \neg\texttt{good\_EMC}$

In this example we will focus on the process (Res_1, Res_3, Res_5) and change it to (Res_2, Res_3, Res_5) – that is, the first element of the process will be changed from Res_1 to Res_2. The original process (Res_1, Res_3, Res_5) is straightforward because it produces a monotonically growing set of options – in other words, the design is more and more specialised but never reverted. As explained in Example 33, Res_1 is not followed by Res_2 because some of the assumptions in Res_1 would be invalidated by Res_2. Res_4 is produced in parallel to Res_3 and merged into the main process in the next step, Res_5. We can use impact_μ to analyse the impact of changing $S = (Res_1, Res_3, Res_5)$ to $S' =$

(Res_2, Res_3, Res_5).

The process involves quite a few arguments and complex calculations. In order to make it easier for the reader to follow along, we have created a diagram of the entire process in Figure 4.3 on page 145, and listed all arguments in Table 4.3 on the same page.

How to read Figure 4.3 The diagram on page 145 shows two versions of the same decision process, namely $S = (Res_1, Res_3, Res_5)$ and $S' = (Res_2, Res_3, Res_5)$. Each column corresponds to a stage (for example, the leftmost column displays Res_1 in S and Res_2 in S'). The three rows show arguments found only in S (top row), arguments found in both processes (middle row) and arguments found in S' only (bottom row). Arrows between arguments indicate attacks, as usual.

It is important to note that the arguments shown are those of emb_S and $\mathsf{emb}_{S'}$, respectively - that is why we also see the three arguments a_A, a_B and a_C, which were not part of S or S', but are a product of the enforcement operation: a_A and a_B attack arguments a_5 and a_7, making explicit the fact that `shim` and `bolts` are incompatible with our choice of `screws` in stage 2 and therefore have to be rejected. a_C attacks argument a_{12} (in order to enforce a_{15}), because by choosing Res_3 we accepted the argument *against* `good_EMC` and thus rejected our earlier argument a_{12} which stated the contrary.

The diagram visualises a number of interesting properties of decision processes and their embeddings. First, we can see that the arguments at each stage are consistent - as evidenced by the fact that all "attacks" arrows are drawn across the boundaries of a decision outcome.

Second, we made use of Definition 54 Cond. 1, which states that in an embedding of a decision process, each stage must contain the knowledge of its predecessor. Because of this condition we only display arguments when they are first introduced, and do not have to repeat them later on. To find out, for example, which arguments are part of emb_S at stage three, we need to consider the first three columns in the top and middle rows. To see which arguments are part of $\mathsf{emb}_{S'}$ at stage two, we need to take the first two columns of the middle and botton rows.

Third, the diagram illustrates how enforce works as part of the Embed operation. This is indicated by a_A to a_C: If we take the argument graph for $\mathsf{emb}_{S'}$ at stage two (con-

taining arguments a_1, a_2, $a_5 \ldots a_{12}$, a_A and a_B), we can see that its grounded extension consists of $\{a_1, a_2, a_9 \ldots a_{12}, a_A, a_B\}$. Specifically, it contains all of Res_2's arguments. Without a_A and a_B, argument a_{11} would not be sceptically acceptable. It is however sceptically acceptable in this case because of the use of enforce in the definition of Embed (page 123).

Finally, we can see from the diagram what the result of extract is for any of the stages in S or S'. For example, to get $\mathsf{extract}(\mathsf{emb}_{S'}(Res_2))$ we have to look at the middle and bottom rows (because they contain arguments for S'), and take the argument graph for the cells in the first two columns (because Res_2 is the second result, and each stage contains all rules introduced at previous stages). We then take the grounded extension of this argument graph to get the result of extract. In our case, the grounded extension to consider is $\{a_1, a_2, a_9 \ldots a_{12}, a_A, a_B\}$, so we can simply take the union of the knowledge of those arguments to get $\mathsf{extract}(\mathsf{emb}_{S'}(Res_2))$.

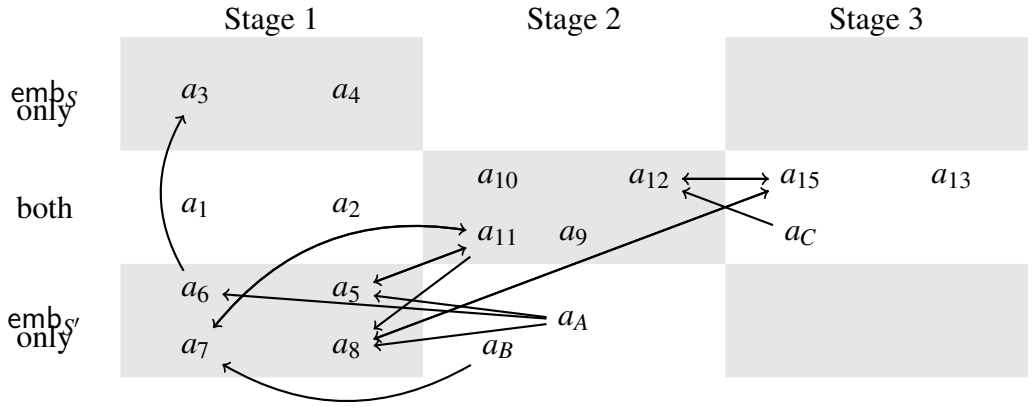


Figure 4.3: Impact of changing from S to S' – page 143

Table 4.3: Arguments for Section 4.4.3.1

Name	Argument	Attacks
a_1	$[\Rightarrow \texttt{al};\texttt{al}]$	-
a_2	$[a_1;\texttt{al} \Rightarrow \texttt{strong};\texttt{strong}]$	-
a_3	$[a_1;\texttt{al} \Rightarrow \texttt{non\_corr.};\texttt{non\_corr.}]$	-
a_4	$[a_1;\texttt{al} \Rightarrow \texttt{high\_weight};\texttt{high\_weight}]$	-
a_5	$[\Rightarrow \texttt{bolts};\texttt{bolts}]$	a_{11}
a_6	$[a_5;\texttt{bolts} \Rightarrow \neg\langle r_2\rangle;\neg\langle r_2\rangle]$	a_3
a_7	$[\Rightarrow \texttt{shim};\texttt{shim}]$	a_{11}
a_8	$[a_5,a_7;\texttt{shim},\texttt{bolts} \Rightarrow \texttt{good\_EMC};\texttt{good\_EMC}]$	a_{15}
a_9	$[\Rightarrow \texttt{no\_coating};\texttt{no\_coating}]$	-
a_{10}	$[a_1,a_9;r_6;\texttt{corrosion\_resistant}]$	-
a_{11}	$[\Rightarrow \texttt{screws};\texttt{screws}]$	a_5,a_7
a_{12}	$[a_1,a_{11};r_7;\texttt{good\_EMC}]$	a_{15}
a_{13}	$[\Rightarrow \texttt{length}(s_2,25cm);\texttt{length}(s_2,25cm)]$	-
a_{15}	$[a_{13};r_9;\neg\texttt{good\_EMC}]$	a_{12},a_8
a_A	$[\Rightarrow \neg\langle\Rightarrow \texttt{bolts}\rangle]$	a_5,a_6,a_8
a_B	$[\Rightarrow \neg\langle\Rightarrow \texttt{shim}\rangle]$	a_7
a_C	$[\Rightarrow \neg\langle r_7\rangle]$	a_{12}

Impact Analysis To measure the impact of changing Res_1 to Res_2, we proceed step by step through the decision sequence. First, we analyse the impact on the first element using the $\mathsf{strengthDiff}(KB, KB')$ measure. The diagram on page 145 is helpful here, too: To see what KB and KB' should be for this invocation of $\mathsf{strengthDiff}$, we can simply set KB (the "old" knowledge base) to contain the knowledge for all arguments in the top left and middle left cells, and KB' (the new knowledge base) to contain all arguments in the middle left and bottom left cells. This results in the following calculation:

$$
\begin{aligned}
\mathsf{impact}_{\mathsf{strengthDiff}}(S, S', 1) &= \mathsf{strengthDiff}(\mathsf{extract}(\mathsf{emb}_S(Res_1)), \mathsf{extract}(\mathsf{emb}_{S'}(Res_2))) \\
&\qquad (\text{By Propage 29}) \\
&= \mathsf{strengthDiff}(Res_1, Res_2) \\
&\qquad (\text{Cf. Example 40 on page 136}) \\
&\approx 0.19
\end{aligned}
$$

Calculating the impact on the second element is a little more complex.

$$
\begin{aligned}
\mathsf{impact}_{\mathsf{strengthDiff}}(S, S', 2) &= \mathsf{strengthDiff}(\mathsf{extract}(\mathsf{emb}_S(Res_3)), \mathsf{extract}(\mathsf{emb}'_S(Res_3))) \\
&\qquad (\text{Definition 55 and } Res_1 \cup Res_3 \text{ conflict-free}) \\
&= \mathsf{strengthDiff}(Res_1 \cup Res_3, \mathsf{enforce}(\mathsf{args}(Res_3), Res_2 \cup Res_3)) \\
&\qquad (Res_1 \subseteq Res_3) \\
&= \mathsf{strengthDiff}(Res_3, \mathsf{enforce}(\mathsf{args}(Res_3), Res_2 \cup Res_3)) \\
&\qquad (K_{23} := \mathsf{enforce}(\mathsf{args}(Res_3), Res_2 \cup Res_3)) \\
&= \mathsf{strengthDiff}(Res_3, K_{23}) \\
&\qquad (G_2 := \mathsf{argGraph}(K_3 \cup K_{23}); \text{ see discussion below}) \\
&= \frac{10.75}{12} - \frac{7.5}{8} \\
&\approx 0.896 - 0.9375 \\
&\approx -0.0415
\end{aligned}
$$

Here, we examine strengthDiff for Res_2 and K_{23}, which together result in the graph $G_2 = (A_2, Att_2) = \mathsf{argGraph}(Res_2 \cup K_{23})$. As we can see in Figure 4.3, this graph contains arguments $A_2 = \{a_1, \ldots, a_{12}, a_A, a_B\}$ and has the following attacks: $a_6 \to a_3$, $a_A \to a_6$, $a_5 \leftrightarrow a_{11}$, $a_7 \leftrightarrow a_{11}$, $a_B \to a_7$, $a_A \to a_5$, $a_A \to a_8$ and $a_{11} \to a_8$.

The value of $\mathsf{strengthDiff}(Res_3, K_{23})$ is negative, because the average strength in $\mathsf{enforce}(\mathsf{args}(Res_3), Res_2 \cup Res_3)$ is lower than the average strength of arguments in Res_3 alone.

We can now compute the impact of changing Res_1 to Res_2 on the last element in the sequence, Res_5. The method is the same as in the previous two examples so we will elide some intermediate steps.

$$
\begin{aligned}
\mathsf{impact}_{\mathsf{strengthDiff}}(S, S', 3) &= \mathsf{strengthDiff}(\mathsf{extract}(\mathsf{emb}_S(Res_5)), \mathsf{extract}(\mathsf{emb}_{S'}(Res_5))) \\
&\qquad (\ldots, G_3 := (A_3, Att_3)) \\
&= \frac{13.25}{15} - \frac{8.25}{11} \\
&= 0.88\overline{3} - 0.75 \\
&\approx 0.13
\end{aligned}
$$

In the last step, the strengthDiff measure increased slightly, because one of Res_3's arguments is defeated - namely a_{12} the arguments for `good_EMC`, which was deactivated in order to accomodate a_{15}. The values 0.19 and 0.13 (first and third calculation) indicate a rather large increase in the average strength of arguments (considering that the strength is between zero and one). The impact on the second step, measured as -0.0415, is smaller. This shows that changing S to S' overall had a positive effect on argument strength. To complete this example, we give the values for $\mathsf{impact}_{\mathsf{argDiff}}(S, S', i)$ without much detail:

$$
\begin{aligned}
\mathsf{impact}_{\mathsf{argDiff}}(S, S', 1) &= |\{a_3, \ldots, a_8\}| &&= 6 \\
\mathsf{impact}_{\mathsf{argDiff}}(S, S', 2) &= |\{a_3, \ldots, a_8, a_A, a_B\}| &&= 8 \\
\mathsf{impact}_{\mathsf{argDiff}}(S, S', 3) &= |\{a_3, \ldots, a_8, a_A, a_B\}| &&= 8
\end{aligned}
$$

The knowledge-based impact (as measured with argDiff) of choosing Res_2 over Res_1 is most strongly evident in the first two elements of the sequence, where it increases from 6 to 8. After that, the impact remains the same, because no additional arguments have to be changed in Res_5 to accomodate Res_2.

4.4.3.2 Impact on Decision Sequences

In order to get an indication of the impact of change on an entire decision sequence, we can simply take the sum of the impact on its elements.

Definition 64 (Impact on Decision Sequence)**.** *Let $S = (Res_1, \ldots, Res_n)$ and $S' = (Res'_1, \ldots, Res'_n)$ be two sequences of decision outcomes and let μ be an impact measure. The impact of changing S to S' is defined as*

$$\mathsf{impact}^*_\mu(S, S') = \sum_{1 \leq i < n} \mathsf{impact}_\mu(S, S', i)$$

where $S' = (Res'_1, Res_2, \ldots, Res_n)$ is the changed sequence.

Example 45. *Given the results from Example 4.4.3.1, we can easily compute* $\mathsf{impact}^*_{\mathsf{strengthDiff}}(S, S')$ *and* $\mathsf{impact}^*_{\mathsf{argDiff}}(S, S')$*:*

$$\mathsf{impact}^*_{\mathsf{strengthDiff}}(S, S') \approx 0.19 - 0.0415 + 0.13 = 0.2785$$
$$\mathsf{impact}^*_{\mathsf{argDiff}}(S, S') = 6 + 8 + 8 = 22$$

Since these values on their own do not have a unit, they gain meaning only by comparing the impact of several different changes. With argDiff*, as with* strengthDiff *before, the impact increases as we proceed in the decision sequence.*

Before we conclude this section we will show that impact_μ is well-behaved: If any two decision sequences S, S' have a prefix in common (that is, some initial decisions are unchanged), then impact_μ will only yield a non-zero value *after* the change was made.

Proposition 35. *Let S, S' be two decision sequences of the same length n, let T be a decision sequence of length k with $1 < k < n$ such that T is a prefix of both S and S'. Let μ be an impact measure with $\mu(KB, KB') = 0$ if $KB = KB'$. (identity property). Then for all $l \leq k$,*

$$\mathsf{impact}_{\mu}(S,S',l) = 0$$

Proof. Follows from Definition 63 and the fact that $\mu(Res_1, Res_2) = 0$ if $Res_1 = Res_2$ (identity). □

The impact measures developed in this section demonstrated the value of our two-tiered model of decision sequences. Example 4.4.3.1 in particular shows how complex argument graphs arise from relatively simple definitions such as decision outcomes, embeddings and the two impact measures.

4.4.4 Progress in Decision Sequences

In the paragraphs above we developed two ways of measuring change in decision outcomes, one based on knowledge that was added or removed, and one based on conflict arising from a change of mind. We applied strengthDiff and argDiff to two outcomes of the same decision, in order to gauge the difference in argument strength and number between them.

Instead of analysing the difference between two outcomes of the same decision, we are now going to look at the difference between two decision outcomes Res_1 and Res_2 where Res_1 was *followed by* Res_2, instead of replaced by it, resulting in a characterisation of the "progress" of a decision sequence. That is, we get an impression of how the coherence (conflict-freeness) and justification of decisions evolves.

The difference between Res_2 replacing Res_1 and Res_2 following Res_1 is that in the first case, the option represented by Res_1 is not part of the eventual design anymore, because Res_1 was completely replaced by Res_2. In the second case, both Res_1 and Res_2 are part of the final design, and Res_2 is not a revision of Res_1 but a refinement of it. When talking about decision outcomes in this section we will always use their embeddings (see Section 4.3.2 on page 118) to ensure that all implicit assumptions are included in the analysis.

Again, we characterise progress in terms of arguments, not for example in terms of how many of requirements have been met and how many are still open. This approach is a good demonstration of the unique contribution that an argumentation-based model such as ours may bring to the management of engineering design processes.

In the following sections (4.4.4.1 to 4.4.4.3), we define three binary relations of

outcomes: monotonic, weakening and alteration. These relations can be applied to decision sequences by applying them to successive pairs of outcomes. For example, in a decision sequence $(Res_1, Res_2, Res_3, Res_4)$ we may find that the transition from Res_1 to Res_2 was monotonic, and the transition from Res_2 to Res_3 was an alteration. We then express each of the three relations in terms of argDiff and strengthDiff.

4.4.4.1 Monotonic

Ideally, decision processes advance linearly towards their target, without any change of requirements, reversals of decisions or other detours. The justifications of individual decisions (i.e. the design documents) can be collated to an overall design without any inconsistencies. Therefore, this kind of progress does not introduce any attacks on previously accepted arguments, and results in conflict-free embeddings.

In our formal model, a monotonic transition from Res_1 to Res_2 has the property that all of Res_1's arguments are part of the grounded extension of $\mathsf{emb}(Res_2)$, that is, they are still acceptable in Res_2.

Definition 65 (Monotonic)**.** *A sequence of decision outcomes $S = (Res_1, Res_2)$ is* monotonic *iff*

$$\mathsf{args}(Res_1) \subseteq \Sigma_{\mathsf{gr}}(\mathsf{argGraph}(\mathsf{emb}_S(Res_2)))$$

Example 46. *(Res_1, Res_3) from Example 34 is monotonic.*

An equivalent definition of *monotonic* is that the argument graph of $\mathsf{emb}_S(Res_2)$ has an empty attacks-relation, as the following result shows.

Proposition 36. *A sequence of decision outcomes $S = (Res_1, Res_2)$ is monotonic if and and only if*

$$\mathsf{attacks}(\mathsf{argGraph}(\mathsf{emb}_S(Res_2))) = \emptyset$$

Proof. ($\Leftarrow$) Let $S = (Res_1, Res_2)$ such that $\mathsf{attacks}(\mathsf{argGraph}(\mathsf{emb}_S(Res_2))) = \emptyset$. Let $a \in \mathsf{args}(Res_1)$. By Definition 54 Cond. 1, $Res_1 \subseteq \mathsf{emb}_S(Res_1)$, and by Definition 54 Cond. 3, $\mathsf{emb}_S(Res_1) \subseteq \mathsf{emb}_S(Res_2)$, so $a \in \mathsf{args}(\mathsf{emb}_S(Res_2))$.

Since $\mathsf{attacks}(\mathsf{argGraph}(\mathsf{emb}_S(Res_2))) = \emptyset$, $a \in \Sigma_{\mathsf{gr}}(\mathsf{argGraph}(\mathsf{emb}_S(Res_2)))$. Therefore S is monotonic.

($\Rightarrow$) Let $S = (Res_1, Res_2)$ be a decision sequence such that S is montonic. Let $G_2 = \mathsf{argGraph}(\mathsf{emb}_S(Res_2))$. Assume that $\mathsf{attacks}(G_2) \neq \emptyset$ (Proof by contradiction). Then there exist two arguments $a_1, a_2 \in \mathsf{args}(\mathsf{emb}_S(Res_2))$ such that $(a_1, a_2) \in \mathsf{attacks}(G_2)$. Since both $\mathsf{argGraph}(Res_1)$ and $\mathsf{argGraph}(Res_2)$ are conflict-free, the attack (a_1, a_2) must be such that

1. $a_1 \in \mathsf{args}(Res_1)$ and $a_2 \in \mathsf{args}(Res_2)$ or
2. $a_1 \in \mathsf{args}(Res_2)$ and $a_2 \in \mathsf{args}(Res_1)$ or
3. a_1 or a_2 were introduced by Definition 55 Cond. 2, reactivate.

In case (1), $a_1 \in \mathsf{args}(G_2)$ (by the assumption that S is monotonic, Definition 65). Then, a_2 is attacked by an argument in the grounded extension and therefore $a_2 \notin \Sigma_{\mathsf{gr}}(G_2)$. This violates condition 2 of Definition 54, so emb_S is not an embedding of S, which contradicts Theorem 5.

In case (2), by Definition 54 Cond. 2, $a_1 \in \Sigma_{\mathsf{gr}}(G_2)$ and therefore $a_2 \notin \Sigma_{\mathsf{pgrr}}(G_2)$. This contradicts the assumption that S is monotonic (Definition 65).

In case of (3), there is an "underlying" attack (a_2, a_2') by. Definition 48 which can be reduced to case (1) or (2). □

Proposition 37. *If a decision sequence $S = (Res_1, Res_2)$ is monotonic then* $\mathsf{strengthDiff}(Res_1, Emb_S(Res_2)) = 0$.

Proof. From Propage 36 we know that $\mathsf{attacks}(\mathsf{argGraph}(\mathsf{emb}_S(Res_2))) = \emptyset$, and we can apply Propage 32 to get $\mathsf{strengthDiff}(Res_1, Emb_S(Res_2)) = 0$. □

The opposite direction of Propage 37 does not hold, because Propage 32 also only holds for the "if-then" case (see the discussion on page 136 for a counterexample).

However, if Res_1 and Res_2 are embedded in some larger decision process $(\ldots, Res_1, Res_2, \ldots)$ then their counterparts Res_1' and Res_2' are not necessarily conflict-free anymore, because attacking arguments may have been introduced by earlier decisions.

4.4.4.2 Weakening

The support for a decision Res_1 is weakened in $\mathsf{emb}_S(Res_2)$ if the option of Res_2 subsumes the option of Res_1, but the arguments of Res_2 defeat at least one argument of

Res_1. In this case, the design that was agreed on in Res_1 has not been altered (only specialised) in Res_2, but its support is weaker.

Definition 66 (Weakening)**.** *In a sequence of decision outcomes $S = (Res_1, Res_2)$, Res_1 is* weakened *by Res_2 if*

1. $\mathsf{option}(Res_1) \subseteq \mathsf{option}(Res_2)$ *and*

2. $\exists a \in \mathsf{args}(Res_1)$ *such that* $a \notin \Sigma_{\mathsf{gr}}(\mathsf{argGraph}(\mathsf{emb}_S(Res_2)))$

Example 47. *In Example 34, the transition (Res_1, Res_2) is a weakening one, because argument $a_3 = [a_1; \texttt{aluminium} \Rightarrow \texttt{non\_corrosive}; \texttt{non\_corrosive}]$ is not part of the grounded extension of Res_2's embedding. It is attacked by argument a_6.*

In a practical application of the theory, occurrences of weakening should be flagged to the user, because they imply that some of the arguments used to justify a decision were attacked (invalidated) later on, even though the decision itself has not been changed. Weakening may be an accidental side effect of decision making.

4.4.4.3 Alteration

As a generalisation of weakening (Definition 66), it is possible that an argument of Res_1 is defeated in $\mathsf{emb}_S(Res_2)$, without $\mathsf{option}(Res_1) \subseteq \mathsf{option}(Res_2)$.

Definition 67 (Alteration)**.** *A sequence of decision outcomes $S = (Res_1, Res_2)$ is an* alteration *iff*

$$\exists a \in \mathsf{args}(Res_1) \text{ such that } a \notin \Sigma_{\mathsf{gr}}(\mathsf{argGraph}(\mathsf{emb}_S(Res_2)))$$

Example 48. *In Example 34, the transition Res_2, Res_3 is an alteration, because the arguments $a_5 = [\texttt{bolts}]$ and $a_7 = [\texttt{shim}]$ are part of Res_2 but not of Res_3.*

The following result shows that any change in the option of a decision outcome results in either an alteration or a monotonic transition.

Proposition 38. *Let Res_1, Res_2 be two decision outcomes with $\mathsf{option}(Res_1) \neq \mathsf{option}(Res_2)$. Then (Res_1, Res_2) is either monotonic or an alteration, but not both.*

Proof. Let $Res_1 = (O_1, K_1)$ and $Res_2 = (O_2, K_2)$ be two decision outcomes with $O_1 \neq O_2$.

Monotonic or alteration... Either $O_1 \subseteq O_2$ or not. If $O_1 \subseteq O_2$, then either (a) $\mathsf{args}(Res_1) \subseteq \Sigma_{\mathsf{pr}}(\mathsf{argGraph}(\mathsf{emb}_S(Res_2)))$ or (b) $\mathsf{args}(Res_1) \nsubseteq \Sigma_{\mathsf{pr}}(\mathsf{argGraph}(\mathsf{emb}_S(Res_2)))$. In case (a), (Res_1, Res_2) is monotonic by Definition 65. In case (b), there exists an argument $a \in \mathsf{args}(Res_1)$ such that $a \notin \Sigma_{\mathsf{gr}}(\mathsf{argGraph}(\mathsf{emb}_S(Res_2)))$. Then, (Res_1, Res_2) is an alteration by Definition 67.

If $O_1 \nsubseteq O_2$ then there exists a literal $l \in O_1$ such that $l \notin O_2$. Hence there is an argument $[l] \in \mathsf{args}(Res_1)$ with $[l] \notin \mathsf{args}(Res_2)$ and (Res_1, Res_2) is an alteration by Definition 67.

... but not both Follows from the logical form of the two defitions. □

In practical applications, alterations may be a sign of a healthy decision process, because they occur if a previous error has been corrected (as shown in Example 48). However, it is clear from the definition that every instance of weakening is also an alteration, so any alterations should be analysed further.

4.4.5 Summary

In the beginning of this section we tried to evaluate the impact that changing one's mind has on past decisions. We studied the impact by comparing the decision process before and after the change was made. We introduced two ways of measuring impact. The first method is *strength-based* and it derives from the change in argument strength (strengthDiff, Definition 61 on page 135). The second method is *knowledge-based* and it counts the number of acceptable arguments added or removed through the change (argDiff, Definition 62 on page 137). In Example 4.4.3.1 we demonstrated both methods in detail, using the case study introduced at the beginning of this chapter.

In the second part of this section, we looked at the difference not between decision sequences, but between their individual steps. We developed a set of properties that characterise progress in decision sequences, namely monotonic, weakening and alteration. The distinction between monotonic and alteration is made purely on the status of the arguments in the first outcome r_1 - either all of them are still acceptable in r_2 (monotonic) or they are not (alteration). Weakening as a subclass of alteration is perhaps the most interesting kind of change from a practical perspective, because it may

indicate that a decision has unintended consequences.

The three properties that characterise progress rely on the notion of decision sequences and embeddings, for making implicit conflicts and assumptions visible.

4.5 Practical Implications

In the last section of this chapter we explain how the theory may be put to use in a practical application and show that our theoretical model implements part of the original vision of project DEEPFLOW. This serves two closely related purposes: First, to demonstrate a novel potentially very fruitful practical application of argumentation theory, and second, to provide motivation for the research efforts still needed to remove the obstacles that stand between the present theoretical model and its practical implementation.

4.5.1 Decision Outcomes Represent Design Documents

The outcome *Res* of a decision stage (Definition 52) forms a conflict-free argument graph, as shown in Propage 24. This graph contains all arguments in favour of the option. *Res* can be seen as the formal representation of a design document that describes why a certain option was chosen. As we discuss in the introduction (Section 1.2.1 on page 12), this is the typical structure of engineering design documents, which focus on the reasons in favour of a particular option (arguments pro). Arguments against that option are not given explicitly, only implicitly in form of counter-arguments supporting the option. This is due to the rhetorical structure of design documents, which aims to persuade the reader that the right option was chosen.

Our definition of the outcome of a decision stage (Definition 52) therefore has the advantage that, in theory, tuples of option and relevant knowledge could be extracted from design documents without having to include *all* available options and all the expert knowledge that was utilised in the decision making. If we define decision processes as sequences of decision outcomes *Res* then our definition matches the textual artifacts produced by actual decision processes.

An important consequence of this finding concerns the use of Natural Language Processing (NLP) techniques for extracting arguments from text: One should not look for arguments, counterarguments, and counter-counterarguments within the same doc-

ument, because most of the times the arguments in one document support the same conclusion. Instead, one should focus on the "positive" relationship between arguments in a single document (that is, the support or sub-argument relationship), and look for inconsistencies in the combination of several documents, for example by embedding them in a decision sequence (see Section 4.3.2).

4.5.2 Impact Analysis

It is common knowledge in engineering (and also in other industries such as programming) that there is a positive correlation between the time it takes for mistakes to be noticed and the cost of fixing them [90]. Using the methodology we developed in Section 4.4, we can not only turn this truism into a formal proposition (see Propage 35) but also, and more importantly, we can give an estimation of how many decisions are affected by change, and even to what degree.

4.5.3 Visualising Decision Processes

Decision sequences (Definition 52) are one-dimensional since the position of each decision outcome in the sequence is defined entirely by its predecessor and successor. Decision outcomes embedded in a decision process may have relationships that are more interesting, for example those examined in Section 4.4.4.

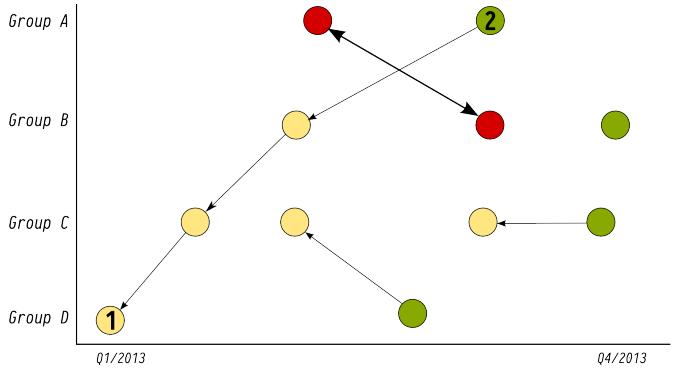


Figure 4.4: DEEPFLOW: Visualisation of a set of design documents.

The original plans for project DEEPFLOW included a visualisation of the set of design documents for an engineering project that would give an impression of the "health" of the design – how many decisions have been reversed or are in conflict with other decsions. A sketch of this visualisation is shown in Figure 4.4.

We see a set of design decisions, represented by circles in three different colours.

In the DEEPFLOW specification, the meaning of the colours was given as follows.

- **Green:** "Good decision", no conflicts with other decisions
- **Yellow:** "Decision Changed", decision was changed by a later decision
- **Red:** "Conflict", decision is incompatible with other decision

This classification of decisions maps directly to the three different kinds of progress we identified in Section 4.4.4. Decisions followed by monotonic transitions can be coloured green, those followed by alterations yellow and decisions that are weakened by subsequent decisions can be coloured red. The arrows in the diagram indicate a "follows' relationship between options, so an arrow from A to B means that B follows A – they are part of a sequence $(\ldots, A, B, \ldots)$.

The two dimensions of the graph in Figure 4.4 are *time* (x-axis) and *organisational responsibility* (y-axis) - that is, which group (of engineers) was responsible for a decision. The very first decision, marked (1), was made by group D. It was subsequently revised several times, most recently by group A, marked (2). The two red arrows are an indication that groups A and B have based their recent decisions on conflicting assumptions, and should get together and restore coherence. The graph also shows that group C has made the most decisions, although three out of its four decisions have been revised later on, as indicated by their yellow colour. If the obstacles described in Section 4.5.1 can be overcome, then our theoretical model of decision processes is able to produce such diagrams automatically from a set of design documents, giving a high-level insight into the consistency of engineering design processes.

4.5.3.1 Characterising Decisions

The analysis so far has been based on sequences of decision outcomes, but not on the different options that were considered for each decision. Due to the nature of design documentation, having a record of those options that were discussed but ultimately rejected is a much stronger assumption (see the discussion on page 154). In case we do have access to this data we can enrich the visualisation with information about options that were considered. For example, we can apply the decision rules from Section 3.3.4 (page 62) of the previous chapter to characterise the optimism of decision makers.

4.5.3.2 Example

We will conclude this section with visualisation of the running example originally introduced on page 118. The example concists of five decision results, Res_1 to Res_5. We did not discuss the organisational structure in which the process took place, except that decision Res_4 was made in parallel to Res_3, and both work streams were merged in Res_5. We will therefore assume two teams: Team A is responsible for decisions Res_1, Res_2, Res_3 and Res_5, and team B is responsible for Res_4 only.

The individual arguments of this process are shown on page 145. The diagram in Figure 4.5 below shows what the process looks like in the DEEPFLOW visualisation. Res_1 and Res_3 are yellow because they were altered by subsequent decisions (Res_1 by Res_3 and Res_3 by Res_5). Res_2 is red because it is a weakening of Res_1.

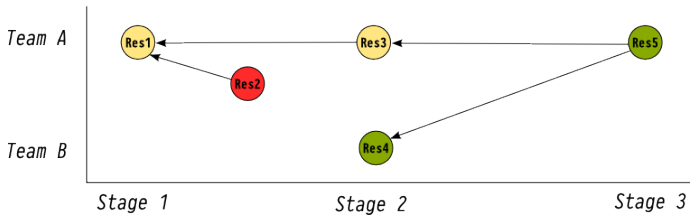


Figure 4.5: Visualisation of example process.

4.5.3.3 Discussion

We started with a minimal representation of decision outcomes (in Definition 52, added a notion of embedding them in the larger context of decision processes (Definition 54), and then looked at the possible relationships of decision outcomes within embeddings (Section 4.4.4). The three properties we identified – no changes, changed decisions, unchanged decisions with weakened support – can be directly mapped to the three categories that had been thought up for project DEEPFLOW, before any of the theory was developed. We believe that this gives additional weight to the validity of our theory.

Of course, there are many problems to be solved before a collection of design documents can be turned into the diagram in Figure 4.4, but our model of decision

processes is a first step in that direction.

4.6 Discussion

4.6.1 Related Work

To the best of our knowledge, this is the first attempt at using formal methods of argumentation for a model of engineering design processes that takes the history and evolution of decisions into account. We can find related work in two areas: Design rationale and measures of argument strength.

4.6.1.1 Design Rationale

Design rationale (DR) is a diagrammatic method for capturing the reasons "why an artifact, or some part of it, is designed the way it is." [91, page 1] Originating in engineering design [9] it has been applied to various other areas including software design [92, 93], architecture [94] and ontology engineering [95].

Each of the cited proposals has its own reference implementation, providing a working definition of DR for a particular context. As a result, there is no consensus on a generally accepted, formal definition.

Notion of Argument in Design Rationale One of the most profound differences between design rationale approaches on the one hand and models based on formal argumentation (including the one in this thesis) on the other hand is in the concept of arguments. A significant proportion[2] of DR implementations are based on the Issue-Based Information System (IBIS [96]) with a "semi-formal graphical format for laying out the structure of arguments" [91, page 5]. In IBIS, decisions are formulated as *issues* with one or more associated *positions*, which in turn are supported to or contradicted by *arguments*. Each position is either accepted (resolved) or rejected from the issue. The format is called semi-formal because there is no information beyond this hierarchical structure of issues, positions and argumentes. For example, it is not possible to infer, say, conflicts between issues based on their arguments. The same applies to other DR implementations, not based on IBIS. This limitation has been recognised as a major drawback of design rationale (see Hu *et al.* [91, page 10] and Lee [8, page 84]).

The INFORAT system [9] is a notable exception because it adds the ability to

[2] About a third of those surveyed by Hu *et al.* [91]

make inference over a design's rationale – for example, it can flag that two conflicting positions are supported by the same arguments. INFORAT captures the history of a decision and it uses a list of argument schemes (called "standard claim vocabulary"), so it is relevant both for the developments in this chapter (decision processes) and the next (argument schemes and meta-argumentation). However, its reasoning capabilities fall short of those of our model in several ways. First, arguments based on a standard claim vocabulary can only be compared syntactically, and not based on an underlying theory (for example, it is not possible to infer that one argument undermines another in INFORAT). This also means that too many arguments may be flagged as conflicting, because it is very well possible that the same argument may be used as support for two conflicting options. In our system, this phenomenon would result in a non-empty intersection of the two sets of arguments (preferred extensions) in support of the options. Second, although the concept of revising decisions exists in INFORAT, decisions are still studied in isolation. In contrast, our model reflects the dynamics of decision processes comprising many individual decisions, enabling for example impact analysis as described in Section 4.4.

Producing Design Rationale Lee [8] lists the methods by which design rationale is produced: Reconstruction (documenting decisions after the fact), record-and-replay (recording all information produced in the design process), methodological byproduct (following a specific design methodology which produces design rationale as an artifact), apprentice (getting the system to ask questions about the process) and automatic generation. Of these, reconstruction produces the highest quality output, but also imposes the highest additional effort on the design process. This classification can clearly be applied to our own approach, as any of these methods could in theory be used to produce formal knowledge bases. The discussion in the previous paragraphs supports our view that the justification for a design should be extracted automatically, if at all possible.

4.6.1.2 Measures of Argument Strength

The impact measure defined in Section 4.4.1.3 uses the change in argument strength (Definition 61 on page 135) as a measure of the weakening or strengthening of a decision justification after choosing a different option. It relies on the game-theoretic

measure of argument strength by Matt and Toni [55].

When we introduced the notion of argument strength on page 130, we distinguished two kinds of strength: Extensional (argument strength is provided as input to the system) and intensional (argument strength is derived from data that is already in the system). One of the goals of our model has been to require as little input as possible beyond the actual arguments that make up decisions, which is why we chose the intensional approach to argument strength, with the specific proposal by Matt and Toni [55].

Let us note here that most approaches to argument strength fall into the "extensional" category. For example, the CARNEADES argumentation system [97] describes argument strength with a partial order of arguments, similar to the ASPIC+ system in its original form (cf. page 31). In the decision-making framework by Amgoud and Prade [33], the strength of arguments pro and con an option is also given as a partial order. The framework for evaluative arguments by Carenini and Moore [98], argument strength is a function of evidence strength which in turn is derived from the preferences of the reader. In the proposal by Dunne *et al.* [99], it is attacks that are weighted, rather than arguments. In preference-based argumentation frameworks by Modgil [100], argument strength is based on preferences of the audience and the values promoted by an argument. Argument strength has been explored in a number of other settings within argumentation, such as compensation-based semantics [101]

Another approach is to equate the strength of an argument with its certainty. The paper by Hunter [102] explores probabilistic argument graphs, graphs equipped with a probability function of their arguments. In structured argumentation, the proposal by Hunter [61] uses a probability distribution on sentences of the logical language to derive probabilities of extensions of arguments. Krause *et al.* [103] define arguments as proofs in a logic of argumentation, and represent uncertainty as a lattice over those proofs. This approach allows for a number of different models of uncertainty.

Inconsistency In our model we define the impact of changing a decision to be the change in argument strength of the decision's support. Alternatively, we can define the impact to be the change in inconsistency of the overall design. There exist a number of ways to quantify inconsistency.

Hunter [104] proposes a measure of conflict for propositional belief bases. Their

measure takes into account both the number of variables involved (contributing to) the inconsistency, and the distribution of the inconsistency over formulae in the knowledge base. A later paper by Grant and Hunter [105] also considers propositional belief bases. Here, the concept of dilation is the basis for several inconsistency measures. Dilating a formula means to look at the nearest neighbours of its models (for example, the formula $a \wedge b$ has a single model $\{11\}$, whose nearest neighbours – differing in no more than one truth assignment – are $\{11, 01, 10\}$, which is the model of $a \vee b$). The amount of dilation required to reach a consistent belief base then forms the basis of the inconsistency measures.

Besides these argumentation-specific measures of inconsistency (and argument strength) there is a large body of work on inconsistency measures in general, for example with probabilistic logics [106], fuzzy logic [107], propositional logic [108, 109]. In [110] the relationship of consistency gain and information loss is discussed, again based on propositional logic, and it is established that inconsistency resolution by deleting or changing formulae always leads to information loss. Our approach is based on inconsistency resolution by expanding the knowledge base, which does not lead to information loss (although the truth of this claim depends on the exact definition of "information loss").

The advantage of measuring inconsistency rather than argument strength is that inconsistency is by definition a function of the knowledge base, and does not need to be supplied as additional input to the system. Inconsistency measures therefore suit our requirement of needing as little input data as possible, in addition to the actual design justification.

Discussion While it seems natural to incorporate a notion of argument strength into argumentation frameworks, extensional definitions of strength are problematic for our use case, because they require additional information that often cannot be obtained straightforwardly. For example, to reason about preferences we must have a formalisation of the audience's preferences or values. Design documents do not contain such data, so it would have to be supplied by means of annotation or meta-data. The history of design rationale (see page 158) has shown that requiring the user of a system to manually formalise design documentation greatly inhibits the adoption of the system. For this reason we have focused on intensional definitions of argument strength.

When extracting arguments from text using natural language processing it is usually possible to quantify the certainty or plausibility that the formalisation accurately represents the text. This data could be used as the basis for a probabilistic or possibilistic argumentation framework. However, the certainty in this case would reflect the quality of the argument extraction process, not of the strength of the extracted arguments. Therefore this data is not a suitable basis for measuring the impact of change on a decision sequence.

4.6.1.3 Argumentation-Based Planning

Our model of design decisions made by different teams over long periods of time bears some resemblance to argumentation-based planning ([111, 112, 113], see [114] for a recent survey), a research area motivated by the question how multiple agents, each with its own knowledge and assumptions, can make a decision to achieve a common goal through the exchange of arguments following some protocol – distinct from other fields in agent-based argumentation [115] such as persuasion [116, 117], inquiry [118] and recommendation [119].

The differences between our work and argumentation-based planning are twofold. First, the collaboration between "agents", or teams of engineers, in our setting is not governed by a formal protocol (a set of rules specifying which moves are allowed at a specific point in the game), and our model only captures the outcome of this collaboration, in the form of design documents and justifications. Second, our main goal is to analyse decision processes that have already happened, for example by computing the quality of decision justifications, or the impact that changing a decision could have. Because of this, decisions in a valid decision sequence are not required to be optimal, and our model cannot be used to determine decide whether an optimal solution for the entire process exists (this is only possible for individual decisions, as covered in Chapter 3, page 43). Instead, we aim to be able to represent a large number of possible decision sequences and analyse them after the fact. In models of agent-based planning, the goal is to arrive at a good decision that realises the agents' goals.

4.6.1.4 Future Work

There are two main areas for future work. First, the theoretical underpinning of our method for impact analysis should be solidified further by studying additional measures

of impact. In this thesis we considered a strength-based measure and a measure of the information contained in a model of decision processes. It would be interesting to incorporate, for example, a notion of preferences or certainty of arguments, and to derive an impact measure from those values.

Both of the measures we studied are based on abstract argumentation. This makes our framework more general (because it is possible to use argumentation systems other than ASPIC+), but it also means that we do not consider any information outside of Dung's argument graphs, for example about sub-arguments and supporting arguments. We plan to investigate the use of non-classical logics, for example probabilistic or fuzzy logics, which would allow us to utilise different measures of inconsistency such as the ones proposed by Thimm [106] or Picado-Muiño [107].

The second area of future work is to turn this proposal into a practical application, as outlined in Section 4.5. This involves two separate tasks: First, the automated extraction of arguments from text. While some progress has been made in this area (e.g. Bex and Bench-Capon 2014 [120], Goudas *et al.* [121], Walton [122]), it is not possible yet to extract arguments with an accuracy sufficient to enable the kind of analysis proposed in this thesis. Second, we see a lot of potential in the visualisation of argumentative information, as explained in Section 4.5.3. The ability to produce visualisations such as Figure 4.4 on page 155 is a major advantage of modeling decision rationale in a formal language, a feature that is genuinely new and not possible with the software currently used in the industry. Research work in this area includes usability studies with focus groups, and an investigation of different layout/diagramming techniques.

4.6.1.5 Conclusion

Our argumentation-based model of decision processes consists of the two components decision outcomes and embeddings.

Decision outcomes (Definition 52) capture the minimal knowledge that is required to justify a decision, that is, to state all arguments in favour of the decision. A decision process, for example a engineering design process, can be seen as a sequence of decision outcomes. Each decision outcome on its own is conflict-free, but the union of several decision outcomes may be inconsistent.

The purpose of embeddings (Definition 54) is to draw out these inconsistencies.

An embedding is a mapping of decision outcomes, which are conflict-free, to the more general concept of ASPIC+ knowledge bases, which may or may not be conflict-free. Embeddings encode a preference for later decisions over earlier ones, ensuring that the latest decision outcome is always sceptically acceptable while retaining all arguments from earlier decisions. Therefore, they ensure that arguments which are not acceptable anymore (for example, because their assumptions no longer hold) are attacked by new, "deactivating" arguments. Embeddings only add knowledge but never delete it, thus making explicit any incompatibilities in the justifications of decisions. The relationship between sequences of decision outcomes and their embeddings is made clear by Figure 4.1 on page 127.

In Section 4.4 of this chapter, we explored the properties of decision sequences in two ways. First we developed a method for analysing the impact of changing an earlier decision by looking at the number of new arguments and conflicts introduced by the changed. In a similar way, we characterised the progress that has been made by transitioning from one decision to the next. Both methods are agnostic of the underlying domain (e.g. of engineering design) and only consider the justifications of decisions on the level of arguments and attacks. The methods thus illustrate the unique advantages that can be gained from a formalisation of decision outcomes in our logical framework.

The practical implications of our work were examined in Section 4.5. The two-tiered model of decision outcomes and decision sequences is consistent with our findings about real-world design documents (see Section 4.5.1). Each of those documents on its own consistently explains why a certain part of the design was chosen, but inconsistency arises when those individual decisions are viewed as parts of the overall design.

In the previous chapter we looked at argumentation-based models of design decisions, and in this chapter we looked at decisions as part of a design, embedded in decision sequences. The following chapter will be concerned with what actually makes an argument, and investigate arguments about arguments (rather than arguments about, say, the merits of aluminium over steel).

Chapter 5

Argument Schemes

5.1 Introduction

In the previous chapters we used the ASPIC+ argumentation system and a "logical language" $\mathscr{L}$. We did not specify an inference mechanism for $\mathscr{L}$, using only defeasible ASPIC+ rules for inference. We only specified the conflict function $\bar{\cdot}$, as required in the definition of ASPIC+ (see Section 2.3.2.2 on page 33). This resulted in a two-tiered system with deductive inferences in $\mathscr{L}$ in one tier, and defeasible rules in ASPIC+ in the other tier. However, an important aspect of argumentation within engineering design has been neglected so far: Decisions are made not only on the basis of logical inferences from some knowledge base, but also on the basis of informal arguments (including personal convictions, experience, and preferences) and experimental data. Any formal model of design debates must therefore be able to handle non-deductive arguments as well as deductive ones.

In this chapter we will study how non-deductive arguments are captured by our framework. In Section 5.2 we start by describing some common patterns of non-deductive reasoning in more detail, in order to get a clear understanding of the different types of arguments and inferences that our model should be able to express.

We then define an argumentation system that is specifically designed to handle non-deductive inferences in Section 5.3. In Section 5.3.2, we introduce the notion of meta-argumentation and show that it has an intuitive interpretation in our system. We conclude with a practical application of the theory, a case study about the interpretation of experimental data within our framework (Section 5.4). The case study is about comparing different drilling methods and involves a software prototype (Section 5.4.4)

as well as meta-argument schemes (Section 5.4.4.5). The chapter concludes with a review of the relevant literature (Section 5.5) and a discussion (Section 5.6).

5.2 Non-Deductive Arguments

Many argumentation systems in the literature have a base logic [123] (the underlying logic $\mathscr{L}$ in ASPIC+), for example propositional or first-order logic, from which arguments are built. The advantage of using a formal logic for arguments is that it provides the means for automatic determination of both the set of arguments (by using the inference relation of $\mathscr{L}$) and the set of attacks (by using $\mathscr{L}$'s notion of inconsistency). When applying the theory of argumentation to practical arguments, for example in our domain of engineering design, one is faced with a challenge: The large majority of real-life arguments can only be formalised in a logical language if either a lot of implicit background knowledge is taken into account, or the result is rather general and the benefits of using a logical language are lost.

With the overall goal of our project in mind (to build an argumentative model on top of what will ideally be automatically processed documents), we will take a different avenue altogether: In this chapter we are completely going to drop the assumption that arguments are built with a base logic. Instead, we will only assume a set of arguments that have no inference relation between them. The formalities of this approach will be discussed in Section 5.3, but first we need to answer the question "What is an argument?"

If we do use a base logic, then the answer is easy: In this case, "an argument is a pair $\langle \varphi, \alpha \rangle$ where φ is a minimal subset of the knowledge base such that φ is consistent and φ entails the claim α." [123]. This definition of an argument relies on the base logic for all three conditions (minimality, consistency and entailment). In order to verify the three conditions and thus to confirm that $a = \langle \varphi, \alpha \rangle$ is an argument, one only has to know the definition of inference in the base logic. Crucially, whether a is an argument does not depend on any other arguments (it is only the acceptability of a that can be influenced by other arguments).

The case for inductive arguments (arguments without a base logic) is not as straightforward. An important difference between deductive and inductive arguments is that the latter can only be validated in a context – additional knowledge about the ar-

guments. To check that a deductive argument is valid, we only need a proof (in the base logic) that the claim can be inferred from the premises. Such a proof only depends on information contained within the argument itself. The validity of inductive arguments depends on additional information external to the argument. For example, natural language arguments often take the form of enthymemes. An enthymeme is an argument that does not contain all of its premises – in other words, it is context-dependent (cf. recent work on natural language arguments [124, 125]). To verify that an enthymeme is an argument (even before one can determine the acceptability of that argument) one has to take into account the context given by implicit assumptions.

Since inductive arguments are context-dependent, there is nothing that can be said about an individual inductive argument on its own (for deductive arguments we can at least decide whether the three conditions are met). We can only make meaningful judgments about an inductive argument if we know its context, that is, if we know its relationship with other inductive arguments. This idea is reminiscent of abstract argumentation in the sense of Dung [42], and indeed we will see in Section 5.3.1 that inductive arguments are built on top of an abstract argument graph. Inductive arguments can be in two possible interactions with other arguments (conflict, and support).[1]

Let us further motivate our theory of inductive arguments by looking at two specific examples, argument schemes (Section 5.2.1) and the interpretation of experimental data (Section 5.4.3).

5.2.1 Argument Schemes

Most arguments produced by humans are not deductive, but they are not completely arbitrary, either. They follow well-known patterns, so-called argument schemes. Argument schemes have been studied extensively by Walton [126] and we will give a brief review of some of the most common schemes listed in his survey.

According to [126], argument schemes are informal patterns of human reasoning. One of the most common argument schemes is *appeal to expert opinion*, in which a claim is supported by evidence of an expert (a person different from the proponent of the argument).

The following list of argument schemes is not exhaustive for two reasons. Because

[1] The relationship of our framework to bipolar argumentation will be discussed in Section 5.5.2.

argument schemes are not formally specified, there is no consensus on how many different schemes there are, and it is impossible to prove that a list of argument schemes is complete, so any attempt at compiling such a list would be vulnerable to claims of its partiality. But our list below only includes a subset of the argument schemes in Walton's book [126]. It has been purposefully selected to draw out the possible implications that argument schemes may have on arguments they are used in, in order to provide a good motivation for the theoretical model developed in Section 5.3. In addition, we placed special emphasis on argument schemes that are commonly used in engineering debates.

Notation Throughout the examples in this chapter we will use free variables $X, Y, \ldots$. These variables are only used as patterns that serve to build an intuition for argument schemes. Whenever we talk about concrete argumentation systems, we assume that all free variables have been eliminated by replacing them with appropriate values. This practice is not only in line with ASPIC+ conventions but it also greatly simplifies the formal notation.

Since a formal definition for meta-ASPIC will only be given in the following section (page 178), we are going to assume the same syntax ASPIC+ for arguments as in other places in this thesis. To see how argument schemes are represented in meta-ASPIC, the reader is referred to Section 5.3.1.1 on page 182.

Appeal to Expert Opinion Recall the following argument from Chapter 4 (page 120). It claims that a structure is non-corrosive because it is made of aluminium:

$$[[\texttt{al}];\texttt{al} \Rightarrow \texttt{non\_corrosive};\texttt{non\_corrosive}] \qquad (A_1)$$

This argument does not provide any evidence in support of the claim, except the defeasible rule $\texttt{al} \Rightarrow \texttt{non\_corrosive}$. But what is our justification for applying this rule? We can imagine an expert E (who specialises in the science of corrosion) claiming that components made from aluminium are not prone to corrosion. The expert E would be able to produce an argument for this rule, that is, an argument whose claim is

$\texttt{al} \Rightarrow \texttt{non\_corrosive}$:

$$[[\texttt{al} \Rightarrow \texttt{Al\_oxide\_coating}]; \texttt{Al\_oxide\_coating} \Rightarrow \texttt{non\_corrosive}; \texttt{non\_corrosive}] \qquad (A_2)$$

Argument A_2 says that aluminium does not corrode because it is coated in a layer of aluminium oxide (this layer is actually caused by corrosion of pure aluminium). If prompted, E would be able to produce additional arguments in support of A_2, for example scientific studies or a more in-depth explanation of the phenomenon involving the reaction of aluminium oxide with water at a molecular level.

But in order to make argument A_1, it is enough to refer to E's expert opinion - we do not actually have to produce argument A_2 as well. This particular argument scheme is therefore a way of pruning the tree of supporting arguments which would otherwise be very difficult to keep track of during a design debate. On the other hand, if we rely on argument A_1, then we open ourselves to attacks of the form "E is not an expert" and so on – regardless of whether E's statement is true or not.

The appeal to expert opinion in schematic terms is shown below, based on the original definition in [127]. The variables X, D and C stand for expert, domain and claim.

$$[\mathsf{Expert}(X,D), \mathsf{Domain}(C,D), \mathsf{Claims}(X,C) \Rightarrow C] \qquad (Exp)$$

To apply the scheme Exp to argument A_1, we can instantiate it as follows:

$$[\mathsf{Expert}(\mathsf{E}, \mathsf{metals}), \mathsf{Domain}(A_1, \mathsf{metals}), \mathsf{Claims}(\mathsf{E}, A_1) \Rightarrow A_1] \qquad (Exp(A_1))$$

Argument $Exp(A_1)$ shows that the claim we made in A_1 is backed up by E, who is an expert in "metals".

Argument from Alternative The argument scheme "Argument from Alternative" consists of stating that there are two options, a and b, which are mutually exclusive and collectively exhaustive. One then has to show that a is not viable, which leaves only b as a logical conclusion. For example:

$$[[\neg\texttt{composites}]; \neg\texttt{composites} \Rightarrow \texttt{al}; \texttt{al}] \qquad (A_3)$$

Argument A_3 says that composites are ruled out as a material, and therefore aluminium has to be chosen. However, the only reason why the defeasible rule $\neg\texttt{composites} \Rightarrow \texttt{al}$ holds is that `composites` and `al` are mutually exclusive, and one of the *has* to be chosen. We can therefore give the following example in support of our claim:

$$[\texttt{alternative}(\texttt{composites},\texttt{al});\texttt{alternative}(X,Y) \Rightarrow (\neg X \Rightarrow Y);$$
$$\neg\texttt{composites} \Rightarrow \texttt{al}] \qquad (A_4)$$

With argument A_4, we can give a reason for inferring `al` from $\neg$`composites` in argument A_3. However, we need to establish first that the two options are mutually exclusive and collectively exhaustive (or at least be ready to defend our argument against attacks on those premises). Again, this argument schemes saves us from having to elaborate on the benefits of `al`, instead relying on it being the only viable option.

$$[\neg\texttt{alternative}(\texttt{composites},\texttt{al})] \qquad (C_1)$$

A simple counter-argument to A_4 is given by C_1. By denying the assumption $\texttt{alternative}(\texttt{composites},\texttt{alu})$, we can derive an attack on A_4. This counter-argument does not draw into question the scheme (argument from alternative), it only attacks the application of the scheme to this particular instance.

Argument from Positive Consequences The next argument scheme, *Argument from Positive Consequences*, is perhaps the most fundamental one as far as decision making is concerned: We should choose an option a because a will bring about positive consequences. For example:

$$[[\texttt{result}(\texttt{al},\texttt{low\_cost})];\texttt{result}(\texttt{al},\texttt{low\_cost}) \Rightarrow \texttt{al};\texttt{al}] \qquad (A_5)$$

In argument A_5, we argue that aluminum is the right choice because it is a low-cost material. We can support this argument by making the underlying scheme explicit:

$$[\texttt{result}(\texttt{al},\texttt{low\_cost}),\texttt{desirable}(\texttt{low\_cost});$$
$$\texttt{result}(X,Y),\texttt{desirable}(Y) \Rightarrow Y;\texttt{result}(\texttt{al},\texttt{low\_cost}) \Rightarrow \texttt{al}] \qquad (A_6)$$

When using the argument scheme from positive consequences, we have to establish that the consequences are indeed good, and that choosing that particular option will bring about the consequences.

Personal (*ad hominem*) Arguments Another common argument scheme is that the appeal at the person (*ad hominem*). This scheme is an argument about the proponent of another argument, not against that other argument. There are many types of the *ad hominem* argument – for example, one can claim that the proponent of an argument is not qualified to make the argument, or that they have been known to make false claims in the past, etc.

$$[\texttt{proponent}(\texttt{al},\texttt{james});\texttt{proponent}(\texttt{al},\texttt{james}) \Rightarrow \neg\texttt{al};\neg\texttt{al}] \qquad (A_7)$$

Argument A_7 is a counter-argument against choosing aluminium, and it is supported by the fact that the argument for aluminium has been produced by James. The inference can be justified by providing the following argument:

$$[\texttt{proponent}(\texttt{al},\texttt{james}),\texttt{unqualified}(\texttt{james});\texttt{proponent}(X,Y),$$
$$\texttt{unqualified}(Y) \Rightarrow (\texttt{proponent}(X,Y) \Rightarrow \neg X);$$
$$\texttt{proponent}(\texttt{al},\texttt{james}) \Rightarrow \neg\texttt{al}] \qquad (A_8)$$

Note that argument A_8 only need one premise related to the claim X, namely that it was put forward by Y – no additional information about X is needed to apply the scheme to obtain $\neg X$. *Ad hominem* arguments can therefore be constructed with very little evidence.

The *ad hominem* example differs from the other schemes in an important way: It can be used to produce arguments *against* an option – in other words, we can use

it to produce counterarguments. The other argument schemes presented here all build up support for an argument, but A_7 introduces an attack. This is important because if we had no attacks, then there would be no reason for using Dung's framework. *Ad hominem* is not the only scheme that can produce attacks. The scheme "Argument from Negative Consequences" for example is very similar to "from Positive Consequences" (5.2.1), except that it has a negative conclusion. A_7 is different from the previous counter-argument C_1, because A_7 is relevant in any scenario where an argument for `al` is made, whereas C_1 does not argue for or against `al` per se, but rather attacks any arguments such as A_4 which rely on the assumption that aluminium and composites are the only two options.

We can contest the use of *ad hominem* by producing an argument against the defeasible rule $ad\_hom = \texttt{unqualified}(Y) \Rightarrow (\texttt{proponent}(X,Y) \Rightarrow \neg X)$:

$$[\neg\texttt{personal\_attacks} \Rightarrow \neg\langle ad\_hom\rangle] \qquad (C_2)$$

C_2 says that because personal attacks are disallowed (they are not valid arguments), the rule *ad_hom* should not be used. C_2 attacks any argument that does *ad_hom*, for example A_8.

Appeal to Authority The last scheme in our survey is the *appeal to authority*. This scheme subsumes *appeal to expert opinion* (Section 5.2.1), since experts are figures of authority, but it includes other types of authority. For example, groups often follow a "seniority principle" in which the opinion of the senior and more experienced members is afforded a higher weight. We can encode the seniority principle as follows:

$$\begin{aligned}[\texttt{proponent}(\texttt{al},\texttt{james}),\texttt{proponent}(\texttt{composites},\texttt{jane});\\ \texttt{proponent}(\texttt{al},\texttt{james}),\texttt{proponent}(\texttt{composites},\texttt{jane}) \Rightarrow \texttt{composites};\\ \texttt{composites}] \qquad (A_9)\end{aligned}$$

Argument A_9 says that we should choose composites over aluminium because the argument for composites was put forward by Jane, whereas the argument for aluminium was produced by James. The reasons for this preference can, for example, be explained

by

$$\begin{aligned}[\texttt{proponent}(\texttt{al},\texttt{james}),\texttt{proponent}(\texttt{composites},\texttt{jane}),\\ \texttt{senior}(\texttt{james},\texttt{jane}),\texttt{senior}(Y,X)\Rightarrow(\texttt{proponent}(X,X'),\\ \texttt{proponent}(Y,Y')\Rightarrow Y');\texttt{proponent}(\texttt{al},\texttt{james}),\\ \texttt{proponent}(\texttt{composites},\texttt{jane})\Rightarrow\texttt{composites}]\end{aligned}\qquad(A_{10})$$

It is clear that the line between appeal to authority and *ad hominem* arguments is blurred - the argument for aluminium is rejected because it was brought forward by James, and not because of the claim `al` (aluminium) as such.

5.2.2 Interpreting Experimental Data

It is possible to treat the interpretation of experimental data as an instance of one of Walton's argument schemes, for example "Argument from Experience". However, the status of arguments from experimental data (and to some extent from other empirical data, e.g. from past obervations) is differrent from those based on other argument schemes. First, such arguments are more difficult to defeat because empirical observations ("numbers") are considered to be more reliable than, say, personal feelings or intuitions. Second, when arguments from empirical data are disputed, they are rarely attacked on the underlying data, and more commonly on the conclusions that can be drawn from this data. In contrast, argument schemes as a whole are more commonly attacked on their assumptions. For example, the scheme "Argument from Expert Opinion" can be attacked by arguing that X is not an expert in domain Y, or that the problem at hand is not part of Y. Experimental results can only be attacked if mistakes were made when conducting the experiment. Otherwise they can almost be considered to have the same status as axioms in logic, because of their reproducibility.

A typical argument from experimental data draws an analogy between the experimental setting and the actual problem at hand. For example, one could make the following argument based on the case study about conventional versus orbital drilling

(see Section 5.4 for more details):

$$[\texttt{goodResults}(\texttt{diameter}\ 5/16, \texttt{conventional}), \\ \texttt{hole\_diameter}(5/16) \Rightarrow \texttt{use\_conventional}] \quad (A_{11})$$

saying that because the experiment using conventional drilling techniques with a $5/16$ inch diameter gave good results, and our current design involves the same diameter, we should use conventional drilling.

The pattern of drawing analogies between the experiment and the problem at hand can be formulated as an argument scheme, too: If the experimental outcome for technique Y and setting X was good, and our decision involves a setting similar to X, then we can expect a similarly good outcome from technique Y.

$$[\texttt{goodResults}(X,Y), \texttt{similar}(X,X') \\ \texttt{currentProblem}(X') \Rightarrow \texttt{choose}(Y)] \quad (A_{12})$$

Of course, the quality of the argument is determined by the meaning of "similar" in this scheme. This is drawn out in the next two arguments.

$$[\texttt{diameter}(\texttt{dec1}, 3/8), \texttt{diameter}(\texttt{exp1}, 5/16) \\ \Rightarrow \texttt{similar}(\texttt{exp1}, \texttt{dec1})] \quad (A'_{12})$$

A'_{12} argues that the difference between the whole diameter in the experiment and the one in our current problem is rather small, so the two settings can be considered similar enough for the results to be transferrable. In A'_{12}, we set $X = \texttt{exp1}$ and $X' = \texttt{dec1}$ to highlight that we are applying the scheme to a concrete setting (current problem `dec1` and experiment `exp1`).

$$[\texttt{environment}(\texttt{dec1}, \texttt{dry}), \texttt{environment}(\texttt{exp1}, \texttt{wet}) \\ \Rightarrow \neg\texttt{similar}(\texttt{exp1}, \texttt{dec1})] \quad (C_3)$$

C_3 rebuts A'_{12}, and it undercuts any instantiation of A_{12} for the two choices of X and X'. The two arguments C_3 and A'_{12} are an example of debates over the meaning of experi-

mental results. The core question in this case is which of the parameters (environment, hole diameter) is more likely to have influenced the outcome of the experiment and thus to have an impact on hole quality. C_3 is an example of a *critical question*, an important concept in argument schemes which we will discuss in the next section.

5.2.3 Critical Questions

In the previous two sections, 5.2.1 and 5.2.2, we discussed arguments and counter-arguments (for example, C_1, C_2 and A_7) together. Prior work on argument schemes (primarily Walton [128, 126] but also formalisations in theory of argumentation such as [127] (expert opinion), [129, 130, 131] (law) and [132] (trust)), makes a clear distinction between critical questions (C_1, C_2, C_3) and regular counter-arguments (A_7), which we will draw out in this section.

In Walton's theory, there exists a list of critical questions for every argument scheme. Each critical question identifies possible attacks on arguments from the scheme it is associated with, by pointing out either a condition that must hold for an argument scheme to be applied, or an exception that renders an argument scheme invalid for a specific instance. The first kind of critical question is called *condition*, and the second kind is called *exception*.

Some critical questions, such as the one expressed in C_1, target the conditions of an argument scheme. Another type of critical questions is aimed at exceptions to the applicability of a scheme, for example C_2. Both kinds have in common that they do not attack the conclusion drawn from the argument scheme directly and do therefore not act as counter-arguments against the original proposition. Instead, they only prevent the application of a particular argument scheme, by claiming either that its conditions are not met, or that it is not applicable at all, regardless of the specific premises.

A good starting point for a formal characterisation of critical questions is thus to look at the attackers of an argument from argument schemes. If the conclusion of an attacking argument a negates one of the attacked argument's premises, then a is an attack on a condition of the scheme. If it is the negation of a rule, then a is an exception to the scheme. This definition also clearly distinguishes critical questions from counter-arguments. The latter is always a rebuttal, the former can be an undercut or an undermining atttack. We will make this definition more precise in Section 5.3.1.1

on page 182.

5.2.4 Summary

After a brief discourse on the nature of non-deductive arguments, we examined a number of argument schemes commonly used in engineering debates. We studied five concrete argument schemes and proposed tentative translations into the ASPIC+ terminology. We also discussed arguments from experimental data and their special status among argument schemes in the engineering domain. What all argument schemes we considered have in common is that they are context-dependent: Each of the five arguments (A_1, A_3, A_5, A_7, A_9) is accompanied by a supporting argument to establish that the premises of the scheme are met, and to turn the scheme into an ASPIC+ rule.

The assumptions underlying argument schemes are arguments on their own, and can therefore be supported, defeated etc. In the following we will formalise the idea of representing argument schemes as defeasible rules produced by other arguments.

5.3 Meta-ASPIC

In this section we will define an argumentation system called "meta-ASPIC" that is designed specifically to reason about argument schemes. Its syntax is similar to the ASPIC+ syntax we have been using throughout this thesis, and its semantics (meaning) are given by a translation to regular argument graphs.

The discussion in Section 5.2 made it clear that inferences are of great importance in non-deductive arguments, because debates in engineering revolve around the question whether a set of premises warrants a particular conclusion. The question whether premises hold or not is of comparatively lesser importance. For our meta-ASPIC system, we consequently take the set of facts F as given and instead focus on the formation of arguments. First, we assume that any fact $a \in F$ gives rise to an argument $[a]$. The reason for this requirement is that it will lead to a simple syntax for arguments, because it will allow us to treat all arguments (atomic and compound) uniformly, without having to distinguish the case that an argument is "only" a fact. Second, we treat inference rules and arguments as one and the same. As a result, there is no set of rules (unlike in ASPIC+) from which arguments are constructed. Instead, it is possible and legal to form an argument from any combination of other arguments as support and with any

other argument as conclusion. The only restriction we do place on arguments is that they are non-cyclic, that is, an argument cannot be used as its own support. This reflects our inuitive understanding that cyclic arguments are rarely used in real-life debates, and leads to some interesting properties in relation to meta-argumentation, which we will discuss in Section 5.3.2. We use the symbol $\rightsquigarrow$ to denote argument formation.

The second component of our system, besides arguments, is contradiction. In meta-ASPIC, the set of facts F may be contradictory. This constraint, or rather lack of constraint, allows us model situations where arguments are built from facts based on faulty observations. Note that any set F endowed with a binary "conflict" relationship is by definition an abstract argument graph. We will therefore speak of the underlying argument graph of a meta-ASPIC system, as opposed to the underlying logic of an ASPIC+ system.

The examples in Section 5.2.1 indicated how argument schemes can be represented in ASPIC+: As arguments whose conclusions are defeasible rules (the rules that allow the argument scheme to be applied). However, in order to get the most general definition of meta-arguments, we cannot assume anything about defeasible rules that is not already part of the syntax. In particular, we cannot assume that a defeasible rule is in any way schematic, that is, contains free variables that may be bound to different values. The reason is that defeasible rules represent inductive arguments and as such do not necessarily stand for an abstract inference rule that applies to anything but the specific case for which they are stated. As a result, the only constraint on what constitutes a meta-argument is given in the syntax – it must have a set of supporting arguments and a conclusion – and there is no requirement for the support to be in any kind of deductive relationship with the conclusion.

This section is organised as follows. We start by defining the syntax (Definition 68 on page 178) and semantics (Definition 72 on page 182) of meta-ASPIC. We then consider to separate issues in meta-ASPIC: The argument schemes that were introduced in the previous section are translated to meta-ASPIC+ on page 182, and expressions of the truth paradox in meta-ASPIC is discussed on page 187. In Section 5.3.2 (page 188) we apply the concept of meta-argumentation to meta-ASPIC.

5.3.1 Definition of Meta-ASPIC

We will define meta-ASPIC as a formal foundation for meta-arguments in two steps. In Definition 68 below we set up the syntax of a language for describing meta-arguments. The subsequent definitions up to Definition 72 on page 182 will establish the meaning of that language, by providing a translation from it to Dung's argument graphs.

Definition 68 (Meta-Argument)**.** *Let $G = (A, Att)$ be an argument graph. A* meta-argument *over G is*

1. *If $a \in A$ then $[a]$ is a meta-argument*
2. *If $\varphi_1, \ldots, \varphi_n, \varphi$ are meta-arguments (with $n \geq 0$) then $[\varphi_1, \ldots, \varphi_n \rightsquigarrow \varphi]$ is a meta-argument*
3. *If $\varphi_1, \ldots, \varphi_n, \varphi$ and are meta-arguments (with $n \geq 0$) then $[\varphi_1, \ldots, \varphi_n \rightsquigarrow \neg\varphi]$ is a meta-argument*

If a meta-argument has an empty set of supporting arguments then we do not write the arrow symbol (for example, instead of $[\rightsquigarrow \neg a]$ we write $[\neg a]$). We use the notation M_A for a set of meta-arguments over an argument graph (A, Att) if there is no risk of ambiguity.

Example 49. *Let $G = (A, Att)$ be an argument graph with $A = \{(a_1, a_2, a_3, a_4)\}$ and $Att = \{(a_1, a_3), (a_2, a_4), (a_4, a_2)\}$. We can define the set $M_A = \{m_1, \ldots, m_6\}$ of meta-arguments over G with*

$$m_1, \ldots, m_4 = [a_1], \ldots, [a_4]$$
$$m_5 = [m_3 \rightsquigarrow m_2]$$
$$m_6 = [m_1 \rightsquigarrow \neg m_4]$$

The first four meta-arguments m_1 to m_4 are simply the meta-level equivalents of a_1 to a_4 on the object level. m_5 and m_6 exist only on the meta-level and represent additional information that is not present in G: First, m_5 states that m_3 supports m_2. Second, m_6 states that m_1 is a reason against (counterargument to) m_4. The attacks-relation Att of the object-level graph is only relevant in the evaluation of sets of meta-arguments. It does not affect which meta-arguments can be formed.

The function $\mathsf{conc}(a) : M_A \to M_A$ returns the conclusion of a meta-argument a and is defined as

$$\mathsf{conc}(a) = \begin{cases} [a'] & \text{if } a' \in A \text{ and } a = [a'] \\ [\varphi] & \text{if } a = [\varphi_1, \ldots, \varphi_n \rightsquigarrow \varphi] \\ [\neg\varphi] & \text{if } a = [\varphi_1, \ldots, \varphi_n \rightsquigarrow \neg\varphi] \end{cases}$$

Likewise, $\mathsf{sup}(a) : M_A \to \mathscr{P}(M_A)$ returns the support of a meta-argument a and is defined as

$$\mathsf{sup}(a) = \begin{cases} \{\varphi_1, \ldots, \varphi_n\} & \text{if } a = [\varphi_1, \ldots, \varphi_n \rightsquigarrow \varphi] \\ \emptyset & \text{otherwise} \end{cases}$$

The final function we need for talking about meta-arguments is obj. If a meta-argument is of the form $\varphi_1, \ldots, \varphi_n \rightsquigarrow \varphi$, then intuitively it is an argument about the arguments φ_1 to φ_n and φ. The function obj captures the intuition that arguments "about" other arguments or about domain-level facts. In the former case, $\mathsf{obj}(a)$ returns the arguments that are referred to by an argument a, and in the latter case $\mathsf{obj}(a)$ returns the empty set.

Definition 69 (Object Function)**.** *Let $\Omega = (G, M_A)$ be a meta-*ASPIC *system with $G = (A, Att)$ The* object function *of Ω,* $\mathsf{obj} : M_A \to 2^{M_A}$*, is defined as*

$$\mathsf{obj}(a) = \begin{cases} \{\varphi_1, \ldots, \varphi_n, \varphi\} & \textit{if } a = \varphi_1, \ldots, \varphi_n \rightsquigarrow \varphi \\ \{\varphi_1, \ldots, \varphi_n, \neg\varphi\} & \textit{if } a = \varphi_1, \ldots, \varphi_n \rightsquigarrow \neg\varphi \\ \emptyset & \textit{if } a = [a'] \textit{ for } a' \in A \end{cases}$$

Example 50. *For arguments m_5 and m_6 from Example 49,* obj *returns the following values:*

$$\mathsf{obj}(m_5) = \{m_3, m_2\} \qquad \mathsf{obj}(m_6) = \{m_1, \neg m_4\}$$

and for all other arguments it returns the empty set.

We are now ready to give a definition of our meta-ASPIC argumentation system. It is simply a set of meta-arguments over an argument graph $G = (A, Att)$, with two conditions. First, all arguments $a \in A$ must have corresponding meta-arguments $[a] \in$

M_{Att}, and second, M_{Att} must be closed under obj - that is, it must contain the supports of all its arguments.

Definition 70. *A* meta-ASPIC *system is a tuple* (G, M_A) *where* $G = (A, Att)$ *is an argument graph and* M_A *is a set of meta-arguments over* G *such that*

1. $M_A \supseteq \bigcup_{a \in A} [a]$
2. *For all* $a \in M_A$, $\mathsf{obj}(a) \subseteq M_A$

Example 51. (G, M_A) *from the previous example (Example 49) is a meta-*ASPIC *system, because every object-level argument (*a_1 *to* a_4*) is represented by a meta-level argument (*m_1 *to* m_4*), and the component arguments of* m_5 *and* m_6 *are also part of* M_A*.*

There are several important differences between standard ASPIC+ systems and meta-ASPIC systems as defined in Definition 70.

First, the logical language $\mathscr{L}$ has been replaced by an argument graph G. This may seem like a radical change, but in reality it is only syntactical. In the original definition of ASPIC+, the logical language $\mathscr{L}$ is only characterised by the existence of a "conflict" function $\overline{\circ} : \mathscr{L} \to 2^{\mathscr{L}}$. It is not required to have an inference relation or any operators. An argument graph (A, Att) on the other hand consists of a set A and an "attacks" relation $Att \subseteq A \times A$. Given that every binary relation of two sets $Rel \subseteq A \times B$ specifies a function $f_{Rel} : A \to 2^B$, we could easily convert Att to a function $f_{Att} : A \to 2^A$, so G would meet the requirements for a logical language $\mathscr{L}$. Conversely, we can create an argument graph $(\mathscr{L}, Rel_{\overline{\circ}})$ for any logical language with "conflict" function. Our Definition 70 therefore does not augment or diminish the expressive power of the underlying language $\mathscr{L}$.

Finally, the negation symbol $\neg$ in Definition 68 does not have the same meaning as it would for example in classical logic. The effect of writing $\neg a$ in the conclusion of a rule is to introduce an attack on the argument a, and not to conclude the opposite of a's conclusion. This means that $\neg\neg a$ is not the same as a. Double negation cannot be expressed directly. Instead one can introduce a third argument which attacks $\neg a$. $\neg$ therefore modifies the relationship between arguments, not between their conclusions.

The advantage of our Definition 70 is that we do not have to translate between "conflicts" in our arguments and "attacks" in the argument graph produced by the sys-

tem. Instead, we can simply use the attacks relation Att to determine whether two arguments are incompatible.

What does meta-ASPIC give us that regular argument graphs do not? To answer this question we need to look at the second component of argument graphs: Attacks. There are two sources of attacks in meta-ASPIC – the attacks relation Att of the underlying graph G, and the negation operator $\neg$ that may be used in conclusions of arguments in M_A.

Definition 71 (Attacks in meta-ASPIC)**.** *Let (G, M_A) be a meta-ASPIC system with $G = (A, Att)$ and let $a, b \in M_A$. a* m-attacks *b if and only if*

1. $\mathsf{conc}(a) = [a'], \mathsf{conc}(b) = [b']$ *and* $(a', b') \in Att$ *or*
2. $\mathsf{conc}(a) = [\neg b'], \mathsf{conc}(b) = [b']$ *or*
3. $b = [\varphi_1, \ldots, \varphi_k \rightsquigarrow \psi]$ *such that there is an* $i \in \mathbb{N}$ *with* $1 \leq i \leq k$ *such that a m-attacks φ_i.*

Example 52. *The meta-*ASPIC *system given by the argument graph $G = (A, Att)$ with $A = \{(a_1, a_2, a_3, a_4)\}$ and $Att = \{(a_1, a_3), (a_2, a_4), (a_4, a_2)\}$ and meta-arguments $M_A = \{m_1, \ldots, m_6\}$ with*

$$m_1, \ldots, m_4 = [a_1], \ldots, [a_4]$$
$$m_5 = [m_3 \rightsquigarrow m_2]$$
$$m_6 = [m_1 \rightsquigarrow \neg m_4]$$

from Example 51 has the following attacks: $\{(m_1, m_3), (m_2, m_4), (m_4, m_2), (m_6, m_4)\}$, as shown in Figure 5.1.

Arguments a and b in meta-ASPIC can attack one another in three ways, as defined in Definition 71. Either there is already a conflict between a and b in the underlying argument graph, or the conclusion of a is $\neg b$, or a meta-attacks a supporting argument of b.

In a meta-ASPIC system (G, M_A), no restrictions are placed on the argument graph G. In particular, there is no requirement for the attacks relation Att to contain any

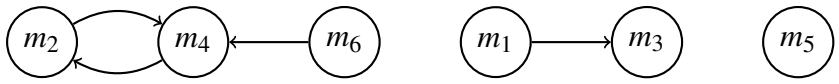


Figure 5.1: Reified graph of the meta-ASPIC system from Figure 5.2 on page 185

elements. If we start with an attacks-free argument graph $G = (A, \emptyset)$, we can can use meta-arguments to introduce conflict in a set that is otherwise conflict-free.

With the previous two definitions in place it is now straightforward to define the reified graph of (the abstract argument graph acting as the interpretation of) a meta-ASPIC system.

Definition 72 (Reified graph of meta-ASPIC)**.** *Let $\Omega = (G, M_A)$ be a meta-ASPIC system. The function* $\mathsf{reify}(M)$ *returns the* reified graph *of M and is defined as*

$$\mathsf{reify}(\Omega) = (M_A, Att^*)$$

where $Att^* = \{(a,b) \in M_A \mid a$ m-attacks *$b\}$.*

Note that $\mathsf{reify}(\Omega)$ returns a Dung-style argument graph, not a meta-ASPIC system.

Example 53. *The reified graph of the system in Example 51 is given by (A, Att^*) with $Att^* = \{(m_1, m_3), (m_2, m_4), (m_4, m_2), (m_6, m_4)\}$. It is visualised in Figure 5.1.*

To summarise, meta-arguments have the standard structure of supports and conclusions, but they differ from standard ASPIC+ arguments in the "type" of their conclusion: In our system, the conclusion of an argument is also an argument, but in classic ASPIC, the conclusion is a sentence of the underlying language $\mathscr{L}$.

5.3.1.1 Argument Schemes

To illustrate how argument schemes are handled by meta-ASPIC we will now describe a discussion between aerospace engineers in the framework, using some of the argument schemes introduced in Section 5.2.1. The discussion is fictional, but it is based on a use case from our industry partner.

Several engineers are designing a rib that is part of a wing. They are currently trying to decide on a material to be used for the rib. While in reality there is a choice of a large number of alloys and composites, we assume here that the principal decision is only that of aluminium or composite materials. The choice will be represented by two

arguments Comp and Al. In this example we assume that the object-level graph consists only of those two arguments and is conflict-free: $G = (\{\texttt{Comp}, \texttt{Al}\}, \emptyset)$. Throughout this example we will represent object-level arguments by their conclusions (as we are not interested in their internal structure, or how that conclusion was derived). We are going describe a set of meta-arguments and give the final result in the form of a meta-ASPIC system at the end of this example.

The two options are mutually exclusive, so the scheme "argument from alternative" (page 170) can be applied. Since aluminium and composites cannot both be chosen at the same time, choosing one means excluding the other. The argument scheme is represented by three meta-level arguments: One to establish that aluminium and composites are mutually exclusive, and then one for each of the two attacks that result from this argument scheme.

$$[\texttt{Alternative}(\texttt{Comp}, \texttt{Al})] \tag{M_1}$$

$$[[\texttt{Alternative}(\texttt{Comp}, \texttt{Al})] \rightsquigarrow (\texttt{Comp} \rightsquigarrow \neg\texttt{Al})] \tag{M_2}$$

$$[[\texttt{Alternative}(\texttt{Comp}, \texttt{Al})] \rightsquigarrow (\texttt{Al} \rightsquigarrow \neg\texttt{Comp})] \tag{M_3}$$

The reader may have noticed that the first meta-argument M_1 makes use of an object-level argument $\texttt{Alternative}(\texttt{Comp}, \texttt{Al})$ which is not part of the graph G. As we build up this discussion we will introduce several new object-level arguments, which stand for additional domain-level knowledge necessary to support the argument schemes we use. The resulting meta-ASPIC graph will therefore be based on an extended object-level graph G', with $G \sqsubseteq G'$. It is important to note that, while M_2 and M_3 resemble the logical axiom *tertium non datur*, the assumption that a third option does not exist can be attacked (and indeed there are more than two possible materials for the component).

Having established the external constraints of the solution, we now turn to the actual debate about the materials. Engineer E is recognised by her peers as an expert

on metallurgy (abbreviated Mly) and suggests to use aluminium.

$$[[\texttt{Expert}(\texttt{E},\texttt{Mly})],[\texttt{Domain}(\texttt{Al},\texttt{Mly})],[\texttt{Claims}(\texttt{E},\texttt{Al})] \rightsquigarrow [\texttt{Al}]] \qquad (M_4)$$

In reality, arguments from expert opinion usually do not just state a conclusion without backing it up with further evidence. Instead, expert arguments summarise the expert's reasoning as well as the conclusion [133]. For example, E might recommend aluminium based on her experience with similar designs. We omit the details in this example because we want to demonstrate additional argument schemes.

The knowledge of E, the expert, may be outdated because E has not published any work on metallurgy recently – $\texttt{NoPub}(\texttt{E},\texttt{Mly})$. This opens the argument from expert opinion to an attack on one of its supports:

$$[[\texttt{NoPub}(\texttt{E},\texttt{Mly})] \rightsquigarrow \neg[\texttt{Expert}(\texttt{E},\texttt{Mly})]] \qquad (M_5)$$

This attack is an example of a common pattern. Every argument scheme is associated with a list of "Critical Questions", questions which point to potential weaknesses of the argument. Some critical questions, such as the one expressed in M_5, target the conditions of an argument scheme. Another type of critical questions is aimed at exceptions to the applicability of a scheme.

Another common pattern of argumentation is to argue from (positive or negative) consequences. In our example, heavy components increase the fuel consumption of airplanes. Minimising weight is therefore very important in aerospace design. The relatively high weight of aluminium is a reason to avoid it. This argument scheme is known as *argument from negative consequences* (bringing about A will result in C, C is negative, therefore A should not be brought about). Conversely, using composites will have positive consequences, since it is lighter. The following two meta-arguments illustrate reasoning about consequences.

$$[[\texttt{BadCons}(\texttt{Al})] \rightsquigarrow \neg\texttt{Al}] \qquad (M_6)$$

$$[[\texttt{GoodCons}(\texttt{Comp})] \rightsquigarrow \texttt{Comp}] \qquad (M_7)$$

M_6 attacks the argument for aluminium Al. M_7 does not attack any arguments in this

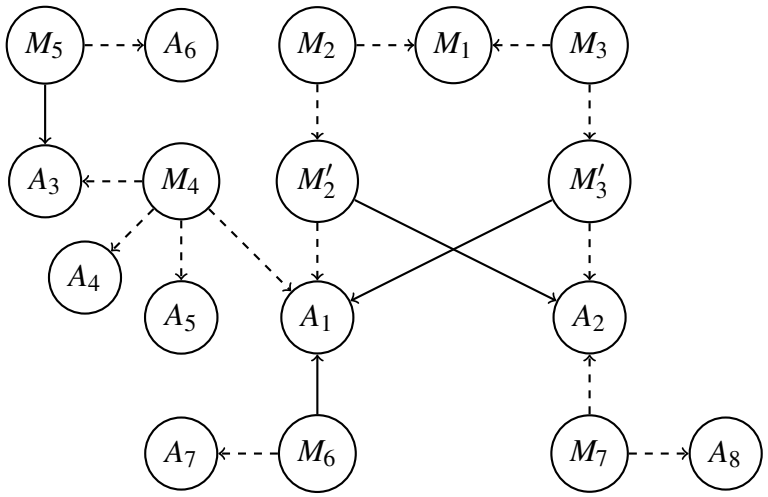


Figure 5.2: Meta-ASPIC graph with argument schemes from Section 5.3.1.1. Its reified graph is shown in Figure 5.3. Dashed arrows represent the obj function for illustrative purposes. Argument definitions are given in Table 5.1 on page 186. All arguments shown here are meta-arguments.

setting, but it would any argument against `Comp` if such an argument existed. In a real-life debate there would be additional arguments about how bad the consequences of choosing aluminium are, and if the benefits outweigh the costs. These kinds of arguments for decision making have been extensively studied in Chapter 3 of this thesis. At this point we can evaluate the meta-ASPIC system to obtain an argument graph, which can then be used to determine the acceptability of A_1 (the meta-argument for `Al`) and A_2 (the meta-argument for `Comp`). The graph is shown in Figure 5.2 on page 185. As mentioned earlier, we introduced several new object-level arguments, which are accounted for in the object-level graph G' with $G' = (\{\texttt{Al}, \texttt{Comp}, \texttt{Alternative}(\texttt{Comp}, \texttt{Al}), \ldots\}, \emptyset)$. In Figure 5.2, those object-level arguments are omitted. We only show their corresponding meta-arguments (for example, we show $A_2 = [\texttt{Comp}]$ but not `Comp`).

The *reified graph* of the system shown in Figure 5.2 is a standard Dung-style argument graph, visualised in Figure 5.3. As shown by the mutual attack between M'_2 and M'_3, attacks on the meta-level may be circular, even though meta-arguments themselves are strictly non-circular (Definition 68).

One effect of the argument from alternative, represented by M_1 to M_3, is that the two options `Comp` and `Al` are mutually exclusive - their meta-arguments A_1 and A_2 cannot be part of the same extension.

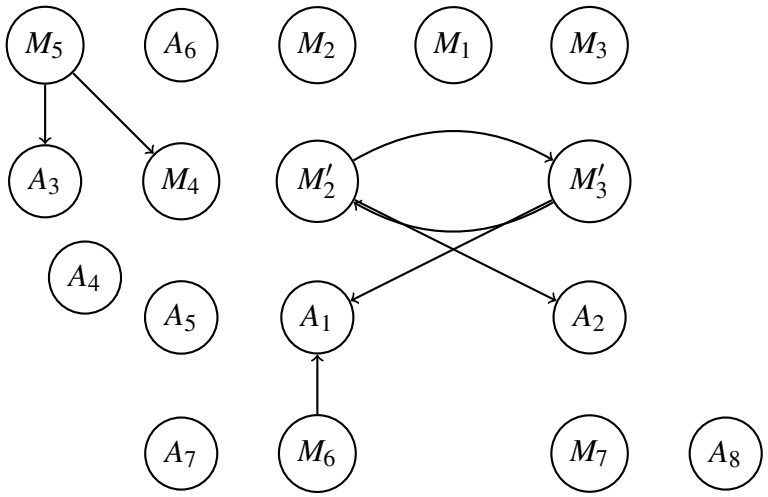


Figure 5.3: Reified graph of meta-ASPIC system from Figure 5.2.

Arg.	Content	Arg.	Content
A_1	$[\texttt{Al}]$	M_1	$[\texttt{Alternative(Comp,Al)}]$
A_2	$[\texttt{Comp}]$	M_2	$[M_1 \rightsquigarrow M'_2]$
A_3	$[\texttt{Expert(E,Mly)}]$	M'_2	$[A_2 \rightsquigarrow \neg A_1]$
A_4	$[\texttt{Domain(Mat,Mly)}]$	M_3	$[M_1 \rightsquigarrow M'_3]$
A_5	$[\texttt{Asserts(E,Al)}]$	M'_3	$[A_1 \rightsquigarrow \neg A_2]$
A_6	$[\texttt{NoPub(E,Mly)}]$	M_4	$[A_3, A_4, A_5 \rightsquigarrow A_1]$
A_7	$[\texttt{BadCons(Al)}]$	M_5	$[A_6 \rightsquigarrow \neg A_3]$
A_8	$[\texttt{GoodCons(Mat)}]$	M_6	$[A_7 \rightsquigarrow \neg A_1]$
		M_7	$[A_8 \rightsquigarrow A_2]$

Table 5.1: List of arguments in Figure 5.2

Critical Questions In Section 5.2.3 on page 175, critical questions were introduced as pointers to potential attacks on argument schemes. Based on that discussion, we can adopt the following definition of critical questions:

Definition 73 (Critical Question)**.** *Let (G, M_A) be a meta-*ASPIC *system and let $a, b \in M_A$ be meta-arguments. a is a* critical question *for b if*

1. $\mathsf{conc}(a) = \neg b$ *(exception) or*

2. $\mathsf{conc}(a) = \neg\varphi$ *for a $\varphi \in \mathsf{sup}(b)$ (condition)*

Example 54. *The following argument is a critical question because it attacks the condition of the scheme* argument from alternative *(cf. C_1 on page 170):*

$$[\rightsquigarrow \neg\texttt{alternative(composites,al)}]$$

It represents (the negative answer to) the critical question "Are there really only two options?". An example of an exception (cf. C_2 on page 172) is given by

$$\neg\texttt{personal\_attacks} \rightsquigarrow \neg[\texttt{unqualified}(\texttt{james}) \rightsquigarrow (\texttt{proponent}(\texttt{al},\texttt{james}) \rightsquigarrow \neg\texttt{al})]$$

*It represents the objection "*ad hominem *arguments are invalid".*

In the above sections we showed how some common argument schemes can be represented in meta-ASPIC, and we gave a definition of critical questions based on the discussion in Section 5.2.3 at the beginning of this chapter. This concludes our evaluation of argument schemes in meta-ASPIC.

5.3.1.2 The Truth Paradox in Meta-ASPIC

As we have effectively defined a new argumentation system, a natural question to ask is whether the results it produces are sound. Our system can be used to determine the acceptability of arguments, and one basic attribute of "soundness" should be that it does give two different answers to an argument's acceptability at the same time. This problem is closely related to the truth paradox in logic: Is the sentence s:"this sentence is false" true or false? We can attempt to translate this sentence into meta-ASPIC with the following argument q: $q = [[a] \rightsquigarrow \neg[a]]$ (for an argument a of the underlying argument graph). q intuitively says "if a is an argument then a is not an argument". However, the crucial difference between argument q and statement s is that q does not refer to itself. It is indeed not possible to construct an argument that refers to itself in meta-ASPIC (see Definition 68). This is why the acceptability of q can be decided unambiguosly in the same way as that of any other meta-argument: By reifying the meta-ASPIC system containing it (using reify, Definition 72) and then determining the acceptability of a in the resulting abstract argument graph. For our example, we know that if q cannot be part of an extension under any semantics because it attacks itself (by Definition 71 Cond. 2 and 3). Any set of arguments with a self-attacking argument is not conflict-free and can therefore not be a subset of an extension.

When considering the truth paradox in this context it is important to remember that an argument's acceptability is different from its truth status. Acceptability is a weaker notion than truth, and it is subjective (as for example in argument graphs with two mutually exclusive preferred extensions).

Our system gives rise to an intuitive and direct notion of meta-arguments, which we will explore in the next section.

5.3.2 Object- and Meta-Level Arguments

Let us consider two of the arguments from Section 5.2.1 again. They were originally introduced on page 171 as arguments A_7 and C_2, but we repeat them here in meta-ASPIC notation:

$$[[\texttt{proponent}(\texttt{al},\texttt{james})] \rightsquigarrow \neg[\texttt{al}]] \tag{A_7'}$$

$$[[\neg\texttt{personal\_attacks}] \rightsquigarrow \neg A_7'] \tag{C_2'}$$

The conclusion of C_2' is $\neg A_7'$ – an attack on another argument. We can see that A_7' is not about the merits of aluminium over composites, but about another *argument*. We can therefore call C_2' a meta-argument. Meta-arguments are quite common in engineering debates. They arise, for example, whenever a critical question in form of an exception is raised (cf. Definition 73, condition 1). In general, we can view argument schemes as arguments on the meta level, and their instantiations as arguments on the object level. There are several reasons why meta- and object-level should be distinguished.

First, this distinction is necessary if we want to classify debates by their topic. The topic of a debate is exactly what arguments are "about", and meta-arguments are more relevant to the topic of arguments (where questions such as "what is a legitimate argument?" are answered) than the topic of, say, materials engineering. Second, by distinguishing the levels in which a debate takes place, we can make our analysis of each level more fine-grained: on the meta-level of argument schemes, we can refer e.g. to Walton's work, and on the level of engineering we can refer to a different source of knowledge.

In this section we will define a method to stratify (clearly separate) the arguments in a meta-ASPIC system by their meta-level, that is, how far they are removed from the object level of domain-specific arguments. We start by considering obj, the object function from page 179 (Definition 69). It returns the arguments one meta-level below the given argument. The principle behind obj can be turned around: The characteristic

object function obj^* (below) maps a set of arguments A to all arguments that are about an argument in A (all direct meta-arguments for a). It is the meta-ASPIC equivalent of an argument graph's characteristic function $\mathscr{F}$.

Definition 74 (Characteristic Object Function)**.** *Let $\Omega = (G, M_A)$ be a meta-*ASPIC *system with $G = (A, Att)$ and let $G' = (M_A, Att') = \mathsf{reify}(\Omega)$. The* characteristic object function *of (G, M_A), $\mathsf{obj}^* : 2^{M_A} \to 2^{M_A}$, is defined as*

$$\mathsf{obj}^*(B) = \{a \in M_A \mid \mathsf{obj}(a) \subseteq B\}$$

Example 55. *Continuing with Example 51, we get $\mathsf{obj}^*(\emptyset) = \{a_1, a_2, a_3, a_4\}$, and $\mathsf{obj}^*(\mathsf{obj}^*(\emptyset)) = M_A$.*

Given a meta-ASPIC system $\Omega = (G, K)$, can we define a sequence $Ob_\Omega = (Ob_1, Ob_2, \ldots)$ with $Ob_1 = \mathsf{obj}^*(\emptyset)$ and $Ob_{n+1} = \mathsf{obj}^*(Ob_n)$. obj^* resembles the characteristic function $\mathscr{F}$ that is of argument graphs. $\mathscr{F}$ is used to define the grounded extension of an argument graph, by computing the fixed point of $\mathscr{F}(\mathscr{F}(\ldots\mathscr{F}(\emptyset)))$. This fixed point exists for obj^* too, but it is not very interesting because it always contains the entire set of arguments A.

The abstraction index of an argument a is the index of the first meta-level that includes a.

Definition 75 (Abstraction Index)**.** *Let $\Omega = (G, M_A)$ be a meta-*ASPIC *system with $Ob_\Omega = (Ob_1, Ob_2 \ldots)$. Let $a \in M_A$. The* abstraction index *of a, abbreviated $I_\Omega(a)$, is the lowest i such that $a \in Ob_i$.*

The abstraction index is defined for every argument in a meta-ASPIC system. It can be calculated directly from the argument a, without needing to know the entire meta-ASPIC system.

Definition 76 (Abstraction Index Function)**.** *Let (G, M_A) be a meta-*ASPIC *system and let $a \in M_A$. The function $\mathsf{abstract}(a)$ is defined as*

$$\mathsf{abstract}(a) = \begin{cases} 1 & \text{if } a = [a'] \text{ for an } a' \in A \\ i+1 & \text{otherwise, where } i = \max\{\mathsf{abstract}(b) \mid b \in \mathsf{obj}(a)\} \end{cases}$$

Proposition 39. *Let* $\Omega = (G, M_A)$ *be a meta-*ASPIC *system with* $G = (A, Att)$ *and let* $Ob_\Omega = (Ob_1, Ob_2, \ldots)$. *Let* $a \in M_A$. *Then*

$$I_\Omega(a) = \mathsf{abstract}(a)$$

Proof. First assume that $a = [a']$ for an $a' \in A$. Then, $\mathsf{obj}(a) = \emptyset$ and $\mathsf{obj}(a) \subseteq \emptyset$, so $a \in \mathsf{obj}^*(\emptyset)$. Therefore, $I_\Omega(a) = 1 = \mathsf{abstract}(a)$.

We prove the second case ($\mathsf{obj}(a) = \{a_1, \ldots, a_n\}$) via induction. Specifically we prove the following statement: $\forall n \in \mathbb{N} \geq 1$: If $\max\{\mathsf{abstract}(b) \mid b \in \mathsf{obj}(a)\} = n$ then $I_\Omega(a) = n + 1 = \mathsf{abstract}(a)$.

Base case, $n = 1$ If $\max\{\mathsf{abstract}(b) | b \in \mathsf{obj}(a)\} = 1$, then by Definition 76, for all $b \in \mathsf{obj}(a)$, $b = b'$ for a $b' \in A$, so $\mathsf{obj}(a) \subseteq Ob_1$ and $I_\Omega(a) = 2 = \mathsf{abstract}(a)$.

Induction case Let $n = k+1$ with $k \geq 1$, and suppose $\max\{\mathsf{abstract}(b) \mid b \in \mathsf{obj}(a)\} = n$. Then for all $b \in \mathsf{obj}(a)$, $\mathsf{abstract}(b) \leq n$, and there exists a $c \in \mathsf{obj}(a)$ such that $\mathsf{abstract}(c) = n$. Now we show that for all $b \in \mathsf{obj}(a)$, $I_\Omega(b) \leq n$. Let $b \in \mathsf{obj}(a)$. Either $\mathsf{abstract}(b) < n$, so $\mathsf{abstract}(b) \leq k$, so we can apply the hypothesis and get $\mathsf{obj}^*_\Omega(b) \leq n$. If $\mathsf{abstract}(b) = n$, then by Definition 76, for all $b' \in \mathsf{obj}(b)$, $\mathsf{abstract}(b') \leq k$ and we can apply the induction hypothesis to get $\mathsf{obj}(b) \subseteq O_k$, so $b \in \mathsf{obj}^*(O_k) = O_{k+1} = O_n$ so $I_\Omega(b) \leq n$. Since there exists an argument c such that $\mathsf{abstract}(c) = n$, we can apply the same reasoning to obtain $I_\Omega(c) = n$. Then $\mathsf{obj}(a) \subseteq O_n$, so $a \in \mathsf{obj}^*(O_n)$ and we get $I_\Omega(a) = n + 1 = \max\{\mathsf{abstract}(b) \mid b \in \mathsf{obj}(a)\} + 1 = \mathsf{abstract}(a)$. □

Example 56. *For* m_4 *and* m_5 *from Example 51, we get* $\mathsf{abstract}(m_5) = \mathsf{abstract}(m_4) = 2$. *All other arguments in that example have an abstraction index of 1.*

Since all arguments eventually appear in Ob_Ω, the abstraction index exists for every argument a. We can use I_Ω to compare the "level of abstraction" of arguments. An argument a is said to be an *object-level* argument if $I_\Omega(a) = 1$.

With the obj function (Definition 69) we can inspect arguments to see which other arguments they talk about. The characteristic object function obj^* of a meta-ASPIC system inverts this perspective as it maps a set of arguments S to the set of arguments whose *domain* is S. Both obj and obj^* have counterparts in Dung's original argumentation system, namely in the attacks-relation Att and in the characteristic function $\mathscr{F}$. There are two principal differences between the two functions.

First, the polarity of obj is always positive. The "is-about" relation is intuitively transitive – if argument a is about b and b is about c then a is also about c, or at least the relationship of a and c is similar to the one of a and b. With attacks on the other hand, the polarity is inverted. If a attacks b and b attacks c then a defends c, so a is in a positive relationship with c and in a negative one with b. For this reason, the characteristic function $\mathscr{F}$ is defined in terms of the "defends" relationship, covering two attacks in each successive application. The characteristic object function obj^* only covers one level of the "talking-about" relation given by obj.

The second difference only comes to light when Dung's argument frameworks are instantiated. In instantiations of argument graphs, attacks are the result of an inconsistency between the conclusion of one argument (the attacker a) and the assumptions or the conclusion of another argument (the attackee b). However, if either a or b was removed from the argument graph, by deleting the knowledge it is based on, then the attack would disappear, but the remaining argument would still be part of the argument graph – assuming a and b do not overlapage The attack by a on b only exists because both a and b are present in the graph, and not because a explicitly refers to b or vice versa. Attacks in instantiations of Dung's argument graphs are incidental. Support in meta-ASPIC systems (as evidenced by the obj function) on the other hand is an essential part of meta-arguments, because they are about other arguments. If $a \in \mathsf{obj}(b)$ then it is impossible to remove a from the system without changing b.

5.3.2.1 A Hierarchy of Meta-Argumentats

The characteristic object function obj^* (Definition 74) makes it possible to stratify the arguments in an argument graph G according to their first appearance in the sequence $Ob_\Omega = (Ob_1, \ldots, Ob_n)$. The first set of arguments, Ob_1, can be called the object level because it contains only arguments that are truly atomic in Dung's sense and thus are making claims about the domain (for example about engineering decisions) rather than about other arguments. The second level, Ob_2, adds arguments that are based on arguments from the first level, and so on.

In this hierarchy we can define some concepts that are commonly part of higher-order argumentation, such as attacks on attacks, and attacks on supports.

Definition 77 (Hierarchy of Argument Graphs)**.** *Let* Ω *be a meta-*ASPIC *system and*

let $Ob_\Omega = (Ob_1, Ob_2, \ldots)$ *and let* $(G, Att^*) = \mathsf{refify}(\Omega)$ *be its reified argument graph. The* hierarchy of argument graphs *of* Ω *is defined as* $G_\Omega = (G_1, G_2, \ldots)$ *where each* $G_i = \mathsf{reify}(\Omega_i)$ *for the meta-*ASPIC *system* $\Omega_i = (G_i', M_A \cap Ob_i)$ *with* $G_i' = (Ob_i, Att^* \cap (Ob_i \times Ob_i))$.

In a hierarchy of argument graphs $(G_1, \ldots,)$, the first graph G_1 is the object-level graph containing all atomic arguments of the underlying meta-ASPIC system.

Proposition 40. *Let* $\Omega = (G, M_A)$ *be a meta-*ASPIC *system with* $G = (A, Att)$ *and let* $G_\Omega = (G_1, \ldots)$. *Then* $G_1 = (A', Att')$ *where*

$$A' = \{[a] \mid a \in A\}$$
$$Att' = \{([a], [b]) \mid (a, b) \in Att\}$$

Proof. To prove the claim it suffices to show that $Ob_1 = A'$, because then the "attacks" part of G_1 is uniquely determined by Att (since $Att_1 = Att^* \sqcap Ob_1$).

$$\begin{aligned}
Ob_1 &= \mathsf{obj}^*(\emptyset) \\
&= \{a \in M_A \mid \mathsf{obj}(a) \subseteq \emptyset\} && \text{(By Definition 74)} \\
&= \{a \in M_A \mid \mathsf{obj}(a) = \emptyset\} \\
&= \{a \in M_A \mid a = [b] \text{ for a } b \in A\} && \text{(By Definition 69)} \\
&= A'
\end{aligned}$$

□

Example 57. *Example 51 gives rise to a hierarchy of argument graphs with depth 2. The graphs* $G_1 = (Ob_1, Att_1)$ *and* $G_2 = (Ob_2, Att_2)$ *are defined as*

$$Ob_1 = \{m_1, m_2, m_3, m_4\} = \{[a] \mid a \in A\}$$
$$Att_1 = \{(m_1, m_3), (m_2, m_4), (m_4, m_2)\} = \{([a], [b]) \mid (a, b) \in Att\}$$
$$Ob_2 = Ob_1 \cup \{m_5, m_6\}$$
$$Att_2 = Att_1 \cup \{(m_6, m_4)\}$$

The graphs are shown in Figure 5.4.

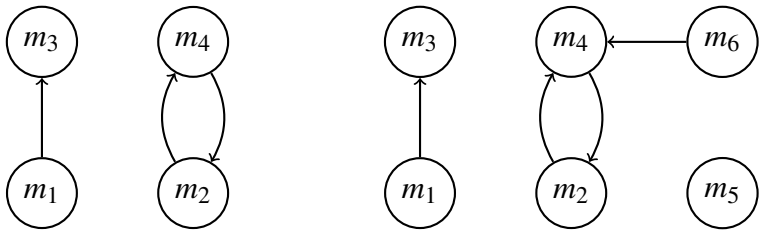


Figure 5.4: Argument graphs for Example 57. G_1 is shown on the left, G_2 (a supergraph of G_1) on the right.

The hierarchy of meta-arguments is monotonic, in the sense that lower-level argument graphs are contained in higher-level ones.

Proposition 41. *Let Ω be a meta-*ASPIC *system and let $G_\Omega = (G_1, G_2, \ldots)$. For every $i \geq 1$, $G_i \sqsubseteq G_{i+1}$.*

Proof. Let Ω, G_Ω as required and let $i \geq 1$. Let $G_i = (A_i, Att_i)$ and $G_{i+1} = (A_{i+1}, Att_{i+1})$. To show that $G_i \sqsubseteq G_{i+1}$, we need to show that $A_i \subseteq A_{i+1}$ and $Att_i \subseteq Att_{i+1}$.

1. $A_i \subseteq A_{i+1}$**:** Let $a \in A_i$, so by Definition 77 $a \in Ob_i$, so $\mathsf{obj}(a) \subseteq Ob_i$. By Definition 74, $a \in \mathsf{obj}^*(Ob_i) = Ob_{i+1} = A_{i+1}$.

2. $Att_i \subseteq Att_{i+1}$ Let $(a,b) \in Att_i$. By Propage 41 part 1, a and $b \in A_{i+1}$, so by Definition 77, $(a,b) \in Att_{i+1}$. □

5.3.2.2 Attacks on Attacks

Meta-arguments with conclusions of the form $\neg\varphi$ result in object-level attacks. As a result, we can argue about those attacks just as we can about other arguments.

Definition 78 (Meta-Attack)**.** *Let $\Omega = (G, M_A)$ be meta-*ASPIC *system and let $a \in M_A$ and let $b \in M_A$ with $b = [\varphi'_1, \ldots, \varphi'_k \rightsquigarrow \neg a]$. Then b is a* meta-attack *on a.*

A meta-attack can itself be attacked like any other meta-argument, resulting in attacks on attacks.

Example 58. *Consider argument $m_6 = [m_1 \rightsquigarrow \neg m_4]$ from Example 51. A meta-attack on m_6 is given by m_7 with $[m_5 \rightsquigarrow \neg m_6]$. In the resulting argument graph, m_7 attacks m_6. The resulting graph is shown in Figure 5.5.*

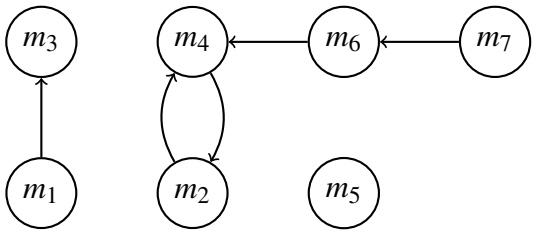


Figure 5.5: Argument graph for Example 58, showing a meta-argument m_7 attacking the meta-attack m_6.

Example 59. *A realistic example of attacks on attacks can be seen in Figure 5.2 on page 185. It involves arguments M_1 to M_3 and A_1, A_2 which are repeated below:*

$$A_1 = [\texttt{Al}] \qquad M_1 = [\texttt{Alternative}(\texttt{Comp}, \texttt{Al})]$$
$$A_2 = [\texttt{Comp}] \qquad M_2 = [M_1 \rightsquigarrow M_2']$$
$$M_2' = [A_2 \rightsquigarrow \neg A_1]$$
$$M_3 = [M_1 \rightsquigarrow M_3']$$
$$M_3' = [A_1 \rightsquigarrow \neg A_2]$$

Here, arguments M_1 to M_3 represent an application of the scheme "argument from alternative", to conclude that A_1 and A_2 are mutually exclusive. Since M_2 and M_3 mutually attack each other and *are meta-attacks by Definition 78, both are examples of attacks on attacks. Please refer to Section 5.3.1.1 for a discussion of the underlying argument scheme.*

Since meta-attacks are themselves arguments, we can compute their abstraction index I_Ω. Every meta-attack on an argument a has a higher abstraction index than a:

Proposition 42. *Let b be a meta-attack on an argument a. Then $I_\Omega(b) > I_\Omega(a)$.*

Proof. Follows from Definition 78 and the proof of Propage 39 □

5.3.2.3 Acceptability of Arguments in Meta-ASPIC

In this section we discuss the relationship between the output of a meta-ASPIC system and its underlying argument graph. Our goal is to show that the two functions meta (taking an argument graph to a meta-ASPIC system) and reify (interpreting a meta-ASPIC system as a regular abstract argument graph) are well-behaved. We show

this by proving two properties: (a) for every object-level graph G there exists a meta-ASPIC system $\mathsf{meta}(G) = (G, M_A)$ whose extensions[2] exactly match the extensions of G, and (b) for any meta-level graph $M = (G, M_A)$ of an object-level graph G, if we first "materialise" all its attacks by computing $\mathsf{reify}(M)$, and then take the minimal graph $\mathsf{meta}(\mathsf{reify}(M))$, we get a meta-ASPIC system whose extensions match those of $\mathsf{reify}(M)$.

Notation Before we look at the two statements in more detail, we would like to clarify the notation used in the following paragraphs. We are going to use the symbol $\simeq$ to mean the "is isomorphic to" relationship between two graphs, specifically between two abstract argument graphs. We will then introduce a similar notion for meta-ASPIC systems (consisting of a graph and a set of rules), which will be denoted by $\cong$. So $\simeq$ relates two graphs and $\cong$ relates two meta-ASPIC systems (not to be confused with $\cong_D$, the equivalence relation on options for a decision frame, introduced in Chapter 3 on page 72).

The first statement (a) trivially holds: We simply choose the "minimal" meta-level system for G, whose only meta-arguments are of the form $[a]$ for the underlying object-level arguments.

Proposition 43. *Let G be an argument graph and let $\mathsf{meta}(G) = (G, M_A)$ with $M_A = \{[a] \mid a \in A\}$. Then,*

$$\mathsf{reify}(\mathsf{meta}(G)) \simeq G$$

Proof. We prove the claim by showing that the output of $\mathsf{meta}(G)$ is a graph $G' = (A', Att')$ with $A' = \{[a] \mid a \in A\} = M_A$ and $Att' = \{([a],[b]) \mid (a,b) \in Att\}$. By Definition 71 Cond. 1, all attacks $(a,b) \in Att$ result in attacks $([a],[b]) \in Att'$. By construction of M_A, none of the other conditions of Definition 71 is met, so $Att' = \{([a],[b]) \mid (a,b) \in Att\}$, and $A' = M_A$ by construction, so the claim holds. □

Proposition 43 shows how every argument graph can be lifted into meta-ASPIC, resulting in the minimal meta-level system M. Since $\mathsf{reify}(M)$ is isomorphic to G, its extensions are exactly the same.

The second statement (b) is formalised similarly, but we first need to define the meaning of "match" - that is, we need the meta-ASPIC counterpart of the equivalence

[2] In this section, when we say "the extensions of the meta-ASPIC system Ω" we mean the extensions of its reified graph $\mathsf{reify}(\Omega)$.

relation $\simeq$ in the world of argument graphs. The following definition generalises equivalence to meta-ASPIC systems:

Definition 79 (Isomorphic meta-ASPIC system)**.** *Two meta-*ASPIC *systems* $M = (G, M_A), L = (G', L_{\mathscr{A}'})$ *with* $G = (A, Att)$ *and* $G' = (A', Att')$ *are* isomorphic*, abbreviated* $M \cong L$*, if*

1. $G \simeq G'$*, witnessed by a bijection* $f : A \to A'$*, and*

2. $L_{\mathscr{A}'} = \{\mathsf{map}_f(r) \mid r \in M_A\}$ *with*

$$\mathsf{map}_f(r) = \begin{cases} \mathsf{map}_f(\varphi_1), \ldots, \mathsf{map}_f(\varphi_n) \rightsquigarrow \mathsf{map}_f(\varphi) & \text{if } r = \varphi_1, \ldots, \varphi_n \rightsquigarrow \varphi \\ \mathsf{map}_f(\varphi_1), \ldots, \mathsf{map}_f(\varphi_n) \rightsquigarrow \neg\mathsf{map}_f(\varphi) & \text{if } r = \varphi_1, \ldots, \varphi_n \rightsquigarrow \neg\varphi \\ [f(a)] & \text{if } r = [a] \text{ for } a \in A \end{cases}$$

In Propage 43, we looked at the result of combining the reify function with the meta function, in effect a round-trip from (Dung's) argument graphs to meta-ASPIC systems to argument graphs. Statement (b) is similar, but it works in the other direction: We start with a meta-ASPIC system, take its output, and then apply the meta function to it, resulting in another meta-ASPIC system. Instead of using graph isomorphism as a notion of equality, we will use isomorphism of meta-ASPIC systems. However, if we simply stated that $\mathsf{meta}(\mathsf{reify}(M)) \cong M$ for some meta-ASPIC M, then our formalism would not be any more expressive than Dung's argument graphs (and indeed this proposition does not hold, because the reify function "reifies" meta-attacks which then become part of the graph that is the argument to meta – see Example 60 below). Therefore, we have to adopt a looser notion of equality: When combining meta with reify, we may not immediately get the same result again, but we are guaranteed to get the same result after at most one iteration. In other words, the operation $\mathsf{meta}(\mathsf{reify}(M))$ is idempotent.

Proposition 44. *Let* $M = (G, M_A)$ *be a meta-*ASPIC *system. Then*

$$\mathsf{meta}(\mathsf{reify}(\mathsf{meta}(\mathsf{reify}(M)))) \cong \mathsf{meta}(\mathsf{reify}(M))$$

Proof. Let $M = (G, M_A)$ be a meta-ASPIC system with an argument graph $G = (A, Att)$. Let $M_1 = (G_1, M_{1_A}) = \mathsf{meta}(\mathsf{reify}(M))$ and let $M_2 = (G_2, M_{2_A}) = \mathsf{meta}(\mathsf{reify}(\mathsf{meta}(\mathsf{reify}(M))))$ be the two meta-ASPIC systems resulting from applying reify and meta to M. To show that $M_1 \cong M_2$ (by Definition 79), we first need to show $G_1 \simeq G_2$, and then $M_{2_A} = f^*(M_{1_A})$ for a bijection f between G_1 and G_2.

1. $G_1 \simeq G_2$ Let $G' = (A', Att') = \mathsf{reify}(M_1)$. Then $\mathsf{meta}(G') = M_2 = (G_2, M_{2_A})$ and $G_2 = G'$. By the definition of meta (in Propage 43), all arguments in M_{1_A} are of the form $[a]$, so by Definition 71, for all attacks in $(a,b) \in Att'$, $a = [a']$ and $b = [b']$ for some arguments $[a]$ and $[b] \in M_{1_A}$ (specifically, a and b are not meta-arguments). So all attacks in Att' have a corresponding attack in Att_1, and hence there exists a bijection between attacks and arguments in G' and attacks and arguments in G_1, and so $G_1 \simeq G'$. Since $G' = G_2$ (by Definition of meta), $G1 \simeq G_2$.

2. $M_{2_A} = f^*(M_{1_A})$ Let $G' = (A', Att') = \mathsf{reify}(M_1)$. Then $\mathsf{meta}(G') = M_2 = (G_2, M_{2_A})$ and $G_2 = G'$, as above. As already established, there exists a bijection f between G_1 and G_2, and by the same reasoning as above, all arguments $b \in M_{2_A}$ are of the form $b = [b']$ for a $b' \in A'$ - so for every $b' \in A'$, there exists a $b \in M_{2_A}$ such that $b = [f(b')]$ and the claim holds. □

Example 60. *To see why it is not the case that* $\mathsf{meta}(\mathsf{reify}(M)) \cong M$ *in Propage 44, consider this counter-example. Let* $G = (\{a_1, a_2\}, \emptyset)$ *be an argument graph with two arguments and no attacks. Let* $M = (G, \{[a_1], [a_2], [[a_1] \rightsquigarrow \neg[a_2]]\})$ *be a meta-*ASPIC *system based on G. Note that the only additional information in M, besides what is required by Definition 70 on page 180, is an attack by* $[a_1]$ *on* $[a_2]$*. Using the symbols*

$m_1 = [a_1]$, $m_2 = [a_2]$, $m_3 = [[a_1] \rightsquigarrow \neg[a_2]]$, *we get*

$$\begin{aligned}
&\mathsf{meta}(\mathsf{reify}(M)) \\
&\qquad (\textit{Definition 72}) \\
&= \mathsf{meta}((\{m_1, m_2, m_3\}, \{(m_3, m_2)\})) \\
&\qquad (\textit{ Let } G' = \mathsf{reify}(M) = (\{m_1, m_2, m_3\}, \{(m_3, m_2)\})) \\
&= \mathsf{meta}(G') \\
&\qquad (\textit{Definition of } \mathsf{meta} \textit{ in Propage 43}) \\
&= (G', \{[m_1], [m_2], [m_3]\})
\end{aligned}$$

We can see that G is not isomorphic to G' (and thus M is not isomorphic to $\mathsf{meta}(G)$*), because G has two arguments and no attacks, whereas G' has three arguments and one attack. On the other hand, M contains two object-level arguments $[a_1]$ and $[a_2]$ and one meta-attack, but* $\mathsf{meta}(G')$ *contains only the object-level arguments $[m_1]$, $[m_2]$ and $[m_3]$. The meta-attack m_3 in M has been reified as an object-level attack in G'.*

The results 43 and 44 together show that meta and reify are "almost" inverses - they always converge after a single round trip (modulo graph isomorphism). This shows us that meta-ASPIC is really a language on top of Dung's argumentation system[3] , and that our framework is well-behaved and has a standard interpretation in the underlying theory of abstract argumentation.

5.4 Case Study: Choosing a Drilling Technique

As part of project DEEPFLOW, the model developed in Section 5.3 was applied to the problem of choosing a drilling technique for the production of a wing component. The decision between conventional and orbitall drilling involves many parameters and has to be made for each case individually, because there is no clear favourite between the two. In Section 5.4.1 we give a brief overview of the subject

[3] In the language of abstract algebra, espage Lawvere [134], we have shown that meta is left adjoint to reify.

5.4.1 Conventional or Orbital Drilling?

An important part of any engineering project is manufacturing design, that is, to decide how the new product will be manufactured. Manufacturing design covers many areas such as shopfloor layout, machining parameters and assembly tasks. A typical choice that has to be made as part of preparing the machinery is whether to use conventional or orbital drilling. In orbital drilling, the cutting tool (drill) is rotates around two axes – its own axis, as in conventional drilling, and an eccentrical axis. As a result, the hole diameter is greater than the drill diameter. Advantages of orbital over conventional drilling are reduced thrust force and a smaller broken-off pieces (chips). The smaller drill also makes it possible to extract the chips with a vacuum during the drilling. As a result, less heat develops and the chippings are prevented from damaging the structure. There are also several reasons against using orbital drilling: It requires specialised machinery and drills and is not suitable for all metals.

While the arguments above play a role in the choice of drilling method, much greater weight is afforded to the concrete parameters of the tasks, for example hole diameter and depth and material. It is common to undertake experiments to determine how the material will react to different drilling methods. The experimental data then forms the basis of arguments used in the decision making process. The same data can be interpreted several ways, leading to different conclusions.

In a case study with Queen's University Belfast[4] we compared the two drilling methods for a variety of different materials, hole dimenions and temperatures.

5.4.2 Experiment Results

For the experiments, conventional and orbital drilling methods were tried compared with varying parameters, resulting in sixty-four different configurations. For each configuration, between one and four holes were drilled and the outcome was assessed in three categories: Tool wear, hole quality and surface finish.

The complete list of parameters is

1. Environment: Dry, coolant, wet, MQL

2. Drill bit: Carbide, PCD, HSS, HSS-Co

[4]To be published; all experimental data courtesy of David Payne, `d.payne@qub.ac.uk`

3. Drill bit coating: Yes/No

Several other parameters remained fixed due to time constraints. These include material (one layer of CFRP composites and one layer of titanium), hole diameter ($\frac{5}{16}$ inch), drill speed and drill bit condition. After each experiment, the results were ranked on a scale of one to five. The results can be found on page 201. Table 5.2 shows a selection of the experimental results. The first two rows show, for example, that switching from a dry environment to one that is cooled with a coolant liquid results in slightly better hole and surface quality, while the tool wear stays the same. However, both values (1 and 2) are relatively weak on a scale of 1 to 5. The next row (experiment 5) shows that orbital drilling performs even worse in the same context. On the other hand, orbital drilling yields adequate results when used with MQL cooling and a carbide drill without coating (experiment 16).

In order to use the results for decision making, one has to choose which configuration is the closest to the actual setting, and interpolate when data is missing – in other words, one has to draw an analogy. This analogy is where arguments and counter-arguments are produced.

Example 61. *Assume that we are working with an aluminium piece and a $\frac{3}{4}$ inch hole diameter. We can make the following assumptions: The changed diameter probably has the same effect on both types of drilling, and (for the purpose of this particular drilling step) aluminium is more similar to titanium than to composites. Based on those assumptions and the results in Table 5.2, we argue that conventional drilling is better suited because it results in a higher quality and lower tool wear.*

5.4.3 Interpreting Experimental Data Using Meta-ASPIC

We treat the experimental data from Table 5.2 as a set $D = \{r_1, \ldots, r_{64}\}$ whose elements are the rows of the table. There is a set of functions number, type, material for the attributes of the table, mapping rows to field values: $\mathsf{type}(r_5) = \text{Orbital}$ and so forth. Arguments are formed using both the field name functions and any number of user-defined terms such as cost or experience_With.

Expage No.[a]	Type	Dia.[b]	Material	Environment	Drill bit	Coating	Tool Wear	Hole Quality	Surface
1	Conventional	$\frac{5}{16}$	CFRP, Titanium	Dry	Carbide	Yes	2	1	1
2	Conventional	$\frac{5}{16}$	CFRP, Titanium	Coolant	Carbide	Yes	2	2	2
5	Orbital	$\frac{3}{16}$	CFRP, Titanium	Dry	Carbide	Yes	1	1	1
15	Orbital	$\frac{3}{16}$	CFRP, Titanium	Wet	Carbide	No	2	2	2
16	Orbital	$\frac{3}{16}$	CFRP, Titanium	MQL	Carbide	No	3	3	3
20	Conventional	$\frac{5}{16}$	CFRP, Titanium	MQL	PCD	Yes	3	4	4
30	Orbital	$\frac{3}{16}$	CFRP, Titanium	Coolant	PCD	No	1	3	2
33	Conventional	$\frac{5}{16}$	CFRP, Titanium	Dry	HSS	Yes	1	1	1
36	Conventional	$\frac{5}{16}$	CFRP, Titanium	MQL	HSS	Yes	3	2	2
40	Orbital	$\frac{3}{16}$	CFRP, Titanium	MQL	HSS	Yes	2	2	2
43	Conventional	$\frac{5}{16}$	CFRP, Titanium	Wet	HSS	No	1	1	1
46	Orbital	$\frac{3}{16}$	CFRP, Titanium	Coolant	HSS	No	1	1	1
49	Conventional	$\frac{5}{16}$	CFRP, Titanium	Dry	HSS-Co	Yes	1	1	1
52	Conventional	$\frac{5}{16}$	CFRP, Titanium	MQL	HSS-Co	Yes	3	2	2
56	Orbital	$\frac{3}{16}$	CFRP, Titanium	MQL	HSS-Co	Yes	2	2	2
64	Orbital	$\frac{3}{16}$	CFRP, Titanium	MQL	HSS-Co	No	2	1	1

Table 5.2: Experimental results for conventional and orbital drilling, as discussed in Section 5.4.2 (page 199). The numbers in "Tool Wear", "Hole Quality" and "Surface" columns are qualitative assessments by an engineer.

[a] Selected results only. Full results available upon request. All experimental data courtesy of David Payne (QUB)
[b] Drill Diameter

5.4.3.1 Gathering Support for Orbital or Conventional Drilling

For example, to say that our machining process does not support application of coolant for orbital drilling, and as a result we cannot use evidence from experiments in which coolant was used, we can write

$$\mathsf{experiment}(X), \mathsf{type}(X, \mathsf{Orbital}), \mathsf{environment}(X, \mathsf{Coolant})$$

$$\rightsquigarrow \neg\mathsf{supportingEvidence}(X, \mathsf{Orbital})$$

where $\mathsf{experiment}(X)$ will be grounded for each row r_1 to r_{64}, type, environment are binary predicates that can be used to check for specific field values, and supportingEvidence is a user-defined term that indicates whether a result X can be used as supporting evidence for its method, that is for $\mathsf{type}(X)$. This schematic rule is instantiated for example with $X = r_{46}$, resulting in the following argument:

$$A_1 = [\mathsf{experiment}(r_{46}), \mathsf{type}(r_{46}, \mathsf{Orbital}), \mathsf{environment}(r_{46}, \mathsf{Coolant})$$
$$\rightsquigarrow \neg\mathsf{supportingEvidence}(r_{46}, \mathsf{Orbital})]$$

The last remaining step is to use arguments with conclusion $\mathsf{supportingEvidence}(\_, X)$ as support for the claim $\mathsf{choose}(X)$:

$$\mathsf{supportingEvidence}(\_, X) \rightsquigarrow \mathsf{choose}(X)$$

The underscore _ signifies a free variable that is not referenced in the rule head. When evaluating arguments, the system only deals with completely grounded arguments such as A_1 above, so the standard definition of the reified graph of a meta-ASPIC system (Definition 72) can be used to determine acceptable arguments.

With the rules described above, we will end up with some arguments with conclusion $\mathsf{choose}(\mathsf{Orbital})$ and some arguments with conclusion $\mathsf{choose}(\mathsf{Conventional})$. Assuming an appropriate encoding of the mutual exclusivity of Orbital and Conventional, the resulting graph will have two preferred extensions, one with arguments in support of conventional drilling and one with arguments in support of orbital drilling. The exact number of arguments depends on the user-defined rules.

However, a decision still has to be made, even if there is no clear favourite emerging from the argument graph. We can resolve such disputes by adding additional information (in form of arguments), as the next section shows.

5.4.3.2 Combining Empirical Evidence With Other Argument Schemes

In addition to arguments that are directly supported by experimental results, we can create arguments from argument schemes that take into account the circumstances of our particular problem.

One example is the resolution of conflicts in group decision making. When the options have been evaluated, for example using the methods described in Chapter 3, but there is no clear winner, one way do resolve the tie is by letting the most senior team member decide, using an "argument from seniority". Because the argument from seniority overrides a junior's argument by virtue it being made by a junior person, rather than the quality of their argument, it is counted as an *ad hominem* argument. While *ad hominem* arguments are often disallowed in professional debates (cf. [126]), arguments from seniority are more readily accepted, especially when a senior decision is the only way to resolve a tie.

The example can be modelled using a standard logic-programming approach, as discussed in [127]:

$$\mathsf{seniorTo}(\mathsf{alice},\mathsf{bob}).$$

$$\mathsf{proponent}(\mathsf{alice},\mathsf{Orbital}).$$

$$\mathsf{proponent}(\mathsf{bob},\mathsf{Conventional})$$

$$\mathsf{proponent}(X,Y),\mathsf{proponent}(R,S),\mathsf{seniorTo}(X,R),\mathsf{neq}(Y,S) \rightsquigarrow \neg\mathsf{choose}(S)$$

As usual, the last rule will be grounded for each valid assignment of the variables (neq is a built-in predicate that holds whenever its two arguments are not equal). As a result, we get an argument

$$A_2 = [\mathsf{proponent}(\mathsf{alice}, \mathsf{Orbital}), \mathsf{proponent}(\mathsf{bob}, \mathsf{Conventional}),$$
$$\mathsf{seniorTo}(\mathsf{alice}, \mathsf{bob}), \mathsf{neq}(\mathsf{Orbital}, \mathsf{Conventional})$$
$$\rightsquigarrow \neg\mathsf{choose}(\mathsf{Conventional})]$$

A_2 now attacks any argument with claim $\mathsf{choose}(\mathsf{Conventional})$ asymmetrically. When looking at the grounded extension of the resulting graph we will find the claim choose only for one drilling method, orbital.

5.4.4 Software Prototype

In this section we describe a prototypical implementation of the case study on orbital and conventional drilling (Section 5.4.3), giving evidence of the practical usefulness of our argumentation system.

5.4.4.1 Overview

The program is an editor and interpreter for argument schemes. Argument schemes are entered using a PROLOG-like syntax with rules, variables and a "negation" operator $\neg$. There are two interpreters: The first produces a graph-based interpretation for visialising arguments and their extensions. The scond produces an SQL query that instantiates schemes with acceptable subsets of the argument graphs.

The program consist of three layers: The user interface, a REST[5] application server and an SQL server. The only purpose of the REST server (middle layer) is to forward SQL queries to the database server, which cannot be accessed directly by the UI because it is behind a corporate firewall. The comptuations are done on the client and by the SQL server.

The programming language Haskell [136] was chosen for implementing the software, because there already exists an open-source implementation of Dung's argument graphs [137, 138], and because its type system, terse syntax and lazy evaluation semantics make it well-suited for implementing domain-specific languages and interpreters. Another reason is that Haskell programs can be compiled to JavaScript, so they can

[5]Representational State Transfer, a standard architecture for communication between servers in the world wide web [135]

```
data Rule = Rule{
 body :: [RulePart],
 headName :: PredicateName,
 headVar :: UnaryVar
}

 data RulePart =
      Predicate{ pr :: PredicateDefinition }
    | Aggr{
        operator :: AggOperator,
        pr :: PredicateDefinition,
        comp :: CompOperator,
        compWith :: String
      }
```

Figure 5.6: Rule parts and rule types in Haskell

run in ordinary web browsers without having to install them. This was an external requirement from SAP under whose supervision the software was developed.

5.4.4.2 Implementation Details

We will now discuss some aspects of the implementation in more detail. We start with the internal representation of arguments.

An argument in our system is a tree. The nodes of the tree are rules $p_1(X_1), \ldots, p_n(X_n) \rightsquigarrow p(X)$ (where the p_i are predicates) and the children of a node are those rules whose conclusion is used in the rule body (a *supports* relationship).

Typed Predicates A special feature of our definition is that the predicates p_i are *typed* in the same way as tables in a relational database. For example, the type of $\texttt{engineer}(X)$ is $(\mathsf{Int}, \mathsf{Text}, \mathsf{Text})$, so each row of the table describes an engineer with ID number, first name and last name. The system has a number of built-in predicates that are based on the tables of the underlying database. Every tables is assigned a "primary" predicate, such as `engineer`. If a table has foreign keys, then each of the foreign-key relations is assigned an additional, binary predicate – for example, if there was a table of "conclusions" and each conclusion was supported by an engineer, a $\mathsf{supports}(X,Y)$ predicate would be generated.

Implementation of Rules The listing in Figure 5.6 shows the definition of a rule in Haskell. A rule consists of a list of rule parts (the rule body), and a unary predicate (the rule head). A rule part is either a predicate, or an *aggregation*. Aggregations are

a special feature of our implementation that allows end users to count the acceptable arguments for a given conclusion.

For example, the scheme *argument from expert opinion* can be attacked by showing that the person making a claim is not actually in expert in a domain. In our software prototype, we can establish "experthood" as follows:

$$\texttt{person}(X), \texttt{COUNT}(\texttt{publication}(X,Y) \leq 3) \rightsquigarrow \neg\texttt{expert}(X,Y)$$

The rule above expresses that X is not an expert in domain Y if they do not have at least four publications in Y. The set of available aggregation functions includes MIN, MAX, AVG, and any other aggregations that are supported by the underlying SQL server implementation.

Argument Creation Users can generate new predicates by writing rules of the form $p_1(X_1), \ldots, p_n(X_n) \rightsquigarrow p(X)$, with $i \geq 1$. This ensures that all user-defined predicates are ultimately derived from system-defined ones, and thus can be turned into SQL queries (see below).

Another important difference to a traditional logic-programming approach to argumentation is that arguments are created from rules without unifying free variables with atoms. There is, though, unification with relational types: The predicate $\texttt{engineer}(X)$ introduced earlier binds the variable X to the record type $(\mathsf{Int}, \mathsf{Text}, \mathsf{Text})$. When attempting to unify another free variable with X, the unification engine checks that the types are compatible.

5.4.4.3 Argument Evaluation

When arguments have been formed as described in the preceding section they are represented internally as abstract syntax trees (ASTs). The program has two evaluators for ASTs: As SQL queries and as argument graphs.

Argument Graph Evaluation The argument graph evaluation is straightforward: The set of arguments is completely determined by user input, and an argument a attacks another argument b if the conclusion of a is the negation ($\neg$) of b's conclusion. Please refer to Definition 81 on page 210 for a formalisation of this evaluation.

```
SELECT * FROM (SELECT * FROM publication AS Y)
JOIN (SELECT * FROM person AS X)
ON X.personId = Y.authorId
```

Figure 5.7: Generated SQL query for $\texttt{publication}(X,Y)$

```
SELECT (X.*, Y.*) FROM (SELECT * FROM publication AS Y)
JOIN (SELECT * FROM person AS X)
ON X.personId = Y.authorId
GROUP BY X.personId
HAVING COUNT(X.personId) <= 3
```

Figure 5.8: Generated SQL query for $\texttt{COUNT}(\texttt{publication}(X,Y) \leq 3)$

SQL Query Evaluation In order to instantiate an argument scheme such as (*Exp*) above, we first translate the preconditions of the rule into SQL. The predicate $\texttt{person}(X)$ translates into

```
SELECT * FROM person
```

When translating the predicates into queries, we proceed from left to right, using the free variables directly as identifiers in SQL. To evaluate $\texttt{COUNT}(\texttt{publication}(X,Y) \leq 3)$, we have to perform two steps. First, we translate the predicate $\texttt{publication}(X,Y)$ to the SQL query shown in Figure 5.7. The translation described in Figure 5.7 is predefined and can be generated automatically from the database schema, by exploiting the fact that the `publication` table has a foreign key `authorId`, which is linked to the primary ID column of the `author` table. The *JOIN* statement in l. 2 directly uses the previous query for $\texttt{person}(X)$.

The second step in the translation of $\texttt{COUNT}(\texttt{publication}(X,Y) \leq 3)$ adds a filter condition to the inner query from Figure 5.7. The result is shown in Figure 5.8.

From Arguments to Graphs The examples demonstrate the principle of translating arguments step-by-step to SQL. To extend the approach from individual arguments to argument graphs, two additional steps are necessary: First, there may be several arguments with the same conclusion (for example, representing different reasons why someone might be an expert). All arguments with the same conclusion are joined using the *UNION* operator in SQL. Second, attacks on arguments should result in the exclusion of records from the query. Attacks stem from negated conclusions ($\neg\texttt{expert}(X)$)

and are implemented using the *EXCEPT* keyword, which is the SQL equivalent of set difference. To build the query for $\texttt{expert}(X)$ arguments, the interpreter first creates a *UNION* query of all arguments whose conclusion is $\texttt{expert}(X)$. It then takes all arguments with conclusion $\neg\texttt{expert}(X)$ and excludes them from the first query. This is done recursively, from the bottom up.

5.4.4.4 Optimising the System

As stated above, we do not generate all possible arguments in order to evaluate the argument graph: Instead of instantiating each rule with all possible database results, we only check whether the set of possible results is empty or not. If it is, then we delete the rule, if it is not (i.e. if the result set is populated) then we take the rule as a proxy for *all* arguments it generates. This simple optimisation means that most of the computational work is offloaded to the SQL server and does not have to be performed on the client. In this section, we will show that this optimisation is correct in the sense that it does not affect the acceptable sets of arguments.

First, let us make precise the meaning of some of the terms we have been using. We will start by defining *database arguments*, which are arguments that consist of predicates on some universe (of rows) $\mathscr{U}$. Technically, $\mathscr{U}$ consists of several sets (one for each table), but we will ignore the fact that rows have a type for now. In the following we also assume a set N of names, or labels.

Definition 80 (Database Argument)**.** *Let $\mathbb{P} = \{P_1, \ldots, P_n\}$ be a set of predicates on $\mathscr{U}$, with $P_i : 2^{\mathscr{U}} \to 2^{\mathscr{U}}$.*

1. *If $P \in \mathbb{P}$ and $n \in N$, then $P \rightsquigarrow n$ is a* database argument
2. *If $A_1, \ldots, A_k$ are database arguments and $n \in N$ then $A_1, \ldots, A_k \rightsquigarrow n$ is a* database argument
3. *If $A_1, \ldots, A_k$ are database arguments and $n \in N$ then $A_1, \ldots, A_k \rightsquigarrow \neg n$ is a* database argument

The function $lbl(A)$ returns the label (in N) of a database argument.

Database arguments are essentially trees whose leaves are predicates in $\mathbb{P}$ and whose nodes are labelled with elements of N.

Example 62. *Assume that $\mathscr{U}$ contains all rows in Table 5.2 on page 201. Then we can define the following predicates:*

$$\begin{aligned}
\texttt{conventional}(X) &\Leftrightarrow \texttt{type}(X) = \text{“}\textit{Conventional}''\\
\texttt{orbital}(X) &\Leftrightarrow \texttt{type}(X) = \text{“}\textit{Orbital}''\\
\texttt{dry}(X) &\Leftrightarrow \texttt{environment}(X) = \text{“}\textit{Dry}''\\
\texttt{coolant}(X) &\Leftrightarrow \texttt{environment}(X) = \text{“}\textit{Coolant}''\\
\texttt{good}(X) &\Leftrightarrow \texttt{holeQuality}(X) > 3\\
\texttt{bad}(X) &\Leftrightarrow \texttt{holeQuality}(X) = 1
\end{aligned}$$

with $\mathbb{P} = \{\texttt{conventional}, \texttt{orbital}, \texttt{dry}, \texttt{coolant}, \texttt{bad}\}$. N contains claims we want to make based on the data, for example $N = \{\texttt{chooseOrbital}, \texttt{chooseConventional}, \ldots\}$. Remember that in the actual application, N is defined by the user (implicitly, through the rules that are entered into the system), and $\mathbb{P}$ is given by the underlying database.

Some database arguments are:

$$\begin{aligned}
A_1 &= [\texttt{conventional}(X), \texttt{good}(X) \rightsquigarrow \texttt{chooseConventional}(X)]\\
A_2 &= [\texttt{orbital}(X), \texttt{good}(X) \rightsquigarrow \texttt{chooseOrbital}(X)]\\
A_3 &= [\texttt{coolant}(X), \texttt{conventional}(X) \rightsquigarrow \neg\texttt{chooseConventional}(X)]\\
A_4 &= [\texttt{coolant}(X), \texttt{orbital}(X) \rightsquigarrow \neg\texttt{chooseOrbital}(X)]
\end{aligned}$$

The rationale behind arguments A_3 and A_4 is that we want to discount results which are based on an environment where coolant was used, because the machines in our workshop are not able to drill in such an environment.

Database arguments act as queries over the database $\mathscr{U}$. To interpret them in the meta-ASPIC framework, we define a translation process consisting of three steps. First, we give a straightforward translation of a set of database arguments A to an argument graph (A, Att) by defining the attacks-relation Att. This graph (A, Att) will serve as the underlying argument graph for the eventual meta-ASPIC system (G, K).

The purpose of the second and third steps is to filter out vacuous arguments, that is, arguments which are not backed by any data in the database. To achieve this we

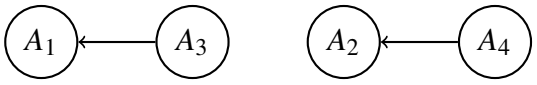
map each database argument to a set of rows in $\mathscr{U}$. We then add an attacker for each vacuous argument, thereby making it unacceptable in (G,K) under any semantics.

The first step in the translation, Definition 81, produces an argument graph (A',Att) from a set of database arguments A by defining the attacks relation Att.

Definition 81 (Argument Graph from Database Arguments)**.** *The* argument graph for database arguments A *over a set of symbols* N *is a graph* (A,Att_N) *where*

$$Att_N = \{(a,b) \in A \times A \mid lbl(a) = \neg lbl(b) \text{ or } lbl(b) = \neg lbl(a)\}$$

Example 63. *Applying Definition 81 to the example 62 we get the following graph:*



It seems that the grounded extension contains A_3 *and* A_4 *(the "negative" arguments), but we will see in Example 65 that some of the arguments are invalid, because they are not backed by any actual experimental data.*

For the second step we map each database argument to a set of rows in the database. This step is what enables the performance gains described above: Because we only need to know whether the set of rows for a database argument is empty or not, we do not need to add every possible instantiation of that argument to our graph - instead, we only check if the query has any results or not. That way, most of the work of evaluating arguments is performed by the SQL server.

Queries are generated for each argument with a positive (non-negated) conclusion, by first taking the intersection of the subsets of $\mathscr{U}$ defined by the predicates in the argument's body, and then subtracting all arguments whose conclusion is a negation of one of the argument's supporting predicates:

Definition 82 (Argument Query)**.** *Let* (A,Att_N) *database argument graph over a set of symbols* N *and a set of predicates* $\mathbb{P}$*. The function* $\mathsf{ins} : A \to \mathscr{P}(\mathscr{U})$ instantiates *arguments and is defined as*

$$\mathsf{ins}(a) = \begin{cases} P(\mathscr{U}) \setminus \mathsf{ins}^-(a) & \text{if } a = [P \rightsquigarrow n] \text{ for a } P \in \mathbb{P} \\ (\mathsf{ins}(A_1) \setminus \mathsf{ins}^-(A_1)) \cap \ldots \cap (\mathsf{ins}^(A_n) \setminus \mathsf{ins}^-(A_n)) & \text{if } a = [A_1,\ldots,A_k \rightsquigarrow n] \end{cases}$$

where ins^- *is a helper function that instantiates the attacking arguments, defined as* $\mathsf{ins}^-(a) = \{\mathsf{ins}(b) \mid b \in A \text{ and } (b,a) \in Att_N\}$.

The recursive definition of *ins* ensures that arguments are instantiated only by those rows for which either no counter-arguments exists (so ins^- is empty), or for which all counter-arguments are in turn countered. Because the only operations used are $\cup$ and $\cap$, the instantiation of an argument can be performed entirely by the database server. To this extent we translate $\cup$ to the *UNION* statement and $\cap$ to the *EXCEPT* statement in SQL.

Example 64. *Let us begin with argument*

$$A_1 = [\texttt{conventional}(X), \texttt{good}(X) \rightsquigarrow \texttt{chooseConventional}(X)]$$

from Example 62. Using the definitions from Example 62, we get the following instantiation (with rows from Table 5.2): $\mathsf{ins}(A_1) = \{r_{20}\} \setminus \mathsf{ins}^-(A_1)$. *Row 20 is the only one that fulfils the criteria* conventional *and* good. *In order to fully evaluate the argument graph we also need to compute* $\mathsf{ins}^-(A_1)$, *by evaluating* $\mathsf{ins}(A_3)$. *This may result in a successful attack on a, if* $\{r_{20}\} \setminus \mathsf{ins}(A_3) = \emptyset$. *However, as there are no rows for* A_3 *(*$\mathsf{ins}(A_3) = \emptyset$*), we get* $ins(A_1) = \{r_{20}\} \setminus \emptyset = \{r_{20}\}$. A_3 *is an example of a vacuous argument (see below).*

Definition 84 distinguishes two cases. For database arguments that correspond directly to a predicate in $\mathbb{P}$, we simply apply the predicate on $\mathscr{U}$. If it is a defeasible rule, we instantiate its body by recursively calculating ins for each supporting argument. However, we also subtract all rows that act as parts of counter-arguments. This means $\mathsf{ins}(a)$ effectively performs an evaluation of a.

It is possible to describe database arguments that do not map to any rows in the database, that is, arguments a with $\mathsf{ins}(a) = \emptyset$. We call such arguments *vacuous* because they are not backed by any data. Vacuous arguments should not be allowed to attack other arguments. We exclude them from our graph, resulting in a sub-graph $(A^+, Att_N^+) \sqsubseteq (A, Att_N)$ of valid arguments:

Definition 83 (Valid Database Argument Graph)**.** *For any argument graph from database arguments* $G = (A, Att_N)$, *the* valid sub-graph *of G is defined as* (A^+, Att_N^+)

with

1. $A^+ = \{a \in A \mid \mathsf{ins}(a) \neq \emptyset\}$
2. $Att_N^+ = \{(a,b) \in \mathscr{R}_{\mathscr{N}} \mid a,b \in A^+\}$.

Example 65. *Argument A_2 from Example 62 is defined as*

$$[\texttt{orbital}(X), \texttt{good}(X) \rightsquigarrow \texttt{chooseOrbital}(X)]$$

It is not valid (in the sense of Definition 83) because there are no rows in the database that satisfy both predicates, orbital *and* good*. For the same reason, A_3 is not valid. We thus get the subgraph (A^+, Att_N^+) of valid arguments, with $A^+ = \{A_1, A_4\}$ and $Att_N^+ = \emptyset$.*

With Definition 81 we get an argument graph and with Definition 84 we can map each argument to a set of rows in $\mathscr{U}$. To instantiate database arguments, we simply apply ins to each database argument, and extend the attacks relation *Att* accordingly.

Definition 84 (Instantiation of Database Argument Graph)**.** *Let A be a set of database arguments and let (A, Att_N) be its argument graph. The* instantiated graph *of A is an argument graph $(\tilde{A}, \tilde{Att}_N)$ with*

1. $\tilde{A} = \bigcup_{a \in A} \mathsf{ins}(a)$
2. $\tilde{Att}_N = \bigcup_{(a,b) \in Att_N} \{(a',b') \in \tilde{A} \times \tilde{A} \mid a' \in \mathsf{ins}(a) \text{ and } b' \in \mathsf{ins}(b)\}$

Example 66. *The argument graph from Example 63 is instantiated in $(\tilde{A}, \tilde{Att}_N)$, where $\tilde{A}$ contains the following rows:*

$$\mathsf{ins}(A_1) = \{r_{20}\} \qquad \mathsf{ins}(A_2) = \emptyset$$
$$\mathsf{ins}(A_3) = \{r_2\} \qquad \mathsf{ins}(A_4) = \{r_{30}, r_{46}\}$$

We finish this section by showing that our optimisation (computing $ins(a)$ for all arguments on the server, instead of computing the preferred extensions of the graph on the client) is valid: A database argument $a \in A$ is credulously acceptable in (G,K) if and only if all of its instantiations $ins(a)$ are acceptable in (G',K') (see Theorem 6 below). Before we can prove the claim, we need a handful of additional definitions.

First, we say that two arguments a and b in a graph G are extension equivalent if and only if every (preferred or grounded) extension of G either contains both a and b, or neither a nor b.

Definition 85 (Extension Equivalence)**.** *Let $G = (A, Att)$ be an argument graph and let $a, b \in A$. a is* extension equivalent *to b (short $a \equiv_E b$) if and only if for all $s \in \{\mathsf{gr}, \mathsf{pr}\}$ and for all $\mathscr{E} \in \Sigma_s(G)$, $a \in \mathscr{E} \Leftrightarrow b \in \mathscr{E}$.*

The notion of extension equivalence is an equivalence relation:

Proposition 45. *$\equiv_E$ is an equivalence relation*

Proof. Follows from the fact that $\Leftrightarrow$ is an equivalence relation on truth values □

In order to show that our optimisation is correct we need to show that the query function $ins(a)$ returns an equivalence class of the instantiated argument graph $(\tilde{A}, \tilde{Att}_N)$, and it is therefore enough to check the acceptability of a single member of $ins(a)$ to determine the acceptability of all instantiated arguments in $ins(a)$.

Theorem 6. *Let (A^+, Att^+) be a valid database argument graph and let $(\tilde{A^+}, \tilde{Att^+})$ be its instantiation. For all $a \in A^+$ and all $a', b' \in ins(a)$, $a' \equiv_E b'$.*

Proof. $G = (A^+, Att^+)$ be a valid database argument graph and let $G' = (\tilde{A^+}, \tilde{Att^+})$ be its instantiation. Let $a \in A^+$ and let $a', b' \in ins(a)$.

Proof by contradition. Assume that there is an $s \in \{\mathsf{pr}, \mathsf{gr}\}$ such that there exists an $\mathscr{E} \in \Sigma_s(G')$ such that $a' \in \mathscr{E}$ and $b' \notin \mathscr{E}$. Then there exists a $c' \in \tilde{A^+}$ such that $(c', b') \in \tilde{Att^+}$ and $(c', a') \notin \tilde{Att^+}$. By Definition 84 there exists a $c \in A$ such that $(c, a) \in Att_N$. However, by the same definition, $(c', a') \in \tilde{Att^+}$, so the assumption is false and the claim holds. □

In this result we made a number of simplifications compared to the software implementation. For example, our rules do not support aggregates, and record types have been omitted as mentioned above. However, the general principle behind Theorem 6 still applies.

To summarise: Our software can be used to define database argument graphs (Definition 81), which are argument schemes that can be instantiated by data in an SQL

database. The acceptability of database arguments is therefore determined by the acceptability of the concrete database rows they map to (Definition 83). However, it is not necessary to load all rows that instantiate database arguments into memory. As Theorem 6 shows, it is enough to check whether each database argument is populated, and then to evaluate the graph of valid database arguments (Definition 83). The size of the argument graph is therefore not bounded by the number of rows in the database, only by the number of database arguments.

5.4.4.5 Meta-Arguments to Counter Vacuous Arguments

An important step in proving Theorem 6 was to discount argument schemes that are not instantiated by any data as "vacuous" (see page 211). Saying that an argument should not count because it is not backed by any data is a meta-argument, and as such we can express it in meta-ASPIC. In this section we describe how such a meta-argument can be transplanted from Definition 83 to a meta-ASPIC argument graph. By formulating this constraint as a meta-argument we open up the possibility of adding counter-arguments or additional meta-arguments within the framework itself.

Definition 86 (Counter-Argument)**.** *Let $G = (A, Att_N)$ be an argument graph from database arguments. The graph $G' = (A', Att_N)$ is G* extended with counter-arguments *and is defined as*

1. $A' = A \cup A_C \cup A_D$
2. $A_C = \{[\mathsf{vacuous}(a)] \mid a \in A \text{ s.t. } ins(a) = \emptyset\}$
3. $A_D = \{[\mathsf{vacuous}(a) \rightsquigarrow \neg a] \mid [\mathsf{vacuous}(a)] \in A_C\}$

Example 67. *Consider the argument graph in our running example (Example 63). As we saw above, $ins(A_2) = \emptyset$, so A_C for the example contains the single argument $[\mathsf{vacuous}(A_2)]$. For A_D we get one attack: $A_D = \{[[\mathsf{vacuous}(A_2)] \rightsquigarrow \neg A_2]\}$.*

Instead of restricting the original set of arguments A (as was done in Definition 83 for valid database argument graphs), Definition 86 adds two new sets to the graph: A_C, with arguments $[\mathsf{vacuous}(a)]$ stating that a is not backed up by any data, and A_D – the actual argument scheme against vacuous arguments – introducing an attack on all arguments that had been singled out as vacuous in A_C.

Both approaches, restricting and adding counter-arguments, result in the same acceptability of database arguments. This result is formalised in the next proposition. However, the acceptable sets of arguments are not identical for G' and G^+, because of the counter-arguments that G' contains. We can therefore only show that the acceptability of arguments in the two graphs is identical up to linear extension, a notion introduced in the previous chapter (page 39).

Theorem 7. *Let $G = (A, Att)$ be an argument graph from database arguments, let $G' = (A', Att_N)$ be G extended with counter-arguments, and let $G^+ = (A^+, Att^+)$ be the valid graph database argument graph of G (Definition 83). Then, for all $s \in \{\mathsf{pr}, \mathsf{gr}\}$,*

$$\Sigma_s(G') \text{ is a linear extension of } \Sigma_s(G^+)$$

Proof. First note that $G^+ \sqsubseteq G'$, so all arguments in G^+ are also in G'. To show that $\Sigma_s(G')$ is a linear extension of $\Sigma_s(G^+)$, we need to prove that

1. For all $E \in \Sigma_s(G^+)$, there is an $E' \in \Sigma_s(G')$ such that $E \subseteq E'$ and
2. For all $E' \in \Sigma_s(G')$, there is an $E \in \Sigma_s(G^+)$ such that $E \subseteq E'$

(1) Let $E \in \Sigma_s(G^+)$. We distinguish two cases, one for $s = \mathsf{pr}$, and one for $s = \mathsf{gr}$. First assume $s = \mathsf{pr}$ so E is a preferred extension of G^+. To prove that there is an $E' \in \Sigma_{\mathsf{pr}}(G')$ with $E \subseteq E'$, it suffices to show that E is an admissible set in G', because then it is the subset of a preferred extension (by Dung's fundamental lemma). E is an admissible set if it is conflict-free and defends itself against all attacks. Since E is a preferred extension in G^+ it is conflict-free (in G^+). It is also conflict-free in G', because additional attacks in G' are introduced only through the set A_D and $A_D \cap E = \emptyset$. Now we prove that E is admissible in G' by contradiction. Assume that there exists an argument $a \in E$ and $b \in A'$ such that $(b,a) \in Att'$ and there is no $c \in E'$ such that $(c,b) \in Att'$. $b \in A'$, so (by Definition 86) $b \in A \cup A_C \cup A_D$. If $b \in A$, then either $b \in A^+$ or $b \notin A^+$. If $b \in A \setminus A^+$, then $ins(b) = \emptyset$ (by Definition 83) so there exists a $c \in A_D$ such that $(c,b) \in Att^+$ (by Definition 86). c itself is not attacked in G' so $c \in E'$. If on the other hand $b \in A^+$ then there exists a $c \in E$ such that $(c,e) \in Att^+$ (because E is admissible) and so $(c,e) \in Att'$. Therefore $b \notin A$, so $b \in A_C$ or $b \in A_D$. As the arguments in A_C do not produce any attacks, $b \in A_D$. So $b \in A_C$ with $b = [\mathsf{vacuous}(a)]$

(by Definition 86). However, since $a \in E$, $ins(a) \neq \emptyset$ so there is no argument $b' \in A_C$ with $b' = [\mathsf{vacuous}(a)]$. This contradicts the assumption that an argument b attacking a exists in G'.

Now assume that $s = \mathsf{gr}$, so E is the grounded extension of G^+. Let E be the grounded extension of G'. We show that E is a subset of E' by structural induction over the characteristic function $\mathscr{F}_{G^+}$. Specifically, we show that for all $D \subseteq E$, if $\mathscr{F}_{G^+}(D) \subseteq E'$ then $\mathscr{F}_{G^+}(\mathscr{F}_{G^+}(D)) \subseteq E'$ (induction step), and we show that $\mathscr{F}_{G^+}(\emptyset) \subseteq E'$ (base case). Since $\mathscr{F}_{G^+}$ is monotonically increasing, and the grounded extension is the least fixed point of $\mathscr{F}_{G^+}$, this suffices to show that $E \subseteq E'$ (because for every $a \in E$ there is a D_0 with $a \in \mathscr{F}_{G^+}(D_0)$). **Base case**: Let $a \in \mathscr{F}_{G^+}(\emptyset)$. To see that a is unattacked in G', assume that there is an argument b such that $(b,a) \in Att'$. Since $a \in \mathscr{F}_{G^+}(\emptyset)$, $b \notin Att^+$. So either $b \in A \setminus A^+$, or $b \in A_D$. In the first case, there exists a $c \in A'$ such that $(c,b) \in Att'$, and c is unattacked in G' so $c \in \Sigma_{\mathsf{gr}}(G') \setminus \{a\}$ so $a \in \Sigma_{\mathsf{gr}}(G')$. Since $a \in A^+$, $ins(a) \neq \emptyset$, so $b \notin A_D$. This contradicts the assumption that $b \in A \cup A_C \cup A_D$. Therefore a is not attacked in G', so $a \in E'$. **Induction step**: Let $D \subseteq E$ such that $\mathscr{F}_{\mathscr{G}^+}(D) \subseteq E'$. To show that $\mathscr{F}_{G^+}(\mathscr{F}_{G^+}(D)) \subseteq E'$, let $D' = \mathscr{F}_{G^+}(\mathscr{F}_{G^+}(D)) \setminus D'$. So D' contains the arguments that were added by the latest application of $\mathscr{F}_{G^+}$. Let $a \in D'$. Assume that there is an argument $b \in A'$ such that $(b,a) \in Att'$. If $b \in A^+$, then b is attacked by $\mathscr{F}_{G^+}(D)$, and thus (by induction hypothesis) b is attacked by $E' \setminus \{a\}$, so a is admissible with respect to $E' \setminus \{a\}$, and thus a is in E' (the grounded extension of G'). If on the other hand $b \in A' \setminus A^+$, then we can derive a contradiction similarly to the other cases of this proof above.

(2) Let $E' \in \Sigma_s(G')$. Again we distinguish $s = \mathsf{pr}$ and $s = \mathsf{gr}$. If $s = \mathsf{gr}$ then E' is the grounded extension of G'. In this case, E is the grounded extension of G^+, and $E \subseteq E'$ can be shown analogously to part (1) of this proof for the case that $s = \mathsf{gr}$.

If $s = \mathsf{pr}$, then E' is a preferred extension of G' and we will show that there exists a preferred extension E in G^+ such that $E \subseteq E'$. Let $E = E' \cap A'$. To prove that E is admissible, we need to show that E is conflict-free and that E defends itself against all attacks. E is conflict-free because $E \subseteq E'$ and E' is conflict-free (since it is a preferred extension of G'). Now assume that there is an argument $a \in E$ and an argument $b \in A^+$ such that $(b,a) \in Att^+$. Either $b \in A'$, or $b \notin A'$. If $b \in A'$ then there exists a $c \in E'$ such that $(c,b) \in Att'$ (since $E \in \Sigma_{\mathsf{pr}}(G')$). By Definition 86, $c \in A \cup A_C \cup A_D$. If $c \in A$

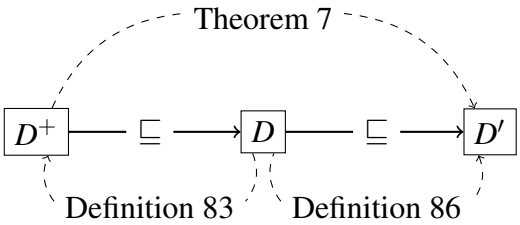


Figure 5.9: Relationship of D^+, D and D' in Theorem 7

then $c \in A^+$, so $c \in E'$. If $c \in A_D$, then $ins(b) = \emptyset$, so $b \notin A^+$ (which contradicts the assumption that $b \in A^+$). Therefore E is an admissible set in G^+. To finish the proof we need to show that E is a preferred extension – a maximal admissible set. Assume that there is an argument $a \in A^+$ such that a is admissible with respect to E and $a \notin E$. Since $A^+ \subseteq A'$, $a \in A'$. $a \in A^+$ so by Definition 83, $ins(a) \neq \emptyset$. Therefore, a is not attacked by any arguments in A_D, so a is admissible with respect to E' too. Since E' is a preferred extension it is a maximal admissible set so $a \in E'$. But then $a \in E' \cap A'$, so $a \in E$ and we have shown that E is a preferred extension in $\mathscr{G}^+$. □

This completes our theoretical evaluation of the case study, closing the circle with the original discussion of meta arguments in Section 5.2. Theorem 7 demonstrates how meta-arguments can be taken from the meta-language of mathematics and informal descriptions (as in Definition 83) and incorporated into meta-arguments in a meta-ASPIC system (as in Definition 86). The relationship between the graphs D^+, D and D' is shown in Figure 5.9. D represents the original graph of database arguments, including those that are potentially invalid (vacuous). D^+ on the left contains only valid arguments and is therefore a subgraph of D. D' on the right contains additional meta-arguments to attack (and make unacceptable) any invalid arguments in D. D' is therefore a supergraph of D. The dashed arrows in the lower half of the diagram show how the two graphs D' and D^+ can be generated from D.

We conclude with a discussion of our approach and a review of the relevant literature.

5.4.5 Summary

We implemented a software prototype based on the meta-ASPIC system introduced in Section 5.3. The prototype has been used to model arguments about drilling methods, combining empirical evidence with arguments from argument schemes. An early

version of the software has been presented at COMMA 2014.

The prototype as described here used arguments to describe database queries. SQL databases do not have a notion of conflict between data rows. Therefore, when viewed as a meta-ASPIC system, the underlying argument graph was a conflict-free graph, and attacks were only introduced on the meta-level. This approach can easily be extended to other data sources. The resulting meta-argumentation system can be used to summarise debates across a variety of media, including text, verbal discussions, experimental data, and requirements specifications.

5.5 Related Work

The work in this chapter brings together two areas of research in formal methods of argumentation: Argument schemes, and meta-argumentation. We have argued that the two are closely related, and that argument schemes are a form of meta-argumentation. However, they have traditionally been treated as separate entities. We will therefore review the state of the art in each area individually.

We can also view meta-ASPIC as "yet another argumentation system", regardless of our motivation to use it for studying meta-arguments. From this perspective it makes sense to compare meta-ASPIC with similar argumentation systems.

5.5.1 Argument Schemes

Previous research has been concerned with the representation of argument schemes in a formal setting [139, 21, 140, 127, 141, 132, 142], in particular for the legal domain [143, 144, 26, 145, 131] (see [27] for a review). Arguments schemes are seen as "generators" of arguments [146], a view that we also take (especially in our case study (Section 5.4) where argument schemes literally generate arguments in the form of SQL queries). However, there is a significant conceptual difference between our work and the literature in that we studied formal representations of argument schemes on the basis of non-deductive arguments (Section 5.2). As a consequence, our system can represent argument schemes (as arguments about arguments), but it cannot *infer* arguments from a set of argument schemes and some formalised knowledge, which has been the primary use case for formalising argument schemes. Instead, the purpose of our system is to extend classical notions of acceptability to the meta-argumentation, and the fact

that meta-ASPIC can represent argument schemes at all is only a consequence of our view that argument schemes are examples of meta-argumentation.

Our work takes quite a unique position within the literature on formalising argument schemes. Our primary intention has been to develop a framework in which meta-arguments can be evaluated using traditional argument graph semantics, and argument schemes are only one example of meta-argumentation.

5.5.2 Meta-Argumentation

Meta-level argumentation is the study of arguments about arguments [147, 148, 149], describing the properties of arguments and attacks in the same language as arguments themselves.

Research by Modgil *et al.* [147] has shown how several extensions to abstract argumentation can be modeled using meta-level constructs in a standard (that is, without any special arguments) abstract argumentation system as defined by Dung [42]. This is achieved by translating each of these additions, such as attacks on attacks, or preferences, into a constellation of several arguments that are only connected by the "attacks" relation. The extensions of the extended abstract argumentation systems are shown to conincide with those of the resulting argument graph. However, this approach to meta-level argumentation does not provide a systematic way of instantiating abstract arguments. The examples in [147] suggest that there is a need for a systematic approach which uses structured arguments to unify the various proposals for abstract argumentation.

In an earlier paper by Modgil [150], a method is introduced for organising several abstract argument graph into a hierarchy. The proposal distinguishes defeats from attacks, both of which are binary relations between arguments. Defeats are a subset of attacks. Whether an attack is also a defeat depends on information in the next-higher level in the hierarchy of argument graphs. Defeats give rise to the notion of resolutions, which are sub-graphs of an argument graph in which all mutual attacks are resolved (so only one of the two edges exists in the subgraph). The main difference between our approach and the work by Modgil is that in [150], attacks are presumed to be caused by contradictions in a formulae of a logical language, and thus symmetric in most cases. The reason for organising argument graphs in a hierarchy is that it allows one to resolve

symmetric attacks by establishing preferences.

As we have mentioned above, one of the distinguishing features of meta-ASPIC is that it allows arguments to appear as premises and conclusions of other arguments, in order to better represent natural language arguments. A recent proposal by Amgoud *et al.* [125] has a similar feature and we will therefore discuss it here as a system for meta-argumentation. We explained in Section 5.2 that the validation of natural language arguments requires context (unlike that of deductive arguments), and that this context is provided by meta-arguments. Amgoud *et al.* introduce a formalism (as a set of inference rules) for reasoning about meta-arguments. Their aim is to answer questions such as "is a a reason for concluding b?", where a and b can be arguments themselves. Their approach is promising, but as the authors say, the question of soundness and completeness of their logic is still open. In our approach we do not have to worry about this problem because we project meta-arguments into the well-known domain of abstract argumentation [42].

The idea that there is a hierarchy of meta-arguments, instead of only one meta-level and one object-level, is not new and has been explored for example by Wooldridge [149]. The link between meta-argumentation and argument schemes has been drawn in [127].

Abstract argumentation [42] provides a graph-based interpretation of argument graphs. Bipolar argumentation [151, 152, 153] is an extension of Dung's abstract argumentation framework, adding a "supports"-relation as a second relation over arguments. Dung's original framework considered this relationship only implicitly, using the concept of defence for the defeaters of an argument's defeaters. Supporting arguments allow additional extension semantics. For example, sets of arguments are considered safe if none of their members depend on (are supported by) an argument outside the extension, which results in a stronger notion of internal coherence than just being conflict-free. Whilst bipolar argumentation is appealing as it offers a range of possibilities for defining the "supports"-relation, there is no formalisation of meta-level arguments, and supports for attacks (i.e. each attack by an argument A on argument B is justified by an argument C) cannot be defined.

5.5.3 Bipolar Argumentation

The central idea in bipolar argumentation [154, 153, 155, 151] is that, besides attacking or conflicting (as in abstract argument graphs), there is a second "supports" relationship that abstract arguments may be in.

It is important to distinguish the later work on bipolarity *in* argumentation (e.g. [154, 153]) from the earlier work on bipolar argumentation frameworks, short BAFs (e.g. [153, 155, 151]). The latter establishes BAFs as the formal foundations for the former. Bipolarity in argumentation characterises different kinds of support, such as deductive support, necessary support and evidential support, in terms of a BAF.

In BAFs, there is no notion of meta-argumentation, because all arguments are on the same level and the "supports" relationship between arguments is not restricted. For example, it is possible to have cyclic supports (although in an earlier version of BAF [151], only acyclic graphs were considered). In BAF it is thus not possible to separate the different domains in which arguments can exist, for example engineering (the object domain) and argument schemes (the meta-domain). This however is an important requirement for our model engineering debates, as discussed in Section 5.2.1. On the other hand, in meta-ASPIC one cannot model self-supporting arguments without resorting to a second argument on a higher level (ie. one has to write $[[a] \rightsquigarrow [a]]$). The reason for these differences is that BAF utilises a binary relation to express support, whereas in meta-ASPIC we chose an algebraic approach (compare our Definition 68 on page 178 with the BAF definition as $(G, Att_{att}, Att_{sup})$ where Att_{sup} is the supports-relation). Our approach prevents, for example, self-attacking arguments (so in this sense it is less expressive than BAF), but on the other hand it enables interesting constructions in the realm of meta-argumentation such as the hierarchy of argument graphs in Definition 77 on page 191, and the characteristic object function $\mathscr{O}$ on page 189.

As far as the comparison between bipolarity in argumentation and meta-ASPIC is concerned, of the three notions of support discussed in [154], meta-ASPIC is most closely related to *evidential support*. This term stems from evidence-based argumentation, which we will discuss in detail in the following section.

5.5.4 Evidence-Based Argumentation

Evidence-based argumentation frameworks (EAFs, [156, 157, 158]) extend Dung's argument graphs in two ways. First, they complement the existing "attacks" relation Att_a with an "evidential supports" relation Att_e. Second, Att_a and Att_e are not binary relations over the set of arguments A, but rather they are subsets of $\mathscr{P}(A) \times A$ - thus allowing supports and attacks by *sets* of arguments. It is easy to see the analogy with meta-ASPIC, where we allow several arguments in the premises of a meta-argument. In this section we first give an interpretation of meta-ASPIC in terms of EAF, and then discuss the relative advantages of each approach.

5.5.4.1 Translation

Let us begin with the formal definition of EAF.

Definition 87 (Evidential Argumentation System [156])**.** *An evidential argumentation system is a tuple* (A, Att_a, Att_e) *where A is a set of arguments,* Att_a *is a relation of the form* $(\mathscr{P}(A) \setminus \emptyset) \times A$*, and* Att_e *is a relation of type* $\mathscr{P}(A) \times A$*, such that within the argumentation system,* $\nexists x \in \mathscr{P}(A), y \in A$ *such that* $xyAtt_a y$ *and* $xAtt_e y$*. We assume the existence of a "special" argument* $\eta \notin A$*.*

The Att_e and Att_a relations encode evidential support and attacks between arguments. The special argument η represents the environment, and is used as a support for "self-evident" arguments, which are not supported by any other arguments. η can be seen as an interface to the world outside the particular EAF under consideration. EAFs give rise to the notion of evidential support:

Definition 88 (Evidential Support [156])**.** *An argument a is* e-supported *by a set S iff*

1. $SAtt_e a$ *where* $S = \eta$*, or*
2. $\exists T \subset S$ *such that* $TAtt_e a$ *and* $\forall x \in T$*, x is e-supported by* $S \setminus \{x\}$*.*

S is a minimum support for a if there is no $T \subset S$ *such that a is e-supported by T.*

By Definition 88, for every supported argument there exists a chain of evidence starting with the special argument η. We can then define evidence-supported attacks as follows:

Definition 89 (Evidence-Supported Attack [156])**.** *A set S carries out an* evidence-supported attack *on an argument a if*

1. *$XAtt_a a$ where $X \subseteq S$, and*

2. *All elements $x \in X$ are e-supported by S.*

An evidence-supported attack by a set S is minimal iff there is no $T \subset S$ such that T carries out an evidence-supported attack on a.

And finally, the traditional notion of acceptability extends to EAFs also:

Definition 90 (E-Acceptability [156])**.** *An argument a is e-acceptable with respect to a set S iff*

1. *S e-supports a and*

2. *Given a minimal evidence-supported attack $X \subseteq A$ against a, $\exists T \subseteq S$ such that $TAtt_a x$, where $x \in X$ such that $X \setminus \{x\}$ is no longer a supported attack on a.*

The second condition of Definition 90 ensures that any argument which is involved in an attack on S it itself attacked by S. A set of arguments S is **conflict-free** iff $\forall y \in S$, $\nexists X \subseteq S$ such that $XAtt_a y$ (i.e. if it does not attack itself). The final definition we need is that of e-admissibility:

Definition 91 (e-Admissible Set of Arguments [156])**.** *A set of arguments S is said to be admissible iff*

1. *All elements of S are e-acceptable with respect to S*

2. *S is conflict-free*

We will now define a mapping T from meta-ASPIC systems to EAFs. It has the property that it preserves admissibility of sets of arguments, in the sense that if a set of meta-ASPIC arguments $S \subseteq M_A$ for some meta-ASPIC system M is admissible in $\mathsf{reify}(M)$, then the closure of M under sup (cf. page 179) is e-admissible in $T(M)$. Further, T is non-trivial, because it does not simply define Att_e to $(\{\eta\}, a)$ for every argument a - instead it uses the sup function to define evidential support.

Definition 92 (meta-ASPIC to EAF)**.** *Let $M = (G, M_A)$ be a meta-*ASPIC *system. The* corresponding EAF *is a tuple $T(M) = (A', Att_a, Att_e)$ with*

1. $A' = M_A$
2. $Att_a = \{(\{a\}, b) \mid a, b \in M_A$ *such that a m-attacks* $b\}$
3. $Att_e = \{(s_a, a) \mid a \in M_A$ *and* $s_a = \mathsf{sup}(a)$ *iff* $\mathsf{sup}(a) \neq \emptyset$, *and* $s_a = \eta$ *otherwise*$\}$

Proposition 46. *Let $M = (G, M_A)$ be a meta-*ASPIC *system. Let $S \subseteq M_A$ such that for all $a \in S$, $\mathsf{sup}(a) \subseteq S$. Let $G' = (M_A, Att) = \mathsf{reify}(M)$ and let $S \subseteq M_A$. S is admissible in G' if and only if S is e-admissible in $T(M)$.*

Proof. Let S, $T(M) = (A', Att_a, Att_e)$ and $G' = (M_A, Att) = \mathsf{reify}(M)$ as defined in the claim.

(⇒) Assume S is admissible in G'. We need to show that S is e-admissible in $T(M)$. By Definition 91, this involves two conditions. **(1)** All elements of S are e-acceptable with respect to S. Let $a \in S$ such that a is not e-acceptable with respect to S. Then by Definition 90, either (1.i) S does not e-support a, or (1.ii) there exists a minimal evidence-supported attack $X \subseteq M_A$ such that there is no $T \subseteq S$ such that $TAtt_a x$ for an $x \in X$ such that $X \setminus \{x\}$ is no longer an e-supported attack on a. Assume that **(1.i)** is the case. Considering Definition 88 Conditions 1 and 2, and Definition 92 Cond. 3, this means that S is not closed under sup - a contradiction with the assumptions on S laid out above. For **(1.ii)**, let $X \subseteq M_A$ be a minmal evidence-supported attack such that there is no $T \subseteq S$ such that $TAtt_a x$ for an $x \in X$ such that $X \setminus \{x\}$ is no longer an e-supported attack on a. By Definition 92 Cond. 2, $X = \{b\}$ for some $b \in M_A$, and b is not attacked by any argument in S. This contradicts the assumption that S is an admissible set in G'.

(2) S is e-conflict-free. Suppose that S is not e-conflict-free, that is, there exists a $y \in S$ such that there is a $X \subseteq S$ such that $XAtt_a y$. By Definition 92 Cond. 2, $Y = \{a\}$ for some $a \in S$, and a m-attacks y. However, if a m-attacks y then by Definition 72, a attacks y in G', so S is not conflict-free in G', which contradicts the assumption that S is admissible.

(⇐) Assume S e-admissible in $T(M)$. We need to show that S is admissible in G'. Assume S not admissible in G'. Then either (1) there exist $x, y \in S$ such that $(x, y) \in Att$, or (2) there is an $a \in S$ such that a is not acceptable with respect to S. **(1)** If S is not

conflict-free in G' then there exist $x, y \in S$ such that $(x, y) \in Att$, so by Definition 92 Cond. 2, $(\{x\}, y) \in Att_a$, so S is not e-conflict-free and thus not e-admissible in $T(M)$, contradicting the assumption. **(2)** If there exists an $a \in S$ such that a is not acceptable with respect to S, then there is a $b \in M_A$ such that $(b, a) \in Att$ but S does not attack b. In this case, by Definition 92 Cond. 2., $\{b\}$ constitutes a minimal evidence-supported attack on a, and there is no $T \subseteq S$ such that $TAtt_a a$, so by Definition 90 Cond. 2, a is not e-acceptable with respect to S, contradicting the assumption that S is e-admissible. □

5.5.4.2 Comparison with meta-ASPIC

If every meta-ASPIC system can be translated to a non-trivial EAF, then one may legitimately question the reasons for choosing meta-ASPIC over EAF. There are two reasons for favouring meta-ASPIC in the scenario we have defined it for – to describe engineering debates.

First (and this point applies to bipolar argumentation too), in our approach we did not re-define the established acceptability semantics for meta-ASPIC, in contrast with EAF, where for example the notion of admissibility is redefined in Definition 91. Instead, the meaning (acceptability of arguments) of a meta-ASPIC system is determined by its translation to a regular abstract argument graph, as evidenced in the function reify. This means we can immediately apply all prior work on abstract argumentation to meta-ASPIC, without having to translate concepts such as different acceptability semantics into our framework first. A very practical consequence of this decision is that we can re-use existing implementations of Dung's argument graphs, for example the one by van Gijzel and Nilsson [137], in implementations of meta-ASPIC.

Second, the primary motivation for meta-ASPIC was the ability to represent meta-argumentation and specifically argument schemes. As a result, we can for example stratify a meta-ASPIC system into different levels of abstraction (see Definition 76 on page 189), and separate argument schemes from domain-specific arguments. This distinction does not exist in EAF, and all arguments reside on the same (object) level. In [142], EAFs were used to reason about argument schemes for normative practical reasoning, however without considering meta-argumentation.

We advocate the position that the choice of argumentation system should be governed by its application, and there are also reasons for choosing EAF over meta-ASPIC.

To give an example, for EAF there exists a further, semantics-preserving translation to AIF, the argument interchange format [157], which does not exist for meta-ASPIC.

5.6 Discussion

In this chapter we presented meta-ASPIC, a modified version of the ASPIC+ argumentation framework with express consideration of meta-arguments. The main semantic difference to the original framework is in the structure of arguments: Arguments in ASPIC+ have sentences of the underlying logic $\mathscr{L}$ as their conclusions, whereas in our system the conclusions of arguments are other arguments. This distinction is essential, because it makes it possible to create arguments about arguments (as opposed to arguments about sentences of $\mathscr{L}$).

As stated in Section 5.2, the purpose of meta-arguments is to justify (or deny) that something is an argument. In our view, meta-argumentation is most interesting in the case of non-deductive arguments, because deductive arguments (of the form $a \vdash b$) are supported on the meta-level by their proofs, which – once they have been formalised – are not subject to debate. This prompted us to replace to replace the "underlying logic" $\mathscr{L}$ in ASPIC with an argument graph in meta-ASPIC, which is the second difference between our system and the original.

In the beginning of this chapter we characterised the difference between deductive and non-deductive arguments as context-dependence: Deductive arguments can be validated without external context as they contain all the information necessary to obtain the conclusion from the premises. Non-deductive arguments (including enthymemes) can only be validated with additional context – for example, we can only say that "`Al_oxide` $\rightsquigarrow$ `non_corrosive`" is an argument because we know that it was put forward by an expert in metal oxidation. In meta-ASPIC, this context is provided by meta-arguments. We gave two concrete examples of meta-argumentation. First, we listed a number of common argument schemes in Section 5.2.1. Argument schemes are common patterns of human reasoning often used in spoken dialogue or written text. From a survey of common argument schemes we concluded that argument schemes are examples of meta-argumentation, because they are reasons that a statement is argument. In addition, some argument schemes talk about other arguments (as opposed to domain-level statements), which is also a form of meta-argumentation. The second

example of meta-argumentation is the interpretation of experimental data. Arguments are formed from this data by relating it to the context of the decision that the experiment was intended to support. As with argument schemes, this context is provided by argument schemes.

The second part of this chapter was dedicated to a case study on interpreting experimental data. We described a software prototype that allows domain experts to use the results of drilling experiments in arguments. In other words, users provided the context necessary to draw conclusions from raw data. The prototype was implemented as a browser application with a SQL server backend and it was backed by data provided by David Payne (of Queen's University Belfast). An interesting aspect of the implementation was covered in Section 5.4.4.4: By transforming arguments into SQL queries we moved a large part of the computational effort required to evaluate argument schemes from the client to the SQL server. This way, argument graphs can be evaluated asynchronously and with full support of the SQL server's query engine.

The case study showed the potential of meta-ASPIC for creating and evaluating arguments across a range of data sources that do not necessarily have an underpinning in formal logic. We believe that argumentation systems such as meta-ASPIC will be crucial for analysing human-generated arguments. In the broader context of this thesis, meta-ASPIC serves as the fundamental representation of arguments on which the developments of the previous chapters can be based. Due to its ability to represent natural language arguments, meta-ASPIC is well suited as a target language for arguments mined from large corpora of engineering documents, thus bringing us one step closer to the original aim of project DEEPFLOW.

The initial application of our framework in the engineering domain was evaluated positively by our colleagues at Queen's University. The main advantages of our approach were perceived to be (a) the ability to quickly summarise experimental data using database arguments and (b) the visualisation of arguments. Future versions of the system should provide a facility for annotating experimental results with user-defined argument data that can then be queried in the same way as the experimental data itself.

Chapter 6

Discussion

6.1 Future Work

Throughout the thesis we tried to show a path towards practical applications of our theory, most extensively in Chapter four where we illustrated how our decision model gives rise to a novel visualisation of decision documentation in a design process (page 154). A number of obstacles remain before our model can be used in a fully automated way, as envisioned in the introduction.

The biggest piece of work is argument extraction. Our model of decision outcomes already reflects the structure of design documents, but the step from textual design documentation to an ASPIC+ knowledge base is still missing. Most of this work is a natural language processing problem: Sentences have to be analysed and their argument structure extracted. Then the conflict between arguments needs to be established. Much decision documentation in engineering design takes the form of semi-structured (as opposed to unstructured) documents. Semi-structured data includes, for example, tables where each column determines the type of data in its cells, and requirements specifications, where sentences follow a pattern such as "The product shall have $\langle$ property $\rangle$". We assume that exploiting this structure will ease the task of extracting arguments.

For the argumentation model itself we see the following possible improvements. First, we would like to investigate whether additional data about arguments – such as strength, plausibility, etc – yields benefits that are large enough to justify the additional work required to formalise this data. Throughout the thesis it has been our goal to minimise the information required by our system (in addition to the actual arguments). However, for future work we would like to relax this requirement a little, especially if it

can be combined with the information from semi-structured data as mentioned above.

Second, as we mentioned earlier we see decision outcomes as representations of design documents. It would be interesting to consider if a set of design documents can be extracted from a formalised decision process, representing the most up-to-date version of the documentation. For this task one could also consider the audience (project managers, engineers, customers, etc.), building on existing work by Dunne *et al.* [159].

For the meta-ASPIC system for meta-argumentation and argument schemes from Chapter 5 we gave a formal comparison with evidence-based argumentation frameworks. This line of work should be continued, by comparing meta-ASPIC with other existing approaches to argumentation. It should, for example, be possible to encode meta-ASPIC graphs in propositional logic *directly*, similar to the proposal by Besnard *et al.* for abstract argument graphs [160], without first computing the reified graph and then applying their encoding.

The source code of the implementation of the case study described in Section 5.4.3 (page 200) cannot be made available to the public because it is owned by SAP, but it would benefit the practical adoption of our system if it was released under an open-source license. We would therefore like to write a new implementation of the code in Haskell and release it.

6.2 Discussion

At the beginning of this thesis (page 14) we listed a number of requirements for an argument-based model of engineering design processes. These requirements were

RQ1 Represent design decisions with the pros and cons for each of their options, including the reasoning that was applied to arrive at the pros and cons and possible worlds in which the underlying assumptions hold.

RQ2 Reason about decisions so represented, specifically by characterising the decision rules used to arrive at the decision, and determining the effect that "choosing an option" has on the knowledge base.

RQ3 Formulate sequences of decision problems in which decisions made at one stage influence the range of options and constraints for decisions made later in the process, and assess the impact of changing a previous decision.

RQ4 Combine various forms of reasoning such as deductive arguments, empirical evidence, intuition and heuristics.

How does our system meet those requirements? To address RQ1 we developed the Argumentation Decision Framework (ADF) in Chapter 3. The pros and cons of each option are expressed as arguments, and the reasoning behind the pros and cons is represented in the structure of those arguments. ADF combines both multi-criteria decision making (MCDM) and decision making under uncertainty (DMU). For MCDM a list of requirements can be specified, and the requirements met by each option are represented by acceptable arguments for that option. The possible worlds of DMU correspond to preferred extensions in an option's argument graph, and the utility of each option is again a set of acceptable arguments. We gave correspondence results for traditional models of DMU and MCDM.

Moving beyond mere representations of DMU and MCDM, we addressed RQ2 with an exploration of decision rules and a formal definition of "accepting a decision" in our model. We translated several DMU decision rules into ADF, and classified them according to their decisiveness (i.e. their ability to assign different rankings to different options). We identified a degree of decisiveness unattainable by traditional means and proposed an argumentation-specific decision rule that meets this highest degree of decisiveness. We further compared decision rules by their optimism (i.e. by their propensity to assume more positive outcomes). Such classifications are useful for characterising decisions after they have been made. By using decision rules to compare a number of past decisions made by different engineers, one can spot patterns which may lead to improvements in the decision making process.

We then investigated the effect of accepting a decision on the knowledge base. We defined an "enforce" operation that promotes a conflict-free set of arguments from credulous to sceptical acceptability. In terms of ADF, we can apply "enforce" to the set of arguments pro the chosen decision, to reflect the fact that they are now more firmly believed than before the decision was made (when they were only one of a number of credulously acceptable sets of arguments). We also showed that enforcement in ASPIC+ relies on the ability to deactivate defeasible rules.

Chapter four was dedicated to the third requirement, RQ3. We started with a definition of the outcome of a decision (its set of arguments pro) and looked at the possible

effects of chaining several outcomes in a sequence. We obseved that, given a sequence $S = (Res_1, Res_2)$ of decision outcomes, the union $Res_1 \sqcup Res_2$ may contain conflicting arguments even if Res_1 and Res_2 on their own are conflict-free. At the same time, Res_2 may contain more, less or different knowledge than Res_1 - there is no relationship between the knowledge bases of two separate decision outcomes. This prompted us to look at a second kind of sequence which we called "decision process". The decision process of a sequence S (obtained through Emb_S) is a sequence of knowledge bases which are not necessarily conflict-free, but monotonically increasing. Each stage contains the knowledge of all previous stages. By transforming S into a decision process we can draw out some of the implicit assumptions in S and make them explicit. This is one example of earlier decisions affecting later one. In a second step we measured the impact of changing a past decision, by choosing a different option for it.

The final requirement, RQ4, has been addressed in Chapter five. Based on a study of argument schemes commonly used in engineering design we proposed meta-ASPIC, an argumentation system loosely based on ASPIC+ that aims to capture meta-argumentation. Arguments in meta-ASPIC can exist on different levels on a hierarchy of abstraction. For example, arguments about a specific engineering problem are on the lowest level, and arguments about those arguments (such as *ad hominem* arguments against their proponents) are located on the level above. We discussed a case study in which drilling techniques are compared based on a number of experiments, and described an implementation of this case study.

From the above discussion it should be clear that our framework addresses the requirements adequately. Some of the techniques we developed are independent from the others. For example you can apply ADF to a single decision without worrying about decision sequences, or meta-argumentation. We believe that modularity is a key factor in the adoption of a framework like ours, because it means that applications do not have to implement the entire system to make use of one of its features.

Appendix A

Functions & Symbols

Table A.1 on page 233 contains a list of important functions and symbols used in this thesis.

Name or Symbol	Description	Defined where
$\bar{\cdot}$	Contrariness function	Definition 7 on page 29
$\langle\cdot\rangle$	Rule name function	P. 29
$A \Delta B$	Symmetric difference between two sets A and B	Definition 19 on page 37
$\Sigma_e(G)$	Extensions E of an argument graph G under semantics $e \in \{\mathsf{pr}, \mathsf{gr}\}$	P. 27
$\mathsf{args}_D(O)$	Arguments of a decision frame D for an option $O \in \mathscr{O}$	P. 55
$\mathsf{args}(KB)$	Arguments for a knowledge base KB	P. 30
$\mathsf{argGraph}(KB)$	Argument graph for a knowledge base KB	P. 33
$\mathsf{attacks}(A)$	Attacks of a set of arguments	P. 34
$\mathsf{attackers}(a, A)$	Attackers of an argument a in a set of arguments A	Definition 13 on page 34
$\mathsf{avg\_strength}(A, G)$	Average strength of a set of arguments A in a graph G	Definition 60 on page 134
$\mathsf{conc}(a)$	Conclusion of an argument a	Definition 9 on page 30
$D = (K, C, R)$	Decision frame with knowledge base K, consequences C and requirements R	P. 54
$\mathsf{deactivate}(r, KB)$	Deactivate a defeasible rule r in a knowledge base KB	Definition 47 on page 86
emb_S	Embedding of a decision sequence S	Definition 55 on page 123
$\mathsf{enforce}(B, KB)$	Enforce a set of arguments $B \subseteq \mathsf{args}(KB)$	Definition 49 on page 94
$\mathsf{extract}(KB)$	Extract assumptions and rules used to form the grounded extension of a knowledge base KB	Definition 56 on page 125
$G = (A, Att)$	Argument graph G with arguments A and attacks $Att \subseteq A \times A$	P. 24
KB	Knowledge base containing defasible rules	P. 33
$L(\Delta)$	Language of a decision rule Δ	Definition 38 on page 72
$\mathsf{meta}(G)$	meta-ASPIC system for an argument graph G	P. 195
$\mathsf{name}(r)$	Name of defeasible rule r (usually written as $\langle r \rangle$)	P. 29
$\mathscr{O}$	Set of all options	P. 54
$\mathscr{P}(A)$	Powerset of a set A	Definition 15 on P. 36
$\mathsf{possibleWorlds}(D, O)$	Possible worlds for an option O in a decision frame D	Definition 32 on page 64
$\mathsf{prem}(a)$	Premises of an argument a	Definition 9 on page 30
$\mathsf{reify}(\Omega)$	Reified graph of a meta-ASPIC system Ω	Definition 72 on page 182
$s_G(x)$	Strength of an argument x in a graph G	Definition 59 on page 133
$\mathsf{rules}(a)$	Rules of an argument a	Definition 9 on page 30
$S = (Res_1, \ldots, Res_n)$	Sequence of decision outcomes Res_i	P. 116
$\mathsf{sat}_D(O)$	Satisfaction function of a decision frame D	Definition 29
$\mathsf{sub}(a)$	Sub-arguments of an argument a	Definition 9 on page 30
$\mathsf{topRule}(a)$	Top rule of an argument a	Definition 9 on page 30

Table A.1: Symbols and Functions

Bibliography

[1] Gerhard Pahl and Wolfgang Beitz. *Engineering Design A Systematic Approach*. Springer, second edition edition, 1996.

[2] Richard Birmingham, Graham Cleland, Robert Driver, and David Maffin. *Understanding Engineering Design: Context, Theory, and Practice*. Prentice Hall, 1997.

[3] Jeremy S. Busby. Effective practices in design transfer. *Research in Engineering Design*, 10:178–188, 1998.

[4] Stuart Pugh. *Total Design. Integrated Methods for Successful Product Engineering*. Addison-Wesley, 1993.

[5] Atila Ertas and Jesse C. Jones. *The Engineering Design Process*. John Wiley and Sons, Inc., 2nd edition, 1996.

[6] Reza Beheshti. Design decisions and uncertainty. *Design Studies*, 14:85–95, 1993.

[7] Frank-Lothar Krause, Kai Mertins, Andreas Edler, Peter Heisig, Ingo Hoffmann, and Markus Helmke. *Computer Integrated Technologies and Knowledge Management*, volume Handbook of Industrial Engineering: Technology and Operations Management, chapter 6, pages 177–226. Wiley-Interscience, New York, 3rd edition, 2001.

[8] Jintae Lee. Design rationale systems: Understanding the issues. *IEEE Expert*, 12(3):78–85, 1997.

[9] Janet Burge and David C. Brown. Reasoning with design rationale. In J. Gero, editor, *Artificial Intelligence in Design*, pages 611–629, Netherlands, 2000. Kluwer Academic Publications.

[10] Leonard J. Savage. *The Foundations of Statistics*. Wiley, 1954.

[11] Simon French. *Decision Theory: An introduction to the matehmatics of rationality*. 1987.

[12] Amos Tversky and Daniel Kahneman. Judgment under uncertainty: Heuristics and biases. *Science*, 185:1124–1131, 1974.

[13] Daniel Kahneman and Amos Tversky. Prospect theory: An analysis of decision under risk. *Econometrica*, 47(2):263–291, 1979.

[14] Neil Stewart, Nick Chater, and Gordon D.A. Brown. Decision by sampling. *Cognitive Psychology*, 53(1):1–26, 2006.

[15] Gerd Gigerenzer. Why heuristics work. *Perspectives on Psychological Science*, 3(1):20–29, 2008.

[16] Sven Ove Hansson. Risk. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Spring 2014 edition, 2014.

[17] Trevor Bench-Capon and Paul Dunne. Argumentation in artificial intelligence. *Artificial Intelligence*, 171:701–729, 2007.

[18] Philippe Besnard and Anthony Hunter. *Elements of Argumentation*. The MIT Press, Cambridge, Massachusetts, 2008.

[19] Kenichi Okuno and Kazuko Takahashi. Argumentation system with changes of an agent's knowledge base. In *International Joint Conference on Artificial Intelligence*, pages 226–232, 2009.

[20] Leila Amgoud and Mathieu Serrurier. Agents that argue and explain classifications. *Autonomous Agents and Multi-Agent Systems*, 16(2):187–209, 2008.

[21] Katie Atkinson, Trevor Bench-Capon, and Peter McBurney. A dialogue game protocol for multi-agent argument over proposals for action. *Autonomous Agents and Multi-Agent Systems*, 11(2):153–171, 2005.

[22] Yannis Dimopoulos, Pavlos Moraitis, and Alexis Tsoukiàs. Argumentation based modeling of decision aiding for autonomous agents. In *IAT*, pages 99–105. IEEE Computer Society, 2004.

[23] Wietske Visser, Koen V. Hindriks, and Catholijn M. Jonker. An argumentation framework for qualitative multi-criteria preferences. In *TAFA 2011*, volume 7132 of *LNAI*, pages 85–98, 2012.

[24] Xiuyi Fan, Robert Craven, Ramsay Singer, Francesca Toni, and Matthew Williams. Assumption-based argumentation for decision-making with preferences: A medical case study. In Leite et al. [161], pages 374–390.

[25] Matthew Williams and Anthony Hunter. Harnessing ontologies for argument-based decision-making in breast cancer. In *Tools with Artificial Intelligence, 2007. ICTAI 2007. 19th IEEE International Conference on*, volume 2, pages 254–261, oct. 2007.

[26] Adam Wyner, Trevor Bench-Capon, and Katie Atkinson. Towards formalising argumentation about legal cases. In *Proceedings of the 13th International Conference on Artificial Intelligence and Law*, pages 1–10. ACM, 2011.

[27] Henry Prakken and Giovanni Sartor. Law and logic: A review from an argumentation perspective. *Artificial Intelligence*, 227:214 – 245, 2015.

[28] Leila Amgoud, Yannis Dimopoulos, and Pavlos Moraitis. Making decisions through preference-based argumentation. In Brewka and Lang [162], pages 113–123.

[29] Yannis Dimopoulos, Pavlos Moraitis, and Leila Amgoud. Extending argumentation to make good decisions. In Francesca Rossi and Alexis Tsoukiàs, editors, *ADT*, volume 5783 of *Lecture Notes in Computer Science*, pages 225–236. Springer, 2009.

[30] Xiuyi Fan, Francesca Toni, Andrei Mocanu, and Matthew Williams. Dialogical two-agent decision making with assumption-based argumentation. In Ana L. C. Bazzan, Michael N. Huhns, Alessio Lomuscio, and Paul Scerri, editors, *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '14, Paris, France, May 5-9, 2014*, pages 533–540. IFAAMAS/ACM, 2014.

[31] Paul-Amaury Matt, Francesca Toni, and Juan R. Vaccari. Dominant decisions by argumentation agents. In Peter McBurney, Iyad Rahwan, Simon Parsons, and Nicolas Maudet, editors, *ArgMAS*, volume 6057 of *Lecture Notes in Computer Science*, pages 42–59. Springer, 2009.

[32] Simon Parsons and Shaw Green. Argumentation and qualitative decision making. In Anthony Hunter and Simon Parsons, editors, *ESCQARU*, volume 1638 of *Lecture Notes in Computer Science*, pages 328–340. Springer, 1999.

[33] Leila Amgoud and Henri Prade. Using arguments for making and explaining decisions. *Artificial Intelligence*, 173:413–436, 2009.

[34] Jann Müller and Anthony Hunter. An argumentation-based approach for decision making. In *ICTAI*, pages 564–571. IEEE, 2012.

[35] Henry Prakken. An abstract framework for argumentation with structured arguments. *Argument & Computation*, 1(2):93–124, 2010.

[36] Jann Müller, Anthony Hunter, and Philip S. Taylor. Meta-level argumentation with argument schemes. In Liu et al. [163], pages 92–105.

[37] Jann Müller and Anthony Hunter. Deepflow: Using argument schemes to query relational databases. In Parsons et al. [164], pages 469–470.

[38] Niall Rooney, Hui Wang, Fiona Browne, Fergal Monaghan, Jann Müller, Alan Sergeant, Zhiwei Lin, Philip S. Taylor, and Vladimir Dobrynin. An exploration into the use of contextual document clustering for cluster sentiment analysis. In Galia Angelova, Kalina Bontcheva, Ruslan Mitkov, and Nicolas Nicolov, editors, *Recent Advances in Natural Language Processing, RANLP 2011, 12-14 September, 2011, Hissar, Bulgaria*, pages 140–145. RANLP 2011 Organising Committee, 2011.

[39] Fiona Browne, David A. Bell, Weiru Liu, Yan Jin, Colm Higgins, Niall Rooney, Hui Wang, and Jann Müller. Application of evidence theory and discounting techniques to aerospace design. In Greco et al. [165], pages 543–553.

[40] Jann Müller and Tobias Trapp. Using argumentation to develop a set of rules for claims classification. In *Proceedings of the 7th KES International Conference on Intelligent Decision Technologies (KES-IDT 2015)*, Intelligent Decision Technologies, pages 459–469. Springer, 2015.

[41] Sanjay Modgil and Henry Prakken. A general account of argumentation with preferences. *Artif. Intell.*, 195:361–397, 2013.

[42] 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.

[43] Hadassa Jakobovits and Dirk Vermeir. Robust Semantics for Argumentation Frameworks. *Journal of Logic and Computation*, 9(2):215–261, 1999.

[44] A. Bondarenko, P.M. Dung, R.A. Kowalski, and F. Toni. An abstract, argumentation-theoretic approach to default reasoning. *Artificial Intelligence*, 93:63–101, 1997.

[45] Leila Amgoud and Jonathan Ben-Naim. Ranking-based semantics for argumentation frameworks. In Liu et al. [163], pages 134–147.

[46] Philippe Besnard, Alejandro Javier García, Anthony Hunter, Sanjay Modgil, Henry Prakken, Guillermo Ricardo Simari, and Francesca Toni. Introduction to structured argumentation. *Argument & Computation*, 5(1):1–4, 2014.

[47] Phan Minh Dung, Robert Kowalski, and Francesca Toni. *Argumentation in AI*, chapter Assumption-based argumentation, pages 25–44. Springer, 2009.

[48] Alejandro J Garcia and Guillermo R Simari. Defeasible logic programming an argumentative approach. *Theory and Practice of Logic Programming*, 4(1):95–138, 2004.

[49] Philipe Besnard and Anthony Hunter. A logic-based theory of deductive arguments. *Artificial Intelligence*, 128(1-2):203–235, 2001.

[50] Philippe Besnard and Anthony Hunter. Constructing argument graphs with deductive arguments: a tutorial. *Argument & Computation*, 5(1):5–30, 2014.

[51] S. Staab and R. Studer, editors. *Handbook on Ontologies*. International Handbooks on Information Systems. Springer, 1 edition, 2004.

[52] Gang Kou, Kaisa Miettinen, and Yong Shi. Multiple criteria decision making: Challenges and advancements. *Journal of Multi-Criteria Decision Analysis*, 18:1–4, 2011.

[53] Hélène Fargier and Régis Sabbadin. Qualitative decision under uncertainty: back to expected utility. *Artif. Intell.*, 164(1-2):245–280, 2005.

[54] Didier Dubois, Hélène Fargier, Henri Prade, and Patrice Perny. Qualitative decision theory: from savage's axioms to nonmonotonic reasoning. *J. ACM*, 49(4):455–495, 2002.

[55] Paul-Amaury Matt and Francesca Toni. A game-theoretic measure of argument strength for abstract argumentation. In Steffen Hölldobler, Carsten Lutz, and Heinrich Wansing, editors, *JELIA*, volume 5293 of *Lecture Notes in Computer Science*, pages 285–297. Springer, 2008.

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

[57] Kristijonas Cyras and Francesca Toni. ABA+: assumption-based argumentation with preferences. In Baral et al. [166], pages 553–556.

[58] Rolf Haenni. Probabilistic argumentation. *J. Applied Logic*, 7(2):155–176, 2009.

[59] Jürg Kohlas. Probabilistic argumentation systems: A new way to combine logic with probability. *J. Applied Logic*, 1(3-4):225–253, 2003.

[60] Hengfei Li, Nir Oren, and Timothy J. Norman. Probabilistic argumentation frameworks. In Sanjay Modgil, Nir Oren, and Francesca Toni, editors, *Theorie and Applications of Formal Argumentation - First International Workshop, TAFA 2011. Barcelona, Spain, July 16-17, 2011, Revised Selected Papers*, volume 7132 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2011.

[61] Anthony Hunter. A probabilistic approach to modelling uncertain logical arguments. *Int. J. Approx. Reasoning*, 54(1):47–81, 2013.

[62] Anthony Hunter. Probabilistic qualification of attack in abstract argumentation. *Int. J. Approx. Reasoning*, 55(2):607–638, 2014.

[63] Pedro Cabalar and Tran Cao Son, editors. *Logic Programming and Nonmonotonic Reasoning, 12th International Conference, LPNMR 2013, Corunna, Spain, September 15-19, 2013. Proceedings*, volume 8148 of *Lecture Notes in Computer Science*. Springer, 2013.

[64] Yuqing Tang, Chung-Wei Hang, Simon Parsons, and Munindar P. Singh. Towards argumentation with symbolic dempster-shafer evidence. In Verheij et al. [167], pages 462–469.

[65] Ringo Baumann and Gerhard Brewka. Expanding argumentation frameworks: Enforcing and monotonicity results. In Baroni et al. [168], pages 75–86.

[66] Ringo Baumann. What does it take to enforce an argument? minimal change in abstract argumentation. In Luc De Raedt, Christian Bessière, Didier Dubois, Patrick Doherty, Paolo Frasconi, Fredrik Heintz, and Peter J. F. Lucas, editors, *ECAI 2012 - 20th European Conference on Artificial Intelligence. Including Prestigious Applications of Artificial Intelligence (PAIS-2012) System Demonstrations Track, Montpellier, France, August 27-31 , 2012*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, pages 127–132. IOS Press, 2012.

[67] Ringo Baumann and Gerhard Brewka. AGM meets abstract argumentation: Expansion and revision for dung frameworks. In Yang and Wooldridge [169], pages 2734–2740.

[68] Ringo Baumann and Gerhard Brewka. Spectra in abstract argumentation: An analysis of minimal change. In Cabalar and Son [63], pages 174–186.

[69] Lucas Carstens, Xiuyi Fan, Yang Gao, and Francesca Toni. An overview of argumentation frameworks for decision support. In Madalina Croitoru, Pierre Marquis, Sebastian Rudolph, and Gem Stapleton, editors, *Graph Structures for Knowledge Representation and Reasoning - 4th International Workshop, GKR 2015, Buenos Aires, Argentina, July 25, 2015, Revised Selected Papers*, volume 9501 of *Lecture Notes in Computer Science*, pages 32–49. Springer, 2015.

[70] Xiuyi Fan and Francesca Toni. Decision making with assumption-based argumentation. In E. Black, S. Modgil, and N. Oren, editors, *TAFA 2013*, volume 8306 of *LNAI*, pages 127–142, 2014.

[71] Pietro Baroni, Marco Romano, Francesca Toni, Marco Aurisicchio, and Giorgio Bertanza. An argumentation-based approach for automatic evaluation of design debates. In Leite et al. [161], pages 340–356.

[72] Pietro Baroni, Marco Romano, Francesca Toni, Marco Aurisicchio, and Giorgio Bertanza. Automatic evaluation of design alternatives with quantitative argumentation. *Argument & Computation*, 6(1):24–49, 2015.

[73] Valentinos Evripidou, Lucas Carstens, Francesca Toni, and David Cabanillas. Argumentation-based collaborative decisions for design. In *26th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2014, Limassol, Cyprus, November 10-12, 2014*, pages 805–809. IEEE Computer Society, 2014.

[74] Didier Dubois, Lluis Godo, Ramon López de Mántaras, and Henri Prade. Qualitative reasoning with imprecise probabilities. *J. Intell. Inf. Syst.*, 2(4):319–363, 1993.

[75] Didier Dubois, Hélène Fargier, and Patrice Perny. Qualitative decision theory with preference relations and comparative uncertainty: An axiomatic approach. *Artif. Intell.*, 148(1-2):219–260, 2003.

[76] Kristijonas Cyras and Francesca Toni. Non-monotonic inference properties for assumption-based argumentation. In Elizabeth Black, Sanjay Modgil, and Nir

Oren, editors, *Theory and Applications of Formal Argumentation - Third International Workshop, TAFA 2015, Buenos Aires, Argentina, July 25-26, 2015, Revised Selected Papers*, volume 9524 of *Lecture Notes in Computer Science*, pages 92–111. Springer, 2015.

[77] Martin Diller, Adrian Haret, Thomas Linsbichler, Stefan Rümmele, and Stefan Woltran. An extension-based approach to belief revision in abstract argumentation. In Yang and Wooldridge [169], pages 2926–2932.

[78] Toby Walsh, editor. *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*. IJCAI/AAAI, 2011.

[79] Pablo Pilotti, Ana Casali, and Carlos Iván Chesñevar. A belief revision approach for argumentation-based negotiation agents. *Applied Mathematics and Computer Science*, 25(3):455–470, 2015.

[80] Guido Boella, Leendert W. N. van der Torre, and Serena Villata. Changing institutional goals and beliefs of autonomous agents. In The Duy Bui, Tuong Vinh Ho, and Quang-Thuy Ha, editors, *Intelligent Agents and Multi-Agent Systems, 11th Pacific Rim International Conference on Multi-Agents, PRIMA 2008, Hanoi, Vietnam, December 15-16, 2008. Proceedings*, volume 5357 of *Lecture Notes in Computer Science*, pages 78–85. Springer, 2008.

[81] Gerardo I. Simari, Paulo Shakarian, and Marcelo A. Falappa. A quantitative approach to belief revision in structured probabilistic argumentation. *Ann. Math. Artif. Intell.*, 76(3-4):375–408, 2016.

[82] Paulo Shakarian, Gerardo I. Simari, and Marcelo A. Falappa. Belief revision in structured probabilistic argumentation. In Christoph Beierle and Carlo Meghini, editors, *Foundations of Information and Knowledge Systems - 8th International Symposium, FoIKS 2014, Bordeaux, France, March 3-7, 2014. Proceedings*, volume 8367 of *Lecture Notes in Computer Science*, pages 324–343. Springer, 2014.

[83] Célia da Costa Pereira, Andrea Tettamanzi, and Serena Villata. Changing one's mind: Erase or rewind? In Walsh [78], pages 164–171.

[84] Bei Shui Liao, Li Jin, and Robert C. Koons. Dynamics of argumentation systems: A division-based method. *Artif. Intell.*, 175(11):1790–1814, 2011.

[85] Jean-Guy Mailly. Dynamic of argumentation frameworks. In Francesca Rossi, editor, *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, pages 3233–3234. IJCAI/AAAI, 2013.

[86] Marcela Capobianco, Carlos Iván Chesñevar, and Guillermo Ricardo Simari. Argumentation and the dynamics of warranted beliefs in changing environments. *Autonomous Agents and Multi-Agent Systems*, 11(2):127–151, 2005.

[87] Marcelo A. Falappa, Alejandro Javier García, and Guillermo Ricardo Simari. Belief dynamics and defeasible argumentation in rational agents. In James P. Delgrande and Torsten Schaub, editors, *10th International Workshop on Non-Monotonic Reasoning (NMR 2004), Whistler, Canada, June 6-8, 2004, Proceedings*, pages 164–170, 2004.

[88] Leila Amgoud and Philippe Besnard. Bridging the gap between abstract argumentation systems and logic. *Scalable Uncertainty Management*, pages 12–27, 2009.

[89] John Von Neumann. Zur theorie der gesellschaftsspiele. *Annalen der Mathematik*, 100:295–320, 1928.

[90] Joe Barkai. Design for excellence. *OEM OffHighway Magazine*, 2014.

[91] Xiaochun Hu, Jun Pang, Yang Pang, Michael Atwood, Wei Sun, and William C. Regli. A survey on design rationale: Representation, capture and retrieval. In *Proceedings of DETC'000 (2000 ASME Design Engineering Technical Conference)*, 2000.

[92] Alex P. J. Jarczyk, Peter Löffler, and Frank M. Shipman III. Design rationale for software engineering: A survey. In *Proceedings of the 25th Hawaii International Conference on System Sciences*, pages 577–586. Springer, 1992.

[93] Muhammad Ali Babar, Antony Tang, Ian Gorton, and Jun Han. Industrial perspective on the usefulness of design rationale for software maintenance: A survey. In *Proceedings of the Sixth International Conference on Quality Software (QSIC'06)*. IEEE Computer Society, 2006.

[94] Antony Tang, Muhammad Ali Babar, Ian Gorton, and Jun Han. A survey of architecture design rationale. Technical report, Swinburne University of Technology and NICTA), 2005.

[95] Klaas Dellschaft, Hendrik Engelbrecht, José Barreto, Sascha Rutenbeck, and Steffen Staab. Cicero: Tracking design rationale in collaborative ontology engineering. *The Semantic Web: Research and Applications*, pages 782–786, 2008.

[96] Werner Kunz, Horst W. J. Rittel, We Messrs, H. Dehlinger, T. Mann, and J. J. Protzen. Issues as elements of information systems. Technical report, 1970.

[97] Thomas Gordon, Henry Prakken, and Douglas Walton. The carneades model of argument and burden of proof. *Artificial Intelligence*, 171:875–896, 2007.

[98] Giuseppe Carenini and Johanna D. Moore. Generating and evaluating evaluative arguments. *Artif. Intell.*, 170(11):925–952, 2006.

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

[100] Sanjay Modgil. Reasoning about preferences in argumentation frameworks. *Artificial Intelligence*, 173:901–934, 2009.

[101] Leila Amgoud, Jonathan Ben-Naim, Dragan Doder, and Srdjan Vesic. Ranking arguments with compensation-based semantics. In Baral et al. [166], pages 12–21.

[102] Anthony Hunter. Some foundations for probabilistic abstract argumentation. In Verheij et al. [167], pages 117–128.

[103] Paul Krause, Simon Ambler, Morten Elvang-Gøransson, and John Fox. A logic of argumentation for reasoning under uncertainty. *Computational Intelligence*, 11:113–131, 1995.

[104] Anthony Hunter and Sébastien Konieczny. On the measure of conflicts: Shapley inconsistency values. *Artificial Intelligence*, 174(14):1007–1026, September 2010.

[105] John Grant and Anthony Hunter. Distance-based measures of inconsistency. In Linda C. van der Gaag, editor, *ECSQARU*, volume 7958 of *Lecture Notes in Computer Science*, pages 230–241. Springer, 2013.

[106] Matthias Thimm. Inconsistency measures for probabilistic logics. *Artif. Intell.*, 197:1–24, 2013.

[107] David Picado-Muiño. Measuring and repairing inconsistency in knowledge bases with graded truth. *Fuzzy Sets and Systems*, 197:108–122, 2012.

[108] Anthony Hunter and Sébastien Konieczny. Measuring inconsistency through minimal inconsistent sets. In Brewka and Lang [162], pages 358–366.

[109] Kedian Mu, Weiru Liu, and Zhi Jin. A general framework for measuring inconsistency through minimal inconsistent sets. *Knowl. Inf. Syst.*, 27(1):85–114, 2011.

[110] John Grant and Anthony Hunter. Measuring consistency gain and information loss in stepwise inconsistency resolution. In *Symbolic and Quantitative Approaches to Reasoning with Uncertainty - 11th European Conference, ECSQARU 2011, Belfast, UK, June 29-July 1, 2011. Proceedings*, pages 362–373, 2011.

[111] Pere Pardo, Sergio Pajares Ferrando, Eva Onaindia, Lluis Godo, and Pilar Dellunde. Cooperative dialogues for defeasible argumentation-based planning. In Peter McBurney, Simon Parsons, and Iyad Rahwan, editors, *Argumentation in*

Multi-Agent Systems - 8th International Workshop, ArgMAS 2011, Taipei, Taiwan, May 3, 2011, Revised Selected Papers, volume 7543 of *Lecture Notes in Computer Science*, pages 174–193. Springer, 2011.

[112] Pere Pardo, Sergio Pajares, Eva Onaindia, Lluis Godo, and Pilar Dellunde. Multiagent argumentation for cooperative planning in delp-pop. In Liz Sonenberg, Peter Stone, Kagan Tumer, and Pinar Yolum, editors, *10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011), Taipei, Taiwan, May 2-6, 2011, Volume 1-3*, pages 971–978. IFAAMAS, 2011.

[113] Sergio Pajares Ferrando and Eva Onaindia. Defeasible argumentation for multiagent planning in ambient intelligence applications. In *International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2012, Valencia, Spain, June 4-8, 2012 (3 Volumes)*, pages 509–516, 2012.

[114] Angel Rolando Medellin Gasque. *Argumentation-based dialogues over cooperative plans*. PhD thesis, University of Liverpool, UK, 2013.

[115] Peter McBurney and Simon Parsons. Dialogue games for agent argumentation. In Guillermo Ricardo Simari and Iyad Rahwan, editors, *Argumentation in Artificial Intelligence*, pages 261–280. Springer, 2009.

[116] Anthony Hunter. Modelling the persuadee in asymmetric argumentation dialogues for persuasion. In Yang and Wooldridge [169], pages 3055–3061.

[117] Anthony Hunter. Persuasion dialogues via restricted interfaces using probabilistic argumentation. In Steven Schockaert and Pierre Senellart, editors, *Scalable Uncertainty Management - 10th International Conference, SUM 2016, Nice, France, September 21-23, 2016, Proceedings*, volume 9858 of *Lecture Notes in Computer Science*, pages 184–198. Springer, 2016.

[118] Elizabeth Black and Anthony Hunter. An inquiry dialogue system. *Autonomous Agents and Multi-Agent Systems*, 19(2):173–209, 2009.

[119] Christophe Labreuche, Nicolas Maudet, Wassila Ouerdane, and Simon Parsons. A dialogue game for recommendation with adaptive preference models. In Ger-

hard Weiss, Pinar Yolum, Rafael H. Bordini, and Edith Elkind, editors, *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, Istanbul, Turkey, May 4-8, 2015*, pages 959–967. ACM, 2015.

[120] Floris Bex and Trevor J. M. Bench-Capon. Extracting and understanding arguments about motives from stories. In Elena Cabrio, Serena Villata, and Adam Wyner, editors, *Proceedings of the Workshop on Frontiers and Connections between Argumentation Theory and Natural Language Processing, Forlì-Cesena, Italy, July 21-25, 2014.*, volume 1341 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2014.

[121] Theodosios Goudas, Christos Louizos, Georgios Petasis, and Vangelis Karkaletsis. Argument extraction from news, blogs, and the social web. *International Journal on Artificial Intelligence Tools*, 24(5), 2015.

[122] Douglas Walton. Using argumentation schemes for argument extraction: A bottom-up method. *IJCINI*, 6(3):33–61, 2012.

[123] Anthony Hunter. Base logics in argumentation. In *Computational Models of Argument (COMMA'10)*, pages 275–278. IOS Press, 2010.

[124] Leila Amgoud, Philippe Besnard, and Anthony Hunter. Logical representation and analysis for rc-arguments. In *ICTAI'2015*, 2015.

[125] Leila Amgoud, Philippe Besnard, and Anthony Hunter. Representing and reasoning about arguments mined from texts and dialogues. In Sébastien Destercke and Thierry Denoeux, editors, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty - 13th European Conference, ECSQARU 2015, Compiègne, France, July 15-17, 2015. Proceedings*, volume 9161 of *Lecture Notes in Computer Science*, pages 60–71. Springer, 2015.

[126] Douglas Walton, Chris Reed, and Fabrizio Macagno. *Argumentation Schemes*. Cambridge University Press, 2008.

[127] Anthony Hunter. Reasoning about the appropriateness of proponents for arguments. In *AAAI 2008*, pages 89–94, 2008.

[128] Douglas Walton. *Fundamentals of Critical Argumentation*. Cambridge University Press, 2006.

[129] Henry Prakken. AI & law, logic and argument schemes. *Argumentation*, 19:303–320, 2005.

[130] Adam Wyner and Trevor Bench-Capon. Argument schemes for legal case-based reasoning. In *International Conference on Legal Knowledge and Information Systems*, pages 139–149, 2007.

[131] Trevor J. M. Bench-Capon, Henry Prakken, Adam Wyner, and Katie Atkinson. Argument schemes for reasoning with legal cases using values. In *International Conference on Artificial Intelligence and Law, ICAIL '13, Rome, Italy, June 10-14, 2013*, pages 13–22, 2013.

[132] Simon Parsons, Katie Atkinson, Zimi Li, Peter McBurney, Elizabeth Sklar, Munindar P. Singh, Karen Zita Haigh, Karl N. Levitt, and Jeff Rowe. Argument schemes for reasoning about trust. *Argument & Computation*, 5(2-3):160–190, 2014.

[133] Douglas Walton. *Appeal to Expert Opinion*. Pennsylvania State University Press, University Park, Pennsylvania, 1997.

[134] F. William Lawvere. Functorial semantics of algebraic theories. In *Proceedings of the National Academy of Sciences*, pages 869–72. Springer, 1963.

[135] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine.

[136] Paul Hudak, Simon Peyton Jones, Philip Wadler, Brian Boutel, Jon Fairbairn, Joseph Fasel, María Guzmán, Kevin Hammond, John Hughes, Thomas Johnsson, Richard Kieburtz, Rishiyur Nikhil, Will Partain, and John Peterson. Report on the programming language haskell, A non-strict, purely functional language. *SIGPLAN Notices*, 27(5):1, 1992.

[137] Bas van Gijzel and Henrik Nilsson. A principled approach to the implementation of argumentation models. In Parsons et al. [164], pages 293–300.

[138] Bas van Gijzel. Tools for the implementation of argumentation models. In Andrew V. Jones and Nicholas Ng, editors, *2013 Imperial College Computing Student Workshop, ICCSW 2013, September 26/27, 2013, London, United Kingdom*, volume 35 of *OASICS*, pages 43–48. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2013.

[139] Katie Atkinson, Trevor Bench-Capon, and Sanjay Modgil. Argumentation for decision support. In S. Bressan, J. Kng, and R. Wagner, editors, *Proceedings of the Seventeenth International Conference on Database and Expert Systems Applications (DEXA 2006)*, number 4080 in Lecture Notes in Computer Science (LNCS), pages 822–831, Berlin, Germany, 2006. Springer.

[140] Floris Bex, Henry Prakken, Chris Reed, and Douglas Walton. Towards a formal account of reasoning about evidence: Argumentation schemes and generalisations. *Artificial Intelligence and Law*, 11:125–165, 2003.

[141] Elizabeth Sklar, Simon Parsons, and Munindar P. Singh. Towards an argumentation-based model of social interaction. In *Proceedings of the Tenth International Workshop on Argumentation in Multi-Agent Systems (ArgMAS 2013)*, 2013.

[142] Nir Oren. Argument schemes for normative practical reasoning. In Elizabeth Black, Sanjay Modgil, and Nir Oren, editors, *Theory and Applications of Formal Argumentation - Second International Workshop, TAFA 2013, Beijing, China, August 3-5, 2013, Revised Selected papers*, volume 8306 of *Lecture Notes in Computer Science*, pages 63–78. Springer, 2013.

[143] Henry Prakken, Adam Wyner, Trevor Bench-Capon, and Katie Atkinson. A formalisation of argumentation schemes for legal case-based reasoning in aspic+. *Journal of Logic and Computation*, in press, 2013.

[144] Thomas Gordon and Douglas Walton. Legal reasoning with argumentation schemes. In *Proceedings of the 12th International Conference on Artificial Intelligence and Law*, pages 137–146, 2009.

[145] Henry Prakken, Adam Wyner, Trevor Bench-Capon, and Katie Atkinson. A formalization of argumentation schemes for legal case-based reasoning in ASPIC+. *J. Log. Comput.*, 25(5):1141–1166, 2015.

[146] Katie Atkinson, Trevor Bench-Capon, and Peter McBurney. Computational representation of practical argument. *Synthese*, 152(2):157–206, 2006.

[147] Sanjay Modgil and Trevor Bench-Capon. Metalevel argumentation. *J. Log. Comput.*, 21(6):959–1003, 2011.

[148] Guido Boella, Leendert van der Torre, and Serena Villata. On the acceptability of meta-arguments. In *Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT '09. IEEE/WIC/ACM International Joint Conferences on*, volume 2, pages 259 –262, sept. 2009.

[149] Michael Wooldridge, Peter McBurney, and Simon Parsons. On the meta-logic of arguments. In Simon Parsons, Nicolas Maudet, Pavlos Moraitis, and Iyad Rahwan, editors, *Argumentation in Multi-Agent Systems, Second International Workshop, ArgMAS 2005, Utrecht, The Netherlands, July 26, 2005, Revised Selected and Invited Papers*, volume 4049 of *Lecture Notes in Computer Science*, pages 42–56. Springer, 2005.

[150] Sanjay Modgil. Hierarchical argumentation. In Fisher et al. [170], pages 319–332.

[151] Claudette Cayrol and Marie-Christine Lagasquie-Schiex. On the acceptability of arguments in bipolar argumentation frameworks. In *Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 378–389. Springer, 2005.

[152] Farid Nouioua and Vincent Risch. Bipolar argumentation frameworks with specialized supports. In *Tools with Artificial Intelligence (ICTAI), 2010 22nd IEEE International Conference on*, volume 1, pages 215–218, 2010.

[153] Claudette Cayrol and Marie-Christine Lagasquie-Schiex. Bipolarity in argumentation graphs: Towards a better understanding. In *Scalable Uncertainty Management*, volume 6929 of *Lecture Notes in Computer Science*, pages 137–148, 2011.

[154] Claudette Cayrol and Marie-Christine Lagasquie-Schiex. Bipolarity in argumentation graphs: Towards a better understanding. *Int. J. Approx. Reasoning*, 54(7):876–899, 2013.

[155] Claudette Cayrol and Marie-Christine Lagasquie-Schiex. *Argumentation in Artificial Intelligence*, chapter Bipolar abstract argumentation systems, pages 65–84. Springer, 2009.

[156] Nir Oren and Timothy J. Norman. Semantics for evidence-based argumentation. In *Computational Models of Argument: Proceedings of COMMA 2008, Toulouse, France, May 28-30, 2008.*, pages 276–284, 2008.

[157] Nir Oren, Chris Reed, and Michael Luck. Moving between argumentation frameworks. In Baroni et al. [168], pages 379–390.

[158] Sylwia Polberg and Nir Oren. Revisiting support in abstract argumentation systems. In Parsons et al. [164], pages 369–376.

[159] Paul E. Dunne and Yann Chevaleyre. The complexity of deciding reachability properties of distributed negotiation schemes. *Theor. Comput. Sci.*, 396(1-3):113–144, 2008.

[160] Philippe Besnard, Sylvie Doutre, and Andreas Herzig. Encoding argument graphs in logic. In Anne Laurent, Olivier Strauss, Bernadette Bouchon-Meunier, and Ronald R. Yager, editors, *Information Processing and Management of Uncertainty in Knowledge-Based Systems - 15th International Conference, IPMU 2014, Montpellier, France, July 15-19, 2014, Proceedings, Part II*, volume 443 of *Communications in Computer and Information Science*, pages 345–354. Springer, 2014.

[161] João Leite, Tran Cao Son, Paolo Torroni, Leon van der Torre, and Stefan Woltran, editors. *Computational Logic in Multi-Agent Systems - 14th International Workshop, CLIMA XIV, Corunna, Spain, September 16-18, 2013. Proceedings*, volume 8143 of *Lecture Notes in Computer Science*. Springer, 2013.

[162] Gerhard Brewka and Jérôme Lang, editors. *Principles of Knowledge Representation and Reasoning: Proceedings of the Eleventh International Conference, KR 2008, Sydney, Australia, September 16-19, 2008*. AAAI Press, 2008.

[163] Weiru Liu, V. S. Subrahmanian, and Jef Wijsen, editors. *Scalable Uncertainty Management - 7th International Conference, SUM 2013, Washington, DC, USA, September 16-18, 2013. Proceedings*, volume 8078 of *Lecture Notes in Computer Science*. Springer, 2013.

[164] Simon Parsons, Nir Oren, Chris Reed, and Federico Cerutti, editors. *Computational Models of Argument - Proceedings of COMMA 2014, Atholl Palace Hotel, Scottish Highlands, UK, September 9-12, 2014*, volume 266 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2014.

[165] Salvatore Greco, Bernadette Bouchon-Meunier, Giulianella Coletti, Mario Fedrizzi, Benedetto Matarazzo, and Ronald R. Yager, editors. *Advances in Computational Intelligence - 14th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, IPMU 2012, Catania, Italy, July 9-13, 2012, Proceedings, Part III*, volume 299 of *Communications in Computer and Information Science*. Springer, 2012.

[166] Chitta Baral, James P. Delgrande, and Frank Wolter, editors. *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifteenth International Conference, KR 2016, Cape Town, South Africa, April 25-29, 2016*. AAAI Press, 2016.

[167] Bart Verheij, Stefan Szeider, and Stefan Woltran, editors. *Computational Models of Argument - Proceedings of COMMA 2012, Vienna, Austria, September 10-12, 2012*, volume 245 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2012.

[168] Pietro Baroni, Federico Cerutti, Massimiliano Giacomin, and Guillermo Ricardo Simari, editors. *Computational Models of Argument: Proceedings of COMMA 2010, Desenzano del Garda, Italy, September 8-10, 2010*, volume 216 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2010.

[169] Qiang Yang and Michael Wooldridge, editors. *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*. AAAI Press, 2015.

[170] Michael Fisher, Wiebe van der Hoek, Boris Konev, and Alexei Lisitsa, editors. *Logics in Artificial Intelligence, 10th European Conference, JELIA 2006, Liverpool, UK, September 13-15, 2006, Proceedings*, volume 4160 of *Lecture Notes in Computer Science*. Springer, 2006.