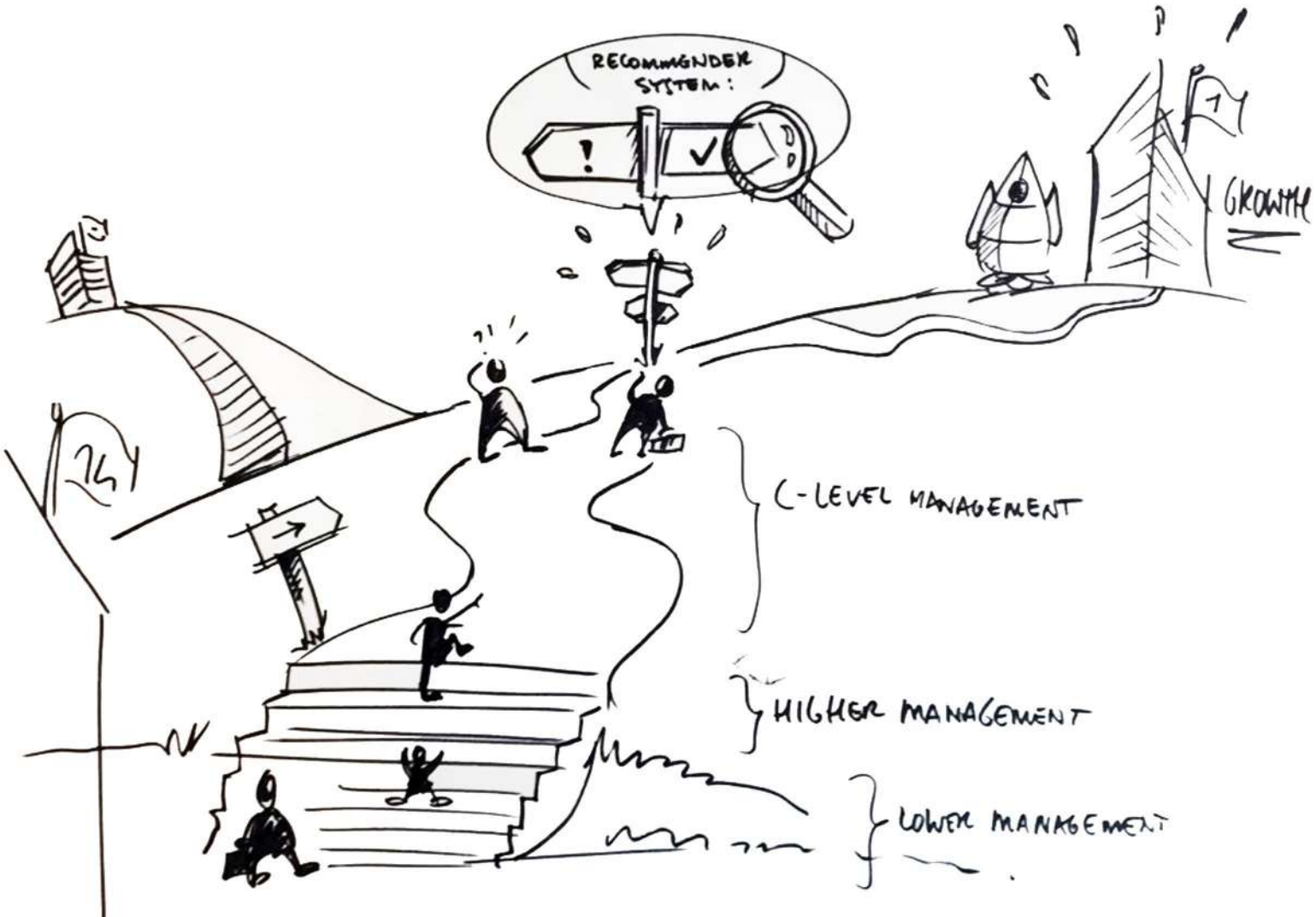
Rational Architecture

Reasoning about Enterprise Dynamics



Supervisor:

Prof. Dr. L. van der Torre (University of Luxembourg)

The author was employed at the University of Luxembourg and received support from the National Research Fund Luxembourg (reference PHD/09/082) in the project “Rational Architecture”.

PhD-FSTC-2017-28
The Faculty of Sciences, Technology and Communication

DISSERTATION

Presented on 06/04/2017 in Luxembourg

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG

EN INFORMATIQUE

by

Marc van Zee

Born on 28 October 1985 in Vlissingen (The Netherlands)

RATIONAL ARCHITECTURE
REASONING ABOUT ENTERPRISE DYNAMICS

Dissertation defense committee

Prof. Pierre Kelsen, chairman
Université du Luxembourg, Luxembourg

Prof. Wiebe van der Hoek, vice-chairman
University of Liverpool, United Kingdom

Prof. Leendert van der Torre, supervisor
Université du Luxembourg, Luxembourg

Prof. Farhad Arbab
CWI Amsterdam, the Netherlands

Dr. Andreas Herzig
IRIT, Université Paul Sabatier, France (CNRS)

Prof. Erik Proper, expert
Luxembourg Institute of Science and Technology, Luxembourg

Dr. Dragan Doder, expert
University of Belgrade, Serbia

To my parents.

Summary

Following the banking crisis in the late 2000s, the Commission Wijffels [Wij13] conducts a major study on how the stability of banks in the Netherlands can be improved. The motivation for this study is that "banks have been insufficiently compliant and stable in the last years." One of their main recommendations is for every bank to use enterprise architecture as a central component. Enterprise architecture is used to model large enterprises in a holistic fashion by connecting their IT infrastructure and applications to the business processes they support. In turn this links them also to the products and services that are realized by those business processes. Where software architecture is analogous to the architecture of a building, enterprise architecture is comparable to the planning of a city. It involves long-term plans, a large amount of (changing) stakeholders, vague and high-level goals, and many types of uncertainty.

The ultimate goal of this thesis is a decision support system for enterprise architects. Although decision support systems have found their way in many fields such as information architecture, software architecture, and business informatics, they have not done so much for enterprise architecture. This is partly due to the relative young age of the profession, and partly due to the complexity of the domain. In this thesis we take a first step in this direction by building a bridge between enterprise architecture and artificial intelligence. Since the field of enterprise architecture is large, our specific focus is on how artificial intelligence can play a role in formalizing reasoning processes in enterprise architecture. We build our bridge from enterprise architecture to artificial intelligence in three parts. We start from the practice of enterprise architecture, but as we progress through the thesis, the contributions will become increasingly more technical and the result more general.

The first part of this thesis consists of clarifying important characteristics of enterprise architecture. We do so through a mixed qualitative/quantitative empirical study among a group of enterprise architects. This results in a list of eight characteristics, which we then use as yardsticks for remaining parts of the thesis. We analyze an existing approach for enterprise architecture decision rationalization and recognize that many of our characteristics are not supported.

In the second part, we focus on reasoning in the early requirement phase of the development of an enterprise architecture. In the early requirements phase, an enterprise architect collaborates with a group of stakeholders to refine the high-level goals and values of the enterprise into more specific goals and tasks. Although there are various languages and tools in existence to support these activities, there is limited support for reasoning about the relation between the resulting models and underlying arguments that were put forward in the discussions. To support this, we develop the RationalGRL framework, which uses techniques from formal argumentation to formally trace back goals and tasks of an information system to underlying arguments.

In the third part, we focus on reasoning about enterprise architecture planning. We store plans and assumptions in a database, and develop a reasoning formalism for the dynamics of this database. The database plays the role of an intelligent calendar, performing consistency checks when new assumptions or plans are added. We formalize the database using a belief-desire-intention logic, a common approach to formalize resource-bounded planning. In order to characterize the dynamic of the database, we develop postulates for rational revision, and we prove representation theorems linking these postulates to a pre-order over semantic models.

Acknowledgments

Ever since I first learned how to beat my brother at the game Snake by changing a few lines of QBasic code, I have been fascinated by computers. But I somehow couldn't see myself working on programming, hardware or network protocols for my entire life. This is why I decided to do a bachelor study Industrial Design, focusing on intelligent product design. During that study, I became more and more interested in artificial intelligence, independently of products. After reading the book "Gödel, Escher, Bach" by Douglas Hofstadter with my friend Jesse, I made the decision to start a master study Artificial Intelligence at Utrecht University. There I was introduced to subjects such as logics of agency (John-Jules Meyer), defeasible logics and argumentation (Henry Prakken), and the computational beauty of nature (Gerard Vreeswijk). Each of these subjects inspired me greatly and laid the basis for most of the things you find in this thesis, so I am forever thankful for the passion and dedication of these teachers.

I consider myself extremely lucky to have been able to do my Ph.D. at the University of Luxembourg at the research group Individual and Collective Reasoning. I have met great researchers, some of whom I now consider to be good friends. Some deserve special mentioning. Agustin, my first office mate and probably the most helpful and genuine person I have ever met, thanks for always being there for me. Silvano, a decent researcher but a terrible chess player, I enjoyed working on social network analysis with you. Pouyan, Aida, Tjitze, Mikolai, Giovanni, Marcos, Livio, Xavier, Cristiana, Ana, Alessia, Robert, Alessandra, and Dov, it was a pleasure getting to know you all, whether it was by doing research together or having a few beers at Liquid.

Thanks to modern technology, royal funding, and a very open-minded supervisor I was able to interact with a very large number of people from diverse disciplines outside Luxembourg. I thank Thomas Icard for working on intention reconsideration during my visits at Stanford, it made me realize working together with a philosopher can be extremely inspiring. I also thank Yoav Shoham for his ideas and discussions during my visit and for laying the conceptual foundations for the last part of my thesis.

Before I started my Ph.D., I had no knowledge of enterprise architecture, and I am grateful to Erik Proper for helping me in this respect by organizing seminars, and bringing me in contact with practitioners. I also thank enterprise architects Martin van den Berg of de Nederlandsche Bank, as well as Saco Bekius and Michiel Borgers of de Belastingdienst for meeting me on various occasions and giving me important insights into the practice of the field. Finally I thank Diana, Dirk, and Georgios for their opinions and for swimming with me at the Coque now and then, as well as writing great papers with me.

Sepideh Ghanavati introduced me to the field of requirements engineering, which plays an important part in my thesis as well. Thanks for always taking time to talk to me. I also thank Floris Bex for spending a lot of time contributing to our papers.

I have also been lucky enough to do an internship at Google Pittsburgh in Pennsylvania. Special thanks there goes to Mitch, who always had time to answer my many questions. A big thanks to the Moka team as well, and to my co-interns Sam, Jacen, Marko, and Angela. It was a lot of fun hanging out with you, and your help in the coding interviews was invaluable.

I should thank three persons in my research group who played the biggest part in shaping this thesis. Emil, thanks for being my daily supervisor. Our many discussions and notes have laid the foundations for many of my ideas in this thesis. But more importantly, you were my ultimate opponent when it comes to defending my own opinions. I frequently left our meetings dazed and confused, but overall it has made a stronger person. Dragan, you are probably the best mathematician I ever met. Truth be told, I haven't met a lot of them, but I still learned a huge amount of things from you. Thanks for taking a lot of time going through the proofs of this thesis with me. And finally Leon, my "official" supervisor. I am forever grateful for the freedom you have given me and the time you spent shaping my ideas. In the coming years I will try, most certainly without success, to attain your gift for combining broad intuition with mastery of technical details.

Besides all these collaborations in research, I couldn't have written this thesis without support from my friends and family. I thank my friends from the football team Ell ("Mir sinn den Eller!") for making me feel very welcome in Luxembourg. I have had a lot of fun with you. Special thanks for Thierry, I'll miss our nights in your car listening to David Bowie and Lou Reed. Frank, we've experiences so many things together, that I cannot imagine our friendship ever to end. Thanks for being such a good friend for such a long time, and thanks for making the figures of the "bridges" in Chapter 1. Jesse and Joost, we may live apart, but our design trio will always live on. Ontwerp tot nut van het algemeen en bijzondere! Christ, I know we are far apart, but you always feel close at heart. Siem, Toon, Luuk, and Timon, our indescribable adventures will stay with me forever.

Finally, I thank my parents, Hans and Wilma, for their continuous love and support, and for raising me in peace and serenity. I know my life isn't exactly as serene as yours, but I thank you for believing in me and for always being there for me. I also thank my brother Jeroen, who has been one of my best friends ever since we've been growing up. I don't think anybody has such a good taste of music as we do. Lastly, Claire, thank you for having the patience to understand my many irrational impulses, for believing in me, and for always being there for me. Home is wherever I'm with you.

Contents

1

Introduction

A characteristic of design that is special to it, besides this gradual emergence of goals, is that the largest task is to generate alternatives. There are lots of theories of decision making, a field that has been heavily cultivated by economists and statisticians. But most theories of decision making start out with a given set of alternatives and then ask how to choose among them. In design, by contrast, most of the time and effort is spent in generating the alternatives, which aren't given at the outset. [. . .] The idea that we start out with all the alternatives and then choose among them is wholly unrealistic.

Herbert A. Simon, *What we know about learning* [Sim98]

In this thesis we study reasoning of enterprise architects. We investigate important characteristics of enterprise architect reasoning and we develop languages and techniques from logic and artificial intelligence in order to formalize them. This thesis contributes to state of the art research in both enterprise architecture and artificial intelligence, and consists of three parts. In the first part, we investigate the characteristics of enterprise architect reasoning, and in the second and third part we develop increasingly advanced logics and languages to formalize some of these characteristics.

In this chapter, we provide a general introduction to enterprise architecture, and we outline the three main parts of this thesis. For each part we provide a methodology, research questions, and a summary of the main results. We also provide a personal motivation, and a reading guide for this thesis.

1.1 Motivation

The idea that increasingly smarter machines can perform manual work is already well-known. In a widely cited study published in 2013 [FO17], Carl Benedikt Frey and Michael Osborne examine 702 jobs in the United States and find that 47% of all workers have jobs at high risk of potential automation. They conclude that recent developments in artificial intelligence will put a significant part of employment at risk in the near future, across a wide range of occupations. Such predictions are becoming a reality fast as the insurance firm Fukoku Mutual Life in Japan will lay off more than 30 employees and replace them with IBM's Watson Explorer AI, which can calculate payouts to policyholders. The insurance firm calculated it will increase productivity by 30% and save about 140m yen ($1.2m) a year after the 200m yen ($1.7m) AI system is installed January 2017. Maintaining it will cost about 15m yen ($130k) a year.[1]

[1] See https://www.theguardian.com/technology/2017/jan/05/japanese-company-replaces-office-workers-artificial-intelligence-ai-fukoku-mutual-life-insurance

According to experts, vulnerability to automation is determined mostly by whether the work concerned is routine, and not so much whether it is manual or white-collar [Eco17]. Machines can already do many forms of routine manual labor, and are now able to perform increasingly complex routine cognitive tasks too. Self-driving cars, Jeopardy! champion supercomputers[2], world champion Go computers[3], and a range of useful robots have all appeared just in the past few years. And these innovations are not lab demos; they are demonstrating their skills in the messy real world. These developments contribute to the feeling that we are at an inflection point where many technologies that used to be found only in science fiction are becoming everyday reality.

Managerial jobs are not insusceptible to this trend either. Many of the routine activities performed by managers are assisted by computers, and our data-driven society is playing an important role in this development. While initially exclusive to analytics experts, tools for data analysis to support decisions are becoming rapidly accessible to the masses. IBM Watson Analytics is just one example. Such systems are generally called *decision support systems* or decision-making software. Decision support systems serve the management, operations, and planning levels of an organisation (usually middle and higher management) and help people make decisions about problems that may be rapidly changing and not easily specified in advance. Decision support systems can be either fully computerized, human-powered or a combination of both.

The ultimate goal of this thesis is a decision support system for enterprise architects. Although decision support systems have found their way in many fields such as information architecture, software architecture, and business informatics, they have not done so much for enterprise architecture. Before we explain why this is the case, we first briefly explain enterprise architecture.

We use the following definition of an enterprise architecture:

> "Those properties of an enterprise that are necessary and sufficient to meet its essential requirements" [GP11]

A commonly used metaphor for an enterprise architect is a city planner. City planners work on long-term visions, providing the roadmaps and regulations that a city uses to manage its growth and provide services to citizens. Using this analogy, we can differentiate the role of the system architect, who plans one or more buildings; software architects, who are responsible for the HVAC (Heating, Ventilation and Air Conditioning) within the building; network architects, who are responsible for the plumbing within the building, and the water and sewer infrastructure between buildings or parts of a city. The enterprise architect however, like a city planner, both frames the city-wide design and other activities into the larger plan.

There are a large number of responsibilities and skills that can potentially be associated with an enterprise architect. One way to frame these responsibilities and skills is to distinguish two main roles of an enterprise architect (Figure 1.1)

1. *The engineer.* The architect (usually a group of architects) develops models of business and IT. These models can be UML-like diagrams, specialized enterprise

[2] http://www.techrepublic.com/article/ibm-watson-the-inside-story-of-how-the-jeopardy-winning-supercomputer-was-born-and-what-it-wants-to-do-next/

[3] https://www.wired.com/2016/03/googles-ai-wins-fifth-final-game-go-genius-lee-sedol/

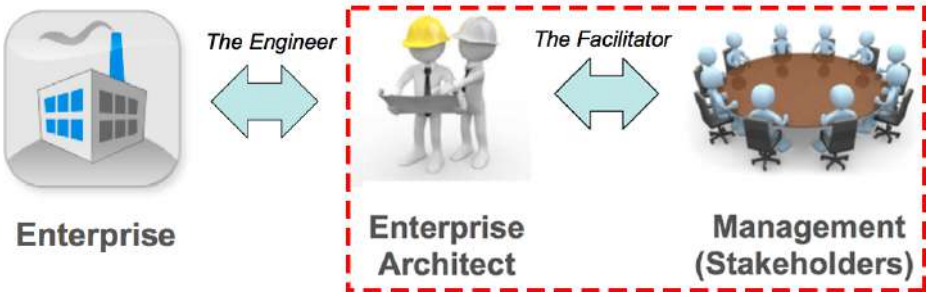


Figure 1.1: Two roles of an enterprise architect, and our scope (red dashed line)

architecture diagrams, risk analysis tools, textual descriptions, or any other representation that the architect feels comfortable with.

2. *The facilitator.* The architect (usually the lead architect) is intermediator between IT and business. Often, the architect attends business meetings and serves as an IT expert, consulted by managers on what specific IT solutions to use.

Remark 1. *We use the terms* management*,* managerial board*,* board*, and* stakeholders *interchangeably throughout this thesis to refer to the group of persons who an enterprise architect advises. While the management and stakeholders may technically be different persons, we do not make this distinction here, since it is not the main focus of our research.*

Enterprise architects often work on large projects with a long duration, and try to steer the enterprise such that the long-term goals and visions of the enterprise are reached, with an emphasize on the IT part of the enterprise. Let us illustrate this through a simple example.

Example 1.1 (University of Luxembourg)**.** *The University of Luxembourg is in the process of merging three separate campuses into a single campus on a new location. This process takes several decades, and requires a complete re-design of the IT landscape. The university would like to ensure that their long-term strategy and vision is aligned with their overall IT strategy, so they hire a team of enterprise architects. The lead architect discusses and refines the strategic goals and vision of the university with the executive board. She then works together with the other architects to develop a long-term IT strategy. This plan involves modeling the current business-IT landscape in ArchiMate, a specialized enterprise architecture modeling language, performing risk analysis on various alternatives, etc. The board, not having expertise nor time to understand the technical details of this problem, are then presented a simplified version of these plans. The members of the board have different – and changing– concerns, discuss these plans with each other and with the architect, who then brings this input back to his team of architects. This process may be repeated any number of times.*

Although the example above is (purposely) simplified, it does give an idea of the large number of varying tasks an enterprise architect should be able to carry out. The enterprise architect is responsible for ensuring the IT strategy and planning are aligned with the company's business goals, and must optimize information management through understanding evolving business needs (*facilitator*), but it also must ensure projects do not duplicate functionality or diverge from each other, and work with solution architects to

provide a consensus based enterprise solution that is scalable and adaptable (*engineer*). As a result, an enterprise architect should have a large number of skills, from technical skills such as comprehensive knowledge of hardware, software, application, and systems engineering, to soft skills such as communication skills and the ability to explain complex technical issues in a way that non-technical people understand it, to managerial skills such as project and program management planning, time management and prioritization.

Currently, there is little research on enterprise architecture decision support, and in particular, there is little support for documenting enterprise architecture decisions in a structured way. We provide an abstract overview of the evolution of decision support for enterprise architecture in Figure 1.2. In the early days, that is, in the time before desktop computers were available to the common man, the tasks of enterprise architects were done on paper (left image, coaster). At the next stage of the evolution, the introduction of the desktop computer made available general tools such as Microsoft Powerpoint (second image from the left, Powerpoint). As the field of enterprise architecture matured, specialized languages and tool support were developed for enterprise architects, such as the ArchiMate language [IJLP12] (third image from the left, ArchiMate). In this thesis, we aim to lie the foundations for the next step (right image, question mark).

Figure 1.2: Evolution of enterprise architecture decision support. From coasters (left), to Powerpoint (second left), to ArchiMate (second right), to the future (right)

We aim to contribute to the development of enterprise architecture decision support by focusing on reasoning about high-level decisions. This is a crucial aspect of enterprise architecting, both for the *engineer* as well as the *facilitator* role. In the role of the facilitator, management may have vague and imprecise goals and wishes, which have to be translated into a concrete IT strategy by the architect. This requires management and the architect to develop a high-level plans and to have a shared conceptualization of the problem. Management generally has limited time to discuss with the architect, so the architect should focus on the most important elements when explaining a solution, and make it as simple as possible. This enables management to understand the solutions well enough in order to make an informed decision. In the engineer role, the architects make plans with each other as well as with other architects, such as software architects, information architects, or system designers.

In this thesis we focus on the *facilitator* role (Figure 1.1, red dashed line). This means we focus on reasoning processes of the enterprise architect (the facilitator) and management. Our aim is to support these processes by developing logical frameworks that store important commitments, or high-level decisions. Such decisions are based on underlying assumptions. Assumptions may pertain to the goals of stakeholders, strategic directions of the enterprise, architecture principles, requirements, arguments put forward in discussions, etc. In practice, enterprises are confronted with frequent changes and challenges to these assumptions. Even more, the assumptions, and their relative priority,

also depend on the specific stakeholders that are involved in creating the architecture of the future enterprise, as well as the actual transformation.

The state of the art research in enterprise architecture is diverse with many different definitions emphasizing different parts of the field. Therefore, the first part of this thesis is concerned with understanding the field of enterprise architecture better by identifying important characteristics of enterprise architecture. In part two and three we then focus on a few of those characteristics in more detail, and we develop logics and languages to formalize them. In the second part we focus on the process of refining high-level goals and values into more specific goals and tasks, and linking these with underlying discussions between the architect and management. In the third part we focus on storing enterprise architecture plans consisting of commitments and underlying assumptions in databases. By storing these elements, the databases serve as guiding instruments steering discussions.

1.2 Part 1: Characteristics of Enterprise Architecture

1.2.1 Background

There are a large number of definitions of enterprise architecture in existence. The definition we use here is by Greefhorst and Proper [GP11]: "*those properties of an enterprise that are necessary and sufficient to meet its essential requirements*". Greefhorst and Proper provide an overview of some of the key definitions of enterprises architecture, while demonstrating that the above definition captures the essence of architecture. They also argue that, in defining this concept, a distinction can be made between the purpose, the meaning and the elements of an enterprise architecture:

Purpose The main purpose of an enterprise architecture is to align an enterprise to its essential requirements. As such, it should provide an elaboration of an enterprise's strategy to those properties that are necessary and sufficient to meet these requirements. These properties will impact the design of the enterprise, and enable the steering and coordination of transformation programmes and projects.

Meaning Given that the main purpose of an enterprise architecture is to align the design of an enterprise to its strategy, the essential meaning of an enterprise architecture is that it provides a normative restriction of design freedom towards transformation projects and programmes (or put more positively: a reduction of design stress).

Elements In the current understanding of the field, key concepts in the field of enterprise architecture include concerns, architecture principles, models, views and frameworks, while these elements are used to express the enterprise as a system in terms of its components and their mutual relationships, and the relationships to its environment.

The many existing definitions of enterprise architecture each have their own focus with regards to purpose, meaning, and elements. For example, the definitions of enterprise architecture provided by IEEE [The00], TOGAF [The09b], ArchiMate [IJLP09] and Giachetti [Gia10], tend to focus on the elements of an enterprise architecture. The Zachman framework [Zac87] and the GERAM framework [BNS03] also have their focus more

on the possible elements (in particular the relevant viewpoints) of an enterprise architecture. Dietz and Hoogervorst [Die08, Hoo09] define enterprise architecture primarily in terms of its meaning. The definition provided by Ross et al. [RWR06] touches both on the purpose and elements aspects, while Op 't Land, et al. [OPW$^+$08] put the focus more on its purpose as a means for informed governance for enterprise transformations.

In line with the different definitions, various languages and techniques for enterprise architecture have been developed in the last decades. In the ArchiMate project [Lan05b], a language for the representation of enterprise architectures was developed, together with visualisation and analysis techniques. The resulting ArchiMate language is an Open Group standard [IJLP09, The09b], and the TOGAF/ArchiMate combination of standards is playing an increasing role in the marketplace [WS10]. An alternative for ArchiMate is the Unified Profile for DoDAF/ MODAF (UPDM) [Obj12] from the OMG, which is based on the USA Department of Defence Architecture Framework (DoDAF) and the UK Ministry of Defence Architecture Framework (MODAF) respectively.

Initial research for the capturing of architectural design decisions was done during the ArchiMate project as well [Lan05b, VHP04]. In [PO10] different lines for reasoning that can be followed in rationalizing architectural decisions are explored in terms of a real world case study. These lines of reasoning include: motivation of stakeholders, abstraction of implementation details, abstraction of construction (black-box / white-box), different aspect systems (business, information, ...) and their dependence, the planned / expected future evolution, and the design horizon with which decisions are made (strategic / tactic / operational). Following this preliminary work, Plataniotis et al. [PdKP13a, PdKP13c, PDKP14a] develop a framework for the capturing and rationalisation of enterprise architecture decisions called EA Anamnesis. This framework formalizes enterprise architecture decisions through meta models.

There are a number of studies investigating how enterprise architecting (or parts of it) is done, but most are limited to understandings of the authors themselves from prior or concurrent industrial roles (cf., [KAV05]), anecdotal evidence gathered in industrial cases (cf. [vGvD14]), specific frameworks (cf. [DPvdMM14]), or single companies (cf. [SR07, NMD14, Nie07]. While such studies are useful for their insights, their are all limited to a specific scope, and it is not directly clear how these findings compare.

Concluding this brief overview, there are various definitions and language of enterprise architecture around, each focusing on different aspects of the practice, and there is some research on understanding the reasoning of architects through case studies or specific empirical research. However, enterprise architecture is a broad field, and it is not directly clear which aspects of enterprise architecture to focus on when formalizing reasoning of enterprise architects.

1.2.2 Methodology

We visualize the overall methodology and contributions of this thesis as a bridge between artificial intelligence and enterprise architecture (Figure 1.3). This bridge consists of two main elements:

- *Foundations*: There are a large number of techniques in artificial intelligence, and as we have seen above, enterprise architecture is a broad field. The foundation of the bridge determines which parts of artificial intelligence to connect to which

parts of enterprise architecture. In other words, it clarifies which parts of enterprise architecture we focus on, and which techniques from artificial intelligence we use to support those parts.

- *Bricks*: Once the foundation is there, the actual bridge can be built. Each brick is a contribution lying somewhere in between the two fields. We start laying our bricks from the direction of enterprise architecture, moving towards artificial intelligence. As a consequence, the chapters in this thesis will become increasingly more technical and focused towards artificial intelligence research, and less tied to specific applications and tools from enterprise architecture.

Figure 1.3: We visualize the methodology, key results, and dependencies between the three parts of this thesis as a bridge that is being built between artificial intelligence and enterprise architecture.

The methodology of this first part of the thesis is to develop the *foundations* of the bridge, and lay the first brick. In this way, we aim to kill two birds with one stone: We aim to contribute to the state of the art in enterprise architecture ("laying the first brick"), and in the process we aim to learn something about which characteristics of enterprise architect reasoning to focus on ("laying the foundations").

We perform two case studies from which we learn aspects of high-level decisions of enterprise architecture. In the first case study, we conduct an empirical study among enterprise architects focusing on how they make decisions and which elements they deem most important. In the second one, we formalize a recently developed framework for enterprise architecture decision rationalisation called *EA Anamnesis* [PdKP13a, PdKP13c, PDKP14a] using first-order logic. In this second study, it is our goal to understand the tools that enterprise architects use by formalizing them, and whether it is straightforward to support their work with an elementary formalization, or whether additional, more advanced, knowledge representation techniques are required.

1.2.3 Research questions

RQ 1. Which aspects of enterprise architect reasoning about high-level decisions can be supported by decision support systems?

This breaks down into the following two sub-questions.

RQ 1.1. Which aspects are characteristics of enterprise architect reasoning about high-level decisions?
RQ 1.2. Which of these aspects will be considered in this thesis?

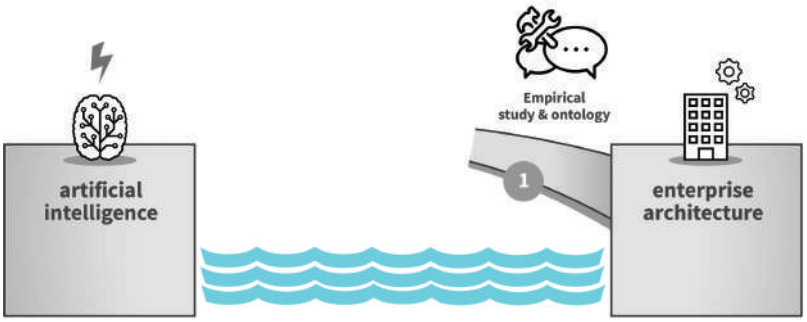


Figure 1.4: The first brick of the bridge: Investigating the relative characteristics of enterprise architect reasoning by formalizing EA Anamnesis and performing an empirical study.

The result of answering these two subquestions forms the foundation of our bridge and the first brick (Figure 1.4), consisting of our two case studies *Empirical study* and an *Ontology*. These two studies are divided into two separate chapters.

In Chapter 2, we report on an empirical study on how the practice of high-level decision making (i.e., decisions in the role of the *facilitator* of Figure 1.1) in enterprise architecture is perceived by professional enterprise architects. We do so through a questionnaire incorporating qualitative and quantitative questions, targeting enterprise architects around the world, in order to determine what they consider to be the important characteristics of enterprise architecture decision making, and whether these characteristics differ considerably from those in closely related fields such as software architecture.

The most important characteristics of enterprise architecture we found are:

1. Translating strategic goals into an IT strategy
2. Communicating plans of action
3. Explaining decisions instead of making them
4. Qualitative before quantitative data
5. Stronger business focus than other disciplines
6. Politics, emotions, and soft skills play a bigger role than in other disciplines

7. Large number of stakeholders with conflicting views

8. Highly uncertain plans in a changing environment

We use these eight characteristics as yardsticks for a formal theory to support enterprise architect reasoning. We observe that approaches based on the idea of *classical rationality* may be less appropriate that those based on *bounded rationality*, which is motivated by the observation that our study shows architects often work with incomplete data and face many types of uncertainty. We propose to use logical theories based on practical reasoning, since such theories have rich concepts for motivational attitudes such as goals and intentions, which appear to be playing an important role.

In Chapter 3 we report on our second case study, in which we formalize EA Anemnesis, a state of the art framework for enterprise architecture decisions rationalization, using first-order logic. We apply the formal framework to ArchiSurance, a well-known illustrative case in the field of enterprise architecture. Our conclusion is that our resulting formal framework, as well as EA Anamnesis, do not consider many of the characteristics we identified before, which are important for enterprise architect reasoning about high-level decisions. For instance, motivational attitudes such as goals are not part of the framework (characteristic 1), different viewpoints of the stakeholders are not taken into account (characteristic 2, 6, 7), and there is no way to cater for various types of uncertainty (characteristic 8).

In order to develop more appropriate support for reasoning about high-level decisions in enterprise architecture, we use various knowledge representation techniques from artificial intelligence research in the next two parts, focusing respectively on goals in Part 2, and planning and scheduling in Part 3.

1.3 Part 2: Goals

1.3.1 Background

Goal modeling

In the conclusion of the previous part we recognized *translating the strategic goals of the enterprise into an IT strategy* as a key activity (characteristic 1). In this process, mainly qualitative data is used (characteristic 2, 3, 4) and there are a large number of stakeholders with conflicting views (characteristic 7).

Translating strategic goals into an IT strategy falls under the the “early-phase” requirements engineering activities of an information system, which include those that consider how the intended system should meet organizational goals, why it is needed, what alternatives may exist, what implications of the alternatives are for different stakeholders, and how the interests and concerns of stakeholders might be addressed [Yu97b]. This is generally referred to as *goal modeling*. Given the number of currently established methods using goal models in the early stage of requirements analysis (e.g., [LY04, Don04, DVLF93, CNYM12, CKM02], overviews can be found in [VL01, KL05]), there is a general consensus that goal models are useful for this purpose. Several goal modeling languages have been developed in the last two decades. The most popular

ones include $i*$ [Yu97a], Keep All Objects Satisfied (KAOS) [vL08], the NFR framework [CNYM12], TROPOS [GMS05], the Business Intelligence Model (BIM) [HBJ+14], and the Goal-oriented Requirements Language (GRL) [AGH+10], which is part of an ITU-T standard, the User Requirements Notation (URN) [ITU08].

A goal model is often the result of a discussion process between a group of stakeholders and an enterprise architect. For small-sized systems for which goal models are constructed in a short amount of time, involving stakeholders with a similar background, it is not often necessary to record all of the details of the discussion process that led to the final goal model. However, enterprise architecture systems - i.e., large-scale IT infrastructures- are complex and are not constructed in a short amount of time, but rather over the course of several meetings. In such situations, failing to record the discussions underlying a goal model in a structured manner may harm the success of the requirement engineering phase of enterprise architecture for several reasons:

1. It is well-known that stakeholders' preferences are rarely absolute, relevant, stable, or consistent [Mar78]. Therefore, it is possible that a stakeholder changes his or her opinion about a modeling decision in between two goal modeling sessions, which may require revisions of the goal model. If previous preferences and opinions are not stored explicitly, it is not possible to remind stakeholders of their previous opinions, thus risking unnecessary discussions and revisions. As the number of participants increases, revising the goal model based on changing preferences can take up a significant amount of time.

2. Other stakeholders, such as new architects on the team, who were not the original authors of the goal model, may have to make sense of the goal model, for instance, to use it as an input in a later stage or at the design and development phase. If these users have no knowledge of the underlying rationale of the goal model, it may not only be more difficult to understand the model, but they may also end up having the same discussions as the previous group of stakeholders [CKM02].

3. Alternative ideas and opposing views that could potentially have led to different goal diagrams are lost. For instance, a group of stakeholders specifying a goal model for a user interface may decide to reduce goals "easy to use" and "fast" to one goal "easy to use". Thus, the resulting goal model will merely contain the goal "easy to use", but the discussion as well as the decision to reject "fast" are lost. This leads to a poor understanding of the problem and solution domain. In fact, empirical data suggest that this is an important reason of requirment engineering project failure [CKI88].

4. In goal models in general, the rationale behind any decision is static and does not immediately impact the goal models when they change. That is, it is not possible to reason about changing beliefs and opinions, and their effect on the goal model. A stakeholder may change his or her opinion, but it is not always directly clear what its effect is on the goal model. Similarly, a part of the goal model may change, but it is not possible to reason about the consistency of this new goal model with the underlying beliefs and arguments. This becomes more problematic if the participants constructing the goal model change, since modeling decisions made by one group of stakeholders may conflict with the underlying beliefs put forward by another group of stakeholders.

1.3.2 Methodology

In this part we lay the second brick of our bridge between artificial intelligence and enterprise architecture. As we mentioned before, our contributions will become increasingly closer to artificial intelligence, and will increasingly move further away from enterprise architecture. The second brick is in between artificial intelligence and enterprise architecture, and therefore will use concepts and techniques from both areas.

Motivated by the characteristics we identified in the previous part, our specific research goal is to apply techniques from artificial intelligence to the goal modeling process, with the expectation that doing so will help resolve issues 1-4 above. We identified several important requirements for our framework:

1. It must be able to formally model parts of the discussion process in the early-requirements phase of an information system,
2. It must be able to generate goal models based on the formalize discussions,
3. It should have formal traceability links between goal elements and arguments in the discussions.

The first requirement might seem imprecise, since it does not specify exactly which parts of the discussion process the framework should capture. However, when taking the second requirement into account, it should be clear that we should at least capture the parts of the discussions that are related to goal models.

1.3.3 Research questions

The overall research question of this part is as follows:

> **RQ 2** Which aspects of enterprise architecture reasoning and goal modeling can be supported by decision support systems?

Given our background discussion and methodology, this breaks down into the following sub questions.

> **RQ 2a** How to formalize the discussions between stakeholders about goal models?
> **RQ 2b** How to generate goal models based on the formalized discussions?

The result of answering these two subquestions forms the second brick of our bridge (Figure 1.6). In Chapter 4 we develop a framework for argumentation and goal modeling called *RationalGRL*. In order to formalize discussions of the early requirements phase of an information system we use *argument schemes* and *critical questions*. We develop a set of argument schemes and critical questions by analyzing transcripts containing discussions about the architecture of an information system. We then develop a formal language for these argument schemes and critical questions, and we develop algorithms to translate argumentation frameworks to goal models.

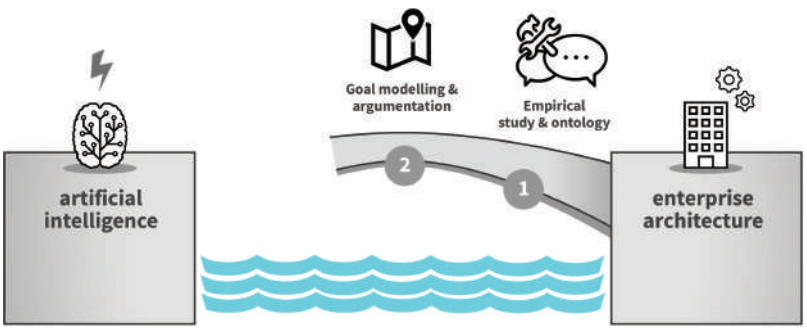


Figure 1.5: The second brick of the bridge: Goal modeling and argumentation.

1.4 Part 3: Planning and Scheduling

1.4.1 Background

One of the earlier practitioners in system architecture Steven H. Spewak defined *enterprise architecture planning* as "the process of defining architectures for the use of information in support of the business and the plan for implementing those architectures." [SH93]. The results of our empirical study in Part 1 show that plans are being used for communication in enterprise architecture (characteristic 2), and that those plans have a high level of uncertainty (characteristic 8). This is not surprising, since enterprise architecture project are often long-term (see Example 1.1). In fact, an important lesson from the ArchiMate project [Lan05b] was that it is inherently difficult to plan architectural design. TAFIM, an enterprise architecture model by and for the United States Department of Defense recommends that in a typical five-year plan, only the first year is detailed, and the other steps are described only in a very abstract way. At each step in the plan, not only must the future abstract plans be further detailed, but architectural designs also have to be reconsidered and possibly revised.

As we already mentioned in Section 1.2.3, we argue that approaches based on *bounded rationality* are suitable to support enterprise architecture planning. Bounded rationality is the idea that the rationality of a decision maker is limited by the tractability of the decision problem, the cognitive limitations of the mind, and the time available to make the decision. Decision-makers in this view act as *satisficers*, seeking a satisfactory solution rather than an optimal one.

In artificial intelligence research, planning research throughout the 1970s and early 1980s was dominated by STRIPS-style classical planning approaches [THA94]. These approaches focused on algorithms for automatic plan generation, that would take as input a specification of the current world state, a goal to be achieved and the actions available to an agent, and would produce as output a plan to achieve the goal state. This style of planning, it was believed, is a central component in rational action. However, already by the mid 1980s a number of researchers, of whom Rodney Brooks is probably the best known [Bro99], began to claim that classical planning approaches were fundamentally flawed, for both pragmatic and philosophical reasons. From a pragmatic point of view, STRIPS-style planning algorithms tend to be computationally intractable, rendering them of limited value to agents that must operate in anything like real-time

environments [Cha87, Byl94]. From a philosophical point of view, it was argued that much of what we regard as everyday intelligence does not arise from abstract deliberation of the kind involved in STRIPS- style planning, but from the interaction between comparatively simple agent behaviors and the agent's environment.

Philosophers often use the concept of *intention* when speaking about plans. They have drawn a distinction between future-directed intentions and present-directed ones. The former guide an agent's planning and constrain their adoption of other intentions, while the latter function *causally* in producing behavior. For example, an architect's future-directed intentions may include talking with the stakeholders tomorrow, and her present-directed intentions may include modifying a presentation now. Bratman [Bra87] argues that intending to do something (or having an intention) plays an important role in an agent's planning: they are high-level plans to which the agent is committed and that she refines step by step, leading to intentional actions. Intentions therefore play a role that is intermediate between goals, plans, and actions. Rationally speaking, an agent is more deeply *committed* to its intentions, e.g. than to its beliefs. If we think of intentions as fitting into more complex plans, then reconsidering an intention potentially requires revising an entire plan, which can be computationally expensive and therefore problematic in real-time. Another aspect of this *relative resistance to revision* is that intentions, and the plans into which they fit, can be very useful for bounded agents who need to select appropriate actions at different times. Both by adopting appropriate policies that can apply at different times to relevantly similar decision problems, and by devising complex plans that can be easily followed at some future time when computation and time may be limited, an agent with intentions may be able to avoid having to perform complex calculations every time a new decision problem arises.

Inspired by the philosophical literature, in the 1990s artificial intelligence researchers started to investigate intention. The first full-fledged BDI paper, with formal details and some results, was the seminal paper by Cohen and Levesque [CL90a]. They laid out a formal logical language, including modal operators for mental states like "belief" and "having as a goal", for temporal expressions, and for descriptions of events such as "A happens" or "x did action a". Although widely cited, the formalism has received its fair amount of criticism as well (e.g., [Sin92a]), and has been revisited various times (e.g., [HL04]). Rao and Georgeff (e.g., [RG91]), offered an alternative, and arguably simpler, formalism based roughly on Computational Tree Logic (CTL). They do not define intentions out of more basic states, and they are also able to handle some of the motivating puzzles more adequately.

Recently, Herzig *et al.* [HLPX16] recognize three fundamental problems with existing BDI theories. One of these problems is that practical and formal approaches evolved separately and neither fertilized the other. More specifically, they argue that BDI is poorly connected to other fields such as automated planning. This is remarkable indeed, given that philosophers regard future-directed intentions as important elements for planning. Shoham [Sho09, Sho16] recognizes this as well, and proposes a more "computationally grounded" approach towards intention revision, which he calls the *database perspective*. The main idea of this approach is to store attitudes such as beliefs and intentions in separate databases, which are in service of some planner. The planner can add or remove entries from the database, while the database should remain internally, and jointly, consistent.

1.4.2 Methodology

In this last part of the thesis we lay the last brick of our bridge from artificial intelligence to enterprise architecture. It is the bridge closest to artificial intelligence, and is therefore considerably more technical than the other parts.

Recall from Section 1.1 that it is our aim to support enterprise architect reasoning by developing a logical framework that stores important commitments made during discussions or meetings. These commitments are based on underlying assumptions, which can be goals of stakeholders, strategic directions of the enterprise, architecture principles, etc. We view a plan abstractly as a sequence of commitments in time, and each commitment in the plan may come with a number of underlying assumptions. If these underlying assumptions change, then parts of the plan may require revision, which in turn may invalidate other parts of the plan, and so on. Therefore, assumptions have an inherently *non-monotonic* character: they are assumed to be true, unless it becomes clear they are false. This is related to the resource-boundedness of our problem domain: an enterprise architect cannot always know all of the assumptions, especially for long term plans.

The methodology of this part is to formalize the dynamics commitments in time with underlying assumptions in a BDI logic. We aim to use a BDI formalism to specify the dynamics of the commitments and assumptions. This will allow the system to serve as a kind of *intelligent calendar*, helping enterprise architects and stakeholders to remember which specific things they have committed themselves to, and whether it is consistent to add new commitments, or whether underlying assumptions are violated.

1.4.3 Research question

> **RQ 3** Which aspects of enterprise architecture planning will be considered in this thesis?

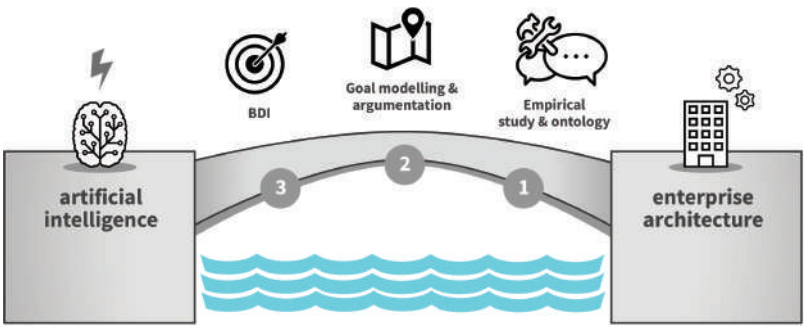


Figure 1.6: The third brick of the bridge: Focusing on planning and scheduling by developing a BDI logic.

Following Shoham's database perspective, we view enterprise architecture planning as a database management problem (see Figure 5.1). In our system, an enterprise architect is in the process of making plans, possibly with a group of stakeholders, and stores commitments and beliefs in two database.

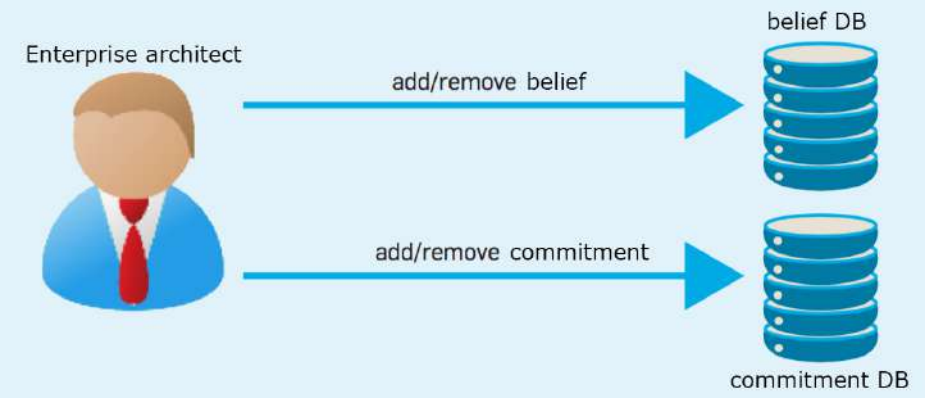


Figure 1.7: We view consistency of commitments and beliefs as a database management problem.

We focus on two main sources for the databases to change:

1. The enterprise architect forms a new belief, e.g. from discussions with stakeholders, or from a piece of data. If the new belief is inconsistent with the existing beliefs, these beliefs will have to be revised to accommodate it. We give general conditions on a single revision with new information that the database has already committed to incorporating using ideas from the classical AGM postulates [AGM85] approach.

2. The enterprise adds a commitment. We formalize these tasks as *future directed atomic intentions*, understood as time-labeled actions pairs (a,t) that might make up a plan. It is assumed the enterprise architect has already committed to the intention, so it must be accommodated by any means short of revising beliefs. The force of the theory is in restricting how this can be accomplished. The job of the database is to maintain consistency and coherence between intentions and beliefs.

We divide the contribution of this part over two chapters. In chapter 5 we develop a logic for beliefs about actions in time. We associate pre-and postconditions with actions. A key element in our approach is the asymmetry we put on assumptions about preconditions and postconditions of actions. First of all, we assume that

> *If an enterprise architect intends to do an action, she assumes the consequences of this action hold.*

However, for preconditions we add a weaker requirement:

> *If an enterprise architect intends to do an action she cannot believe that its preconditions do not hold.*

This requirement is sometimes called *strong consistency*, and is weaker than Bratman's *means-end coherence* requirement [Bra87]. The result of this weakened requirement is that preconditions of actions are treated as *assumptions*: An autonomous agent forms intentions under the assumption that these preconditions will be made true somewhere

in the future. The main technical result of this part is that we axiomatize our logic and prove it is sound and strongly complete with respect to our semantics.

In Chapter 6 we then study the revision of intentions and beliefs. We develop a set of postulates in AGM style, and the main technical result is that we prove the Katsuno-Mendelzon representation theorem. To this end, we define a revision operator that revises beliefs up to a specific time point. We show that this leads to models of system behaviors which can be finitely generated, i.e. be characterized by a single formula.

1.5 Thesis outline and publications

As mentioned earlier, this thesis contributes to state of the art research in both enterprise architecture and artificial intelligence. However, different research fields use different research methodologies. In fields related to the enterprise architecture side of this thesis (such as information systems, requirements engineering, or business informatics), frameworks and theories are usually validated through empirical research, which can for instance be case studies, interviews, or user tests. These theories are often developed in iterations, using methodologies such as *design science* or *grounded theory*. In fields related to artificial intelligence side of this thesis (such as knowledge representation and reasoning), empirical validation is less crucial, and results are often theories or frameworks that have certain provable properties.

Part	**Research fields**	**Output**	**Evaluation**
Part 1	Enterprise architecture, Information systems, Software architecture, Decision theory	Logical framework, Empirical study	Illustrative case
Part 2	Requirement engineering, Goal modeling, Argumentation	Logical framework, Algorithms	Transcripts
Part 3	BDI logic, Belief revision, Temporal logic	Logical framework, Completeness result, Representation result	Formal proofs

Table 1.1: Relevant research fields, main results, and evaluation methods of the three parts in this thesis

As a result, the different parts of this thesis have different outputs and evaluation methods, depending on what audience they are written for. We list the related fields, main results, and evaluation methods of the three parts in Table 1.1. Moreover, we briefly summarize each chapter below.

- **Chapter 2: Enterprise Architects High-Level Decision Making: an Empirical Study**

  We report on an empirical study on how high-level decision making is perceived by professional enterprise architects. Our study consists of a questionnaire incorporating qualitative and quantitative questions, and is answered by about 30

enterprise architects. The outcomes is a list of characteristics of enterprise architecture, which we use as a starting point of the remainder of the thesis.

This chapter is based on joint work with Dirk van der Linden [vdLvZ15].

- **Chapter 3: A Logical Framework for EA Anamnesis**

  We analyze a recent framework for capturing enterprise architecture design decisions called *EA Anamnesis* and recognize various ambiguities and flaws in the specification. We propose a more precise formalisation of EA Anamnesis using first-order logic. Our main conclusion is that our formalism does not offer much support for the type of reasoning processes specific to enterprise architecture we found in the previous chapter. More notably, reasoning about goals, considering various viewpoints of stakeholders, and reasoning about plan dynamics are lacking.

  This chapter is based on joint work with Georgios Plataniotis, Diana Marosin, and Dirk van der Linden [vZPvdLM14].

- **Chapter 4: RationalGRL: A Framework for Argumentation and Goal Modeling**

  We investigate to what extent argumentation techniques from artificial intelligence can be applied to goal modeling. We develop argument schemes and critical questions for goal modeling by analyzing transcripts. We formalize the argument schemes in logic and develop algorithms for the critical questions, we call our framework *RationalGRL*.

  This chapter is based on joint work with Floris Bex, Sepideh Ghanavati, and Diana Marosin [vZG14, MvZG16, vZMBG16, vZMGB16]

- **Chapter 5: A Logic for Beliefs about Actions and Time**

  We develop a logic for temporal belief bases, containing expressions about temporal propositions (tomorrow it will rain), possibility (it may rain tomorrow), actions (the robot enters the room) and pre- and post-conditions of these actions. This logic is motivated by Shoham's database perspective.

  This chapter is based on joint work with Dragan Doder, Mehdi Dastani, and Leon van der Torre [vZDDvdT15a, vZDDvdT15b]

- **Chapter 6: The Dynamics of Beliefs and Intentions**

  We introduce intentions to our logic and we formalize a coherence condition on beliefs and intentions. In order to do this we separate strong beliefs from weak beliefs. We provide AGM-style postulates for the revision of strong beliefs and intentions: strong belief revision may trigger intention revision, but intention revision may only trigger revision of weak beliefs. After revision, the strong beliefs are coherent with the intentions. We show in a representation theorem that a revision operator satisfying our postulates can be represented by a pre-order on interpretations of the beliefs, together with a selection function for the intentions.

  This chapter is based on joint work with Dragan Doder, Mehdi Dastani, and Leon van der Torre [vZDSvdT14, vZ15, vZD16]

- **Chapter 7: Conclusions**

  We draw conclusions and summarize the contributions of this thesis by answering the research questions.

- **Appendix: Proofs**

  The appendix consists of five parts. First, we provide the design instructions used in the transcripts of Chapter 4. Secondly, we provide excerpts of those transcripts. Thirdly, we provide a specification of a goal model in our formal language, which also comes from Chapter 4. Fourth, we provide full proofs of all the main theorems of Chapter 5, and fifth, we provide some simulation results of initial future work of Chapter 6.

Part I

Characteristics of Enterprise Architecture

2

Enterprise Architects High-Level Decision Making: an Empirical Study

Abstract In this chapter we report on an empirical study on how high-level decision making is perceived by professional enterprise architects. Our study consists of a questionnaire incorporating qualitative and quantitative questions, targeting enterprise architects. The result is a list of characteristics of enterprise architecture high-level decision making, including goal-oriented decision-making, the importance of communication, making decisions with much uncertainty and changing environments, emotional decision-making, high level of politicization, and subordination to business management. We suggest approaches based on bounded rationality are suitable to formalize these characteristics.

2.1 Introduction

There are large number of theories and frameworks for enterprise architecture (see previous chapter), but these frameworks rarely document design decisions and underlying assumptions explicitly. This is surprising, given the importance of decisions in related disciplines. For instance, in the domain of software architecture, Tyree and Akerman [TA05] recognized that architecture decision capturing plays a keys role in what they call "demystifying architecture". They stress that architecture decisions should have a permanent place in the software architecture development process. Moreover, it facilitates traceability from decisions back to requirements and it provides agile documentation, which is crucial in an agile process where a team does not have the time to wait for the architect to completely develop and document the architecture.

The empirical study we report on in this chapter aim to take a first step into this direction by contributing to the following aspects:

1. *Developing an ontology for enterprise architecture high-level decision making*. As a first step towards providing support for decision rationalisation in enterprise architecture, it is useful to determine an ontology: What are the elements we have to consider, what are the relationships between these elements? While many characteristics of enterprise architecture are common knowledge among practitioners, they are less explicitly available in literature. At the very least, our results can be used as a validation for existing (and new) work aiming to support architects by showing that the problem they address is one important to practitioners, or helping such work to consider additional elements.
2. *Understanding the issues enterprise architects face in decision making*. Both by

explicitly asking what aspects practitioners perceive to be most critical during their decision making process and investigating the characteristics of that process, we can have a more empirically grounded list of focal points for research (and practical) efforts to address. While many issues are stipulated and approached from a theoretical point of view, investigating what issues of decision making are most salient to practitioners in the field might paint a different picture and thus require refocusing of enterprise architecture research efforts.

3. *Making explicit the differences with other fields.* By understanding practitioners' perceptions, we can also investigate how similar and different they perceive decision making in other related fields to be, like for example software or information architecture. For software architecture this is of particular interest, as some frameworks and (design) research efforts in enterprise architecture make implicit assumptions that the two fields are strongly similar. Given the wider scope of enterprise architecture, involving also organizational structures, business processes, value exchanges, and so on, such claims needs proper support. As there is much insight into what software architects do [Kru08] (in contrast to work stipulating what they ought to do [Ber08, McB07]), and recent work has investigated so empirically [SUS14, SH15], drawing comparisons between how similar the fields are experienced by their practitioners becomes a feasible exercise.

Our research objective is to study these aspects and in doing so elicit data that gives insights into the general practice of enterprise architecture as well. We will do so by performing qualitative work with a diverse amount of participants active as enterprise architects. Our specific focus in on *high-level decisions*. This means we focus on the decisions made when the enterprise architect is in the role of *facilitator* (figure 2.1). There clearly are many other types of decisions, but our current focus is on these types of decisions. The reason for this choice is that we expect this role is what sets enterprise architecture apart from other disciplines.

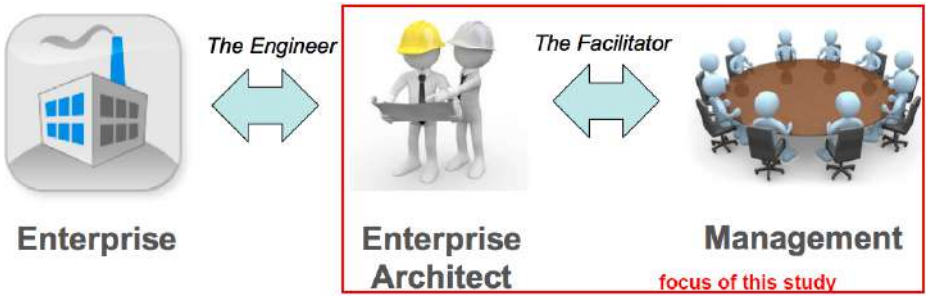


Figure 2.1: Two roles of an enterprise architect from the introduction

Section 2.2 explains the method and details of the empirical study, and an overview of the results is given in Section 2.3. These are discussed in detail in Section 2.4, and their implications for decision support are discussed in Section 2.5. Section 2.6 finally discusses related work, thread to validity, and issues that are left open.

2.2 Methodology

In this section we give an overview of how we gathered the participants, the procedure of the study, used questions, and how we processed the results.

2.2.1 Participants

We specifically targeted enterprise architecture practitioners by posting an invitation to the study on several LinkedIn groups centered around (the use of) enterprise architecture, enterprise architecture methods, or tools (e.g., groups such as *The Enterprise Architecture Network*[1], *TOGAF(R)*[2], *Enterprise Architecture Forum*[3], *Enterprise Architecture*[4], *ArchiMate*[5], *Enterprise Architecture & Processes*[6], and others). Our approach is motivated by other studies explaining how to gather participants from online communities [GT14]. By doing so, it was possible to target a diverse number of participants from both geographical as professional background. This prevented us from having a limited professional context focusing on single companies or geographical areas. Participants were asked to fill out the questionnaire online, and were offered no reward except a copy of the research results, when available.

2.2.2 Procedure

The study consists of a questionnaire with three main parts, which we now discuss in turn.

Professional profile

The profile of the participants was built based upon the following questions, partially inspired by other recent studies investigating similar issues in software architecture difficulties [TGA13].

- What are your main activities as an enterprise architect during the decision making process?
- What modeling languages and techniques do you use?
- Are you internal or external to the company you perform enterprise architecture-related activities for?
- Do you have experience with other architecture fields such as software or information architecture? If so, to what degree do you find the decision making process to be different than in enterprise architecture?

[1] https://www.linkedin.com/groups/36781
[2] https://www.linkedin.com/groups/60545
[3] https://www.linkedin.com/groups/36248
[4] https://www.linkedin.com/groups/54337
[5] https://www.linkedin.com/groups/50758
[6] https://www.linkedin.com/groups/118419

Aspects of decision making (open questions)

In the second part we asked open questions about the difficulties they face in their role as an architect, their involvement and views on design decisions, and what kind of data they use.

- What is your role in the process of making enterprise architecture-related decisions?
- What makes an enterprise architecture decision difficult for you?
- Related to the last question, what are the most important (or critical) aspects of an enterprise architecture decision for you?
- What kinds of input do you use for enterprise architecture decisions, and of those, do you favor qualitative or quantitative data to base your decisions on?

Group vs. individual decision	**Final vs. refined decisions**	**Quantitative vs. qualitative data**
I take a decision by myself	Time constraints do not allow me to consider all decision alternatives	I prefer numerical data to base my decisions on
I take decisions after consulting others	I take a decision without knowing exactly what the outcome will be	I prefer discussions with people to base my decisions on
Decisions are taken by a committee	Decisions often have to be reconsidered, which also affects other decisions	It is easier to make decisions that are based on hard data
Decisions are taken by a group of architects	Decisions are often refined	In general there is sufficient numerical data available to make decisions
The final decision comes down to a single person	When I make a decision, it is final	Discussions with stakeholders offer more insight than numerical data

Table 2.1: Used Likert Scale Statements. Each statement was answered with “strongly disagree” (1) to “strongly agree” (5)

Aspects of decision making (multiple-choice)

In the third and last part, we asked participants to judge to what extent they agreed with a number of statements on a 5 point Likert scale (ranging from ‘strongly disagree’ to ‘strongly agree’). These were created to give insight into how participants feel about the decision making aspects detailed below.

- Whether decisions are group decisions or individual decisions.
- Whether decisions are final or iteratively refined due to time constraints or changes in underlying assumptions.

- Whether the data used for decision making is mostly qualitative (e.g., discussions, opinions) or quantitative (e.g., statistical data).

For each of these three aspects we showed five statements that could each be answered with a number 1-5, ranging from "strongly disagree" (1) to "strongly agree" (5) 2.1.

2.2.3 Analysis method

The results from all open questions were gathered and classified per question. We then used counted occurrences of common threads between participants, both on single word and phrase basis (e.g., multiple occurrences of the term "time constraints" for the question "what makes an enterprise architecture decision difficult?"). This technique was used to build an overview of the general trend for the answers. After doing so we went through the answers again to find answers that specifically conflicted with this trend, and use them to discuss the attitudes of the participants towards the questionnaire.

To estimate the general tendency for each answer in the Likert scale we calculated the median of each question's answers (given the ordinal nature), which we used to determine whether the majority of participants had a polarized (i.e., strong agreement or disagreement) or neutral attitude towards them.

2.3 Results

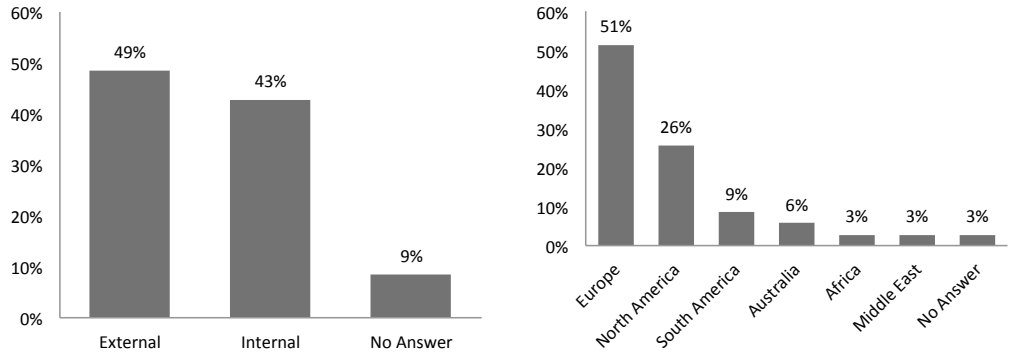
All (anonymized) results of this study can be found on the Github page of this thesis: `https://github.com/marcvanzee/RationalArchitecture/`, in the folder "Ch2 empirical study".

We received **35** full responses to the questionnaire, with many more partial or empty responses discarded. The textual answers were analyzed and coded, and will be discussed in more detail in the next section. There was no strong bias towards external or internal employees, with 17 indicating being external to the companies they provided EA services for, 15 being internal, and the remaining 3 gave no answer. The location of the participants was diverse, with many countries represented. A total of 18 participants were from (> 5) European countries, 9 from North America, 3 from South America, 2 from Australia, 1 from Africa, 1 from Middle East, and finally 1 from an unknown origin. See Fig. 2.2 for an overview.

For the Likert scale, we selected the statements with strong responses (either positive and negative), and emphasized those with a low response variation in their responses (indicating consensus among the participants). These statements are not used as statistically generalizable findings, but as verification for the analysis of the qualitative data, and to ensure they both corroborate each other (table 2.2).

2.4 Analysis

In this section we analyze the results of the participants by describing the dominant views held by participants for the different aspects we studied. We try as much as pos-



(a) Ratio of internal/external

(b) Originating locale

Figure 2.2: Overview of participant information: The ratio of internal/external architects (a) and in which region they operate (b).

Table 2.2: Strongly polarized (≥ 4 and ≤ 2, positive and negative) Likert scale items. Statements in **bold** had particularly low variation and were thus most strongly (dis)agreed on.

Statement	Polarity
I take a decision by myself	
When I make a decision, it is final	Disagree
In general there is sufficient numerical data available to make decisions	
I take decisions after consulting others	
Decisions are taken by a committee	
Time constraints do not allow me to consider all decision alternatives	
Decisions are often refined	Agree
I prefer discussions with people to base my decisions on	
It is easier to make decisions that are based on hard data	
Discussions with stakeholders offer more insight than numerical data	

sible to let the participants speak for themselves, showing their actual responses.

2.4.1 Main activities

Translating strategic goals into an IT strategy Unsurprisingly, there is a clear focus on ensuring the IT strategy is aligned with the company's business goals. Thus, a large portion of the enterprise architect's time is spent on working towards *future* states of the enterprise, less so on the current state (e.g., modeling it, analyzing it). As stated, they spend a lot of time and effort to:

> "Seek the strategy, the strategic goals (qualitative) and objectives (quantitative) and then derive the information required to achieve them."

> "To discover, understand and verify the business goals/objectives as set by upper management."

> "I provide a degree of analysis of our businesses needs and requirements"

> "Gather business strategy & goals, Gather current pain points and limitations (e.g. budget, timelines, organizational barriers)"

> "Relate de content and the message to de strategic goals of the organisation. Describe the implication, risks and alternative."

Communicating plans of action Related to the previous point, enterprise architects often guide discussions with management about the IT strategy.

> "the use of Architecture Artifacts from a variety of disciplines and professions need to be harmonized into a consumable form (easily digestible and understandable) to guide / aid the decision process for a stakeholder (presumably a senior business person) who is responsible for strategy."

> "creating support within the enterprise for a specific solution or specific solution paradigm, so that the decisionmakers are confronted with this paradigm - second opinions about given architectures or decision proposals"

> "communication with stakeholders and 'deciders' investigation and comparison of scenario's"

> "Leading discussions to prepare decision making. In these discussions EA models are used or drawn up. Results are written down in decision memo that list pro's and cons and formulates an advise."

> "Organising collective decision making on proposed changes, based on the proposals and know how available in the organisation."

Enterprise architects seem to concertize the strategic goals, by providing concrete actions to be taken.

> "...and then use those as inputs to model a set of potential courses of action."

> "...providing a recommended course of action if possible"

> "Helping investment decision makers consider alternative future change to their business, and monitoring the impact of the change as its being created and implemented."

Explaining decisions instead of making them This, already in describing their main activities it becomes clear that while enterprise architects work on the future state of an enterprise, there is a clear difference between those who propose (designs, decisions, strategies for) the future of the enterprise, and those who have the power to actually take it there. See, for example:

> "Often I frame the decisions to be made and then propose various options with supporting data. Usually the option that I feel is the best is clear through that data. *However, the senior leaders who own the decisions need to be the ones who actually make it.*" (emphasis added)

> "The Architect is more a facilitator even if one of the stakeholders."

It seems thus that the enterprise architect is not making the strategic decisions, but merely playing the role of an expert or adviser. Sometimes the data only has to be integrated and simplified so it can be understood by management.

> "models and data already exist, and need to be integrated and used towards the future state"

2.4.2 Modeling languages and techniques

When asked about the modeling languages and techniques participants used in their daily work, the whole gamut of languages came by. The usual suspects such as UML, BPMN, ArchiMate (for Western European EAs, at least) were represented, as well as long existing techniques like Flowcharts, IDEF languages, and so on, but just as well less known languages such as IBM and Oracle suites, ScIAM, SAINT, DNDAF, SCOR, RDF, Rummler-Brache, and so on.

Simple diagrams for communication However, more important than *what* language or technique they used was how well models created with it are received:

> "[I favor] simple diagrams, text and tables that can be understood by all stakeholders"

Multiple participants make a distinction between the audience of models and information, and that a distinct purpose followed from that: modeling to capture knowledge, and modeling to communicate knowledge. Practically speaking, very little formal or complicated modeling languages and techniques were actually shown to the business stakeholders when communicating with them:

> "Primary tool for communicating is PowerPoint."

> "...but really powerpoint, excel and visio are more suitable for a non-technical audience."

> "In dialogue with management I do not use modeling languages or techniques."

2.4.3 Qualitative/quantitative data

Qualitative data before quantitative Participants reiterate it is often not possible to obtain quantitative data directly, and even more so, that quantitative data is meaningless without the qualitative data. In other words, it is necessary to clarify the motivation (e.g., strategic goals of the enterprise) before starting quantitative analysis.

> "Kinds of inputs: strategies, directives, policies discussions and workshops with stakeholders I do not have a particular favor"

> "You cannot derive quantitative data unless you know what qualitative goals need to be satisfied (e.g. a strategy is rarely quantitative)."

> "All kinds of input, but mainly discussions with stakeholder and the documentation they take with them."

> "Discussions, theories, case studies are mostly giving qualitative data. Real quantitative data about business processes are scarce in my organisation."

In order to make decisions, architects confirm that both types of data are required. This ranges from quantitative data about the operation of the enterprise, to qualitative data involving the actual people making up the enterprise. Both kinds of data are needed:

However, they also mention quantitative data is the end-goal to base the future state of the enterprise on.

> "You really need both types. Some are hard to quantify but are crucial to future success. Discussions with stakeholders, analyst reports and risk analyses are most often used to compare alternative roadmaps and solutions."

> "Quantitative criteria-based evaluation matrices for the component/solution recommendation decisions; qualitative data for the policy decisions."

> "Ultimately, you should be able to translate all qualitative data into quantitative data. Otherwise, decisions will be based on good intentions and results are extremely hard to measure."

2.4.4 Differences with other architecture fields

Many participants had experience working in other digital architecture fields (e.g, software, information, data architecture). One participant argued that the primary difference between these fields arose simply from the professional culture of their domain:

> "The other disciplines of architecture vary by 'consulting practice' as this is still an emerging and evolving discussion amongst architects and practitioners."

Stronger business focus than other disciplines Participants generally found enterprise architecture to have a broader focus and depth, with the scope and impact of design decisions potentially far greater in enterprise architecture. These differences were often explained by enterprise architecture having a much stronger business focus than comparable fields, from which also a higher abstraction level followed:

> "The 2 main differences between Enterprise and domain specific decision making is that the enterprise level is judgemental as to what are the better components that align with the business goals (i.e. you can't 'prove' it is the 'best' fit) and that the perspective is more holistic and at a higher level of abstraction."

> "The decisions are at much more of a business level in Enterprise Architecture (business capabilities, strategy, etc.). Decisions for software architects are typically much more technical (frameworks, interfaces, etc). [...] they are two different levels of abstraction."

> "EA is socially constructed and is, in this regard, more of a business than a technological discipline."

Politics, emotions, and soft skills play a bigger role than in other disciplines While some participants state that software architecture is not fundamentally different from enterprise architecture (at least in regards to the decision making process), they do showcase the different nature of achieving support for a future state or design, corroborating points made earlier by other participants that enterprise architecture has many more human and 'soft' aspects that need to be dealt with:

> "Fundamentally the decision making process is not different. In software and information architecture, most decision proposals have a high degree of rationality and [a] blue print character. In enterprise architecture more psychological techniques play a role, e.g. creating support by addressing emotions, sharing ideas, creative collaboration etc."

> "EA decision making process has more political, personal etc. influences. Demands more communication and soft-skills. Software architecture decision making is (much) more straightforward fact based."

> "[...] the stakeholders involved in making the enterprise architecture are more likely to include political motives in their decision making process which makes it much more 'fuzzy'. Creating a software architecture for example is in my experience a far more 'objective' process."

This focus on addressing emotions and dealing with people during design come forward quite clearly in responses by participants describing what an Enterprise Architect should be able to do:

> "[...] EA decisions tend to be more conceptual and thus affected by levels of understanding, and personal bias. Hence softer skills are required e.g. language skills, emotional intelligence, consensus drawing."

To be fair and include all opinions given by participants, not all agree on the unique nature of enterprise architecture compared to other digital architecture fields, let alone any architecture (although not going in much more detail to support such stance):

> "There is no difference. Architecture is architecture is architecture. Only the target changes. ...If there were differences, it wouldn't be architecture."

2.4.5 Difficult aspects of design decisions

As participants stated already in other aspects, high-level design decisions are not simple to make, especially when compared to fields they perceive as more technical and rational like software architecture.

Large number of stakeholders with conflicting views Related to the communication aspect, an important difficulty in the decision making are the large number of stakeholders with different interests and different backgrounds, and dealing with conflicting goals.

> "So many stakeholders...."

> "First and foremost, a lack of clear input on the business goals and objectives, unclear or poorly formed needs analysis from directly effected stakeholders, short time frames and unclear or undefined financial commitment"

> "[...] inability of business leaders to articulate their goals/objective specifically enough."

> "Real commitment of boardmembers and other stakeholders. Words as transparency and compliancy are often political issues in practice."

> "The politics. Making a design decision based on principles and best practices is not difficult. Making it such that my stakeholders see the value in where I'm going, and see the benefit of going there with me, is much more difficult and interesting."

Participants perceived such politics to not be only limited to the decision takers or high-level business and convincing them to support a particular future state, but expressed the views that the entire context of enterprise architecture is highly politicized:

> "In a political environment, clarity and direction are not always liked."

> "Conflicting interests from different stakeholders, some more politically motivated than fact based."

> "Conflicting goals, e.g. save money and implement quickly. Lack of knowledge about the business or available products that meet the needs of the business."

High uncertainty and a changing environment The fact that design decisions are often long-term gives them a high amount of uncertainty. Moreover, since the uncertainty is high, there are many assumptions on which these decisions are based, which are often subject to change.

> "Variability, Number of Variables, Amount of Uncertainty, Lack of Information, Delays in decision making, Shifting Dates & Boundaries"

> " Dealing with great uncertainty Dealing with complex organization structure and politics where lots of different stakeholders have different interests"

> "[...] unclear or poorly formed needs analysis from directly effected stakeholders, short time frames and unclear or undefined financial commitment."

> "EA design decisions are always difficult. Lack of information and time constraint are typical reasons."

> "Lack of information is the biggest obstacle to making decision."

2.5 Characteristics of enterprise architecting

2.5.1 List of characteristics

The analysis above contains a list of italic statements summarizing our findings. Recall that our specific focus in this study was on the role of the facilitator (Figure 2.1). The characteristics are as follows:

1. Translating strategic goals into an IT strategy
2. Communicating plans of action
3. Explaining decisions instead of making them
4. Qualitative before quantitative data
5. Stronger business focus than other disciplines
6. Politics, emotions, and soft skills play a bigger role than in other disciplines

7. Large number of stakeholders with conflicting views

8. Highly uncertain decisions in a changing environment

None of these characteristics should be very surprising for anyone working in the field of enterprise architecture, be it in practice or in research. They are in line with many of the observations, definitions, and case studies from the field. However, when put together like this, they allow us use them as yardstick for determining which approaches may be appropriate in order to develop decision support using artificial intelligence techniques.

One direction of research that we deem suitable for modeling some of these characteristics is *bounded rationality*, which we briefly discuss in the next part, starting with *rationality*.

2.5.2 Rationality

Rationality is the habit of acting by reason, which means in accordance with the facts of reality.[7] Rationality implies the conformity of one's beliefs with one's reasons to believe, or of one's actions with one's reasons for action. The term has different specialized meanings in philosophy, economics, sociology, psychology, evolutionary biology, and political science. For instance, in economy, rationality is often understood as conforming to some set of postulates describing rational behavior. Consider for instance the *subjective expected utility* model, where uncertainty is represented by a set of *states of the world* $E_1, ..., E_n$. Possible outcomes are set of *consequences* and a decision alternative, known as an *act*, is a function from states of the world to consequences, denoted by $x = (x_1, ..., x_n)$, where x_i is the consequence of state E_i.

The decision maker's beliefs are represented by a *subjective probability distribution* $p = (p_1, ..., p_n)$, and her values for consequences are represented by a *utility function* $v(x)$. Its *subjective expected utility* (SEU) is then defined as:

$$SEU(\mathbf{x}) = \mathbf{E}_p[v(x)] = \sum_{i=1}^{n} p_i v(x_i).$$

This idea was first proposed by Daniel Bernoulli in 1738, but the idea of seeking to maximize expected value was discarded, but revived by von Neumann and Morgenstern (see [VNM07] for an overview), who considered the case that states of the world have objectively known probabilities, and later by Savage in 1954 [Sav72], who showed that the expected-utility model could be derived from simple axioms of consistent preferences under risk and uncertainty. The model of von Neumann and Morgenstern is extended to include situations where probabilities are subjectively determined by the decision maker.

In approaches such as those discussed in the previous paragraph, rationality is captured by a minimal set of postulates describing rational behavior, such that it corresponds to a unique decision principle. The candidate actions in these approaches are assumed to be feasible. Various groups of research in artificial intelligence dealing with decisions follow the above type of approach. For instance, research using Bayesian

[7] http://www.importanceofphilosophy.com/Ethics_Rationality.html

networks [Pea88], or planning under uncertainty (e.g., [DKKN95]) fall into this category, as well as more qualitative frameworks for decisions, but still in the same line of thoughts (e.g., [DT99, DP95, TP94]). Planning under uncertainty is often formalized using Markov decision processes, a mathematical framework for modeling (single-agent) decision making in situations where outcomes are partly (probabilistically) random and partly under control of the decision making. Markov decision processes are useful for studying a wide range of optimization problems, solved via dynamic programming or reinforcement learning. Yet another example based on classical decision theory is multiple-criteria decision analysis [BS02, GFE05]. This is a sub-area of operations research that explicitly evaluated multiple criteria in decision making, where cost or price are often one of the main criteria, and some measure of quality is typically another criteria.

2.5.3 Bounded Rationality

Bounded rationality is the idea that the rationality of a decision making is limited by the tractability of the decision problem, the cognitive limitations of the mind, and the time available to make the decision. Decision-makers in this view act as *satisficers*, seeking a satisfactory solution rather than an optimal one. Herbert A. Simon proposed bounded rationality as an alternative basis for the mathematical modeling of decision-making, as used in economics, political science and related disciplines. It complements classical rationality, which views decision-making as a fully rational process of finding an optimal choice given the information available. Simon used the analogy of a pair of scissors, where one blade represents "cognitive limitations" of actual humans and the other the "structures of the environment", illustrating how minds compensate for limited resources by exploiting known structural regularity in the environment [GS02].

Studies about bounded rationality have flourished in the last decades and have shed light on the meaning of artificial intelligence itself: an intelligent program should not (or even cannot) provide the optimal solution, but the best solution net of the computational costs [RW91, ZR93, BD94, SW02]. Doyle [Doy88] expresses this aim as follows: "the agent reflects on its circumstances, abilities, and limitations to rationally guide its own reasoning and internal organization [. . .]. Rational self-government includes direct control of the path and amount of reasoning performed".

It is now possible to design "systems" that are capable of taking their own computational resources into consideration during planning and problem solving [BD94, p.245]. In particular, the role of meta-reasoning has been formalized: it "consists of running a decision procedure whose purpose is to determine what other decision procedures [e.g., planners] should run and when" (*ibid.*, p.248). The time used by the decision procedure is important since, in real time problem situations "the utility of a given action varies significantly over the time necessary for a complete solution of the decision problem" [RW91]. In parallel, the agent paradigm has provided a unifying point of view over many branches of AI. Some of the agent models have received particular attention from the formal point of view: in particular, the BDI agent model, a 'cognitive' model of agenthood which assumes that agents, as well as humans, are driven by their mental representations; the leading mental attitudes are beliefs, desires and intentions. To face the resource boundedness problem, Bratman [Bra87] proposes to use the currently intended plans to limit the set of actions to be considered.

We believe such type of formalisms, based on rich psychological human attitudes, are appropriate when formalizing high-level decisions in enterprise architecture reasoning. The characteristics show that mental attitudes such as goals and intentions play an important role in the discussions. Moreover, due to the complexity of the domain, originating partially from a large number of stakeholders and a long perspective, bounded rationaliy is an important factor.

2.6 Discussion

2.6.1 Related work

There is research focusing on what makes an architect a *good* architect – but those studies often still leave in the middle just what the investigated people do in a regular day's work (cf. [SP08, Got13]). As such, while prescribing skills and characteristics architects ought to have, they offer little empirical insight into what architects currently do, especially in regards to decision making. Understanding the practice is important, because the multitude of existing views on enterprise architecture have been said to "provide heterogeneous and fragmented views of the domain" [KTZT12] and because existing frameworks for decision making in enterpise architecture are not grounded on empirical research in enterprise architecture, but rather on work in related areas such as software architecture [vZPvdLM14]. Extensive literature studies detailing the use and benefits of enterprise architecture (e.g., [TSSR11]) offer little further insight into the actual daily practice, involving no grounding of enterprise architecture in the context of its practitioner's perceptions.

Other studies do attempt to investigate how enterprise architecting (or parts of it) is done, but are limited to understandings of the authors themselves from prior or concurrent industrial roles (cf. [KAV05], whose study was "based on our experience, and discussions with other enterprise system architects"), or anecdotal evidence gathered in industrial cases (cf. [vGvD14]) and therefore are not based on primary empirical work. Some studies are limited to specific literature (cf. [DPvdMM14], whose focus only on closed questions based on one specific enterprise architecture framework). Some studies investigate actual companies, but usually in limited scope, for example earlier studies focusing on the use of enterprise architecture in a single aspect of Federal Government [SR07], Czech companies [NMD14], or Finnish companies [Nie07]. While such studies are useful for their insights, comparing the findings in order to gain a general understanding of the enterprise architecture practice brings additional issues of interpretation along. For example, the investigation method, context, and questions posed in these studies are not similar, and thus makes it difficult to relate results to each other.

Some of the few contributions attempting to investigate how enterprise architecture is perceived by those practicing it are for example Dankova [Dan09] and Mentz *et al*. [MKvdM12]. However, Dankova's work is limited in this regard by being essentially a corpus analysis of existing definitions. Mentz *et al*.'s more ambitious attempt incorporating hermeneutic phenomenology to compare understandings of enterprise architecture between practitioners and researchers also only focuses on existing definitions and frameworks and does not actively investigate the views these people have themselves.

2.6.2 Open issues

Our empirical study is a first attempt into understanding how practitioners in enterprise architecture perceive their work. In the context of this thesis it plays the role of an underlying motivation, but the main contributions of thesis can be in found other chapters. Therefore, the empirical study we present here is not in-depth, and one could spend an entire Ph.D. thesis on this subject. One of the main open issues is that this type of research should be done in a more principled and elaborate way.

Another interesting point of research is to investigate to what extent the fields enterprise architecture and software architecture are different. We tried to do so in a very preliminary way, but much more work can be done here. In particular, in current times software architecture is much more integrated into the business domain, with much more flat hierarchies in modern software companies such as Google, Facebook, and Amazon. It is therefore the question if there still is a big difference between the two fields, and if we should perhaps distinguish between "traditional software architecture" and "software architecture 2.0", which is much closer to enterprise architecture.

In the remainder of this section we focus on the internal validity and external validity of this study.

Internal validity: We did not presume a prior 'correct' interpretation of enterprise architecture on participants, instead allowing them to share their perception in order to avoid biasing participants into a specific view on enterprise architecture. We treated the Likert scale answers ordinally, calculating the median in order not to assume knowledge about the relative distance between individual answer pairs. While this gives less information than calculating averages or means, they still offer a clear insight into the general polarity of each group we investigated. The polarity of the answers of this scale (Table 2.2) was in line with the findings of the qualitative work, further ensuring the consistency of the participants' responses.

External validity: While qualitative research as ours does not have the aim to generate statistical inferences to general groups, ensuring that the sample of participants is representative of enterprise architecture on a community as a whole is still an important aspect. To do this we reached out via LinkedIn to ensure targeting enterprise architecture practitioners worldwide, and in our analysis also ensured that no particular country become dominant in the views, as can be seen in Figure 2.2(b). This helped in achieving a saturation of information which is most vital for these types of studies [Mas10]. Given the diversity of responses and drop-off of new information, this seems to have been reached. Furthermore, our sample size seems to be well in range of the mean sample size (=31) for qualitative studies [Mas10].

However, it should be kept in mind that the view of practitioners that software architecture is somehow more rational and hard than enterprise architecture seems to not be in line with other recent findings such as Sherman and Hadar's study [SH15], as their results reveal that (software) architects perform a variety of human-centered activities such as mentoring, leadership, reviewing and management.

2.6.3 Conclusion

We give an outline of the practice of high-level decision making in contemporary enterprise architecture based on a qualitative study of how enterprise architects perceive their professional work. This leads to a list of eight characteristics, which we use as yardsticks for a formal theory underlying a decision support system for enterprise architect reasoning. We observe that approaches based on bounded rationality may be appropriate to model these characteristics. The reason for this is that our characteristics show that enterprise architecture has many types of uncertainty due to its long-term perspective and large number of stakeholders.

3

A Logical Framework for EA Anamnesis

Abstract In this chapter we perform our second case study into the characteristics of enterprise architecture. We analyze a recent framework for capturing enterprise architecture design decisions called *EA Anamnesis* from a formal point of view and recognize various ambiguities and flaws in the specification of the framework. We then propose a more precise formalisation of EA Anamnesis using basic notions of first-order logic. Our main conclusion is that our formalism, and thus EA Anamnesis as well, does not offer much support for the type of reasoning processes specific to enterprise architects we found in the previous section. More notably, reasoning about goals, considering various viewpoints of stakeholders, and reasoning about plan dynamics are lacking.

3.1 Introduction

In this thesis we explore the possibility of explicitly linking architecture-level design decisions with their underlying assumptions. In this chapter, we contribute to this in two ways:

1. We formalize a recently proposed framework for decision making in enterprise architecture by Plataniotis *et al.* [PdKP13b, PDKP14b] called *EA Anamnesis* using first-order logic. This is the first contribution of this thesis, and contributes to the state of the art research in enterprise architecture.

2. We use this formalisation exercise as a case study in order to understand the domain of enterprise architecture better. In the previous chapter we concluded with a list of characteristics of enterprise architecture decision making. In this chapter, we explore whether we are able to capture these characteristics using our simple formalism.

EA Anamnesis is an approach towards capturing high-level decisions and issues in enterprise architecture. It is able to capture various aspects of decisions and provides connections with the existing enterprise architecture modeling language ArchiMate. In this modeling language, one can create the architecture of an enterprise. This is in some sense comparable to the architecture of a piece of software, with the important different that the relations between the information system and the business domain is made more explicit. One of the shortcomings of ArchiMate is that it is not possible to document which decisions were made when creating the models. This is the main motivation for the EA Anamnesis framework.

EA Anamnesis consists of a metamodel that serves as a basis for decision design graphs composed of enterprise architecture decisions, issues, observed impacts, and several

types of dependency relations. In this chapter we analyze the correspondence between the metamodel and the decision design graphs, and we note various technical flaws. We propose a formal framework that captures the decision design graphs more precisely. Moreover, motivated by providing a better guidance on the use of the framework for a priori decision analysis and support, we extend the framework to cater for a more expressive notation of decision state, and we make precise several informally introduced concepts of Plataniotis *et al.* using integrity constraints. We apply our framework to a well-known artificial example called ArchiSurance, and show the benefits of our formal approach by demonstrating the possibility for *a priori* decision analysis through consistency checks on the integrity constraints.

The rest of this chapter is structured as follows: In Section 3.2 we provide an illustrative case that we use throughout the chapter. In Section 3.3 we discuss the framework of Plataniotis *et al.* and its various shortcomings; In Section 3.4 we use this discussion as a motivation to present our formal framework; In Section 3.5 we validate our framework for *a priori* decision analysis by applying it to our ArchiSurance use case; Finally, in Section 3.6 we provide a discussion, containing open issues, related work, and a conclusion.

3.2 Illustrative case: ArchiSurance

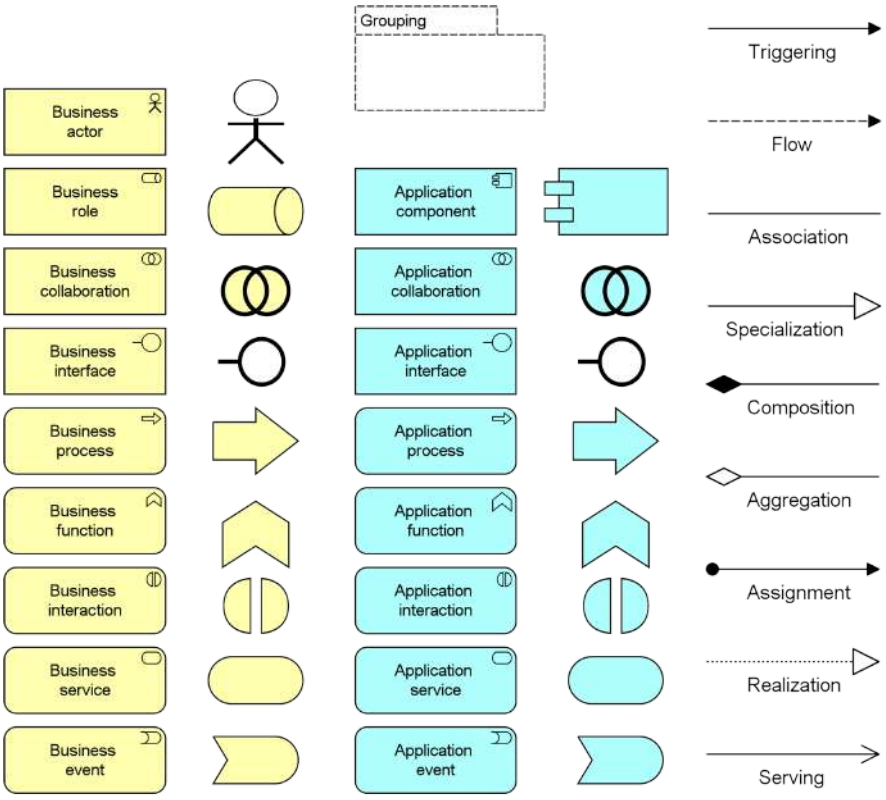


Figure 3.1: Subset of the ArchiMate elements and relationships. Yellow colored elements belong to the *business* layer, and cyan colored elements below to the *application* layer. Elements belonging to the *technology* layer have been omitted.

In this section we introduce the *ArchiSurance* example, that we will use to validate our

logic-based framework in Section 3.5. Since we model this case in ArchiMate, we first explain the main concepts of this language.

3.2.1 ArchiMate

ArchiMate is an Open Group[1] standard enterprise architecture modeling language. It distinguishes itself from other languages such as Unified Modeling Language and Business Process Modeling and Notation (BPMN) by its scope on enterprise modeling as a whole. It is a language for describing the construction and operation of business processes, organizational structures, information flows, IT systems, and technical infrastructure. This is comparable to an architectural drawing in classical building where the architecture describes the various aspects of the construction and use of a building. This insight helps the different stakeholders to design, assess, and communicate the consequences of decisions and changes within and between these business domains.

The Archimate framework divides the enterprise architecture into a business, application and technology layer. In each layer, three aspects are considered: active elements that exhibit behavior (e.g. Process and Function), an internal structure and elements that define use or communicate information. One of the main objectives of the ArchiMate language is to define the relationships between concepts in different architecture domains.

We provide the meaning of several ArchiMate elements and relationships in Figure 3.1. Elements pertaining to the *business* layer are colored yellow, and elements pertaining to the *application* layer are color cyan. The implementation layer has been omitted, since we do not consider it in our example. Various elements can be used to connect the elements in different layers, and in this way one can provide an overview of the dependencies between business and IT.

3.2.2 ArchiSurance

This ArchiSurance case is inspired by a paper on the economic functions of insurance intermediaries [CD06], and is the running case used to illustrate the ArchiMate language specifications [JBQ12]. *ArchiSurance* is the result of a recent merger of three previously independent insurance companies, that now sells car insurances products using direct-to-customer sales model. The goal of the newly created company is to reduce the cost of operation and products.

The merger has resulted in a number of integration and alignment challenges for the new company's business processes and information systems. These challenges appear in the *ArchiSurance* baseline business, application, data and technology architecture.

Figure 3.2 presents the partial (business and application layers) ArchiSurance's direct-to-customer sales model, modeled with the enterprise architecture modeling language ArchiMate.

Two business services support the sales model of ArchiSurance: "Car insurance registration service" and "Car insurance service". ArchiMate helps us to understand the dependencies between different perspectives on an enterprise. For example, in Fig-

[1] See `http://www.opengroup.org/standards/ea`

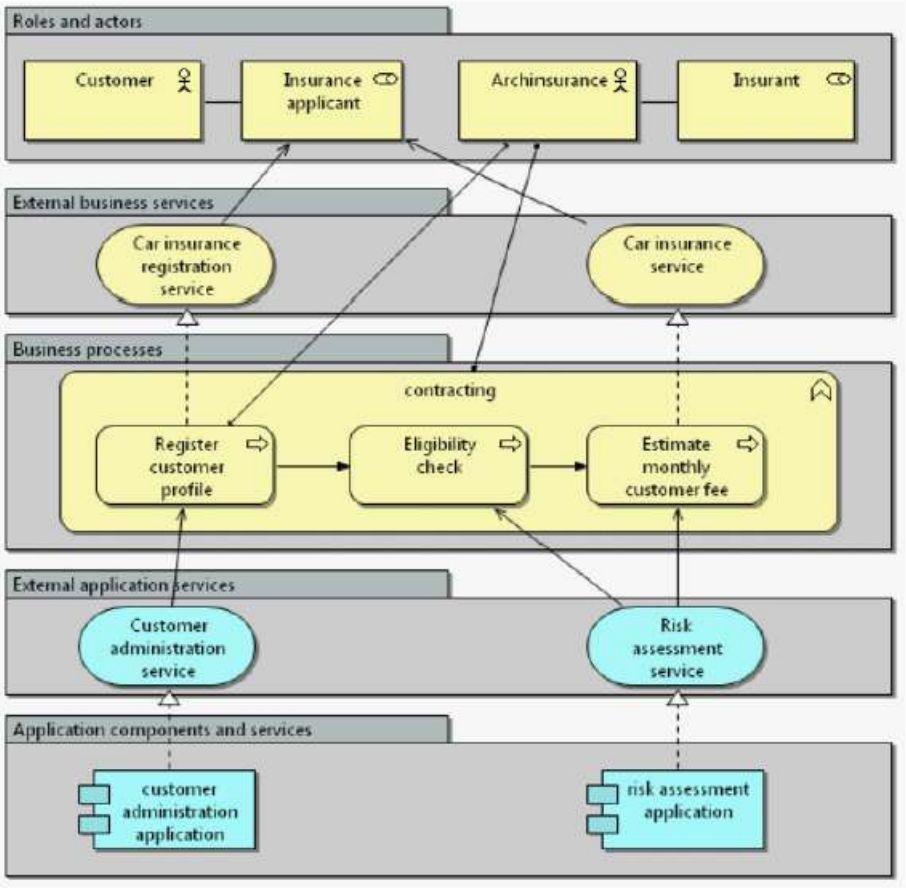


Figure 3.2: ArchiSurance direct-to-customer EA model [PdKP13b]

ure 3.2 we see that the business service "Car insurance registration service" is realized by a business process "Register customer profile". In turn, we also see that this business process is supported by the application service "Customer administration service".

Although removing intermediates in the supply chain leads to a decrease of operation costs, it also increases harmful risk profiles [CD06]. Such profiles lead insurance companies to calculate unsuitable premiums or, even worse, to wrongfully issue insurances to customers. As a response, ArchiSurance decides to use intermediaries to sell its insurance products. After all, compiling accurate risk profiles is part of the core business of an intermediary [CD06]. In our scenario, an external architect call *John* is hired by ArchiSurance to help guide change to an intermediary sales model. John uses ArchiMate to capture the impacts that selling insurance via an intermediary has in terms of business processes, IT infrastructure and more.

For illustration purposes we will focus on the translation of the new business process "Customer profile registration" to EA artifacts in the application layer. The resulting ArchiMate model is depicted in Figure 3.3. Here we see for example how a (new) business process "customer profile registration", owned by the insurance broker (ownership being indicated by a line between the broker and the business process), is supported by the IT applications "customer administration service intermediary" and "customer administration service ArchiSurance".

When we compare Figure 3.2 with Figure 3.3 we see that various changes have been made, but we do not know *why* these changes are made. We do not know which *is-*

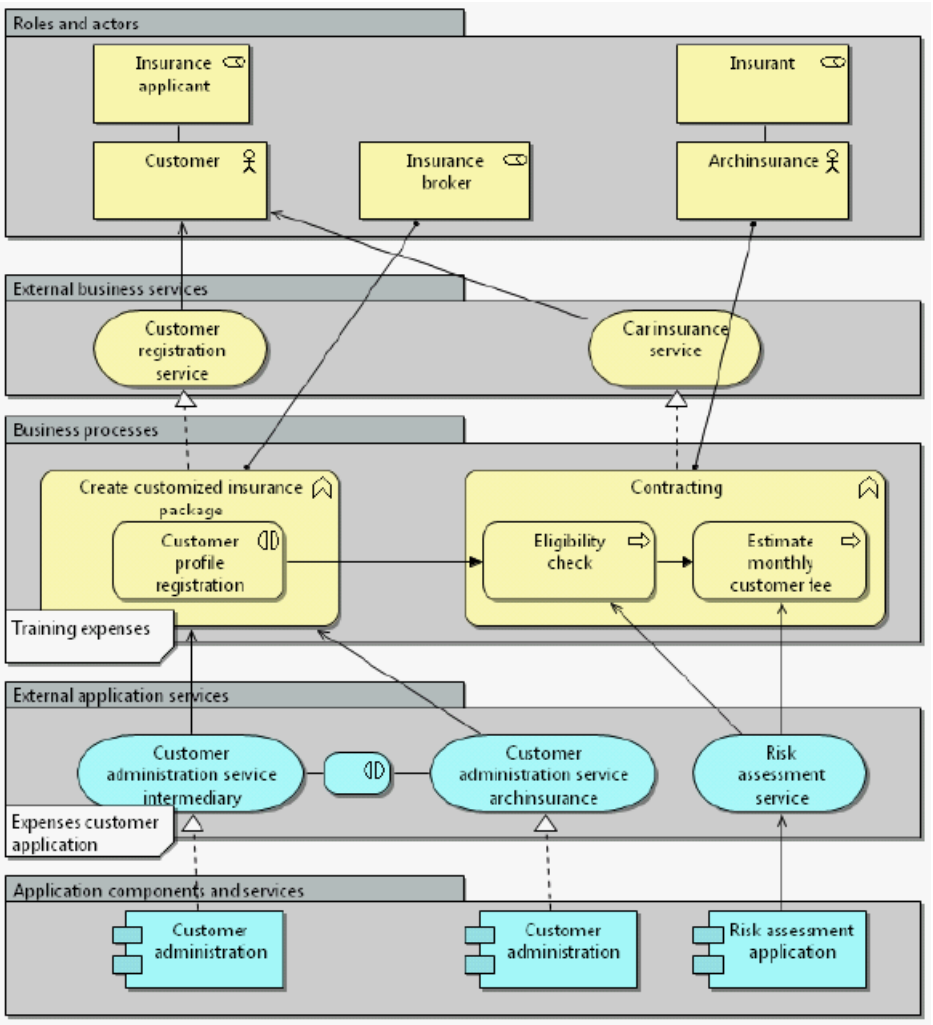


Figure 3.3: ArchiSurance intermediary EA model [PdKP13b]

sues the modelers faced, nor do we know which decisions they made to resolve them. This is what EA Anamnesis aims to provide: a *rationalization* of the decisions made in ArchiMate.

3.3 EA Anamnesis

In this section we briefly review the key components of the EA Anamnesis framework [PdKP13b], followed by a discussion of its limitations. We use these observations as a basis for the formal framework that we introduce in the next section.

3.3.1 Metamodel and decision design graphs

Plataniotis *et al.* [PdKP13b, PDKP14b] recently presented an approach for relating enterprise architecture decisions. Using a metamodel and a decision design graph, they explain how decisions from different enterprise domains (business, application, and technology) relate to each other. For example, how decisions taken on a business level affect IT decisions and vice versa. Their approach is inspired by well-known mechanisms

for capturing architectural rationales in software architecture. The metamodel that was presented by Plataniotis *et al.* is depicted in Figure 3.5. This metamodel serves as an underlying model for design decision graphs. The legend for these design decision graphs is given in Figure 3.4, and an example of a design decision graph is shown in Figure 3.8.

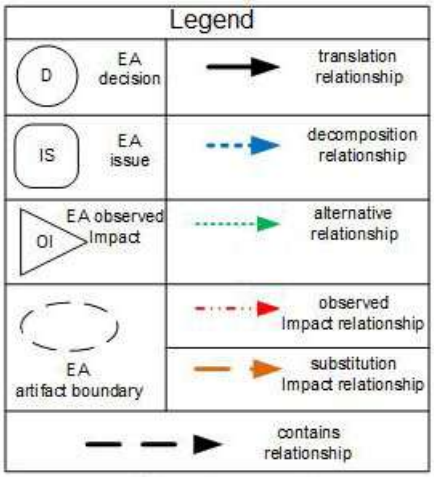


Figure 3.4: Legend for the design decision graphs of EA Anamnesis

From now on, we will use the following naming convention: We will refer to the metamodel in Figure 3.5 as "the metamodel", and the decision design graph in Figure 3.8 as "the decision graph".

The metamodel consists of the following elements:

- **EA Decision** represents a decision that has been made or rejected in order to resolve an issue. An EA decision shows decisions that are captured in the context of an Enterprise Transformation [POtL10]. The decision graph contains a total of 13 decisions, from EA Decision D01 to EA Decision D13.

- **EA Issue** represents an architectural design problem that enterprise architects have to address during the Enterprise transformation process. In this way, they can be regarded as a motivation for the design decisions. The decision graph contains 6 issues, from EA Issue IS01 to EA Issue IS06.

- **EA Artifact** serves as a bridging concept towards the EA modeling language ArchiMate, whereby an EA artifact links EA decisions to ArchiMate concepts. For instance, EA Decision D01 in the decision graph is related to EA artifact "Customer profile registration Business processes". EA issues are not related to artifacts.

- **Layer** is in line with the ArchiMate language [IJLP12]: An enterprise is specified in three *layers*: *Business, Application* and *Technology*. Using these layers, an enterprise architect is able to model an enterprise *holistically*, showing not only applications and physical IT infrastructure (which are contained in the application and technology layers), but also how the IT impacts/is impacted by the products, services and business strategy and processes. EA Decisions are related to layers, for instance in the decision graph EA Decision D01 is related to the Business Layer, while EA Decision D06 is related to the Application Layer.

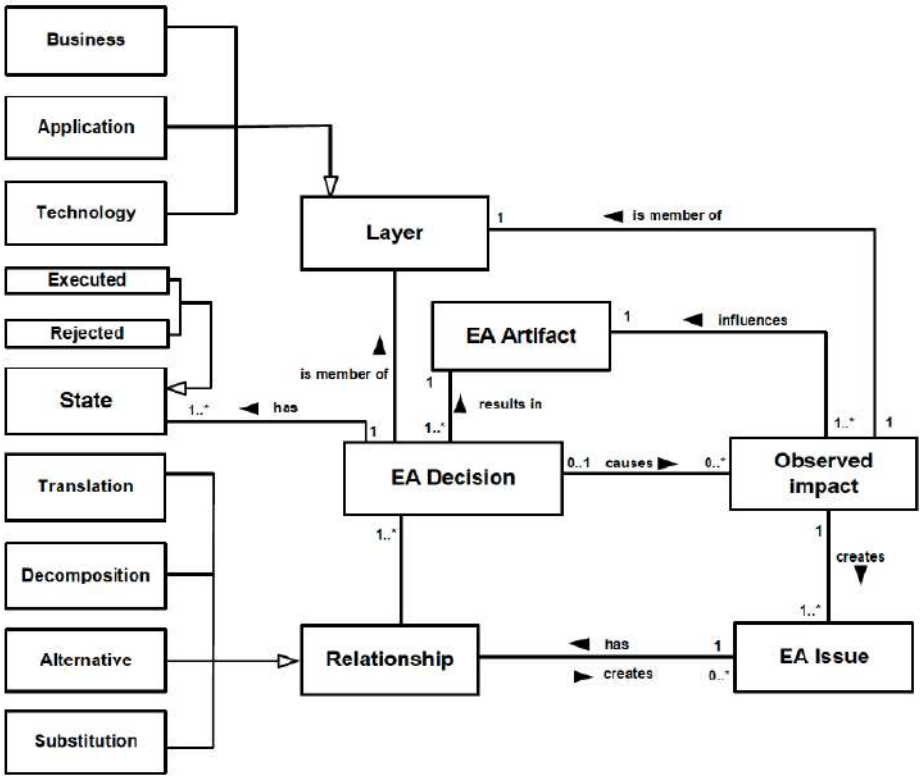


Figure 3.5: EA decisions relationship metamodel [PdKP13b]

- **State** represents the state of an EA Decision, which is either *Executed* or *Rejected*. In an executed state, an EA decision has already been made and was accepted. A rejected decision, on the other hand, is a decision that was considered as an alternative during the decision making process but was rejected because another decision was more appropriate. In the decision graph, the state of a decision is not explicitly represented but it can be inferred from the relationships. A decision that has an *alternative* relation with an issue is rejected, while all other decisions are executed.

- **Relationship** makes the different types of relationships between EA decisions explicit. Based on ontologies for software architecture design decisions, Plataniotis *et al.* define four relationships. The *Translation* relationship illustrates relationships between decisions and issues that belong to different EA artifacts. During the enterprise transformation process architects translate the requirements that new EA artifacts impose (EA issues) to decisions that will support these requirements by means of another EA artifact. *Decomposition* relationships signify how generic EA decisions decompose into more detailed design decisions within an EA artifact. *Alternative* relationships illustrate the EA decisions that were rejected (alternatives) in order to address a specific EA issue. *Substitution* relationships illustrate how one EA decision replaces another EA decision. An EA decision can be replaced when it creates a negative observed impact in the enterprise architecture.

- **Observed Impact** signifies an *unanticipated* positive/negative consequence of an already made decision to an EA artifact. This is opposed to anticipated consequences, as indicated by the *Translation* and *Decomposition* relationships. The

main usefulness of capturing observed impacts is that they can be used by architects to avoid decisions with negative consequences in future designs of the architecture [PdKP13b].

For instance, in the decision graph Decision D10 decomposes to decision D11 through issue IS06. D11 turns out to have a negative observed impact OI1, which is translated to a decision D13 through issue IS07 (alternative D12 for IS07 is rejected). D13 addresses the negative observed impact of D11 by substituting D11.

3.3.2 Limitations of the metamodel

The metamodel of Figure 3.5 should serve as the underlying formalism for the decision graph, but in this subsection we motivate why this is not sufficient by discussing the differences and ambiguities between the metamodel and the decision graph. We will take these remarks into account when formalizing the decision graph in the next section.

- First of all and most importantly, it is very difficult to asses whether the metamodel correctly formalizes the decision graph, simply because EA Anamnesis only provides a single example of the decision graph. Therefore, we are left to guess what the meaning is exactly. The only thing we can do is observe various relationships in the decision graph, and guess what their intended meaning is.

- According to information shown in the decision graph, the creation of a translation/decomposition relationship between two EA Decisions implies the creation of two separate relationships of the same type: one for the EA Decision to EA Issue and another one for the EA Issue to EA Decision. This creates information redundancy issues because this is not captured in the metamodel. The definition of at least one relationship of a specific type should imply that the other relationship should be of he same type. For example, in the decision graph EA Decision 01 is related with EA Issue 03 through a translation relationship. Similarly, EA Issue 03 is related with EA Decision 06 through a translation relationship. The definition of the relationship type between EA Issue 03 and EA Decision 06 should imply the same relationship type between EA Decision 01 and EA Issue 03, but this is currently not captured in the metamodel.

- More generally, the metamodel specification of the four types of relationships seems to allow a number of unwanted models. For instance, it is possible for the relationships to be reflexive (an EA decision contains a Relationship, which again may contain the same EA decision), and it is possible for an EA decision to be related with an EA Issues through more than one relationship. Furthermore, an EA Decision can contain a Relationship, which then can contain more than one EA Issues again. Thus, an EA Decision can be related with multiple EA Issue through a single relationship, which is clearly unwanted.

- The metamodel provides two different types of states (executed, rejected) per EA Decision. Despite the fact that these two states adequately describe the state of an EA Decision during the *a posteriori* analysis, they don't provide enough expressivity in the *a priori* case. In the latter case, there is the need to express that

an EA Decision is in "open" state while enterprise architects examine the alternatives [KLvV06].

- Whereas the metamodel provides the notion of "Observed impact", it does not explicitly distinguish between "positive observed impact" and "negative observed impact". For instance, in the decision graph EA Decision D11 has Observed Impact OI1, which creates an issue IS07. Thus, it seems that this observed impact is negative, but neither the metamodel or the graph are able to distinguish positive impacts from negative ones.

- Finally, there are a number of assumptions on the design graph that have not been made explicit in the metamodel. Firstly, all issues in the graph have been resolved. Secondly, there is always a single decision that is executed in order to solve an issue, while the others are rejected. Finally, a decision that creates a negative observed impact is assumed to be replaced by a decision that addresses this impact. These three assumptions are not formalized, and we propose to do so using integrity constraints. Note that we do not argue that these three assumptions should always hold in any problem domain. Rather, they are assumptions of the work of Plataniotis *et al.* and our current goal is simply to formalize it.

3.4 A formal model for EA decision modeling

In the previous section we showed that the metamodel of Figure 3.5 is not restrictive enough to characterize the design decision graph of Figure 3.8 correctly. In order to resolve this issue and to obtain a consistent formalisation for the decision design graphs that allow for *a priori* decision modeling, we will introduce a formal model in this section.

3.4.1 Elementary definitions for EA decision modeling

Basic concepts from first-order logic with equality and the unique names assumption are adequate to define the entities in the metamodel and the relations between them. We begin with representations for the metamodel elements *EA Decision*, *EA Issue*, and *Observed Impact*.

Definition 3.1 (EA Issue)**.** *Let I be a set of EA Issues, where each issue $i \in I$ is a sentence representing the issue.*

Rationale and example: An EA decision issue (short: issue) represents a single design concern. For now, we follow Plataniotis *et al.* and we do not add any attributes to the issues, but we recognize that this is certainly possible and a necessary extension. For instance, Zimmerman *et al.*[ZKL+09], attribute a total of 18 properties to issues that can be used to characterize them. Because such attributes do not have a specific purpose in our formal model, we leave them out for ease of exposition. The issues in the decision graph are $I = \{IS01, \ldots, IS07\}$.

Definition 3.2 (EA Decision)**.** *Let Art be a set of ArchiMate artifacts, and let $\varepsilon \in Art$ be the "empty" artifact. Let D be a set of EA Decisions, where each decision in D*

is a tuple (d,s,a,l) *consisting respectively of a sentence* d *representing the decision, a state* $s \in \{open, executed, rejected\}$*, an Artifact* $a \in Art$*, and a layer* $l \in \{business, application, technology\}$*. We also write* s_d, a_d*, and* l_d *to refer to respectively the state, the artifact, and the layer of decision* d*. If* $a = \varepsilon$*, the decision is made independent of an ArchiMate artifact.*

Rationale and example: An EA Decision presents a possible solution to the design issue that is expressed by an EA Issue. The state s represents the current state of the decision. While Plataniotis *et al.* distinguish two different states of a decision (a decision is either "executed" or "rejected"), we extend this with an additional state "open". As we mentioned in the previous section, this is motivated by the fact that we aim to capture *a priori* decision analysis, which is different from the *ex post* approach of Plataniotis *et al.* The EA artifact a of an EA Decision represents the EA artifact to which this decision is related. Finally, the layer l is the layer on which the decision is made. Similar to EA Issues, we leave out additional attributes that do not have a specific purpose in our model. In the decision graph, Decision D06 can be represented with (d,s,a,l), where d =D06, $s = executed, a =$ "Customer administration intermediary application service", and $l = application$. Decision D04 can be represented with (d,s,a,l), where d =D04, $s = rejected$, $a = \varepsilon$, and $l = application$.

Definition 3.3 (Observed Impact)**.** *Let* O *be a set of observed impacts, where each observed impact* $o = (oi, v, a, l)$ *consists of a string* oi *describing the observed impact, a value* $v \in \{positive, negative\}$*, an EA Artifact* a*, and a Layer* l*. When* $v = positive$ *we say that* o *is a* positive observed impact*; when* $o = negative$ *we say that* o *is a* negative observed impact*. We also write* v_o, a_o*, and* l_o *to refer respectively to the value, the artifact, and the layer of observed impact* o*.*

Rationale and example: An observed impact is either positive or negative, where negative observed impacts create new issues. This formalization allows for an explicit distinction between positive and negative observed impacts. In the decision graph, the only observed impact is $OI1$, which is negative, so we can formalize this as (oi, v, a, l), where oi =OI1, $v = negative, a =$ "Customer profile registration Business process", and $l = business$.

Definition 3.4 (Contains relation)**.** *Let* $\prec_D \subseteq I \times D$ *be a* contains *relation between issues and decisions,* $\prec_I \subseteq D \times I$ *be a* contains *relation between decisions and issues,* $\prec_{O_{in}} \subseteq D \times O$ *be a* contains *relation between decisions and observed impact, and* $\prec_{O_{out}} \subseteq O \times I$ *a* contains *relation between observed impact and issues. We set the general* contains *relation* $\prec = \prec_D \cup \prec_I \cup \prec_{O_{in}} \cup \prec_{O_{out}}$*. If* $(a \prec b)$*, then we say that* a *contains* b *or that* b *is contained in* a*. We sometimes abbreviate* $(a \prec b) \wedge (b \prec c)$ *with* $a \prec b \prec c$*.*

Rationale and example: The *contains* relation is also used in Zimmerman *et al.* and allows us to define a single hierarchical structure, which serves as a table of content, allowing the user to locate issues and alternatives easily in the enterprise architectural knowledge and helping the knowledge engineer to avoid undesired redundancies. The *contains* relation is the underlying dependency relation that we use to build decision design graphs (which are simply trees in our formalization). We will use this relation to later define the four types of *Relationship* entities that were introduced in the metamodel. These four relationships are relatively complex, so it helps to have a simple underlying representation of the decision hierarchy. Intuitively, the *contains* relations can be obtained by

treating all arcs in the decision graph as of the same type. It contains for instance the following relations: $D01 \prec IS01 \prec D02, D01 \prec IS02 \prec D03, D01 \prec IS03 \prec D04$ (see figure 3.6), but also $D11 \prec OI1 \prec IS07$.

Definition 3.5 (Substitution relation)**.** *Let* $Sub : D \times D$ *be a* substitution *relation between decisions. If* $Sub(d_1, d_2)$ *we say that decision* d_1 substitutes *decision* d_2*, or that decision* d_2 *is* substituted by *decision* d_1*. We also denote this by* $d_1 \xrightarrow{S} d_2$*,*

Rationale and example: We separate replacing decisions from the *contains* relation of the previous definition, so that we are able to formalize the decision graphs of EA Anamnesis as *trees* with a replacement relation. The only place where decisions are replaced is when a negative observed impact is addressed. For instance, the decision graph has one replacement relation, namely $D13 \xrightarrow{S} D11$.

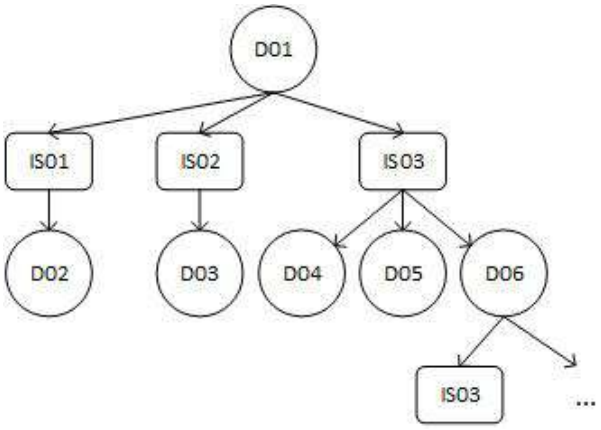


Figure 3.6: The *contains* relation $\prec$ for part of the decision graph.

Definition 3.6 (Decision Design Tree)**.** *A* decision design tree $\mathbb{D} = (D \cup I \cup O, \prec, Sub)$ *consists of a set of decisions D, a set of issues I, a set of observed impacts O, a* contains *relation* $\prec$ *that induces a tree containing issues, decisions, and observed impacts of decisions, and a substitution relation Sub.*

Rationale and example: Modeling architectural decisions in itself is not new: Ran and Kuusela also propose (but do not formalize) the notation of design decision trees [RK96]. Zimmermann *et al.* propose a formalization that is comparable to ours, but our is specifically for enterprise architecture decision making, and we separate the substitution relations, which allows us to use trees instead of graphs.

3.4.2 Layered decision model and logical relations

The elementary definitions from Section 3.4.1 allow knowledge engineers to capture decisions and organize the knowledge in a decision hierarchy. However, the resulting ordered architectural decision tree does not yet support the vision of an active, managed decision model taking a guiding role during architecture design. More relations between decisions, issues and observed impacts must be defined. In this section, we introduce the four relationship of Plataniotis *et al.* and formalize several logical constraints.

Definition 3.7 (Translation relation)**.** *The* translation relationship $R_T \subseteq D \times I \times D$ *is a three-placed decision-issue-decision relationship* $R_T(d_1, i, d_2)$*, also denoted with* $d_1 \xrightarrow{T(i)} d_2$*, that connects two decisions through an issue where these decisions are related to different EA artifacts. Formally, given some decision design tree* $\mathbb{D}$[2]*,*

$$d_1 \xrightarrow{T(i)} d_2 \text{ iff } (d_1 \prec_I i \prec_D d_2) \wedge (a_{d_1} \neq a_{d_2}).$$

Rationale and example: Translation relationships indicate how a decision on one artifact translates to a decision on another artifact through an issue. Thus, having a translation relationship requires three entities: a decision, a issue, and another decision. For instance, the design graph contains the translation relationship $D01 \xrightarrow{T(IS03)} D06$. This is a valid relationship, since we have $D01 \prec_I IS03 \prec_D D06$, and we also have $a_{D01} \neq a_{D06}$ because $a_{D01} =$"Customer profile registration Business process" and $a_{D06} =$"Customer administration intermediary application service". Note that the condition $a_{d_1} \neq a_{d_2}$ already ensures that $d_1 \neq d_2$, which avoids the relation to be reflexive. Furthermore, note that we now require the *unique names assumptions*, which assumes that elements with different names always refer to different entities in the world.

Definition 3.8 (Decomposition relation)**.** *The* decomposition relationship $R_D \subseteq D \times I \times D$ *is a three-placed decision-issue-decision relationship* $R_D(d_1, i, d_2)$*, also denoted with* $d_1 \xrightarrow{D(i)} d_2$*, that connects two decisions through an issue where these decisions are related to the same EA artifact. Formally:*

$$d_1 \xrightarrow{D(i)} d_2 \text{ iff } (d_1 \prec_I i \prec_D d_2) \wedge (a_{d_1} = a_{d_2}) \wedge (d_1 \neq d_2).$$

Rationale and example: Decomposition relationships are similar to translation relationships, with the only difference that in decomposition relationships the two artifacts belonging to the decisions in the relation should be the same. The last condition ensures that the relation is not reflexive.[3] For instance, the design graph contains the decomposition relationship $D01 \xrightarrow{T(IS01)} D02$, which is valid because we have $D01 \prec_I IS01 \prec_D D02$, $a_{D01} = a_{D02} =$"'Customer profile registration Business process", and $D01 \neq D02$.

Definition 3.9 (Alternative relation)**.** *The* alternative relationship $R_A \subseteq I \times D$ *is a two-placed issue-decision relationship, also denoted with* $i \xrightarrow{A} d$*, that connects an issue with a rejected decision. Formally:*

$$i \xrightarrow{A} d \text{ iff } (i \prec_D d) \wedge (s_d = rejected).$$

Rationale and example: The alternative relationship indicates decisions that have been rejected in the decision process. For instance, in the design graph we have $IS03 \xrightarrow{A} D04$ and $IS03 \xrightarrow{A} D05$.

[2]Note that all variables in the formulas are implicitly universally quantified. We have left the quantifiers out for readability.

[3]Actually, the fact that we define $\mathbb{D}$ as a tree induced by $\prec$ already ensures this, but we have left it in for completeness.

Definition 3.10 (Observed Impact relation)**.** *The* observed impact relationship $R_O \subseteq D \times O \times I \times D$ *is a four-placed decision-impact-issue-decision relationship, also denoted with* $d_1 \xrightarrow{O(o,i)} d_2$*, which describes the effect of a negative observed impact on a decision, which is addressed by an issue and subsequently resolved by a decision. Formally:*

$$d_1 \xrightarrow{O(o,i)} d_2 \textit{ iff } (d_1 \prec_{O_{in}} o \prec_{O_{out}} i \prec_D d_2) \wedge (v_o = negative) \wedge (d_2 \xrightarrow{S} d_1).$$

Rationale and example: The observed impact relation is the only relation in the design graph that is not characterized by the metamodel. In the decision graph, EA Decision D11 causes a negative Observed Impact OI1, which is addressed by EA Issue IS07, that is subsequently resolved by EA Decision D13.

With these relations introduced, we will now define three logical constraints on our decision models. We stress that this list is by no means meant to be exhaustive. Instead, it is a list of constraints that are suggested by Plataniotis *et al.*. These constraints are used to check the decision graph for consistency. If the graph is not consistent, we can locate the inconsistency by determining what constraint is violated and for which element. This is useful input for the architect in the decision making process.

Integrity Constraint 1. *All issues should be resolved; For each issue, there should be a decision that is contained in this issue and that is executed*[4]*:*

$\forall_{i \in I} \exists_{d \in D} ((i \prec_D d) \wedge (s_d = executed))$

Rationale and example: An issue represents an architectural design problem that enterprise architects have to address during the enterprise transformation process. Having a consistency check for the status of the issue by verifying whether a decision has been executed to resolve it can assist the architect in detecting "loose ends". This is particularly useful in large and complex graphs with many interdependent nodes [KS93].

Integrity Constraint 2. *If a decision that is contained in an issue is executed, then all other decision that have a relation with that issue should be rejected:*

$$\forall_{i \in I} : (\exists_{d \in D} : ((i \prec_D d) \wedge (s_d = executed))) \Rightarrow (\forall_{d' \neq d} : (i \prec_D d') \Rightarrow (s_{d'} = rejected))$$

Rationale and example: This constraint describes a dependency between decisions that are contained in the same issue. The decision graph suggests that issues are solved by a single decision. This means that when a decision is executed that is contained in an issue, all other decision that are contained in this issue should be rejected. For instance, because decision D06 is executed, both decision D04 and D05 are rejected. Note that it would be possible to combine Constraint 1 and 2 into a single constraint. However, we separated them since Constraint 1 can be met while Constraint 2 is violated, which allows us to provide more detailed error messages.

Integrity Constraint 3. *If a decision contains a negative observed impact, then this decision should be replaced by a decision addressing the negative impact:*

$\forall_{d \in D, o \in O} : ((d \prec_{O_{in}} o) \wedge (v_o = negative)) \Rightarrow \exists_{d' \in D, i \in I} : (d \xrightarrow{O(o,i)} d')$.

[4]Note that we now do add quantifiers since we mix universal quantification and existential quantification.

Rationale and example: The goal of having negative observed impacts is to be able to reconsider decisions that have caused this impact. This constraint addresses this idea by stating that negative observed impacts should result in the substitution of the decision that has caused the impact. For instance, decision D11 contains observed impact OI1. This constraint is satisfied for this impact because we have $D11 \xrightarrow{O(OI1,IS07)} D13$, indicating that decision D13 substitutes decision D11.

3.5 Validation with ArchiSurance

In this section we demonstrate how the formal framework introduced in Section 3.4 supports *a priori* decision analysis of design graphs by consistency checks using the integrity constraints.

Recall the illustrative example of ArchiSurance that we introduced in Section 3.2. Our external architect John is in the process of transforming the ArchiMate model of Figure 3.2 into Figure 3.3. For the implementation of these EA artifacts a number of EA decisions have to be made. John, in parallel with ArchiMate modeling language, uses our approach to capture the relationships of decisions and check the consistency of the decision graph.

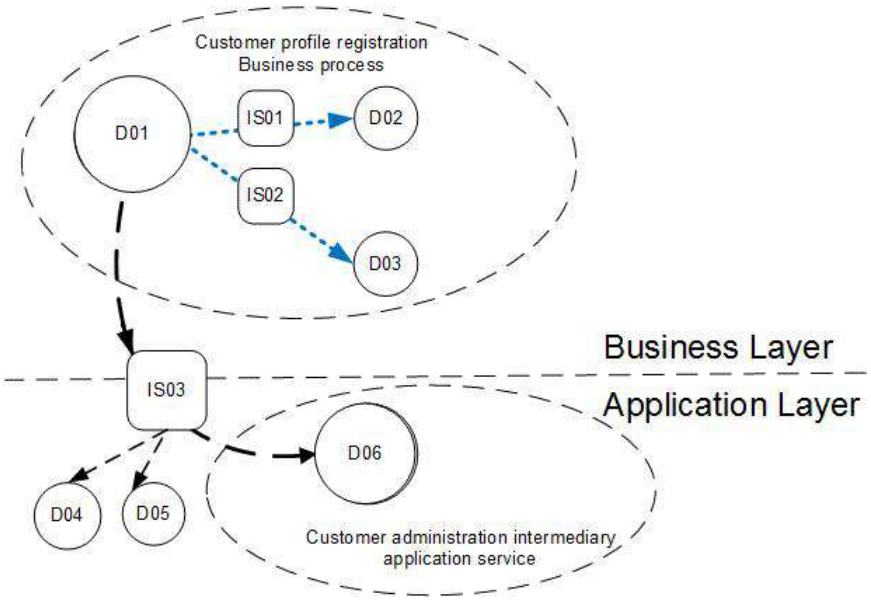


Figure 3.7: ArchiSurance scenario: Integrity constraint 1 is violated because EA Issue IS03 is not resolved

John starts by adding the main decision: "Make customer profile registration via intermediary" (D01) to the decision design graph. This decision belongs to the EA artifact "'Customer profile registration Business process". After the enterprise has decided to make this decision, three new issues arise, IS01, IS02, and IS03. Both IS01 and IS02 are addressed by making a decision that related to the same artifact. For IS03, which stands for "Create an appropriate application service to support new business process", there are three different decisions that can be made in the Application Layer, namely D04, D05, and D06 (see Figure 3.7, the legend of the relations is in Figure 4.5). At this moment, none of these three decisions have been made, so the status of these three

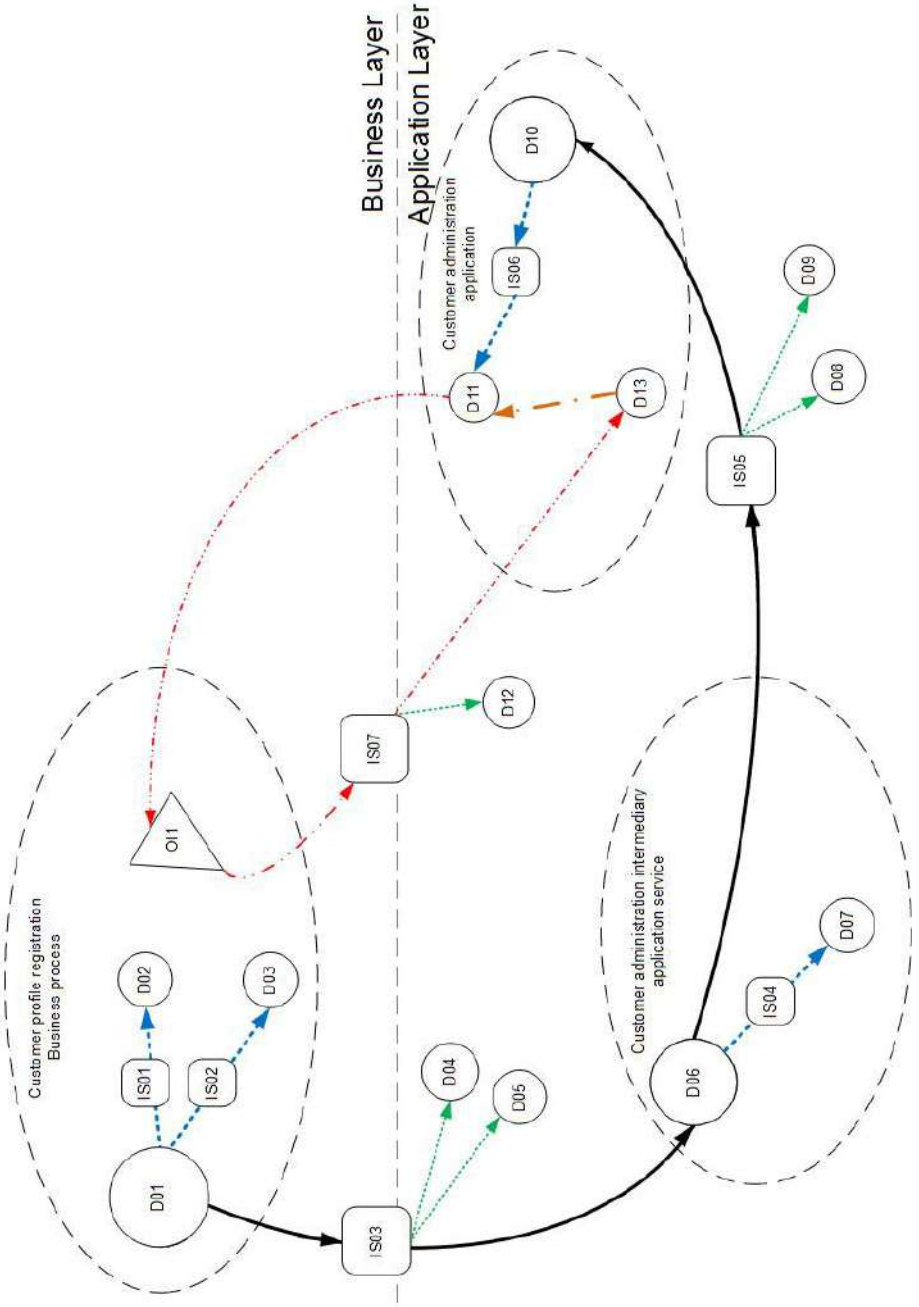


Figure 3.8: EA decisions relationships visualization[5]

decisions is still open. Thus, in figure 3.7 there are two *Decomposes* relations, namely $D01 \xrightarrow{D(IS01)} D02$ and $D01 \xrightarrow{D(IS02)} D03$, and the other relations are simply *Contains* relations: $D01 \prec IS03, IS03 \prec D04, IS03 \prec D05, IS03 \prec D06$. After John has created the graph of Figure 3.7, he checks it for consistency. It turns out that integrity constraint 1 is violated: Not all issues are resolved because for issue IS03 there is no decision d such that $IS03 \prec_D d$ and $s_d = executed$. John can choose between these three decisions and selects decision D06, which stands for "Introduce application service A", as the executed decision.

After having changed the status of decision D06 from "open" to "executed", John checks the consistency of the graph again. This time, another inconsistency arises, namely that integrity constraint 2 is violated. The reason for this is that since decision D06 is contained in issue IS03 (i.e., we have $IS03 \prec_D D06$) and D06 is executed (i.e. $s_{D06} = executed$), all other decisions that are contained in IS03 (that is, decision D04 and D05) should be rejected. Therefore, John decides to change the status of both these decision from "open" to "rejected". When John checks the graph for consistency now, he finds that the graph is consistent.

Decision D06 results in two new issues, of which "Find an appropriate application to interface with the intermediary" (IS05) is solved by "Acquisition of COTS application B" (D10), resulting in the EA artifact "Customer administration application". Decision D10 decomposes through issue IS06 in the decision "Application interface type 1" (D11).

Using the concept of an Observed impact, John formalizes that users of "Customer administration application" had difficulties using this new application interface. This is signified by the negative observed impact 01 "Degraded user experience in the application use" (OI1). As such, EA decision 11 "Application interface 1" has a negative observed impact on the business process "Customer profile registration".

According to integrity constraint 3, a negative observed impact should be addressed by a decision replacing the original decision that causes the observed impact. Therefore, John translates the observed impact "Degraded user experience in the application use" via EA issue 07 "have fitting application interface" into "replace existing application interface with an interface similar to the old one" (EA decision 13), after having rejected the alternative decision "Training of users on the new application". The last step John has to take is to replace EA decision 11 "Application interface type 1" with EA decision 13 "Application interface type 2".

When the transformation has finished and all decisions have been made, John obtains the graph that is depicted in Figure 3.8. This graph is consistent according to the integrity constraints.

3.6 Discussion

3.6.1 Related work

In the domain of software architecture, which is closely related to enterprise architecture, several design rational approaches have been developed: *argumentation* based approaches such as Issue-Based Information System (IBIS) [KR70], Design Rationale

[5]Figure adapted from Plataniotis *et al.* [PdKP13b]

Language (DRL) [Lee91], *template* based approaches, such as [TA05] and *model* based approaches, such as [JB05, TJH07]. Most of them capture textually the architecture decisions, the rationales, the issues and the implications. In addition, the model based approach provides means to relate those decisions with the software artifacts and with other decisions.

About twenty years ago, Ran *et al.* [RK96] proposed a systematic approach to document, refine, organize and reuse the architectural knowledge for software design in the form of a Design Decision Tree (DDT) that is a partial ordering on decisions put in the context of the problem requirements and the constraints imposed by earlier decisions. More recently, Tyree and Akerman [TA05] recognized that architecture decision capturing plays a keys role in what they call "demystifying architecture". They stress that architecture decisions should have a permanent place in the software architecture development process. Moreover, it facilitates traceability from decisions back to requirements and it provides agile documentation (which is crucial in an agile process where a team does not have the time to wait for the architect to completely develop and document the architecture).

Both Zimmerman *et al.* [ZKL+09] and Tan *et al.* [TJH07] recently proposed a comprehensive framework for decision capturing in software architecture. Zimmermann *et al.* also provide a formal framework, focusing mostly on the re-usability of decision by distinguishing between alternatives and outcomes.

In the field of enterprise architecture the literature is significantly more scarce. To our knowledge, the work of Plataniotis *et al.* [PdKP13b] is one of the first approaches towards decision rationalization in enterprise architecture. While most methods for decision modeling and analysis use visual notations from existing modeling methods like UML and the likes, their underpinnings still inherently benefit from mathematical formalizations. Communicating these formalizations to end-users a like does not require a steep level of training, and can be easily communicated to them [Hal90], nor does the focus on a more rigorous specification of these mathematical underpinnings forsake using the other tools and notations that build and rely on them [BH95].

Finally, goal-oriented modeling frameworks (e.g. i*[6], Tropos[7]) provide means to deal with the motivations of designs, being more expressive than the ArchiMate 2.0. motivation layer. Even so, their main focus is not to provide decision rationales. We turn to goal modeling in more detail in the next chapter.

3.6.2 Open issues

We demonstrated how some intuitive and simple constraints can be formalized in first-order logic to check a decision graph for consistency. However, we did not yet present a framework that will actively search for solutions to inconsistencies and in this way support the architect in its decision making process. To do so, a more elaborate representation of decision quality is needed, such that different decision can be compared with each other. We see this are promising future work.

The integrity constraints that we have defined in this work are not meant to be a complete list. As we discussed above, each decision in the metamodel of Plataniotis *et al.* is either

[6] http://www.cs.toronto.edu/km/istar/

[7] http://www.troposproject.org/

Executed or *Rejected*. Kruchten *et al.* [KLvV06] argue that design decisions evolve in a manner that may be described by a state machine or a statechart. They distinguish between seven different states, which are *idea, tentative, decided, approved, rejected, challenged,* and *obsolete*. Having such an expressive representation of a decision allows for more complex constraints on the decision making process. This is another direction of future work.

Finally, one of the biggest challenges in decision capturing is the problem of return of capturing effort. The fact that it takes architects much time to capture design making strategies without having a direct benefit might be a discouraging factor. We believe that our approach simplifies the capturing effort by assisting the architect in its decision making process. Part of our future research will focus on evaluating the actual practical usefulness of our approach.

3.6.3 Conclusion

In this chapter we introduce a logic-based framework for capturing relationships between enterprise architecture decisions. This framework is based on the recent EA Anamnesis by Plataniotis *et al.*. We show that EA Anamnesis has a number of flaws an ambiguities, and we propose a formalization in first-order logic that does not suffer from these flaws. With this formalization, we allow for capturing decision relationship dependencies and consistency checks on additional logical dependencies that we formalized using integrity constraints.

In the previous chapter we ended with a list of eight characteristics of high-level decision making in enterprise architecture, and we suggested to use theories based on bounded rationality as a starting point. Based on the results of the previous chapter, we now provide two directions of research that we feel are not represented in EA Anamnesis, as well as in our logical formalization sufficiently, while they do play an important role in enterprise architecture reasoning about high-level decisions. These two directions of research are:

- *Goals* Both our formalisation and EA Anamnesis do not consider the concept of a goal, or any related motivational attitude such as desires or preferences. This is surprising, given the important role that goals, and especially the process of translating high-level goals into an IT strategy plays in enterprise architecture decision making (see previous chapter). In fact, the ArchiMate language has included the *Motivational Extension*, which allows the user to model concepts such as goals, principles, and values.

- *Planning and Scheduling* Planning can be defined as the process of thinking about and organizing the activities required to achieve a desired goal. Due to the high level of uncertainty, plans in enterprise architecture come with many assumptions and are subject to change. However, both EA Anamnesis and our formalization provide no means of reasoning about the dynamics of commitments in time, nor does it cater for any type of uncertainty in planning.

In the remaining two parts of this thesis we will address both of these shortcomings in turn.

Part II

Goals

4

RationalGRL: A Framework for Argumentation and Goal Modeling

Abstract In this chapter we focus on one of the main activities of an enterprise architect, namely to translate high-level goals into more concrete goals and tasks. We investigate to what extent argumentation techniques can be applied in order to trace back goals and task to underlying arguments. To this end, we develop argument schemes and critical questions by analyzing transcripts of discussions about an information system. We develop the *RationalGRL* logical framework to formalize these argument schemes, and we develop algorithms for the instantiating the argument schemes and answering the critical questions.

4.1 Introduction

Recall from the introduction that the focus of this chapter is on *requirements engineering*, which is an approach to assess the role of a future information system within a human or automated environment. Translating strategic goals into an IT strategy (Characteristics 1 of Chapter 2) falls under the the "early-phase" requirements engineering activities of an information system, which include those that consider how the intended system should meet organizational goals, why it is needed, what alternatives may exist, what implications of the alternatives are for different stakeholders, and how the interests and concerns of stakeholders might be addressed [Yu97b]. This is generally referred to as *goal modeling*.

Goal models are often the result of a discussion process between a group of stakeholders and the enterprise architect. In the introduction of this thesis, we recognized various shortcoming of leaving details of the discussion process out of the resulting goal model. Our goal in this chapter is therefore to develop a framework to include discussions between stakeholders into the goal model, with formal traceability links between elements of the goal model to relevant parts of the discussions.

For completeness, we reiterate the success criteria of our framework, which are also listed in the introduction. Please refer to the introduction for more details on our motivation:

1. It must be able to formally model parts of the discussion process in the early-requirements phase of an information system,

2. It must be able to generate goal models based on the formalized discussions,

3. It should have formal traceability links between goal elements and arguments in the discussions,

In order to formalize discussions of the early requirements phase of an information system (Requirement 1), we use a technique from argumentation and discourse modeling called *argument schemes* [WRM08]. An instantiation of an argument scheme gives a presumptive argument in favor of something. This argument can then be attacked in various way by answering *critical questions*. We analyzing transcripts containing discussions between stakeholders about the architecture of an information system. We annotate the arguments and critical questions we find back in these transcripts. The result is a semi-structured representation of a discussion.

In order to generate goal models based on formalized discussions (Requirement 2), we formalize the argument schemes and critical questions from Requirement 1 in a formal argumentation framework. By using argumentation semantics, we can compute which arguments are accepted and which are rejected. Based on the accepted arguments we can generate a goal model. This goal model has formal traceability links to the underlying (accepted) arguments (Requirement 3). We propose algorithms to apply critical questions, which generate new arguments and may possibly attack previous arguments.

This chapter is organized as follows. Section 4.2 is introductory; it introduces our running example, the Goal-oriented Requirements Language, and argument schemes. In Section 4.3, we develop argument schemes and critical questions by annotating transcripts from discussions about an information system. In Section 4.4 we give various examples of the argument schemes and critical questions we found in the transcripts, and in Section 4.5 we develop a formal model for argument schemes and critical questions. We discuss related work, open issues and conclusions in Section 4.6.

4.2 Background: Goal-oriented Requirements Language and argument schemes

In this section, we first introduce our running example, after which we introduce the Goal-oriented Requirements Language (GRL) [AGH+10], which is the goal modeling language we use to integrate with the argumentation framework. Lastly, we introduce argument schemes, and in particular, we discuss the *practical reasoning argument scheme (PRAS)* [ABC07], which is an argument scheme that is used to form arguments and counter-arguments about situations involving goals. This will be our starting point in the next section.

4.2.1 Running example: Traffic Simulator

Most of the examples in this chapter, as well as the topic of discussion in the transcripts we analyze, come from the traffic simulator design exercise. In this exercise, designers are provided a problem description, requirements, and a description of the desired outcomes. The problem description is given in full in Appendix A, and is summarized as follows: The client of the project is Professor E, who teaches civil engineering at UCI. It is the task of the designer to specify a system in which the professor can teach students how various theories around traffic lights work, such as queuing theory. To this end, a

piece of software has to be developed in which students can create visual maps of an area, regulate traffic, and so forth. The original version of the problem descrption [UCI] is well known in the field of design reasoning since it has been used in a workshop[1], and transcripts of this workshop have been analyzed in detail [PH13]. Although the concepts of traffic lights, lanes, and intersections are common and appear to be simple, building a traffic simulator to represent these relationships and events in real time turns out to be challenging.

4.2.2 Goal-oriented Requirements Language (GRL)

GRL is a visual modeling language for specifying intentions, business goals, and *non-functional requirements* of multiple stakeholders [AGH+10]. GRL is part of the User Requirements Notation, an ITU-T standard, that combines goals and non-functional requirements with functional and operational requirements (i.e. use case maps) in one. GRL can be used to specify alternatives that have to be considered, decisions that have been made, and rationales for making decisions. A GRL model is a connected graph of intentional elements that optionally are part of actors. All the elements and relationships used in GRL are shown in Figure 4.1.

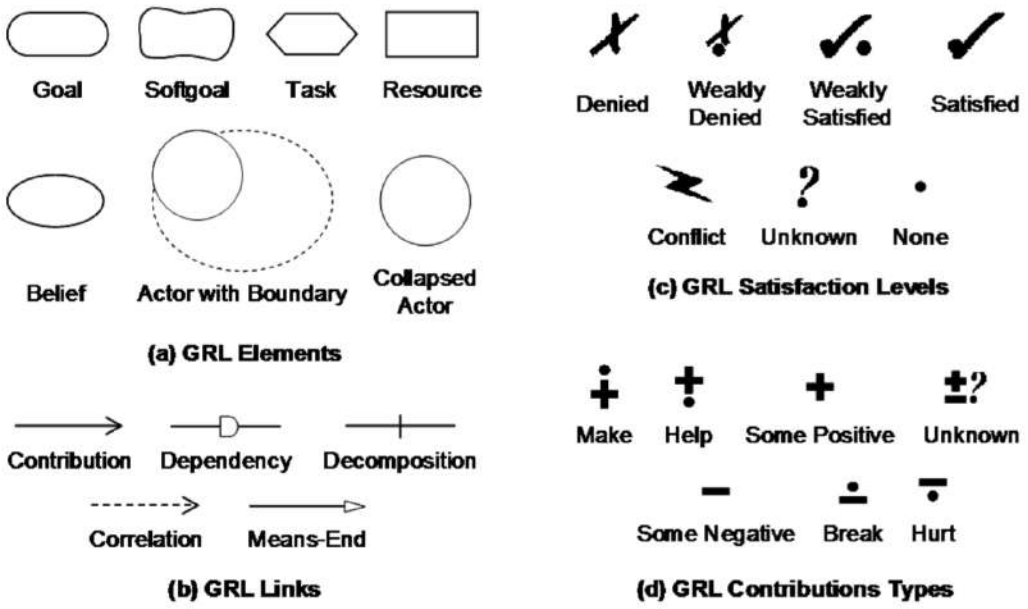


Figure 4.1: Basic elements and relationships of GRL

Figure 4.2 illustrates a GRL diagram from the traffic simulator design exercise. An actor ( ) represents a stakeholder of a system (`Student`, Figure 4.2), or the system itself (`Traffic Tycoon`, Figure 4.2). Actors are holders of intentions; they are the active entities in the system or its environment who want goals to be achieved, tasks to be performed, resources to be available, and softgoals to be satisfied. Softgoals ( ) differentiate themselves from goals ( ) in that there is no clear, objective measure of satisfaction for a softgoal whereas a goal is quantifiable, often in a binary way. Softgoals (e.g. `Realistic simulation`) are often more related to non-functional requirements, whereas goals (such as `Generate cars`) are more related to functional requirements.

[1] `http://www.ics.uci.edu/design-workshop/`

Tasks (⬭) represent solutions to (or operationalizations of) goals and softgoals. In Figure 4.2, some of the tasks are `Create new cars` and `Keep same cars`. In order to be achieved or completed, softgoals, goals, and tasks may require resources (▭) to be available (e.g., `External Library`, Figure 4.2).

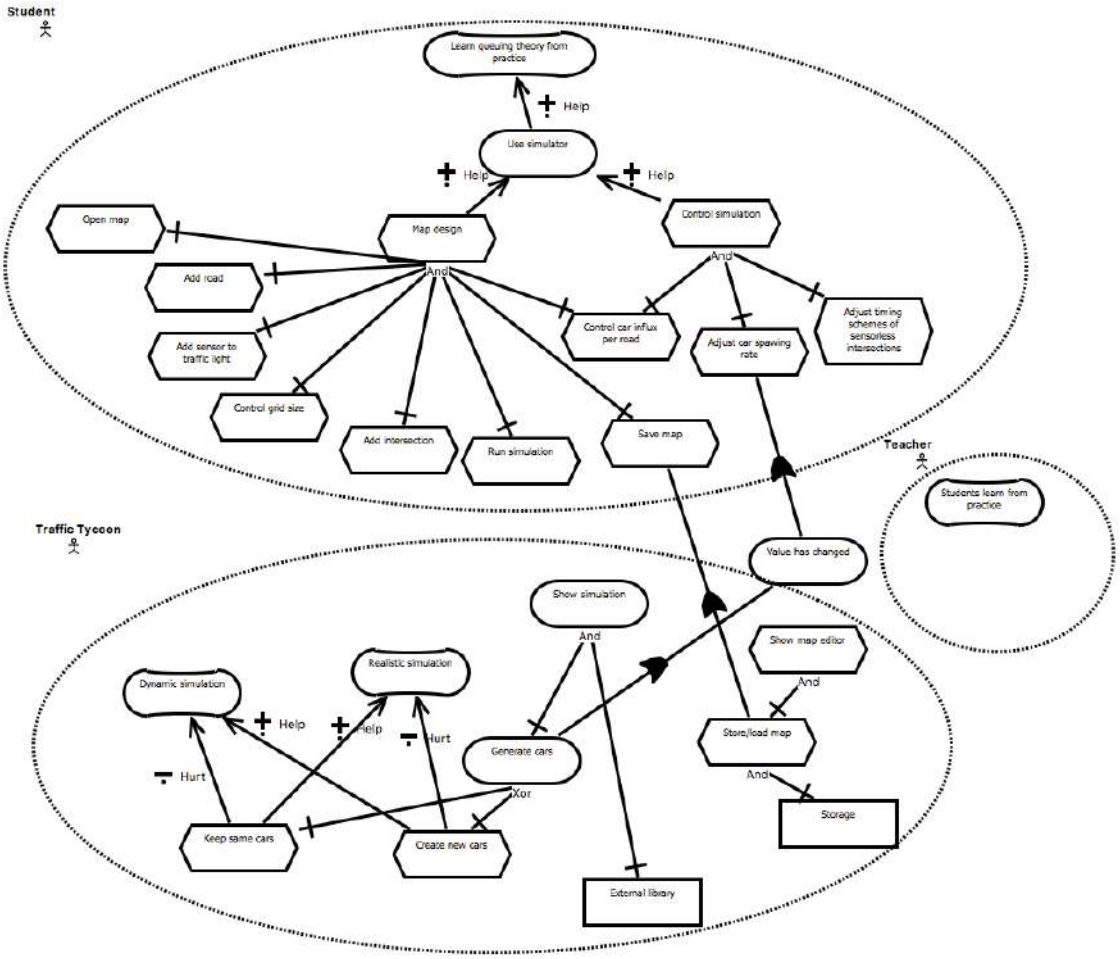


Figure 4.2: GRL Model for the traffic simulator.

Different links connect the elements in a GRL model. AND, IOR, and XOR decomposition links (⊣—) allow an element to be decomposed into sub-elements. In Figure 4.2, the goal `Generate cars` is XOR-decomposed to the tasks `Create new cars` and `Keep same cars`. Contribution links (⟶) indicate desired impacts of one element on another element. A contribution link has a qualitative contribution type or a quantitative contribution. Task `Create new cars` has a *help* qualitative contribution to the softgoal `Dynamic simulation`. Dependency links (⊢●—) model relationships between actors. For example, actor `Traffic Tycoon` depends on the actor `Student` to perform the task `Adjust car spawning rate` to fulfill its task `Generate cars`.

GRL is based on $i*$ [Yu97a] and the NFR Framework [CNYM12], but it is not as restrictive as $i*$. Intentional elements and links can be more freely combined, the notion of agents is replaced with the more general notion of actors, i.e., stakeholders, and a task does not necessarily have to be an activity performed by an actor, but may also describe properties of a solution. GRL has a well-defined syntax and semantics, which are necessary if we want to incorporate it into a formal framework (requirements 1 and 2 as described in the introduction). Furthermore, GRL provides support for providing

a scalable and consistent representation of multiple views/diagrams of the same goal model (see [Gha13, Ch.2] for more details). GRL is also linked to Use Case Maps via URNLink (▶) which provides traceability between concepts and instances of the goal model and behavioral design models. Multiple views and traceability are a good fit with our current research: we aim to add traceability links between intentional elements and their underlying arguments.

GRL has six evaluation algorithms which are semi-automated and allow the analysis of alternatives and design decisions by calculating the satisfaction value of the intentional elements across multiple diagrams quantitatively, qualitatively or in a hybrid way. The satisfaction values from intentional elements in GRL can also be propagated to use case maps elements. jUCMNav, GRL tool-support, also allows for adding new GRL evaluation algorithms [MA09]. GRL also has the capability to be extended through metadata, links, and external OCL constraints. This allows GRL to be used in many domains without the need to change the whole modeling language. This feature also helps us to apply our argumentation to other domain such as compliance, which we explain in more detail in Section 4.6.2.

The GRL model in Figure 4.2 shows the softgoals, goals, tasks and the relationship between the different intentional elements in the model. However, the rationales and arguments behind certain intentional elements are not shown in the GRL model. Some of the questions that might be interesting to know about are the following:

- Why does actor `Teacher` have only a single softgoal `Students learn from practice`? Why is this, for instance, not connected to any of the elements of `Student`?
- What does `Adjust timing schemes of sensorless interactions` mean?
- Why does task `Keep same cars` contribut positively to `Realistic simulation` and negatively to `Dynamic simulation`?
- How does the `Student` control the `Traffic Tycoon`?
- Why does `Map design` have so many decompositions into other tasks?

These are the type of the questions that we cannot answer just by looking at the GRL models. The model in Figure 4.2 does not contain information about discussions that let up to the resulting elements of the model, such as various clarification steps for the naming, or alternatives that have been considered for the relationships. In this chapter we aim to address this shortcoming.

4.2.3 Argument Scheme for Practical Reasoning (PRAS)

Reasoning about which goals to pursue and actions to take is often referred to as *practical reasoning*, and has been studied extensively in philosophy (e.g. [Raz78, Wal90]) and artificial intelligence [Bra87, ABC07]. One approach is to capture practical reasoning in terms of arguments schemes and critical questions [Wal90]. The idea is that an instantiation of such a scheme gives a presumptive argument in favor of, for example, taking an action. This argument can, then, be tested by posing critical questions about,

for instance, whether the action is possible given the situation, and a negative answer to such a question leads to a counterargument to the original presumptive argument for the action.

A formal approach to persuasive and deliberative reasoning about goals and actions has been presented by Atkinson *et al* [ABC07], who define the *practical reasoning argument scheme* (PRAS). PRAS follows the following basic argument structure.

> We have goal *G*,
>
> Doing action *A* will realize goal *G*,
>
> Which will promote the value *V*
>
> *Therefore*
>
> We should perform action *A*

So, for example, we can say that

> We have goal `Generate traffic`,
>
> `Keep same cars` will realize goal `Generate traffic`,
>
> Which will promote the value `Simple design`
>
> *Therefore*
>
> We should perform action `Keep same cars`

Practical reasoning is defeasible, in that conclusions which are at one point acceptable can later be rejected because of new information. Atkinson *et al.* [ABC07] define a set of critical questions that point to typical ways in which a practical argument can be criticized by, for example, questioning the validity of the elements in the scheme or the connections between the elements. Some examples of critical questions are as follows.

1. Will the action bring about the desired goal?
2. Are there alternative ways of realizing the same goal?
3. Are there alternative ways of promoting the same value?
4. Does doing the action have a side effect which demotes some other value?
5. Does doing the action promote some other value?
6. Is the action possible?
7. Can the desired goal be realized?
8. Is the value indeed a legitimate value?

These critical questions can point to new arguments that might counter the original argument. Take, for example, critical question 4: if we find that `Keep same cars` actually negatively influences the value `Realistic simulation`, we have a counterargument to the above argument. Another way to counter an argument for an action is to suggest an alternative action that realizes the same goal (question 2) or an alternative goal that promotes the same value (question 3). For example, we can argue that `Create new cars` also realizes the goal `Generate traffic`, which gives us a counterargument to the original argument – to generate traffic by simply keeping the cars that disappear off the screen and have them wrap around to the other side of the screen – that also follows PRAS.

In argumentation, counterarguments are said to *attack* the original arguments (and sometimes vice versa). In the work of Atkinson et al. [ABC07], arguments and their attacks are captured as an *argumentation framework* of arguments and attack relations as introduced by Dung [Dun95][2]. Figure 4.3 shows an argumentation framework with three arguments from the above example: the argument for `Keep same cars` (A1), the argument for `Create new cars` (A3), and the argument that `Keep same cars` demotes the value `Realistic simulation` (A2). The two alternative PRAS instantiations are A1 and A3. These arguments mutually attack each other, as `Keep same cars` and `Create new cars` are considered to be mutually exclusive. Argument A2 attacks A1, as it points to a negative side-effect of `Keep same cars`.

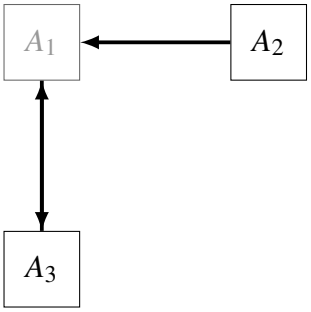


Figure 4.3: Example argumentation framework.

Given an argumentation framework, the acceptability of arguments can be determined according to the appropriate argumentation semantics. The intuition is that an argument is acceptable if it is *undefeated*, that is, any argument that attacks it, is itself defeated. In the argumentation framework in Figure 4.3, argument A2 is undefeated because it has no attackers. This makes A1 defeated, because one of its attackers, A2, is undefeated. A3 is then also undefeated, since its only attacker, A1, is defeated by A2. Thus, the set of undefeated (justified) arguments given the argumentation framework in Figure 4.3 is {A2, A3}, corresponding to arguments for `Realistic simulation` and `Create new cars`.

Practical Argumentation and Goal Modeling

Practical reasoning in the PRAS framework as described above provides a formal framework for defeasible reasoning about goals and actions that adheres to the acceptability

[2]Full definitions of Dung's [Dun95] frameworks and semantics will be given in section 4.3. In this section, we will briefly discuss the intuitions behind these semantics.

semantics of Dung [Dun95] and its various extensions [AC02, Mod09]. The usefulness of PRAS for the analysis of practical reasoning situations has been shown in different areas such as e-democracy [CA09], law [ABC05], planning [MGABCM13] and choosing between safety critical actions [TMA+12]. In this chapter, we aim at capturing the stakeholder's discussions as formal argumentation based on PRAS to decide whether intentional elements and their relationships are shown in the resulting goal model. This give a rationalization to the elements of the goal model in terms of underlying arguments, and furthermore, it allows one to understand why certain other elements have been rejected.

Argumentation schemes and their associated critical questions are very well suited for modeling discussions about a goal model: as Murukannaiah et al. [MKTS15a] have shown, they can guide users in systematically deriving conclusions and making assumptions explicit. This can also be seen from the obvious similaries between PRAS (actions, goals, values) and GRL (tasks, goals, softgoals) in the example above.

However, there are also some differences between PRAS and GRL. Not all elements and relationships of GRL fit into PRAS. For instance, PRAS does not have a notion of "resource", and many of the relationships of GRL do not occur in PRAS. Furthermore, it is not directly clear whether the critical questions as proposed by Atkinson actually apply to GRL. Therefore, we develop our own set of argument schemes and critical questions in the next section by analyzing transcripts of discussions about the traffic simulator.

4.3 Argument Schemes for Goal Modeling

In this section, we develop a set of *argument schemes for goal modeling* and associated critical questions. We start from an initial list that we derive from PRAS, containing argument schemes and critical questions that are specific to elements and relationships of GRL. We then refine this list incrementally by annotating transcripts, meaning we identify and count instantiations of arguments schemes. If we find new argument schemes or critical questions, we add them to the list. If some of them do not occur at all, we remove them from the list. We then manually construct goal models from the arguments and counter arguments we find in the transcripts, and add traceability links between the goal model and underlying arguments. An example is shown in Figure 4.4, which is the same as the example in Figure 4.2, but now including tracability links to underlying arguments. A red dot indicates the underlying argument is rejected, and a green dot means the underlying argument is accepted.

The final list of argument schemes and critical questions that we end up with after our analysis is shown in Table 4.1. The initial list of argument schemes, i.e., those we start with before annotating the transcripts, consists of AS1-AS4, AS6-AS9 (Table 4.1), and their corresponding critical questions. The first four argument schemes (AS0-AS4) are arguments for an element of a goal model, the next seven (AS5-AS11) are about relationships, the next two (AS12-AS13) are about intentional elements in general, and the last is (Att) is a generic counterargument for any type of argument that has been put forward.

We found that answering critical questions can have varying effects on the model, and for each critical question, the right column in Table 4.1 shows the effect of answering the

critical questions affirmatively. Answering a critical questions can create an argument disabling the corresponding GRL element of the attacked argument scheme (DISABLE); it can create an argument introducing a new GRL element (INTRO); it can replace the GRL element corresponding to the original argument (REPLACE), or it can simply attack an argument directly (ATTACK).

In the first subsection of this section, we provide details of the transcript annotation process with concrete examples (see Appendix B for transcript excerpts), after which we analyze our results in the second subsection.

Argument scheme		**Critical Questions**		**Effect**
AS0	Actor a is relevant	CQ0	Is the actor relevant?	DISABLE
AS1	Actor a has resource R	CQ1	Is the resource available?	DISABLE
AS2	Actor a can perform task T	CQ2	Is the task possible?	DISABLE
AS3	Actor a has goal G	CQ3	Can the desired goal be realized?	DISABLE
AS4	Actor a has softgoal S	CQ4	Is the softgoal a legitimate softgoal?	DISABLE
AS5	Goal G decomposes into tasks $T_1, \ldots, T_n$	CQ5a	Does the goal decompose into the tasks?	DISABLE
		CQ5b	Does the goal decompose into any other tasks?	REPLACE
AS6	Task T contributes to softgoal S	CQ6a	Does the task contribute to the softgoal?	DISABLE
		CQ6b	Are there alternative ways of contributing to the same softgoal?	INTRO
		CQ6c	Does the task have a side effect which contribute negatively to some other softgoal?	INTRO
		CQ6d	Does the task contribute to some other softgoal?	INTRO
AS7	Goal G contributes to softgoal S	CQ7a	Does the goal contribute to the softgoal?	DISABLE
		CQ7b	Does the goal contribute to some other softgoal?	INTRO
AS8	Resource R contributes to task T	CQ8	Is the resource required in order to perform the task?	DISABLE
AS9	Actor a depends on actor b	CQ9	Does the actor depend on any actors?	INTRO
AS10	Task T_1 decomposes into tasks $T_2, \ldots, T_n$	CQ10a	Does the task decompose into other tasks?	REPLACE
		CQ10b	Is the decomposition type correct? (AND/OR/XOR)	REPLACE
AS11	Task T contributes negatively to softgoal S	CQ11	Does the task contribute negatively to the softgoal?	DISABLE
AS12	Element IE is relevant	CQ12	Is the element relevant/useful?	DISABLE
AS13	Element IE has name n	CQ13	Is the name clear/unambiguous?	REPLACE
-	-	Att	Generic counterargument	ATTACK

Table 4.1: List of argument schemes (AS0-AS13, left column), critical questions (CQ0-CQ13, middle column), and the effect of answering them (right column).

4.3.1 Details experiment

The transcripts we used are created as part of two master theses on improving design reasoning [Sch16, Riz16].

Subjects The subjects for the case study are three teams of Master students from the University of Utrecht, following a Software Architecture course. Two teams consist of three students, and one team consists of two students.

Experimental Setup The assignment used for the experiments is to design a traffic simulator. Participants were asked to use a think-aloud method during the design session. The assignment was slightly adjusted to include several viewpoints as end products in order to conform to the course material [BCK12]. The full problem descriptions can be found in Appendix A of this thesis. All groups were instructed to apply the *functional architecture method*, focusing on developing the *context*, the *functional*, and the *informational* viewpoints of the traffic simulator software. The students had two hours for the tasks, and the transcripts document the entire discussion. The details of the transcripts are shown in Table 4.2.

	transcript t_1	transcript t_2	transcript t_3
participants	2	3	3
duration	1h34m52s	1h13m39s	1h17m20s

Table 4.2: Number of participants and duration of the transcripts.

Annotation Method We started with an initial list of 8 argument schemes and 18 critical questions that we derived from PRAS (AS1-AS4, AS6-AS9 of Table 4.1). We annotated transcripts with the arguments and critical questions from this list. If we found arguments or critical questions that did not appear in the original list, we added them and counted them as well. Argument schemes that did not appear were removed from the list, but critical questions were not removed (see discussion in Section 4.3.2). Most of the occurrences were not literally found back, but had to be inferred from the context. This can be seen in the various examples we will discuss.

It is generally known in the argumentation literature that it can be very difficult to identify arguments in natural language texts [WRM08]. Arguments are often imprecise, lack conclusion, and may be supported by non verbal communication that is not captured in the transcripts. However, there is hardly any research on on argument extraction in the requirement engineering domains, so despite this potential weakness in our approach, we believe it nevertheless is at least useful as something that others can build further on (see Section 4.6.2).

Results All original transcripts, annotations, and models are available on the Github page of this thesis:

`http://www.github.com/marcvanzee/RationalArchitecture`

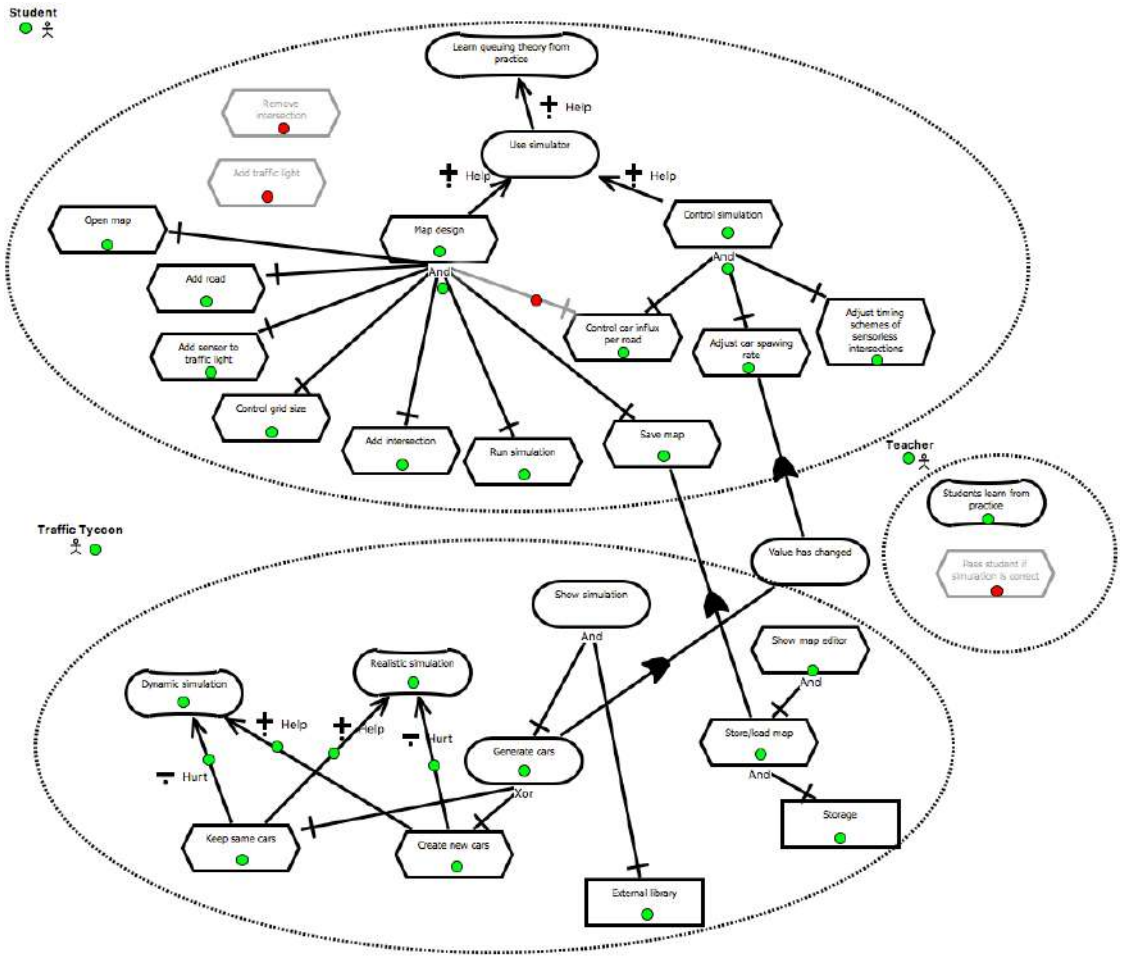


Figure 4.4: The GRL model manually constructed from transcript t_1. Green dots indicate accepted underlying arguments, red dots indicate rejected underlying arguments. Elements and relationships with no dot have been inferred by us.

in the folder "Ch4 transcript analysis". We also provide excerpts of the annotation in Appendix B, and most of the examples we use in this chapter come from the transcripts.

We found a total of 159 instantiations of the argument schemes AS0-AS11 in the transcripts. The most used argument scheme was AS2: "Actor A has task T", but each argument scheme has been found back in the transcripts at least twice (Table 4.3). A large portion (about 60%) of the argument schemes we found involved discussions around tasks of the information system (AS2, AS10).

We annotated 41 applications of critical questions. Many critical questions (about 55%) involved clarifying the name of an element, or discussing the relevance of it (CQ12, CQ13).

For each transcript, we manually created a GRL model from the argument schemes and critical questions we found in them, in order to verify whether the arguments put forward by the participants were sufficiently informative. An example of such a model is shown in Figure 4.4. We added green and red dots to various elements and relationships in the figure. A green dot indicates there is an underlying argument for the element that is accepted, while a red dot indicates a rejected underlying argument. Note that if the underlying argument is rejected, the corresponding GRL element has been disabled. Some

Scheme/Question		t_1	t_2	t_3	total
AS0	Actor	2	2	5	**9**
AS1	Resource	2	4	5	**11**
AS2	Task/action	20	21	17	**58**
AS3	Goal	0	2	2	**4**
AS4	Softgoal	3	4	2	**9**
AS5	Goal decomposes into tasks	4	0	4	**8**
AS6	Task contributes to softgoal	6	2	0	**8**
AS7	Goal contributes to softgoal	0	1	1	**2**
AS8	Resource contributes to task	0	4	3	**7**
AS9	Actor depends on actor	0	1	3	**4**
AS10	Task decomposes into tasks	11	14	11	**36**
AS11	Task contributes negatively to softgoal	2	1	0	**3**
CQ2	Task is possible?	2	2	1	**5**
CQ5a	Does the goal decompose into the tasks?	0	1	0	**1**
CQ5b	Goes decomposes into other tasks?	1	0	0	**1**
CQ6b	Task has negative side effects?	2	0	0	**2**
CQ10a	Task decompose into other tasks?	1	2	0	**3**
CQ10b	Decomposition type correct?	1	0	1	**2**
CQ12	Is the element relevant/useful?	2	3	2	**7**
CQ13	Is the name clear/unambiguous?	3	10	3	**16**
-	Generic counterargument	0	2	2	**4**
TOTAL		69	80	69	**222**

Table 4.3: Number of occurrences of argument schemes and critical questions in the transcripts. Critical questions not appearing in this table were not found back in the transcripts.

elements do not have a corresponding green or red dot. In that case, we have inferred the elements from the discussion, but we could not explicitly find back arguments for it.

We found that answering a critical questions can have four different effects on the original argument and the corresponding GRL element:

- INTRO: Introduce a new goal element or relationship with a corresponding argument. This operation does not attack the original argument of the critical question, but rather creates a new argument. For instance, suppose argument scheme AS5 is instantiated as follows:: "Goal `Generate cars` OR-decomposes into tasks `Keep same cars` and `Create new cars`". Suppose now the critical question CQ5b: "Does the goal `Agreeable meeting dates` decompose into other tasks?" is answered with "yes, namely `Choose randomly`". This results in a new instantiation of AS5, namely: "Goal `Generate cars` OR-decomposes into tasks `Keep same cars`, `Create new cars`, and `Choose randomly`. As a result, the goal model will contain the corresponding task `Choose randomly`, as well as an OR-decomposition relation from the goal `Generate cars` to that task.

- DISABLE: Disable the element or relationship of the argument scheme to which the critical questions pertains. This operation does not create a new argument, but only disables (i.e., attacks) the original one. For instance, suppose argument scheme AS0 is instantiated with: "Actor `Teacher` is relevant". This argument can be attack with critical question CQ0: "Actor `Teacher` is not relevant". As a result,

the argument for the actor is attack, and actor `Teacher` is disabled in the goal model.

- REPLACE: Replace the element of the argument scheme with a new element. This operation both introduces a new argument and attacks the original one. For instance, suppose argument scheme AS2 is instantiated with: "Actor `Student` can perform task `Choose a pattern preference`. This argument can be attacked with critical question CQ13: "The task `Choose a pattern preference` is unclear, it should be `Choose a road pattern`". This results in replacing the original argument with the new argument "Actor `Student` can perform task `Choose a road pattern`. In the goal model, the description of the task should change accordingly.

- ATTACK: Attack any argument with an argument that cannot be classified as a critical question. For instance, suppose argument scheme AS0 is instantiated with "Actor `Teacher` is relevant". Suppose this argument is attacked with critical question CQ0: "Actor `Teacher` is not relevant", because she does not create the system. However, this critical question may in turn be attacked again, if new evidence comes up. For instance, it may turn out that the teacher will work on developing the application herself. In this case, the generic counter argument "Actor `Teacher` is relevant, because does develop for the application" attack the argument "Actor `Teacher` is not relevant". As a result, the original argument "Actor `Teacher` is relevant" is accepted again, and as a result is shown in the goal model.

In Section 4.4 we provide examples we found in the transcripts for all of these four effects.

4.3.2 Analysis

Analysis of the Argument Schemes Recall that our initial list of argument schemes consists of AS1-AS4, AS6-AS9 (Table 4.1). Therefore, the difference between the initial list of argument schemes and those found back in the transcripts is quite small. We found it surprising that we were able to find back all the schemes in the transcript at least twice, even more since the topic of discussion was not goal models, but more generally the architecture of an information system. This gives us an indication that it is possible to capture (parts of the) arguments used in those type of discussions using argument schemes.

We observed that our initial list is rather limited, which is a consequence of the fact that it is derived from PRAS. Since PRAS only considers very specific types of relationships, we are not able to capture many other relationships existing in GRL. GRL has four types of intentional elements (softgoal, goal, task, resource) and four types of relationships (positive contribution, negative contribution[3], dependency, decomposition), allowing theoretically $4^3 = 64$ different types of argument schemes, of which we currently only consider 11. Our analysis however shows that many of these schemes are not often used, and thus, gives us some confidence in the resulting list. However, if

[3]In fact, a contribution can be any integer in the domain [-100,100], but for the sake of simplicity we only consider two kinds of contributions here.

needed, additional argument schemes and critical questions can be added, and our list is not meant to be exhaustive.

Analysis of the Critical Questions The difference between the initial list of critical questions and those we found back in the transcripts is much larger than for the critical questions. We found few of the critical questions we initially proposed. However, this does not mean that they were not implicitly used in the minds of the participants. If a participants for instance forms an argument for a contribution from a task to a softgoal, it may very well be that she was asking herself the question "Does the task contribute to some other softgoal?". However, many of these critical questions are not mentioned explicitly. If we assume this explanation is at least partially correct, then this would mean that critical questions may still play a role when formalizing the discussions leading up to a goal model, and it would be limiting to leave them out of our framework. In the context of tool support, we believe that having these critical questions available may stimulate discussions.

4.4 Examples

We now discuss various instantiations of argument schemes and the result of answering critical questions in more detail. For each example we provide transcript excerpts, a visualization of the arguments, and the corresponding goal model elements. We provide a legend for our visualization notation in Figure 4.5.

Example 1: Disable task `Traffic light`

The transcript excerpt of this example is shown in Table B.2 in the appendix and comes from transcript t_1. In this example, participants are summing up functionality of the traffic simulator, which are tasks that the student can perform in the simulator. All these task can be formalized and are instantiations of argument scheme AS2: "Actor *Student* has tasks T", where $T \in \{$Save map, Open map, Add intersection, Remove intersection, Add road, Add traffic light$\}$".

Once all these tasks are summed up, participant `P1` notes that the problem description states that all intersections in the traffic simulator have traffic lights, so the task `Add traffic light` is not useful. We formalized this using the critical question CQ12: "Is task `Add traffic light` useful/relevant?".

We visualize some of the argument schemes, critical questions, and traceability links with the GRL model in Figure 4.6. On the left side of the image, we see three of the instantiated argument schemes AS2. The bottom one, "Actor `Student` has task `Add traffic light`", is attacked by another argument generated from applying critical question CQ12: "`Add traffic light` is useless (*All intersections have traffic lights*). As a result, the corresponding GRL task is disabled. The other two tasks are enabled and have green traceability links.

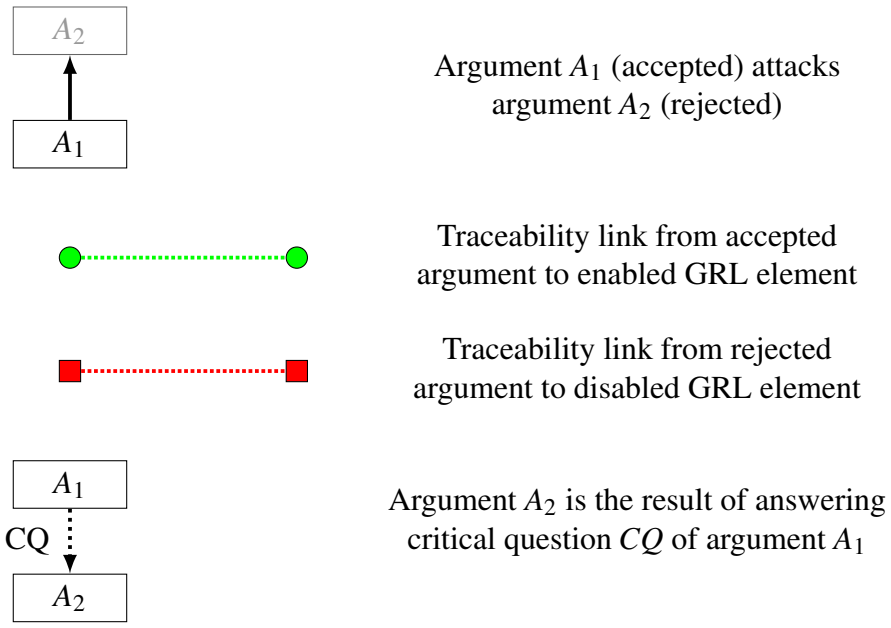


Figure 4.5: Legend of the various elements and relationships we use for the examples in this article.

Example 2: Clarify task `Road pattern`

The transcript excerpt of the second example is shown in Table B.1 in Appendix B and comes from transcript t_3. It consists of a number of clarification steps, resulting in the task `Choose a road pattern`.

The formalized argument schemes and critical questions are shown in Figure 4.7. The discussion starts with the first instantiation of argument scheme AS2: "Actor `Student` has task `Create road`". This argument is then challenged with critical question CQ12: "Is the task `Create road` clear?". Answering this question results in a new instantiation of argument scheme AS2: "Actor `Student` has task `Choose a pattern`". This process is repeated two more times, resulting in the final argument "Actor `Student` has task `Choose a road pattern`". This final argument is unattacked and has a corresponding intentional element (right image).

What is clearly shown in this example is that a clarifying argument attacks all arguments previously used to describe the element. For instance, the final argument on the bottom of Figure 4.7 attacks all three other arguments for a name of the element. If this was not the case, then it may occur that a previous argument is *reinstatiated*, meaning that it becomes accepted again because the argument attacking it is itself attacked. Suppose for instance the bottom argument "Actor `Student` has task `Choose a pattern preference`" did not attack the second argument: "Action `Student` has task `Choose`

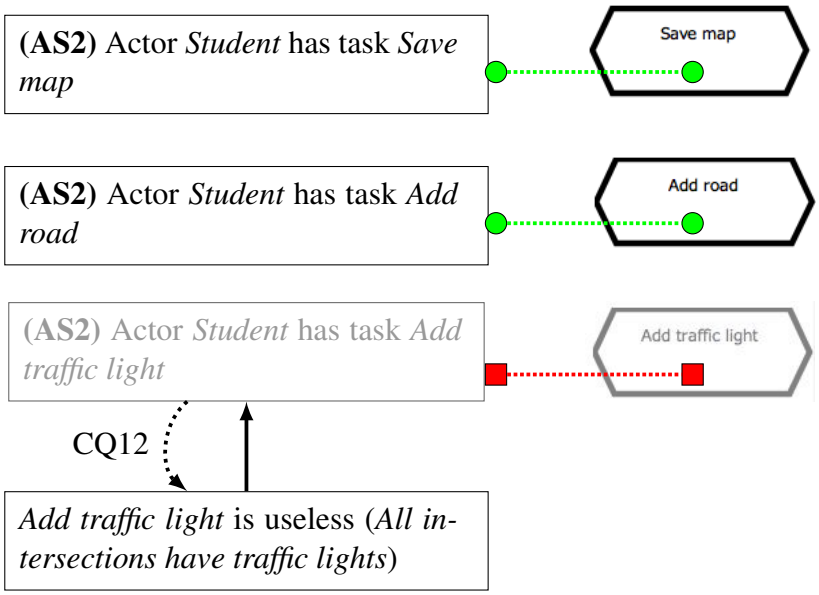


Figure 4.6: Argument schemes and critical questions (left), GRL model (right), and traceability link (dotted lines) for the traffic light example.

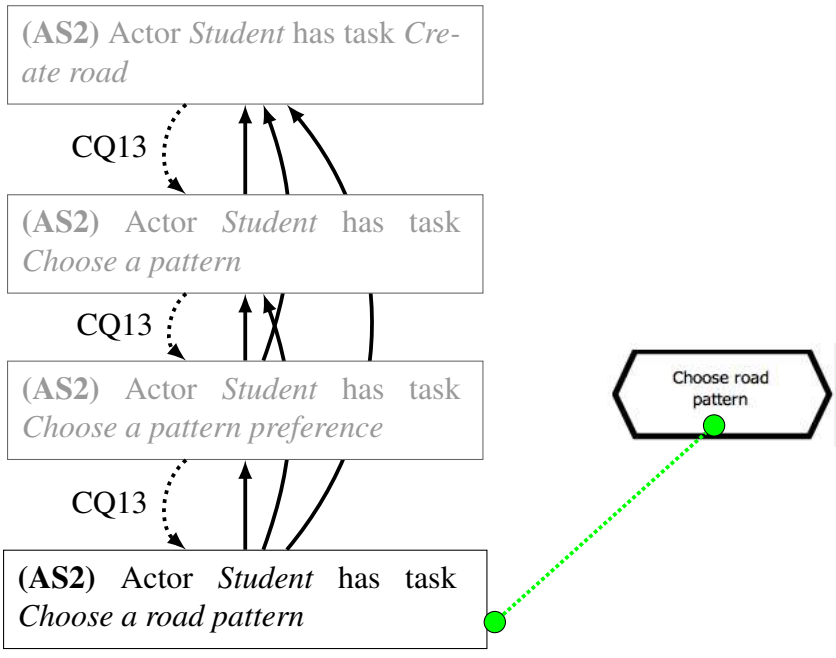


Figure 4.7: Argument schemes and critical questions (left), GRL model (right), and traceability link (dotted line) of the road pattern example.

`a pattern`". In that case, this argument would be reinstated, because its only attacker "Actor `Student` has task `Choose a pattern preference`" is itself defeated by the bottom argument.

Example 3: Decompose goal `Simulate`

The transcript excerpt of this example is shown in Table B.3 in the appendix and comes from transcript t_3. It consists of a discussion about the type of decomposition relationship for the goal `Simulate`.

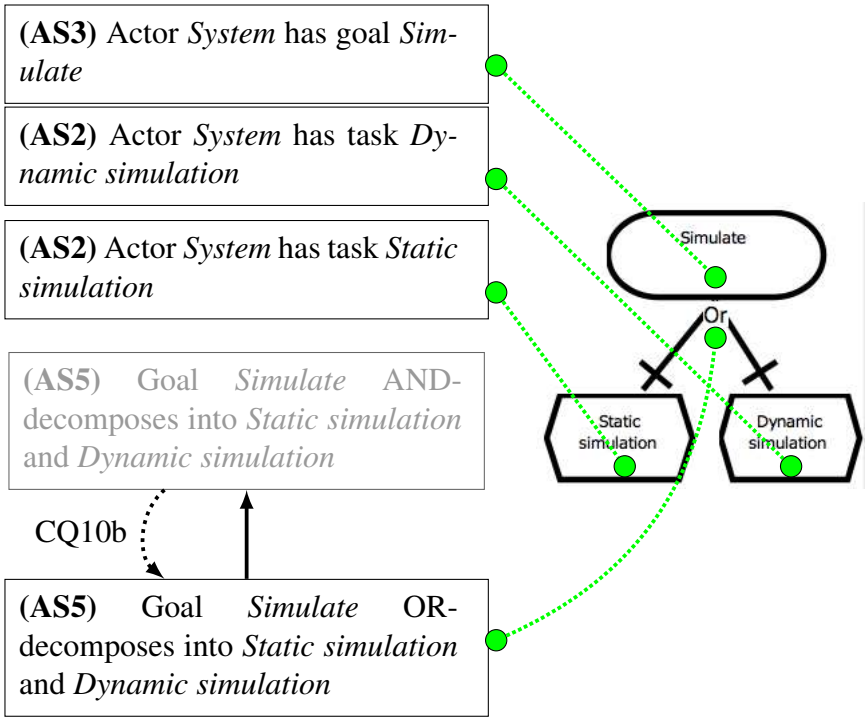


Figure 4.8: Argument schemes and critical questions (left), GRL model (right), and traceability link (dotted line) of the goal decomposition example.

The visualization of this discussion is shown in Figure 4.8. Each GRL element on the right has a corresponding argument on the left. Moreover, the original argument for an AND-decomposition is attacked by the argument for the OR-decomposition, and the new argument is linked to the decomposition relation in the GRL model.

Example 4: Reinstate actor `Development team`

The transcript excerpt of this example is shown in Table B.4 in the appendix and comes from transcript t_3. It consists of two parts: first participant `P1` puts forth the suggestion to include actor `Development team` in the model. This is, then, questioned by participant `P2`, who argues that the professor will develop the software, so there won't be any development team. However, in the second part, participant `P2` argue that the development team should be considered, since the professor does not develop the software.

We formalize this using a *generic counterargument*, attacking the critical question. The first part of the discussion is shown in Figure 4.9. We formalize the first statement as an instantiation of argument scheme AS0: "Actor `development team` is relevant". This argument is, then, attacked by answering critical question CQ0: "Is actor `development team` relevant? with *No*. This results in two arguments, AS0 and CQ0, where CQ0 attacks AS0. This is shown in Figure 4.9, left image.

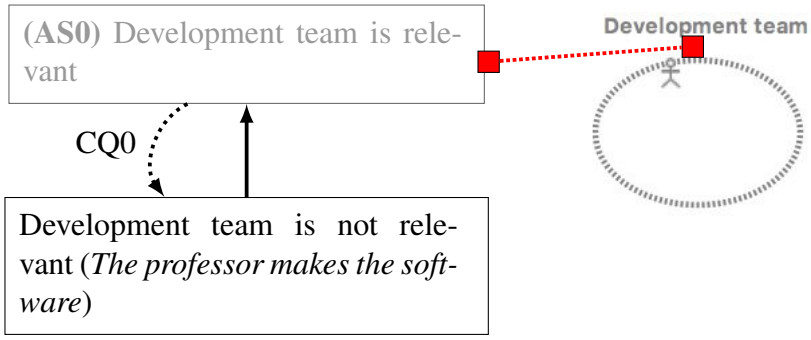


Figure 4.9: Argument schemes and critical questions (left), GRL model (right), and traceability link (dotted line) of a discussion about the relevance of actor Development team.

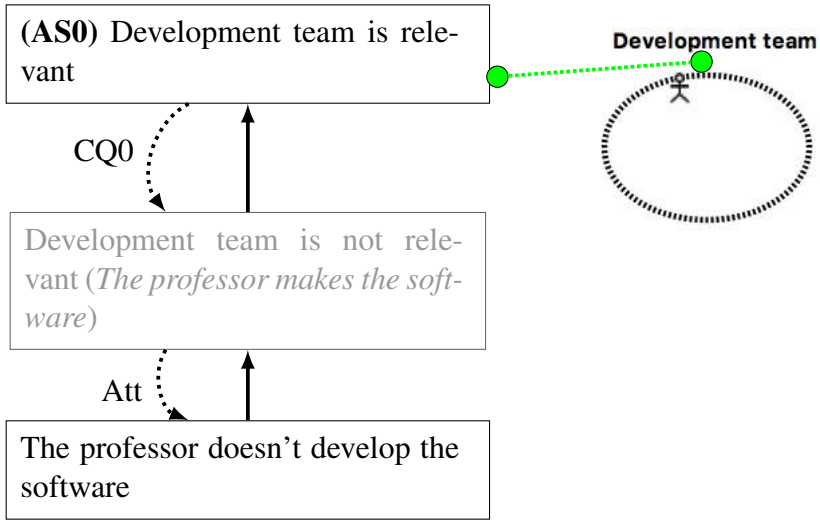


Figure 4.10: Argument schemes and critical questions (left), GRL model (right), and traceability link (dotted line) of a discussion about the relevance of actor Development team.

Figure 4.10 shows the situation after the counter argument has been put forward. The argument "Actor `Professor` doesn't develop the software" now attacks the argument "`Development team` is not relevant (*The professor makes the software*)", which in turn attacks the original argument "`Development team` is relevant". As a result, the first argument is reinstated, which causes the actor in the GRL model to be enabled again.

4.5 RationalGRL: the logical framework

In the previous section we developed a list of critical questions and argument schemes by analyzing transcripts of discussions about the development of a traffic simulator. The resulting list is shown in Table 4.1. We also discussed various examples of applications of the different critical questions and all four different effects (right column of Table 4.1): INTRO, DISABLE, REPLACE, and ATTACK.

The examples and corresponding visualizations of the previous section provide some insight on how to formalize argument schemes, critical questions, and their relationship

with goal models. However, if we are to implement our framework in a tool, we require a more precise formalization of these concepts. Motivated by our approach in Chapter 2 we choose to use formal logic to specify this. This is also a good fit with formal argumentation, which has flourished in the past decades, leading to a large number of different semantics that we can choose from.

The rest of this section is as follows: In the first subsection, we develop a formal language to specify a GRL model. This language consists of atomic sentences. In the second subsection, we develop our notion of an argument: an argument is simply a set of atoms from our language. In the third subsection, we develop algorithms for all the argument schemes and critical questions.

4.5.1 Logical Language for RationalGRL

We define our language as a set of atoms, that is, grounded formulas (without variables) with no logical connectives or negation.

Definition 4.1 (GRL Atoms)**.** *Let P be a set of sentences representing names for GRL elements. The* set of GRL atoms *At is defined as follows:*

$$At = \{actor(i), softgoal(i), goal(i), task(i), resource(i), decomp(k,i,J,dtype), \\ dep(k,i,j), contr(k,i,j,ctype), has(i,j), disabled(i), name(i,p)\},$$

where

- $\{i,j,k\} \cup J \subseteq \mathbb{N}$
- $p \in P$
- $dtype \in \{and, or, xor\}$
- $ctype \in \{+,-\}$

The non-negative integers associates with each element and relation are identifiers. We thus choose to identify each element and relationship in a GRL model with an identifier. This allows us to change the name of an element while still being able to make different statements about this element, for instance by applying critical question CQ12a (clarification, see the example "Clarify task `Road pattern`" above).

We now briefly explain each atom in more detail.

- $actor(i)$: Identifier i is an actor.
- $softgoal(i)$: Identifier i is a softgoal.
- $goal(i)$: Identifier i is a goal.
- $task(i)$: Identifier i is a task.
- $resource(i)$: Identifier i is a resource.
- $decomp(k,i,J,dtype)$: Identifier k is a $dtype$-decomposition (and/or/xor) from the element corresponding to identifier i to the set of elements corresponding to identifiers J.

- $dep(k,i,j)$: Identifier k is a dependency link from the element corresponding to identifier i to the element corresponding to identifier j.
- $contr(k,i,j,ctype)$: Identifier k is a $ctype$-contribution (+/-) from the element corresponding to identifier i to the element corresponding to identifier j.
- $has(i,j)$: Identifier i (which is an actor) has the element corresponding to identifier j.
- $disabled(i)$: The element or relationships corresponding to identifier i is disabled.
- $name(i,p)$: The name of the element corresponding to identifier i is p.

Remark 2. *From these descriptions one can observe that many of the atoms come with a number of assumptions and dependencies. For instance, if $has(i,j)$ is true, then i is an actor and j is an element. We could formalize this as follows:*

$$has(i,j) \rightarrow (actor(i) \wedge (softgoal(j) \vee goal(j) \vee task(i) \vee resource(j))).$$

It would be possible to enumerate all such properties in order to correctly specify a GRL model. One could then formally verify whether a set of atoms violates these constraints. If not, it is a "valid" representation of a GRL model. Since the focus of this chapter is not on a logical analysis, but rather on developing a framework for empirical data, we leave such a formal analysis for future work (see Section 4.6.2).

Using this formalization, it is rather straightforward to specify a GRL model. An example of the specification of the GRL model in Figure 4.8 is shown in Table 4.4. The specification has been written in logic programming style. In this chapter we do not make use of any logic programming techniques but this would be interesting future work (see Section 4.6.2). A more elaborate example of a specification is shown in Appendix C, showing a complete specification of the traffic simulator GRL model in Figure 4.2.

```
goal(0).
task(1).
task(2).
name(0,simulate).
name(1,static_simulation).
name(2,dynamic_simulation).
decomp(3,0,[1,2],or).
```

Table 4.4: Specification of the GRL model in Figure 4.8

4.5.2 Formal argumentation semantics

In the previous subsection we introduced an atomic language to specify a GRL model. In this subsection we give a formal definition of an argument, which is simply a set of atoms from our language. We introduce Dung's acceptability semantics as well, which allows us to determine sets of acceptable arguments.

Definition 4.2 (Argument)**.** *An* argument $A \subseteq Att$ *is a set of atoms from* Att.

This simple definition allows us to form arguments about (parts of) a GRL model. For instance,

$$\{goal(0), name(0, development\_team)\},$$

is an argument. We next introduce an argumentation framework, which is a set of arguments an attack relations between arguments.

Definition 4.3 (Argumentation framework [Dun95])**.** *An argumentation framework* $AF = (Args, Att)$ *consists of a set of arguments* $Args$ *and an attack relationship* $Att : Args \times Args$*, where* $(A_1, A_2) \in Att$ *means that argument* $A_1 \in Args$ *attacks arguments* $A_2 \in Args$*.*

Now that we have defined arguments and their attacks, we are going to define a semantics to determine which arguments are acceptable. The following notions are preliminary to this.

Definition 4.4 (Attack, conflict-freeness, defense, and admissibility [Dun95])**.** *Suppose an argumentation framework* $AF = (Arg, Att)$*, two sets of arguments* $S \cup S' \subseteq Arg$*, and some argument* $A \in Arg$*. We say that*

- S attacks A *if some argument in* S *attacks* A*,*
- S attacks S' *if some argument in* S *attacks some argument in* S'*,*
- S *is* conflict-free *if it does not attack itself,*
- S defends A *if* S *attacks each attack against* A*.*
- S *is* admissible *if* S *is conflict-free and defends each argument in it.*

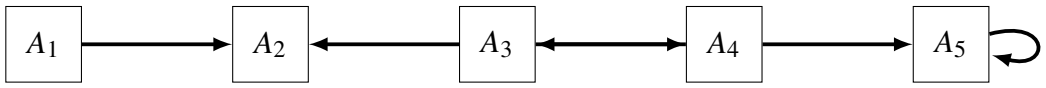


Figure 4.11: Example argumentation framework.

Let us explain these definitions using the example argumentation framework in Figure 4.11. We say that the set $\{A_1, A_4, A_5\}$ *attacks* argument A_2, because A_1 attacks A_2. However, $\{A_1, A_4, A_5\}$ is not *conflict-free*, because A_4 attacks A_5, so this set of arguments attacks itself. If we remove A_5 from this set, then it is conflict-free. In total, all the following sets are conflict-free in Figure 4.11:

$$\{A_1, A_3\}, \{A_1, A_4\}, \{A_2, A_4\}, \{A_1\}, \{A_2\}, \{A_3\}, \{A_4\}, \emptyset.$$

However, not all of these sets are *admissible*. For instance, the set $\{A_2, A_4\}$ is not admissible, because A_1 attacks this set, but the set does not defend itself against this attack. The set $\{A_1, A_3\}$ is admissible, because its only attacker, A_4 is attacked by A_3.

There are a large number of different semantics using these notions to determine which arguments are acceptable, such as the *grounded*, the *preferred*, the *stable*, and the *complete* semantics. However, in this chapter our argumentation frameworks are very simple, in the sense that they do not introduce cycles. In future work, we aim to extend this by using preferences (see open issues in Section 4.6.2). An advantage of our current approach is that all semantics coincide when there are no cycles, which simplifies our case. We use the preferred semantics here, but we could equivalently have chosen any other version.

Definition 4.5 (Preferred semantics [Dun95])**.** *Suppose an argumentation framework* $AF = (Arg, Att)$*. A set of arguments* $S \subseteq Arg$ *is a* preferred extension *if and only if*

- *S is an admissible set of argument,*
- *For each admissible set of arguments* $T \subseteq A$*:* $S \not\subset T$*.*

In our example in Figure 4.11, the preferred extensions are $\{A_1, A_3\}$ and $\{A_1, A_4\}$.

4.5.3 Algorithms for argument schemes and critical questions

Let us briefly review what we have done so far. Up until now we have achieved the following:

- We (informally) developed a set of argument schemes and critical questions in Section 4.3, by annotating transcripts.
- We developed a visual notation to illustrate some argument schemes and critical questions in Section 4.4.
- We proposed a logical language consisting of atomic sentences to describe a GRL model in Section 4.5.1.
- We formalized an "argument" as a set of atoms from our language, and we introduced *argumentation semantics* to compute sets of accepted arguments in Section 4.5.2.

We see that we are becoming increasingly more formal in each step of the progress. What we haven't formalized properly yet are the argument schemes and critical questions, which is what we turn to now.

Both the application of an argument schemes and the answering of critical questions are *procedural* or *dialectic* in nature. Therefore, a natural choice of formalizing them seems to develop procedures, or algorithms for them, which is what we do here.

In the following algorithms, we assume the following global variables:

Definition 4.6 (Global variables)**.** *The following variables are intended to have a* global *scope and have a* static *value, meaning the lifetime of the variable extends across the entire run of the program.*

- *id: the current highest identifier of the elements. This variable is increased for each new element that is added.*
- *Args: contains the set of all arguments.*
- *Att: contains the set of all attack relations.*

Algorithm 4.1 Applying AS0: Actor a is relevant

1: **procedure** $AS_0(a)$
2: $id \leftarrow id + 1$
3: $A \leftarrow \{actor(id), name(id, a)\}$
4: $Args \leftarrow Args \cup A$
5: **end procedure**

Algorithms for argument schemes

The algorithms for the argument schemes are very simple, because they simply consist of forming a new argument and adding it to the set of arguments. No attack relations are introduced.

Algortihm 4.1 for argument scheme AS0: The algorithm takes one argument, namely the name of the actor a. On line 2 of the algorithm, the global variable *id* is increased by one. This ensures that each new argument has a unique identifier. On line 3, the argument for the actor is formed, consisting of two statements, stating respectively that *id* is an actor, and that the name of this actor is a. Finally, on line 4 this argument is added to the global set of arguments *Args*. In Figure 4.10, the application of the argument scheme AS_0(Development team), results in one argument:

$$Args = \{\{actor(0), name(0, \text{Development team})\}\}.$$

Algorithm 4.2 Applying AS1: Actor a_{id} has resource n

1: **procedure** $AS_1(a_{id}, n)$
2: $id \leftarrow id + 1$
3: $A \leftarrow \{resource(id), name(id, n), has(a_{id}, id)\}$
4: $Args \leftarrow Args \cup A$
5: **end procedure**

Algorithm 4.2 for argument scheme AS1: This argument scheme takes two arguments, the identifier a_{id} of the actor and the resource name n. The algorithm is similar to the previous one, with the difference that the newly added argument contains the statement $has(a_{id}, id)$ as well, meaning that the actor with id a_{id} has element *id* (which is a resource). As an example, let us formalize the argument corresponding to resource *External library* of actor *Traffic tycoon* in Figure 4.4. First, we assume some id is associated with the actor:

$$\{actor(0), name(0, traffic\_tycoon)\}.$$

Then we can formalize the argument for the resource as follows:

$$\{resource(1), name(1, external\_library), has(0, 1)\}.$$

Argument scheme AS1 to AS4 are all very similar, in the sense that they all assert that some element belongs to an actor. Therefore, we only provide the algorithm for AS1 and we assume the reader can easily construct the remaining algorithms AS2-AS4.

Algorithm 4.3 for argument scheme AS5: The procedure in Algorithm 4.3 takes three arguments: g_{id} is the identifier of goal G, $T = (T_1, \ldots, T_n)$ is a list of decomposing

Algorithm 4.3 Applying AS5: Goal g_{id} decomposes into tasks $T_1, \ldots, T_n$

```
1:  procedure AS5(g_id, {T_1, ..., T_n}, type)
2:      T_id = ∅
3:      for T_i in {T_1, ..., T_n} do
4:          if ∃_{A∈Args} {task(t_id), name(t_id, T_i)} ⊆ A then
5:              T_id ← T_id ∪ {t_id}
6:          else
7:              id ← id + 1
8:              A ← {task(id), name(id, T_i)}
9:              Args ← Args ∪ A
10:             T_id ← T_id ∪ {id}
11:         end if
12:     end for
13:     id ← id + 1
14:     A ← {decomp(id, g_id, T_id, type)}
15:     Args ← Args ∪ A
16: end procedure
```

task names, and $type \in \{and, or, xor\}$ is the decomposition type. The difficulty of this algorithm is that each of the tasks are stated in natural language, and it is not directly clear whether these tasks are already in the GRL model or not. Therefore, we have to check for each tasks whether it already exists, and if not, we have to create a new task. On line 2, the set T_{id} is initialized, which will contain the ids of the tasks $T_1, \ldots, T_n$ to decompose into. In the for loop, the if statement on line 4 checks whether some argument already exists for the current task T_i, and if so, it simply adds the identifier of the task (t_{id}) to the set of task identifiers T_{id}. Otherwise (line 6), a new task is created and the new identifier *id* is added to the set of task identifiers. After the for loop on line 13, an argument for the decomposition link itself is constructed, and it is added to the set of arguments *Args*.

Let us explain this algorithm with the XOR-decomposition of goal `Generate cars` of Figure 4.4. Suppose the following arguments are constructed already:

- $\{goal(0), name(0, generate\_cars)\}$,
- $\{task(1), name(1, keep\_same\_cars\}$.

Suppose furthermore that someone wants to put forward the argument that goal `Generate cars` XOR-decomposes into tasks `Keep same cars` and `Create news cars`. Formally: $AS_5(0, \{generate\_cars, keep\_same\_cars\}, xor)$. The algorithm will first set $T_{id} = \emptyset$, and then iterate over the two task names. For the first task *generate_cars*, there does not exist an argument $\{task(t_{id}), name(t_{id}, generate\_cars)\}$ yet, so a new argument is created. Suppose the following argument is created: $\{task(2), name(2, generate\_cars)\}$. After this, 2 is added to T_{id}. For the second task an argument exists already, namely $\{task(1), name(1, keep\_same\_cars)\}$, so 1 is simply added to T_{id}. After the for loop, we have $T_{id} = \{1, 2\}$. Next, the decomposition argument is created, which is $\{decomp(3, 0, \{1, 2\}, xor)\}$. This argument is added to *Args* and the algorithm terminates.

Algorithm 4.4 for argument scheme AS6: The procedure in Algorithm 4.4 takes two arguments: t_{id} is the identifier of task T, and s is the softgoal name that is contributed to.

Algorithm 4.4 Applying AS6: Task t_{id} contributes to softgoal s

procedure $AS_6(t_{id}, s)$
  if $\exists_{A \in Args}\{softgoal(i), name(i,s)\} \subseteq A$ **then**
    $s_{id} \leftarrow i$
  else
    $id \leftarrow id + 1$
    $A \leftarrow \{softgoal(id), name(id,t)\}$
    $Args \leftarrow Args \cup A$
    $s_{id} \leftarrow id$
  end if
  $id \leftarrow id + 1$
  $A \leftarrow \{contr(id, t_{id}, s_{id}, pos)\}$
  $Args \leftarrow Args \cup A$
end procedure

The idea behind this algorithm is very similar to the previous one. First, the if statements check whether the softgoal exists already, and if not, an argument is added for it. This ensures that all softgoals have corresponding arguments. After the if statement, the argument for the contribution link is created and it is added to the set of arguments $Args$.

Let us again illustrate this with a simple example from Figure 4.4. Suppose the following argument exists already: $\{task(0), name(0, keep\_same\_cars\}$, and suppose someone would like to add an argument that the task `Keep same cars` contributes positively to softgoal `Dynamic simulation`, i.e. $AS_6(0, dynamic\_simulation)$. The algorithm first checks whether an argument already exists for the softgoal, and when it finds out it does not exist, creates the argument $\{softgoal(1), name(1, dynamic\_simulation)\}$ and adds it to $Args$. Then, the argument for the contribution is added to $Args$ as well, which is $\{contr(2,0,1,pos)\}$.

Algorithms for argument schemes AS7-AS11: The algorithms for $AS7$ to $AS11$ all have a very similar structure as Algorithm 4.4 and have therefore been omitted. Again, we assume the reader can reconstruct them straightforwardly.

Algorithms for critical questions

We now develop algorithms for our critical questions. Recall that answering a critical question can have four effects, and we discuss each of these effects separately.

Algorithm 4.5 Applying DISABLE: Element i is disabled

procedure DISABLE(i)
  $id \leftarrow id + 1$
  $A \leftarrow \{disabled(i))\}$
  $Args \leftarrow Args \cup A$
end procedure

Algorithm 4.5 (DISABLE) for critical questions CQ0-CQ5a, CQ6a, CQ7a, CQ8, CQ11, and CQ12: The disable operation is very straightforward: It simply consists of adding an argument stating the GRL element with identifier i is disabled. Let us reconsider the

example of Figure 4.9. This example consists of an instantiation of argument scheme AS0, which is attacked by an argument that resulted from answering critical question. The formalization of this scenario is shown in Figure 4.12. Interestingly, we see that the DISABLE operation no longer leads to an attack between the two arguments. Instead, the only thing that this operation does is adding another argument stating the element is disabled.

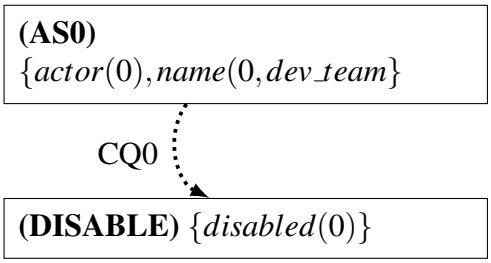


Figure 4.12: Formalization of the arguments in Figure 4.9.

Algorithm 4.6 Answering CQ5b: "Does goal G decompose into any other tasks?" With: "Yes, namey into tasks $t_1,\ldots,t_k$"

procedure CQ5B($g_{id},\{i_1,\ldots,i_n\},type,\{t_1,\ldots,t_k\}$)
  $T_{id} = \{i_1,\ldots,i_n\}$
  for t_i in $\{t_1,\ldots,t_k\}$ **do**
    if $\exists_{A\in Args}\{task(t_{id}),name(t_{id},t_i)\} \subseteq A$ **then**
      $T_{id} \leftarrow T_{id} \cup \{t_{id}\}$
    else
      $id \leftarrow id+1$
      $A \leftarrow \{task(id),name(id,t_i)\}$
      $Args \leftarrow Args \cup A$
      $T_{id} \leftarrow T_{id} \cup \{id\}$
    end if
  end for
  $id \leftarrow id+1$
  $A_{new} = \{decomp(id,g_{id},T_{id},type)\}$
  for A in $\{decomp(\_,g_{id},\_,\_)\} \subseteq A \mid A \in Args\}$ **do**
    $Att \leftarrow Att \cup \{(A_{new},A)\}$
  end for
  $Args \leftarrow Args \cup \{A_{new}\}$
end procedure

Algorithm 4.6 (REPLACE) for critical questions CQ5b: This algorithm is executed when critical question CQ5b is answered, which is a critical question for argument scheme AS5. Therefore, it assumes an argument for a goal decomposition already exists of the following form (see Algorithm 4.3):

$$\{decomp(d,g_{id},\{i_1,\ldots,i_n\},type).$$

The goal of the algorithm is to generate a new argument of the form $decomp(d,g_{id},\{i_1,\ldots,i_5\}\cup\{j_1,\ldots,j_k\},type)$, where $\{j_1,\ldots,j_k\}$ are the identifiers of the additional decomposing tasks $\{t_1,\ldots,t_k\}$.

The algorithm task as input the goal identifier g_{id}, the set of existing decomposing task identifiers $i_1,\ldots,i_n$, the decomposition type, and the names of the new tasks $t_1,\ldots,t_k$ that should be added to the decomposition. The first part of the algorithm is already familiar: For each task name we check whether it already exists as an argument (line 4), and if it doesn't (line 6) we add a new argument for it. After the for loop (line 13), a new argument is created for the new decomposition relation (14). After this, the for loop on line 15 ensures that the new argument attacks all previous arguments for this decomposition (note that the variable "_" means "do not care"). Only at the very end the new argument is added (line 18), to ensure it does not attack itself after the for loop of line 15-17.

An example of this algorithm is shown in Figure 4.13.[4] Before the critical question is applied, the following arguments have been put forward:

- $\{goal(0), name(0, show\_simulation)\}$
- $\{task(1), name(1, generate\_traffic)\}$
- $\{task(2), name(2, compute\_lights)\}$
- $\{decomp(3, 0, \{1,2\}, and)\}$.

Next, Algorithm 4.6 is called as follows: $CQ5b(0, \{1,2\}, and, \{show\_controls\})$. That is, the existing decomposition is challenged by stating that goal *show_simulation* not only decomposes into *generate_traffic* and *compute_lights*, but it also decomposes into *show_controls*. Since this task does not exist yet, it is created by the algorithm, which also ensures the new argument for the decomposition link attacks the previous argument for the decomposition link.

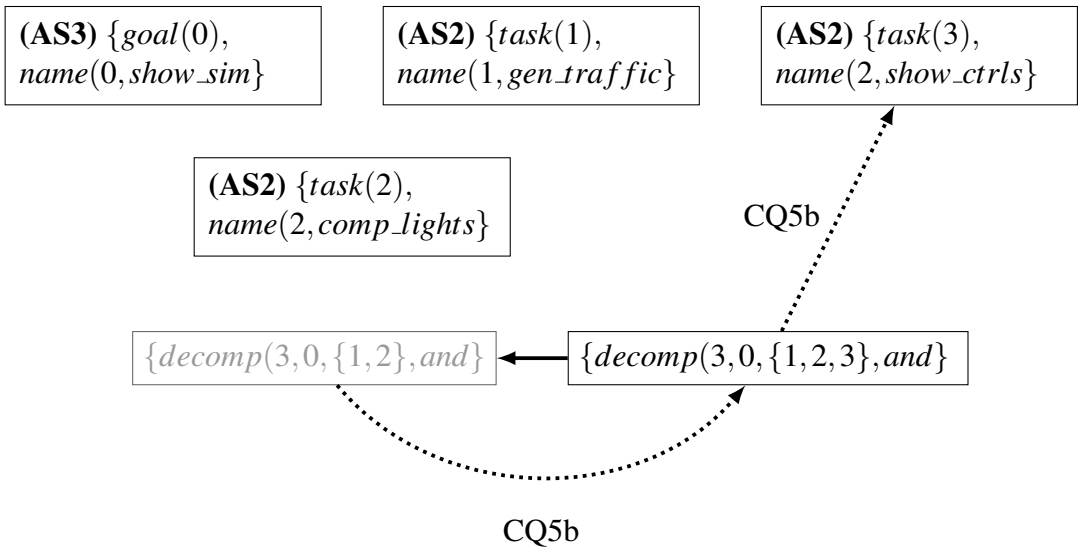


Figure 4.13: Example of applying critical question CQ5b (Algorithm 4.6)

Algorithms for critical questions CQ10a and CQ10b (REPLACE): These algorithms have a very similar structure as Algorithm 4.6 and have therefore been omitted.

[4]Note that part of the arguments (the statements about actors) have been omitted from the figure for readability.

Algorithm 4.7 Answering CQ13: "Is the name of element i clear?" With: "No, it should be n"

1: **procedure** CQ13(i,n)
2: $ArgsN \leftarrow \{A \in Args \mid name(i,x) \in A\}$
3: $B \leftarrow B' \setminus \{name(i,\_)\}$ with $B' \in ArgsN$
4: $A \leftarrow B \cup \{name(i,n)\}$
5: $Args \leftarrow Args \cup \{A\}$
6: **for** B in $ArgsN$ **do**
7: $Att \leftarrow Att \cup \{(A,B)\}$
8: **end for**
9: **end procedure**

Algorithms for critical question CQ13 (REPLACE): This algorithm is used to clarify/change the name of an element. It takes two parameters: the element identifier i and the new name n. The idea behind the algorithm is that we construct a new argument for n from the previous arguments, and we only replace the *name* atom. We also have to ensure that we attack all previous arguments for a name. On line 2, all arguments that have been put forward for this element and contain $name(i,x)$ are collected into the set $ArgsN$. On line 3, some arguments $B' \in ArgsN$ minus the *name* statement is assigned to B, and on line 4 B is joined with the new *name* statement and stored in A, which is then added to the set of arguments $Args$. The for loop on lines 6-8 ensures all previous arguments for names of the element are attacked by the new argument.

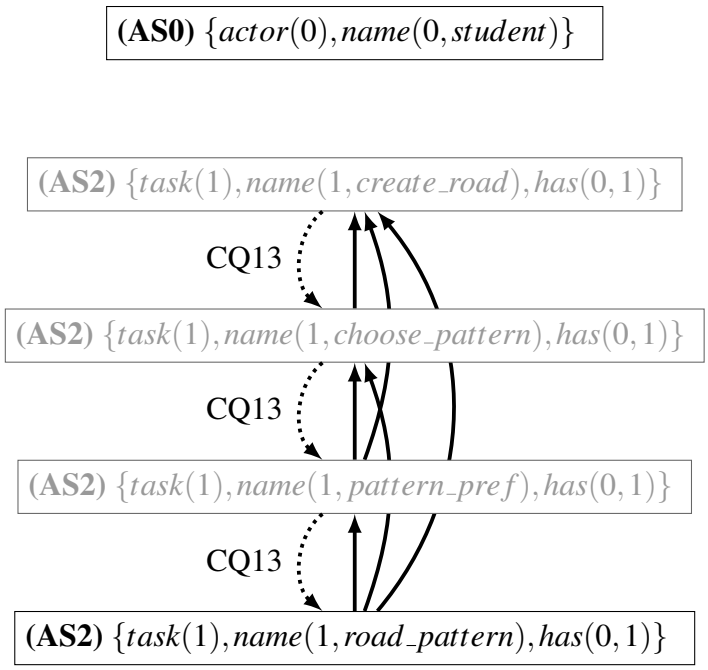


Figure 4.14: Applying critical question CQ13 (Algorithm 4.7) to the example in Figure 4.7.

An example of the working of Algorithm 4.7 is shown in Figure 4.14. Let us consider the last application of CQ13 (bottom argument). Before this application, the following arguments have been put forward:

- A_1: $\{actor(0), name(0, student)\}$

- A_2:$\{task(1), name(1, create\_road), has(0,1)\}$
- A_3 $\{task(1), name(1, choose\_pattern), has(0,1)\}$
- A_4:$\{task(1), name(1, pattern\_pref), has(0,1)\}$

The algorithm is now called as follows: $CQ13(1, road\_pattern)$, i.e., the new name of the element should be *road_pattern*. Let us briefly run through the algorithm. After executing line 2 we obtain $ArgsN = \{A_2, A_3, A_4\}$, since only those arguments contain $name(1, \_)$. Next, on line 3, $B = \{task(1), has(0,1)\}$, i.e., B is the general argument for the task without the *name* statement. After line 4 we have

$$A = \{task(1), has(0,1), name(1, road\_pattern),$$

which is added to $Args$ and attacks arguments A_2, A_3, and A_4.

Algorithms for critical questions CQ6b, CQ6c, CQ6d, CQ7b, and CQ9 (INTRO): The introduction algorithms for the critical questions are all very similar to the INTRO algorithms for argument schemes (Algorithm 4.2). They have therefore been omitted.

Algorithm 4.8 Generic counterargument to argument A

```
1: procedure ATTACK(A)
2:     A_new = {}
3:     Args ← Args ∪ {A_new}
4:     Att ← Att ∪ {(A_new, A)}
5: end procedure
```

Algorithm for Att (Generic counter argument: Applying a generic counter argument is very simple, and simply results on an attack on the original argument. We illustrate this by continuing our example from Figure 4.15 (Algorithm 4.1). The example is shown in Figure 4.15, where we see that a generic counter argument simply attacks the argument to disable the actor.

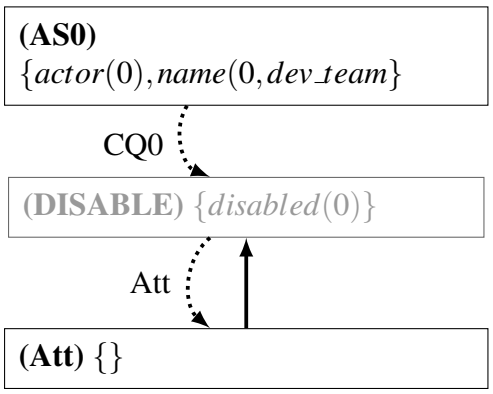


Figure 4.15: Formalization of the arguments in Figure 4.10.

4.5.4 Constructing GRL models

Constructing GRL models from the arguments is extremely simple: We simply compute the extensions of the argumentation frameworks, and collect all atomic sentences in the

accepted arguments. This forms out GRL model. Let us briefly do so for the examples of the previous subsection:

- Figure 4.12: Since there are no attacks between the arguments, all atomic sentences are accepted. This results in the following specification:

```
actor(0).
name(0,dev_team).
disabled(0).
```

  This again corresponds to the GRL model on the right-hand side of Figure 4.9.

- Figure 4.13: There is one rejected argument and five accepted ones. The resulting specification is:

```
goal(0). name(0,show_simulation).
task(1). name(1,generate_traffic).
task(2). name(2,compute_lights).
task(3). name(3,show_controls).
decomp(3,0,{1,2,3},and).
```

- Figure 4.14: There are only two accepted arguments. The resulting specification is:

```
actor(0). name(0, student).
task(1).  name(1,road_pattern). has(0,1).
```

  This corresponds to the right-hand GRL model of Figure 4.7.

- Figure 4.15. There are two accepted arguments, but the *generic counterargument* does not contain any formulas. Therefore the resulting specification is:

```
actor(0).
name(0,dev_team).
```

  This corresponds to the right-hand GRL model of Figure 4.10.

4.6 Discussion

4.6.1 Related work

The need for justifications of modeling choices plays an important role in different requirements engineering methods using goal models. High-level goals are often understood as reasons for representing lower-level goals (in other words, the need for low-level goals is justified by having high-level goals) and other elements in a goal model such as tasks and resources. Various refinements and decomposition techniques, often used in requirements engineer (See [Van01] for an overview), can be seen as incorporating argumentation and justification, in that sub-goals could be understood as arguments supporting parent goals. In that case, a refinement alternative is justified if there are no conflicts between sub-goals (i.e., it is consistent), as few obstacles as possible sub-goals

harm sub-goal achievement, there are no superfluous sub-goals (the refinement is minimal), and the achievement of sub-goals can be verified to lead to achieving the parent goal (if refinement is formal [DvL96]). This interpretation is one of the founding ideas of goal modeling. However, while this interpretation may seem satisfactory, argumentation and justification processes differ from and are complementary to refinement in several respects, such as limited possibilities for rationalization and lack of semantics (see Jureta [JFS08] for more details).

There are several contributions that relate argumentation-based techniques with goal modeling. The contribution most closely related to ours is the work by Jureta *et al.* [JFS08]. Jureta *et al.* propose "Goal Argumentation Method (GAM)" to guide argumentation and justification of modeling choices during the construction of goal models. One of the elements of GAM is the translation of formal argument models to goal models (similar to ours). In this sense, our RationalGRL framework can be seen as an instantiation and implementation of part of the GAM. The main difference between our approach and GAM is that we integrate arguments and goal models using argument schemes, and that we develop these argument schemes by analyzing transcripts. GAM instead uses structured argumentation.

The RationalGRL framework is also closely related to frameworks that aim to provide a design rationale (DR) [SSS+06], an explicit documentation of the reasons behind decisions made when designing a system or artefact. DR looks at issues, options and arguments for and against the various options in the design of, for example, a software system, and provides direct tool support for building and analyzing DR graphs. One of the main improvements of RationalGRL over DR approaches is that RationalGRL incorporates the formal semantics for both argument acceptability and goal satisfiability, which allow for a partly automated evaluation of goals and the rationales for these goals.

Arguments and requirements engineering approaches have been combined by, among others, Haley *et al.* [HMLN05], who use structured arguments to capture and validate the rationales for security requirements. However, they do not use goal models, and thus, there is no explicit trace from arguments to goals and tasks. Furthermore, like [JFS08], the argumentative part of their work does not include formal semantics for determining the acceptability of arguments, and the proposed frameworks are not actually implemented. Murukannaiah *et al.* [MKTS15b] propose Arg-ACH, an approach to capture inconsistencies between stakeholders' beliefs and goals, and resolve goal conflicts using argumentation techniques.

4.6.2 Open issues

We see a large number of open issues that we hope will be explored in future research. We discuss five promising directions here.

Architecture principles

One aspect of enterprise architecture that we did not touch upon in this thesis are *(enterprise) architecture principles*. Architecture principles are general rules and guidelines, intended to be enduring and seldom amended, that inform and support the way in which an organization sets about fulfilling its mission [Lan05a, OP07, The09a]. They reflect a

level of consensus among the various elements of the enterprise, and form the basis for making future IT decisions. Two characteristics of architecture principles are:

- There are usually a small number of principles (around 10) for an entire organization. These principles are developed by enterprise architecture, through discussions with stakeholders or the executive board. Such a small list is intended to be understood *throughout the entire organization*. All employees should keep these principles in the back of their hard when making a decision.
- Principles are meant to guide decision making, and if someone decides to deviate from them, he or she should have a good reason for this and explain why this is the case. As such, they play a normative role in the organization.

Looking at these two characteristics, we see that argumentation, or justification, plays an important role in both forming the principles and adhering to them:

- Architecture principles are *formed* based on underlying arguments, which can be the goals and values of the organization, preferences of stakeholders, environmental constraints, etc.
- If architecture principles are *violated*, this violation has to be explained by underlying arguments, which can be project-specific details or lead to a change in the principle.

In a previous paper, we [MvZG16] propose an extension to GRL based on enterprise architecture principles. We present a set of requirements for improving the clarity of definitions and develop a framework to formalize architecture principles in GRL. We introduce an extension of the language with the required constructs and establish modeling rules and constraints. This allows one to automatically reason about the soundness, completeness and consistency of a set of architecture principles. Moreover, principles can be traced back to high-level goals.

It would be very interesting future work to combine the architecture principles extension with the argumentation extension. This would lead to a framework in which principles cannot only be traced back to goals, but also to underlying arguments by the stakeholders.

Extensions for argumentation

The amount of argumentation theory we used in this chapter has been rather small. Our intention was to create a bridge between the formal theories in argumentation and the rather practical tools in requirements engineering. Now that the initial framework has been developed, is it worth exploring what tools and variations formal argumentation has to offer in more detail.

For instance, until now we have assumed that every argument put forward by a critical questions always defeats the argument it questions, but this is a rather strong assumption. In some cases, it is more difficult to determine whether or not an argument is defeated. Take, for example, the argumentation framework in Figure 4.16 with just A1 and A2. These two arguments attack each other, they are alternatives and without any explicit

preference, and it is impossible to choose between the two. It is, however, possible to include explicit preferences between arguments when determining argument acceptability [AC02]. If we say that we prefer the action `Create new cars` (A2) over the action `Keep same cars` (A1), we remove the attack from A1 to A2. This makes A2 the only undefeated argument, whereas A1 is now defeated. It is also possible to give explicit arguments for preferences [Mod09]. These arguments are then essentially attacks on attacks. For example, say we prefer A3 over A1 because 'it is important to have realistic traffic flows' (A4). This can be rendered as a separate argument that attacks the attack from A1 to A3, removing this attack and making $\{A3, A4\}$ the undefeated set of arguments.

Allowing undefeated attacks also make the question of which semantics to choose more interesting. In our current (a-cyclic) setting, (almost) all semantics coincide, and we always have the same set of accepted arguments. However, once we allow for cycles, we may choose accepted arguments based on semantics which, for instance, try to accept/reject as many arguments as possible (preferred semantics), or just do not make any choice once there are multiple choices (grounded). Another interesting element of having cycles is that one can have multiple extensions. This corresponds to various *positions* are possible, representing various sets of possibly accepted arguments. Such sets can then be shown to the user, who can then argue about which one they deem most appropriate.

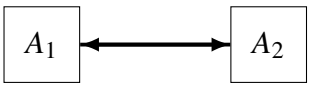


Figure 4.16: Preferences between arguments

Finally, in this chapter we have only explored one single argument scheme, but there are many other around. In his famous book " Argumentation schemes", Walton *et al.* describe a total of 96 schemes. Murukannaiah *et al.* [MKTS15a] already explain how some of these schemes may be use for resolving goal conflicts, and it is worth studying what this would look like in our framework as well.

Tool support

GRL has a well-documented and well-maintained tool called jUCMNav [MA09]. This tool is an extension to Eclipse. Although it is a rich tool with many features, we also believe it is not very easy to set it up. This seriously harms the exposure of the language, as well as the ability for practitioners to use it. We have started to implement a simple version of GRL as an online tool in Javascript. This makes it usable from the browser, without requiring the installation of any tool. The tool can be used from the following address:

`http://marcvanzee.nl/RationalGRL/editor`

A screenshot of the tool is shown in Figure 4.17. As shown, there are two tabs in the tool, one for "Argument" and one for "GRL". The argument part has not been implemented yet, and the GRL part only partly, but the idea behind the tool should be clear. Users are

able to work on argumentation and on goal modeling in parallel, where the argumentation consists of forming arguments and counterarguments by instantiating argument schemes and answering critical questions.

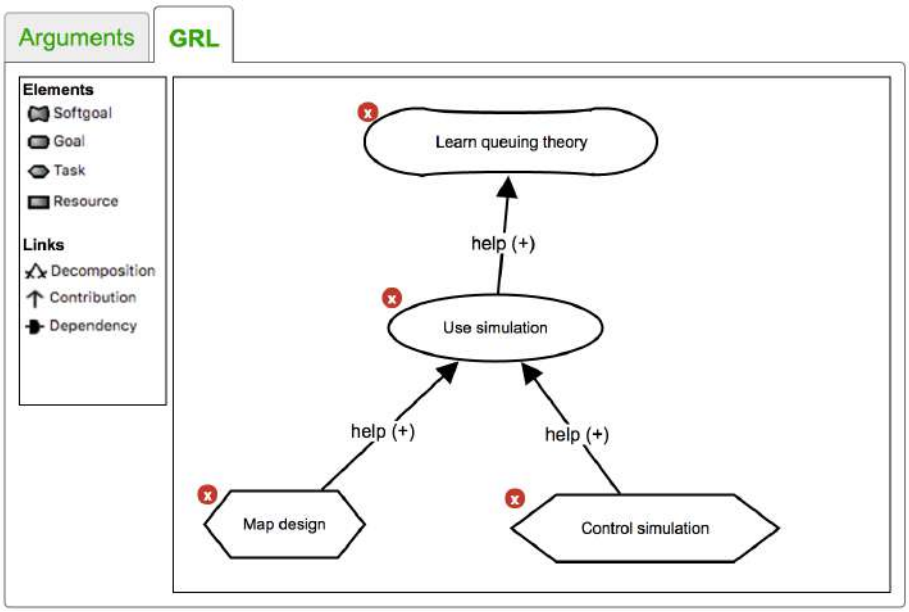


Figure 4.17: Screenshot of the prototype tool

An important aspect of the tool is that users can switch freely between these two ways of modeling the problem. One can model the entire problem in GRL, or one can do everything using argumentation. However, we believe the most powerful way to do so is to switch back and forth. For instance, one can create a simple goal model in GRL, and then turn to the argumentation part, which the users can look at the various critical questions for the elements, which may trigger discussions. These discussions results in new arguments for and against the elements in the goal model. Once this process is completed, one may switch to the goal model again, and so on. We believe that in this way, there is a close and natural coupling between modeling the goals of an organization as well as rationalizing them with arguments.

Empirical study

Although we develop our argument schemes and critical questions with some empirical data, we did not yet validate the outcome. This is an important part, because it will allow us to understand whether adding arguments to goal modeling is actually useful. We have developed an experimental setup for our experiment, which we intend to do during courses at various universities. However, we cannot carry out this experiment until the tool is finished.

Formal framework

The formal framework we present in this chapter is very simple, and does not provide a lot of detail. In line with our approach in Chapter 2, we believe it would be interesting to develop a more robust characterization of a GRL model using logical formulas. Right

now, we have no way to verify whether the goal models we obtain through out algorithms are actually valid GRL models. This is because we allow any set of atoms to be a GRL model, which is clearly very permissive and incorrect. Once we develop a number of such constraints, we can ensure (or even proof) our algorithms do not generate invalid GRL models.

For instance, suppose we assert that an *intentional element* is a goal, softgoal, task, or resource:

$$(softgoal(i) \vee goal(i) \vee task(i) \vee resource(i) \rightarrow IE(i).$$

We can then formalize an intuition such as: "Only intentional elements can be used in contribution relations" as follows:

$$contrib(k,i,j,ctype) \rightarrow (IE(i) \wedge IE(j) \wedge IE(j).$$

Interestingly, such constraints are very comparable to *logic programming* rules. We therefore see it as interesting future research to explore this further, specifically in the following two ways:

- Develop a set of constraints on sets of atoms of our language, which correctly describe a GRL model. Show formally that using our algorithms, each extension of the resulting argumentation framework corresponds to a valid GRL model, i.e., a GRL model that does not violate any of the constraints.
- Implement the constraints as a logic program, and use a logic programming language to compute the resulting GRL model.

4.6.3 Conclusion

In this chapter, we develop the RationalGRL framework to trace back elements of GRL models to arguments used in discussions between stakeholders. The contributions of this chapter have become increasingly more formal. First, we analyze transcripts of meetings about the development of an information system. We count occurrences of argument schemes and critical questions, and categorize them by the effect of instantiating them. Then, we create a visual notation to link arguments and questions to elements of a goal model. If an argument for an element of a goal model is rejected, then the corresponding element is disabled. Similarly, if the argument is accepted, then the corresponding element is enabled. Finally, we formalize the argument schemes and critical questions in a logical framework. We propose a formal language to describe a GRL model, and we develop various algorithms for applying our argument schemes and critical questions.

Our framework is one of the first attempts that try to combine the formal theory of argumentation with the practical frameworks of goal modeling. History has shown that it is remarkably difficult to combine formal results with practice. By taking argument schemes as a starting point, we believe we have chosen the right level of formality, but as the open issues section shows, much work is left to be done.

Part III

Planning and Scheduling

5

A Logic for Beliefs about Actions and Time

Abstract The overall aim of the third and final part of this thesis is to study the dynamics of enterprise architecture plan commitments. We do so by developing a temporal logic for beliefs and intentions. The logic is called Parameterized-time Action Logic, and contains expressions about temporal propositions (tomorrow the hardware will arrive), possibility (the hardware may arrive tomorrow), actions (the meeting takes place) and pre- and post-conditions of these actions. In this chapter, we lay out the methodology of this part, which is based on Shoham's *database perspective*. We then introduce Parameterized-time Action Logic, axiomatize it and proof completeness. We use this logic in the next chapter to characterize the dynamics of enterprise architecture commitments.

Preliminary remark

To avoid confusing the reader, we should start this part with a remark about our approach. Recall from the introduction that the overall aim of this thesis is to build a bridge from artificial intelligence to enterprise architecture. As we progress through the thesis, our contributions become increasingly technical and less related to enterprise architecture. This third and last part of the thesis is therefore the most technical one, and the direct relation between enterprise architecture is less clear.

Motivated by our findings in Chapter 2 and 3, our methodology is to develop a belief-desire-intention logic– a specific approach towards formalizing *resource-bounded* agents– to characterize the dynamics of enterprise architecture commitments. Because of the level of technicality, this part has mostly been written for an audience familiar with knowledge representation, commonsense reasoning, and belief-desire-intention logics. Since one of the most-well known approaches in belief-desire-intention logics is due to Cohen and Levesque [CL90b], we start out by contrasting our approach with their by using their example, which is about a robot. This may seen strange from an enterprise architecture point of view, but we believe it does make sense from a historical point of view.

5.1 Introduction

Sometime in the near future you will tell your household robot: "Bobby, get me some beer from the store". Bobby confirms your request, but when its walking to the store it

encounters your partner, who says: “Bobby, our house is a mess, go home and clean”. Bobby returns home, takes the mop out of its closet and prepares to start cleaning. Just as it is ready to make its first swipe, one of your friends walks in asking: “Bobby, it has been snowing outside, could you clean my car?”. In the meantime, you are getting increasingly frustrated by your lack of beer, and when you see Bobby in the kitchen you shout: “You still didn’t get my beers? Go get them immediately!”. After letting Bobby run around for a few days you return the robot to the factory complaining it does not finish any of the tasks it starts with.

When Bobby 2.0 arrives, the manufacturer happily tells you the new settings will no longer cause Bobby to drop its commitments so quickly. Delighted, you exclaim: “Bobby, I have some friends coming over tonight, get some ingredients and cook diner so we can eat at 7pm tonight”. Realizing the shop closes only at 5pm, Bobby 2.0 delays going to the grocery story until the very last moment, hereby keeping its schedule free for other possible tasks. Unexpectedly, on its way to the grocery store Bobby is delayed by an open bridge and arrives at the store minutes after closing time. It returns to your home empty-handed, leaving you and your friends hungry. It turns out Bobby 2.0 is delaying every task until just before the deadline. Since tasks often have unexpected delays, this means that most of the tasks are finished too late, or not at all. Frustrated, you return it to the factory again, complaining that Bobby is procrastinating its commitments.

On return, the manufacturer ensures you that Bobby 3.0 will no longer postpone fulfilling its commitments, nor will it drop them quickly. At this time you are rather skeptical, but still you ask: “Bobby, I’d like to have diner tonight again. Please buy the ingredients at 2pm, and cook for me at 6pm”. Around 12pm, your partner again realizes the house hasn’t been cleaned properly for a long time, and therefore tells Bobby to clean the house intensively. At 6pm, you sit at your kitchen table wondering where diner is, so you call Bobby asking what happened. Bobby explains it had to clean the house at 12pm, which took three hours, so it couldn’t fulfill its commitment to go shopping at 2pm.

Disappointed, you return it to the manufacturer where it is dismantled.

5.1.1 Commitment to time

The example above is very similar to the example of Willie the robot, due to [CL90b]. In their article entitled “Intention = Choice + Commitment”, Cohen and Levesque were interested in specifying the rational balance of autonomous agents, focusing on the role that intention plays in maintaining this balance. Their approach has typified much subsequent research on belief-desire-intention (BDI) logics, namely to understand and study intentions as commitment in relation to goals and desires. For instance, [RG91] define various commitment strategies of an agents, from blindly committed towards goals, to open mindedly committed. A popular approach is to specify a temporal logic such as linear-time logic (LTL) or computational-tree logic (CTL*) and use modal operators for mental states and use expression of the form “some time in the future”, or ”in the next time moment” to reason about the temporal behavior.

In approaches following the ideas of Cohen and Levesque ([RG91, MvdHvL99] are two examples), intention is often defined as commitments towards goals. However, being committed towards a goal is only one dimension of a commitment, while they have

many more aspects. One important aspect of a commitment that we focus on in this thesis is *commitment towards time*, i.e., *when* these commitments will be fulfilled. In the example above, Bobby the robot is an *online system*; it is receiving orders, forming plans, scheduling tasks, and executing them, all in parallel and in real-time. Bobby plans its commitments at appropriate moments, making sure the different plans do not overlap or are incompatible, while it at the same time may receive new instructions from users.

When specifying a system *exogenously*, which has typified much work on philosophical logic, it may be sufficient to reason about behaviors with notions such as "Bobby believes it should do shopping some time in the future", or "in the next time moment, Bobby intends to make diner". However, if we specify a system *endogenously*, which typifies much work in computer science and planning, one has to be more precise about when commitments will be fulfilled. Indeed, Bobby should know it has the commitment to go shopping at 2pm, and not some time in the future, so that its planner can continue planning on this assumption. By focusing on commitments towards time we therefore aim to bridge the gap between logics of agency on the one side and computer science and planning on the other side.

We will see that even in a simplified setting where we only consider commitments towards time, already non-trivial complications arise. Commitments play the role of *assumptions* on which further plans are based. The three versions of Bobby the household robot each use a different behavior when it comes to commitment to time. The first version uses a stack-like data structure in order to execute its tasks: it adds each new commitments on the stack, and then executes the tasks on top of its stack. The second version rather uses a queue, and moreover delays executing these tasks until the very last moment. Finally, the last version is able to schedule its commitments in time, but it cannot reschedule them. None of these versions seem to be able to fulfilled commitments in a desired way. Instead, Bobby should be able to make plans, store the commitments and use these as assumptions in further planning.

5.1.2 Methodology

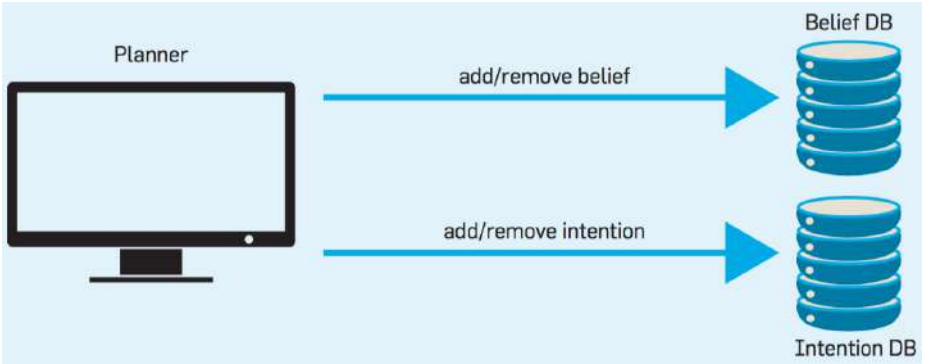


Figure 5.1: The database perspective

We view the problem of intention revision as a database management problem (see [Sho09] for more on the conceptual underpinnings of this standpoint). At any given moment, an agent must keep track of a number of facts about the current situation. This includes beliefs about the current state, beliefs about possible future states, which actions are available now and in the future, and also what the agent plans to do at future moments.

It is important that all of this information be jointly consistent at any given moment and furthermore can be modified as needed while maintaining consistency.

In this chapter we introduce a simple logic that formally models such a "database", and in the next chapter we study its dynamics. *Consistency* in this logic is meant to represent not only that the agent's beliefs are consistent and the agent's future plan is consistent, but also that the agent's beliefs and intentions together form a *coherent* picture of what may happen, and of how the agent's own actions will play a role in what happens. Our primary contribution in this thesis is to focus also on how the database is to be modified, and in the process to provide a clear picture of how intentions and beliefs relate.

What can cause an agent's database to change? In this thesis, we focus on two main sources (Figure 5.1):

1. The agent makes some observation, e.g. from sensory input. If the new observation is inconsistent with the agent's beliefs, these beliefs will have to be revised to accommodate it. While we are fully aware of their shortcomings, in particular when it comes to iterated revision, our account of belief revision follows the classical AGM postulates [AGM85] rather closely. The goal is thus to give general conditions on a single revision with new information that the agent has already committed to incorporating.

2. The agent forms a new intention. Here we focus on future directed intentions, understood as time-labeled actions pairs (a,t) that might make up a plan. Analogously to belief revision, it is assumed the agent has already committed to a new intention, so it must be accommodated by any means short of revising beliefs. The force of the theory is in restricting how this can be accomplished. To be more precise, we purport to model an intelligent database, which receives instructions from some planner (e.g. a STRIPS-like planner) that is itself engaged in some form of practical reasoning. The job of the database is to maintain consistency and coherence between intentions and beliefs.

This simple description, however, obscures some important subtleties in the interaction between beliefs and intentions. The following will serve as a running example that we will use frequently throughout this and the next chapter.

Example 5.1 (Running Example)**.** *Bobby the household robot is considering to buy groceries in the morning and to buy cleaning equipment in the afternoon. Although it is instructed to do both tasks, it only has sufficient budget to do either one of the two, but not both of them. Bobby thus believes it is possible to buy cleaning equipment and to buy food, but it also believes it is impossible to buy both. If Bobby decides to buy food, then it would like to cook in the afternoon.*

Upon adopting the intention to buy food, Bobby will come to have new beliefs based on the predicted success of this intention, e.g., the he will be able to cook afterwards. These further beliefs are important when planning when or how to cook. The intention is also supported by the absence of certain beliefs. It would be irrational for Bobby to adopt the intention to buy food if it believed it did not have sufficient money. Likewise, even if it originally believed it has sufficient money, upon learning it does not, the intention to cook food should be dropped. Yet, when dropping this intention, other beliefs, such as that he will be able to cook, have to be dropped as well, which may in turn force other intentions and beliefs to be dropped. And so on.

5.1.3 Strong and weak beliefs

To deal with issues described above, we separate between beliefs depending on intentions, i.e. *weak beliefs*, and beliefs that do not, i.e. *strong beliefs*.[1] Strong beliefs concern the world as it is, independent of the agent's future plans, but including what (sequences of) actions are possible. Thus, additionally to atomic facts, the agent may have beliefs about what the preconditions and postconditions of actions are, and about which sequences of actions are jointly possible.

A key element in our approach is that we treat preconditions of actions as *assumptions*. This leads to an the asymmetry on beliefs about preconditions and postconditions of actions. First of all, we assume that

> *If an agent intends to take an action at some time t, it weakly believes that its postconditions hold at time $t+1$.*

Since we are not considering actions whose effects are uncertain or dependent on the conditions that obtain when the action is taken, if an action is planned the planner believes whatever follows from it.

However, for preconditions we add a weaker requirement:

> *If an agent intends to take an action at time t, it cannot believe that its preconditions do not hold.*

This requirement is sometimes called *strong consistency*, and is weaker than Bratman's *means-end coherence* requirement [Bra87]. The result of this weakened requirement is that preconditions of actions are treated as *assumptions*: An autonomous agent forms intentions under the assumption that these preconditions will be made true somewhere in the future. Treating preconditions as assumptions is a good fit with how real-time planning agents operate: Intended actions may be added as long as they are consistent with beliefs, and once they are accepted they can be used as additional assumptions to further plans [Sho09]. For instance, the household robot Bobby has the intention to cook diner tonight, which is based on the intention to buy the ingredients, which is in turn based on the assumptions that your friends will attend tonight, even if it does not know this for sure yet. It is only when Bobby finds out your friends are not coming, it should drop its intentions.

5.1.4 Results and overview

We develop a branching-time temporal logic, called Parameterized-time Action Logic (PAL) in order to formalize beliefs. The language of this logic contains formulas to reason about possibility, preconditions, postconditions, and the execution of actions. The semantics of this logic is close to CTL*, and in this way follows the tradition of BDI logics of [RG91]. An important difference is that we do not use modal operators to reason about time, but we use explicit time points. We axiomatize this logic and proof that the axiomatization is sound and strongly complete with respect to our semantics.

[1] The terminology of strong and weak beliefs is due to [vdHJW07].

The structure of this chapter is as follows. In Section 5.2 we introduce the syntax, and in Section 5.3 the semantics. In Section 5.4 we axiomatize our logic, and in Section 5.5 we prove completeness. We discuss related work, open issues, and a conclusion in Section 5.6.

5.2 PAL syntax

Our aim in this section is to develop a logical system that represents an agent's beliefs about the current moment and future moment and actions that may be performed. Beliefs are represented by the formal language $\mathscr{L}$.

Definition 5.1 (Language)**.** *Let*

- $\mathsf{Act} = \{a, b, c, \ldots\}$ *be a finite set of deterministic primitive actions, containing the special "no operation" action NOP;*
- $\mathsf{Prop} = \{p, q, r, \ldots\} \cup \{pre(a, b, \ldots), post(a) \mid \{a, b, \ldots\} \subseteq \mathsf{Act}\}$ *be a finite set of propositions. We denote atomic propositions with* χ.

The sets Prop *and* Act *are disjoint. The language* $\mathscr{L}$ *is inductively defined by the following BNF grammar:*

$$\varphi ::= \chi_t \mid do(a)_t \mid \Box_t \varphi \mid \varphi \wedge \varphi \mid \neg \varphi$$

with $\chi \in \mathsf{Prop}, a \in \mathsf{Act}$*, and* $t \in \mathbb{N}$*. Furthermore, we abbreviate* $\neg \Box_t \neg$ *with* $\Diamond_t$*, and we define* $\bot \equiv p_0 \wedge \neg p_0$ *and* $\top \equiv \neg \bot$.

Intuitively, p_t means that the atomic formula p is true at time t, $do(a)_t$ means that action a is executed at time t. To every finite sequence of actions $(a, b, \ldots)$ and every time point t we associate a formula $pre(a, b, \ldots)_t$, which is understood as the precondition for subsequently executing actions $a, b, \ldots$ at time t. We define preconditions for sequences of actions explicitly, because it is not generally possible to define the precondition for a sequence of actions using preconditions for individual actions. This can already be witnessed in our running example: Bobby believes the preconditions to buy food and cleaning equipment are true separately, but still does not believe the precondition for performing both actions subsequently is true. These type of formulas will play a crucial role when we formalize the coherence condition in the next chapter.

To every action a and every time point t we associate a formula $post(a)_t$, which is understood as the postcondition of action a at time t. The modal operator $\Box_t$ is interpreted as necessity, indexed with a time point t. Intuitively, a formula of the form $\Box_t p_{t+1}$ means "it is necessary at time t that p is true at time $t+1$". The other boolean connectives are defined as usual.

Example 5.2 (Running example (Ctd.))**.** *Let our language contain:*

- $\mathsf{Act} = \{food, equip, cook, NOP\}$*, where* $food$ *is the action "buy food", equip is the action "buy cleaning equipment", and cook is the action "cook", and NOP is the special "no operation" action,*

- $\mathsf{Prop} = \{pre(a,b,\ldots), post(a) \mid \{a,b,\ldots\} \subseteq \mathsf{Act}\}$

Some examples of formulas in the language generated from Act *and* Prop *are:*

- $pre(food)_0 \wedge do(nop)_0 \wedge do(equip)_1$ *(the precondition to buy food at time 0 is true, no action is performed at time 0, and Bobby buys cleaning equipment at time 1)*
- $\Diamond_0(do(food)_0 \wedge \neg do(cook)_1)$ *(it is possible at time 0 to buy food at time 0 and not to cook at time 1),*
- $\Diamond_0 do(food)_0 \wedge \Diamond_0 do(equip)_1 \wedge \neg\Diamond_0(do(food)_0 \wedge do(equip)_1)$ *(it is possible to buy food at time 0 and it is possible to buy equipment at time 1, but it is not possible to do both),*
- $pre(food, cook)_0$ *(the precondition to buy foot at time 0 and then cook at time 1 is true).*
- $do(equip)_1$ *(Bobby will buy cleaning equipment at time 1)*
- $\Diamond_0\neg\Diamond_1 do(cook)_1$ *(it is possible at time 0 that it is not possible at time 1 to cook).*
- $\bigvee_{x\in\mathsf{Act}} pre(food, x, equip)_0$ *(the precondition to buy food at time 0 and to buy equipment at time 2 is true, if a right action is performed at time 1).*

The following definition collects all formulas up to some time t in a set $Past(t)$, which will turn out to be convenient when we axiomatize our logic. Note that formulas up to time t only contain $do(a)_{t'}$ statements with $t' < t$. A formula of the form $do(a)_t$ will be semantically defined as a transition from t to $t+1$. Therefore it does not belong to the formulas true up to time t. We will make this more precise when we introduce the semantics in the next subsection.

Definition 5.2. *$Past(t)$ is the set of all PAL formulas generated by boolean combinations of $p_{t'}, pre(a,b,\ldots)_{t'}, post(a)_{t'}, \Box_{t'}\varphi$, and $do(a)_{t'-1}$ where $t' \leq t$ and φ is some PAL formula.*

5.3 PAL semantics

The semantics of our logic is similar to CTL* [Rey02], namely a tree structure containing nodes and edges connecting the nodes. A tree can equivalently be seen as an enfolded transition system, thereby representing all the possible runs through it. We choose to represent our semantics using trees because it simplifies the completeness proofs. See [Rey02] for an overview of different kinds of semantics and conceptual underpinnings.

With each natural number $i \in \mathbb{N}$ we associate a set of states S_i such that all these sets are disjoint. We then define the accessibility relation between states such that it generates an infinite, single tree.

Definition 5.3 (Tree)**.** *A tree is quadruple $T = (S, R, v, act)$ where*

- $S = \bigcup_{n\in\mathbb{N}} S_n$ *is a set of states, such that each* S_t *is the set of states at time* t, $S_i \cap S_j = \emptyset$ *for* $i \neq j$;
- $R \subseteq \bigcup_{n\in\mathbb{N}} S_n \times S_{n+1}$ *is an accessibility relation that is serial, linearly ordered in the past and connected (so* S_0 *is a singleton);*
- $v : S \to 2^{Prop}$ *is a valuation function from states to sets of propositions;*
- $act : R \to \mathsf{Act}$ *is a function assigning actions to elements of the accessibility relation, such that actions are deterministic, i.e. if* $act((s,s')) = act((s,s''))$, *then* $s' = s''$.

We evaluate formulas on a path in a tree. A path is a sequence of states in a tree, connected by the accessibility relation R.

Definition 5.4 (Path)**.** *Given a tree* $T = (S,R,v,act)$, *a* path $\pi = (s_0, s_1, \ldots)$ *in* T *is a sequence of states such that* $(s_t, s_{t+1}) \in R$. *We write* π_t *to refer to the* t*'th state of the path* π. *We use elements of the path as arguments for the valuation function and the action function:*

- $v(\pi_t)$ *are the propositions true on path* π *at time* t;
- $act((\pi_t, \pi_{t+1}))$ *is the next action on path* π *at time* t. *We abbreviate* $act((\pi_t, \pi_{t+1}))$ *with* $act(\pi_t)$, *since* π_{t+1} *is uniquely determined by the action.*

Intuitively, $v(\pi_t)$ are the propositions true at time t on path π, and $act(\pi_t)$ is the next action a on the path. We next define an equivalence relation $\sim_t$ on paths, which is used to give semantics to the modal operator.

Definition 5.5 (Path equivalence)**.** *Two paths* π *and* π' *are equivalent up to time* t, *denoted* $\pi \sim_t \pi'$, *if and only if the states up to time* t *are the same, i.e.*

$$\pi \sim_t \pi' \text{ iff } (\forall t' \leq t).(v(\pi_{t'}) = v(\pi'_{t'})) \text{ and } (\forall t' < t).(act(\pi_{t'}) = act(\pi'_{t'})).$$

Formulas in PAL are evaluated on a path. Therefore, a model for a formula is pair consisting of a tree and a path in this tree. This, together with some additional constraints related to the pre- and post-conditions of actions, is our definition of a *model*.

Definition 5.6 (Model)**.** *A* model *is a pair* (T,π) *with* $T = (S,R,v,act)$ *such that for all* $\pi \in T$ *the following holds:*

1. *If* $act(\pi_t) = a$, *then* $post(a) \in v(\pi_{t+1})$,
2. *If* $pre(a) \in v(\pi_t)$, *then there is some* π' *in* T *with* $\pi \sim_t \pi'$ *and* $act(\pi'_t) = a$,
3. *If* $pre(a,b,\ldots) \in v(\pi_t)$, *then there is some* π' *in* T *with* $\pi \sim_t \pi'$, $act(\pi'_t) = a$, *and* $pre(b,\ldots) \in v(\pi'_{t+1})$.
4. *If* $pre(\ldots,a,b) \in v(\pi_t)$, *then* $pre(\ldots,a) \in v(\pi_t)$,

We refer to models of PAL with $m_1, m_2, \ldots$, *we refer to sets of models with* $M_1, M_2, \ldots$, *and we refer to the set of all models with* $\mathbb{M}$.

Most of the conditions on models are there to formalize the asymmetry we put on pre-and-postconditions, as discussed in the introduction. Condition 1 is straightforward: we simply expect postconditions to hold in a state after an action has been executed. Condition 2 and 3 put the weaker requirement on the preconditions for actions: If the precondition holds, then there is *some* path in which the action is executed. The opposite direction, stating that preconditions are necessary for executing actions, will be formalized with a coherence condition on beliefs and intentions in the next chapter. Condition 4 of a model simply ensures that if the precondition of a sequence of action is true in a state, then the precondition for any subsequence by removing actions from the end of the sequence is also true in that state.

Example 5.3 (Running example (Ctd.))**.** *Consider the partial PAL model (T,π') of the beliefs of Bobby the household robot from time 0 to time 2 in figure 5.2, where the thick path represents the actual path. Note that we have omitted pre-and postconditions for the NOP action. We provide some examples of the conditions of our model (definition 5.6):*

- *since $act(\pi_1) = equip$, $post(equip) \in v(\pi_2)$ holds as well (condition 1),*
- *since $pre(cook) \in v(\pi_1'')$, there is some path, namely π' with $\pi' \equiv_t \pi''$ and $act(\pi_1') = cook$ (condition 2),*
- *since $pre(food, cook) \in v(\pi_0)$, there exists some path, namely π' with $\pi \equiv_0 \pi', act(\pi_0') = food$, and $pre(cook) \in v(\pi_1')$ (condition 3),*
- *since $pre(food, cook) \in v(\pi_0), pre(food) \in v(\pi_0)$ holds as well (condition 4).*

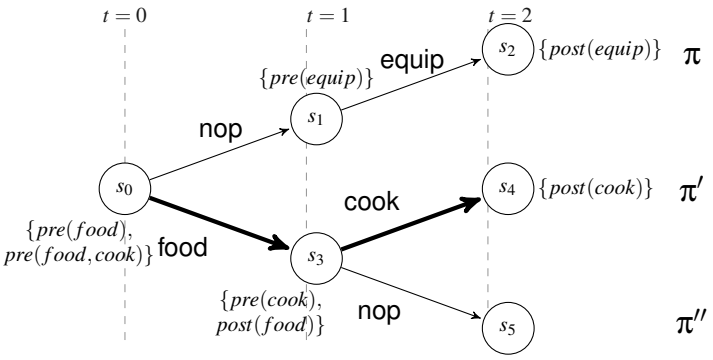


Figure 5.2: Example PAL Model (T,π') from $t = 0$ to $t = 2$. Pre-and postconditions for *NOP* actions are omitted for readability.

We now provide the truth definitions. Recall that formulas are evaluated on a path as a whole, and not in a state.

Definition 5.7 (Truth definitions)**.** *Let $m = (T,\pi)$ be a model with $T = (S,R,v,act)$:*

$T,\pi \models \chi_t$ *iff* $\chi \in v(\pi_t)$ *with* $\chi \in Prop$

$T,\pi \models do(a)_t$ *iff* $act(\pi_t) = a$

$T,\pi \models \neg\varphi$ *iff* $T,\pi \not\models \varphi$

$T,\pi \models \varphi \wedge \varphi'$ *iff* $T,\pi \models \varphi$ *and* $T,\pi \models \varphi'$

$T,\pi \models \Box_t\varphi$ *iff for all* π' *in* T*: if* $\pi' \sim_t \pi$*, then* $T,\pi' \models \varphi$

The truth definitions state that propositions are simply evaluated using the valuation function v, but *do* statements are different. They are about state transitions, and therefore use the action function act. Since we are evaluating formulas from a state, the modal operator $\Box_t$ is indexed with a time point t, and corresponds to the equivalence relations $\sim_t$.

Example 5.4 (Running Example (Ctd.))**.** *We provide some example of applications of the truth definition for the model in figure 5.2:*

$T, \pi \models pre(food)_0 \wedge do(nop)_0 \wedge do(equip)_1$

$T, \pi \models \Diamond_0(do(food)_0 \wedge \neg do(cook)_1)$

$T, \pi \models \Diamond_0 do(food)_0 \wedge \Diamond_0 do(equip)_1 \wedge \neg\Diamond_0(do(food)_0 \wedge do(equip)_1)$

$T, \pi' \models pre(food, cook)_0$

$T, \pi' \not\models do(equip)_1$

$T, \pi'' \models \Diamond_0 \neg\Diamond_1 do(cook)_1$

Next we turn to the notions of validity, satisfiability, and semantic consequence.

Definition 5.8 (Validity, satisfiability, and semantic consequence)**.**

- φ *is* valid*, i.e.* $\models \varphi$ *iff for all models m:* $m \models \varphi$.
- φ *is* satisfiable *iff there exists a model m such that* $m \models \varphi$.
- φ *is a* semantic consequence *of a set of formula* Σ*, i.e.* $\Sigma \models \varphi$ *iff* $\forall m : m \models \Sigma \Rightarrow m \models \varphi$.

Definition 5.9 (Model of a formula)**.** *We say that a model m is a model of a formula* φ *if* $m \models \varphi$*. We denote the set of all models of a formula* φ *by* $Mod(\varphi)$*, i.e.,*

$$Mod(\varphi) = \{m \in \mathbb{M} \mid m \models \varphi\}.$$

5.4 PAL axiomatization

In this part we present the axiomatization of our logic, and we explain the most important axioms in turn.

Propositional tautologies	**(PROP)**
$\Box_t(\varphi \to \varphi') \to (\Box_t\varphi \to \Box_t\varphi')$	**(K)**
$\Box_t\varphi \to \varphi$	**(T)**
$\Diamond_t\varphi \to \Box_t\Diamond_t\varphi$	**(5)**

Axioms PROP, K, T, and 5 together ensure our modal operator is an equivalence relation. This is simply the modal logic system KT5 or KD45.

$\chi_t \to \Box_t\chi_t$, where $\chi \in \mathsf{Prop}$	**(A1)**
$\Diamond_t\chi_t \to \chi_t$, where $\chi \in \mathsf{Prop}$	**(A2)**

Axioms A1 states that if a proposition is true in a state on a path, then it is necessarily true at that time, i.e., it is true in all equivalent paths. The contraposition of Axiom A2 states the same for negated propositions. These axioms follow from the definition of the equivalence $\sim_t$ between paths: if two paths are equivalent up to time t, then the same propositions are true in time t as well.

$$do(a)_t \rightarrow \Box_{t+1} do(a)_t \quad \textbf{(A3)}$$
$$\Diamond_{t+1} do(a)_t \rightarrow do(a)_t \quad \textbf{(A4)}$$

Axioms A3 and A4 are similar to A1 and A2, but then for the case of actions. Recall that *do* statements are semantically represented as transitions between states (definition 5.7). Therefore, the modal operator is indexed with the next time point $t+1$.

$$\Box_t \varphi \rightarrow \Box_{t+1} \varphi \quad \textbf{(A5)}$$

Axiom A5 is a result of the fact that for some path π the number of paths equivalent with π can only decrease as time moves forward. Therefore, if something is true on all paths equivalent up to time t, then it is necessarily true on all paths equivalent up to the next time moment $t+1$.

$$\bigvee_{a \in \mathsf{Act}} do(a)_t \quad \textbf{(A6)}$$
$$do(a)_t \rightarrow \bigwedge_{b \neq a} \neg do(b)_t \quad \textbf{(A7)}$$

Axioms A6 and A7 together state that exactly one action is executed at every time moment.

$$do(a)_t \rightarrow post(a)_{t+1} \quad \textbf{(A8)}$$
$$pre(a)_t \rightarrow \Diamond_t do(a)_t \quad \textbf{(A9)}$$
$$(pre(a,b,\ldots)_t \wedge do(a)_t) \rightarrow pre(b,\ldots)_{t+1} \quad \textbf{(A10)}$$
$$pre(\ldots,a,b)_t \rightarrow pre(\ldots,a)_t \quad \textbf{(A11)}$$

Axioms A8-A11 directly correspond to properties 1-4 of a model (definition 5.6).

$$\Diamond_t (do(a)_t \wedge \alpha) \rightarrow \Box_t (do(a)_t \rightarrow \alpha) \quad \textbf{(A12)}$$
where $\alpha \in Past(t+1)$

Axiom A12 ensures actions are deterministic. If something holds immediately after performing action a in time t (which is why $\alpha \in Past(t+1)$), then it necessarily holds after performing that action in time t. Note this formula does not hold without the restriction of α to $Past(t+1)$, because because formulas containing time points greater than $t+1$ may depend on actions performed after time t.

In addition to these axioms, PAL has two inference rules, a variant of Necessitation and Modus Ponens:

From φ, infer $\Box_t\varphi$ **(NEC)**
From $\varphi, \varphi \to \varphi'$, infer φ' **(MP)**

Remark 3. *In our current axiomatization and semantics, preconditions are sufficient conditions of actions to be possible, but they are not necessary (Axiom A9, respectively condition 2 of definition 5.6). Alternatively, one may strengthen this axiom as follows:*

$$pre(a)_t \leftrightarrow \Diamond_t do(a)_t \qquad (A9^*)$$

The correspondence is preserved by changing the condition (2) of definition 5.6 from an "if" to an "if and only if". However, our choice for A9 is an implementation of Shoham's idea of "opportunistic planning": a planner may form intentions, even though at the moment of planning it may not be clear whether preconditions are true [Sho09].

We next formalize the notion of theorems and derivability.

Definition 5.10 (Theorems in PAL)**.** *A derivation of φ within PAL is a finite sequence $\varphi_1, \ldots \varphi_m$ of formulas such that:*

1. $\varphi_m = \varphi$;
2. *every φ_i in the sequence is either*
   (a) *(an instance of) one of the axioms*
   (b) *the result of the application of Necessitation or Modus Ponens to formulas in the sequence that appear before φ_i.*

If there is such a derivation for φ we write $\vdash \varphi$. and we say φ is a theorem of PAL.

We define theorems and derivability separately because we restrict the application of the Necessitation rule to theorems only.

Definition 5.11 (Derivability in PAL)**.** *A derivation for a formula φ from a set of formulas Σ is a finite sequence $\varphi_1, \ldots \varphi_m$ of formulas such that:*

1. $\varphi_m = \varphi$;
2. *every φ_i in the sequence is either a theorem, a member of Σ, or the result of the application of Modus Ponens to formulas in the sequence that appear before φ_i.*

If there is such a derivation from Σ for φ we write $\Sigma \vdash \varphi$. We then also say that φ is derivable from the premises Σ.

Furthermore, a set of formulas Σ is *consistent* if we cannot derive a contradiction from it, i.e., $\Sigma \nvdash \bot$, and a set of formulas Σ is *maximally consistent* if it is consistent and every superset is inconsistent.

We denote by $Cn(\Sigma)$ the set of consequences of Σ, i.e.

$$Cn(\Sigma) = \{\varphi \mid \Sigma \vdash \varphi\}.$$

5.5 Soundness and completeness

In this section we prove the axiomatization of PAL is sound and strongly complete with respect to its semantics.

Theorem 5.1 (Completeness Theorem)**.** *The logic PAL is sound and strongly complete, i.e.* $\Sigma \vdash \varphi$ *iff* $\Sigma \models \varphi$.

We provide a proof sketch of the theorem. The full proofs of all the results in this chapter can be found in the appendix.

Proof Sketch. We prove the following formulation of completeness: each consistent set of formulas Σ has a model. We prove the Lindenbaum lemma, stating that each consistent set can be extended to a maximally consistent set Σ', i.e. Σ' is consistent and each proper superset of Σ' is inconsistent. In the first step we extend Σ to a maximally consistent set Σ^0.

Then for each t we define an equivalence relation $\equiv_t$ on maximally consistent sets in the following way:

$$\Sigma_1^* \equiv_t \Sigma_2^* \text{ iff } \Sigma_1^* \cap Past(t) = \Sigma_2^* \cap Past(t).$$

Let us denote the corresponding equivalence classes by $[\Sigma^*]_t$, which means $\{\overline{\Sigma}^* \mid \Sigma^* \equiv_t \overline{\Sigma}^*\}$.

In the second part of the proof, using the maximally consistent superset Σ^* of Σ (which exists by the Lindenbaum lemma), we define the tree $M_{\Sigma^*} = (S, R, v, a)$:

1. $S = \bigcup_{t\in\mathbb{N}} S_t$ where $S_t = \{[\overline{\Sigma}^*]_t \mid \overline{\Sigma}^* \equiv_t \Sigma^*\}$
2. sRs' iff $(\exists \overline{\Sigma}^*, t \in \mathbb{N}).(s = [\overline{\Sigma}^*]_t \wedge s' = [\overline{\Sigma}^*]_{t+1})$
3. $\chi \in v(s)$ iff $(\exists \overline{\Sigma}^*, t \in \mathbb{N}).(s = [\overline{\Sigma}^*]_t \wedge \chi_t \in \overline{\Sigma}^*)$.
4. $a = act((s,s'))$ iff $(\exists \overline{\Sigma}^*).(s = [\overline{\Sigma}^*]_t \wedge s' = [\overline{\Sigma}^*]_{t+1} \wedge do(a)_t \in \overline{\Sigma}^*)$.

Given a mcs T^*, we construct a path $\pi_{T^*} = (s_0, s_1, \ldots)$ from it by letting $s_t = [T^*]_t$. So $p \in v_p(\pi_{T^*t})$ iff $p_t \in T^*$ and $a = act(\pi_{T^*t})$ iff $do(a)_t \in T^*$.

If $\pi(\Sigma^*) = (s_0, s_1, \ldots)$, where $s_t = [\Sigma^*]_t$, then one can show that $(M_{\Sigma^*}, \pi(\Sigma^*))$ is a model. Finally, we prove that for each φ, $(M_{\Sigma^*}, \pi(\Sigma^*) \models \varphi$ iff $\varphi \in \Sigma^*)$, using induction on the depth of the proof. Consequently, $M_{\Sigma^*}, \pi(\Sigma^*) \models \Sigma$. □

5.6 Discussion

Since this part consists of two separate chapter, we only discuss the related work, open issues and conclusions of this chapter here. For related work and open issues on intentions, please see the discussion in the next chapter.

5.6.1 Related work

Time in temporal logics can be defined in an *implicit* or *explicit* manner. A time model is implicit when the meaning of formulas depends on the evaluation time, and this is left implicit in the formula. Standard LTL and CTL define time implicitly. For instance, $\Box\Phi$ means that $\forall t \in [T_0, \infty].\Phi(t)$, where T_0 is the evaluation time (the so-called current time instant). A standard way of introducing real time into the syntax of temporal languages constrains the temporal operators with time intervals [ET99, Koy90, AH90, AFH96]. In order to model such time intervals, timed automata may be used. These automata model the behavior of time-critical systems. A timed automaton is in fact a program graph that is equipped with a finite set of real-valued clock variables, called *clocks* for short [AD94]. Timed CTL (TCTL, for short) is a real-time variant of CTL aimed to express properties of timed automata. In TCTL, the until modality is equipped with a time interval such that $\Phi U^J \Psi$ asserts that a Ψ-state is reached within $t \in J$ time units while only visiting Φ-states before reaching the Ψ-state. The formula $\exists\Box^J\Phi$ asserts that there exists a path for which during the interval J, Φ holds; $\forall\Box^J\Phi$ requires this to hold for all paths [BK08]. While such logics allow one to express timed constraints on the modalities in TCTL, there is no way to refer explicitly to the states at which a certain formula holds.

When time is explicit, the language represents the time through a variable. For example, in the following formula an explicit model of time is used:

$$\forall t.\Box(E \wedge T = t) \rightarrow \Diamond(A \wedge T - t < 10)$$

where E is an event [BMN00]. This is for instance formalized by Ostroff when solving control problems using real-time temporal logic (RTTL) [Ost89].

The logic of strategic abilities ATL* (Alternative-Time Temporal Logic), introduced and studied by Alur *et al.* [AHK98], is a logical system, suitable for specifying and verifying qualitative objectives of players and coalitions in concurrent game models. Formally, ATL* is a multi-agent extension of the branching time logic CTL* with *strategic path quatifiers* $\langle\langle C \rangle\rangle$ indexed with coalitions C of players. Bulling and Goranko [BG13] propose a quantitative extension of ATL*, in which it is possible to express temporal constraints as well. For instance, the expression $\phi \wedge x = t$ denotes that ϕ will be true after t transitions, where each transition adds 1 to x.

Many logical systems have been developed for reasoning about the pre and postconditions of actions with explicit time points, such as the Event Calculus [Mue10], Temporal Action Logics [Kva05], extensions to the Fluent Calculus [Thi01], and extensions to the Situation Calculus [PP03] (see [Pat10, Ch.2] for an overview). Our logic is considerably simpler, but the reason for this is because of the type of revision we aim to characterize in the next chapter. Although there are a number of correspondences between AGM postulates and some of the approaches above, none of them prove representations theorems linking revision to a total pre-order on models, which is what we aim to do in the next section.

The logic PAL is also closely related to the logic developed by Icard *et al.* [IPS10], who study the joint revision of beliefs and intentions using AGM-like postulates. While many of the ideas are similar to ours, our logic has a standard branching time semantics, while the path semantics of Icard is less familiar.

5.6.2 Open issues

The most interesting open issues of this last part of the thesis can be found in the next chapter, but when looking at this chapter in isolation one can already identify some directions from here that can be explored.

The frame problem is one of the most fundamental problems in reasoning about action and change. The challenge is how to specify the non-effects of actions succinctly. For instance, if a proposition such as "The table is red" is true at some time t, and no action occurs that affects the truth value of this proposition, then it seems plausible that the table is still red at time $t+1$. Currently, our framework does not contain so-called "frame axioms".

Other mental attitudes, such as intentions, goals, desires and preferences, we have left out completely. This is not because we assume that they are unimportant, but because it was our goal to focus on formalizing the database perspective in a temporal setting.

5.6.3 Conclusion

We present a temporal logic for reasoning about beliefs and action in time. Semantically, our logic is close to CTL* but has a number of important differences. First, PAL contains time-indexed modalities, which allows one to express statements such as "It is possible in February that I will attend IJCAI in July". Secondly, PAL allows for explicit reasoning about pre-and postconditions of deterministic actions. Thirdly, PAL only contains one type of modality, while CTL* contains a next operator and an until operator as well. The last two reasons arguably make PAL less expressive that CTL*, but in return the axiomatization is straightforward, we are able to obtain strong completeness, and it is possible to prove both representation theorems. It seems that this is not possible for CTL* in general.

The main motivation for developing the logic in the chapter is to obtain a representation result for belief and intention revision in the next chapter.

6

The Dynamics of Beliefs and Intentions

Abstract In this chapter we study the dynamics of beliefs and intentions in Parameterized-time Action Logic. In order to develop a coherence condition on belief and intention, we separate strong beliefs from weak beliefs. Strong beliefs are independent from intentions, while weak beliefs are obtained by adding intentions to strong beliefs and everything that follows from that. We provide AGM-style postulates for the revision of strong beliefs and intentions: strong belief revision may trigger intention revision, but intention revision may only trigger revision of weak beliefs. After revision, the strong beliefs are coherent with the intentions. We show in a representation theorem that a revision operator satisfying our postulates can be represented by a pre-order on interpretations of the beliefs, together with a selection function for the intentions.

6.1 Introduction

In the previous chapter we developed Parameterized-time Action Logic, which is a logic for reasoning about beliefs and actions in time. We developed a branching-time semantics for the logic, axiomatized it, and proved it is sound and strongly complete. However, this was only the first step of our final goal, namely to study the dynamics of beliefs and intentions based on Shoham's database perspective (see Chapter 5, Section 5.1 for a more detailed explanation).

In this chapter we do so in the following way. First, where we limited ourselves to developing a logic for the belief database of an agent in the previous chapter, we now add an intention database to the system as well. We then separate strong beliefs from weak beliefs as described in Chapter 5, Section 5.1. Strong beliefs are beliefs that occur in the belief database, and they are independent of intentions. Weak beliefs are obtained from strong beliefs by adding intentions to the strong beliefs, and everything that follows from that. In this way, the weak beliefs represent the *assumptions* that an agent may use to form new intentions. We then formalize a *coherence condition* on the beliefs and intentions. This condition states that the agent weakly believes it is possible to jointly perform all of its intended actions. The main technical result of this chapter is that we develop a set of postulates for the joint revision of belief and intentions, and that we prove a variation of the Katsuno and Mendelzon [KM91] representation theorem. To this end, we define a revision operator that revises beliefs up to a specific time point. We show that this leads to models of system behaviors which can be finitely generated, i.e. be characterized by a single formula.

This chapter is organized as follows: In Section 6.2 we add an intention database to the system, we separate strong and weak beliefs, and we propose a coherence condition on beliefs and intentions. In Section 6.3 we then turn to the main topic of this part:

the revision of beliefs and intentions. In Section 6.4 we study iterated revision, and in Section 6.5 we discuss related work, open issues, and a conclusion.

6.2 Adding intentions

In the previous chapter we developed a logic for the belief database of Shoham's database perspective (Figure 5.1). We did not yet take intentions into account, which is what we do in this section. Recall intentions are formalized as *discrete atomic action intentions* of the form (a,t). We focus on two main tasks: separating beliefs depending on intentions (*weak beliefs*) from those that are not (*strong beliefs*), and formalizing a coherence condition on beliefs and intentions. These two tasks correspond to the two subsections of this section.

6.2.1 Separating strong and weak beliefs

The idea behind strong beliefs (the terminology due to van der Hoek *et al.* [VdHW03]) is that they represent the agent's ideas about what is inevitable, no matter how it would act in the world. In our setting, a set of strong beliefs is a set of formulas starting either with $\Diamond_0$ or $\Box_0$, and all consequences that follow from it. First, we define a language for strong beliefs.

Definition 6.1 (Strong belief)**.** *The* set of all of strong beliefs $\mathbb{SB}$ *for* $\mathscr{L}$ *is inductively defined by the following BNF grammar:*

$$\varphi ::= \Box_0\psi \mid \varphi \wedge \varphi \mid \neg\varphi,$$

where $\psi \in \mathscr{L}$*. A* strong belief *is an element of* $\mathbb{SB}$*.*

We next provide some examples of strong beliefs for our running example.

Example 6.1 (Running example, Ctd.)**.** *Some examples of strong belief formulas are:*

- $\Diamond_0(do(food)_0 \wedge \neg do(cook)_1)$
- $\Diamond_0 do(food)_0 \wedge \Diamond_0 do(equip)_1 \wedge \neg\Diamond_0(do(food)_0 \wedge do(equip)_1)$
- $\Diamond_0\neg\Diamond_1 do(cook)_1$
- $\Box_0\Diamond_0 do(cook)_1$

Next we define a set of strong beliefs, which is generated from the set of all strong beliefs, and closed under consequence.

Definition 6.2 (Set of strong beliefs)**.** *A* set of strong beliefs *SB is a subset of formulas from* $\mathbb{SB}$*, closed under consequence, i.e.* $SB = Cn(\Sigma)$ *where* $\Sigma \subseteq \mathbb{SB}$*.*

The following example shows that a set of strong beliefs may also contain formulas which are not in $\mathbb{SB}$, since they are closed under consequence.

Example 6.2 (Set of strong beliefs)**.** *Let* $\Sigma = \{\neg\Diamond_0 p_3, \Box_0 q_2\} \subset \mathbb{SB}$*, and let the set of strong beliefs* $SB = Cn(\Sigma)$*. From Axioms A1 and A2 we obtain* $\neg p_3 \in SB$*, as well as* $q_2 \in SB$*.*

The reader may already have noted that, semantically, strong beliefs are independent of the specific path on which they are true. Indeed, strong beliefs are true in a tree rather than on a single path. Therefore, if a model (consisting of a tree and a path) is a model for a strong belief formula φ, then all possible models with the same tree are models of the strong belief formula φ. We make this idea precise in the following definition.

Definition 6.3 (Set of models of strong beliefs (msb set))**.** *A* set of models of strong beliefs $MSB \subseteq \mathbb{M}$ *(i.e., an* msb set*) is a set of models satisfying the following condition:*

$$\textit{If } (T,\pi) \in MSB\textit{, then } (T,\pi') \in MSB \textit{ for all } \pi' \in T.$$

The set $\mathbb{MSB}$ *contains all msb sets.*

Definition 6.3 ensures that if some model (T,π) is in a set of models of a strong belief, then all other models (T,π') are also in this set. The following proposition shows a direct correspondence between a set of strong beliefs and its models.

Proposition 6.1. *Given a set of strong beliefs SB, the set of models of SB is an msb set, i.e.,* $Mod(SB) \in \mathbb{MSB}$*.*

We now explain the semantics of strong beliefs models with our running example.

Example 6.3 (Running example (Ctd.))**.** *Consider the tree T of Figure 5.2 and let* $\overline{\pi} \in \{\pi,\pi',\pi''\}$*. The following statements hold:*

- $T,\overline{\pi} \models \Diamond_0(do(food)_0 \wedge \neg do(cook)_1)$
- $T,\overline{\pi} \models \Diamond_0 do(food)_0 \wedge \Diamond_0 do(equip)_1 \wedge \neg\Diamond_0(do(food)_0 \wedge do(equip)_1)$
- $T,\overline{\pi} \models \Diamond_0\neg\Diamond_1 do(cook)_1$
- $T,\overline{\pi} \models \Box_0\Diamond_0 do(cook)_1$

We obtain a belief-intention database by adding intentions to the strong beliefs. By intentions we assume action-time pairs, and an intention database is a set of intentions. We also add the constraint that at most one action is intended for a given time moment.

Definition 6.4 (Belief-Intention database)**.** *An* intention (a,t) *is a pair consisting of an action* $a \in \mathsf{Act}$ *and a time point t.*

An intention database $I = \{(a_1,t_1),(a_2,t_2),\ldots\}$ *is a set of intentions such that no two intentions exist at the same time point, i.e if* $i \neq j$ *then* $t_i \neq t_j$*.*

A belief-intention database (SB,I) *consists of a set of strong beliefs SB closed under consequence, i.e.* $SB = Cn(SB)$*, and an intention database I.*

We define weak beliefs by adding intentions to the strong beliefs, and closing the result under consequence.

Definition 6.5 (Weak Beliefs)**.** *Given a belief-intention database (SB,I), the weak beliefs are defined as follows:*

$$WB(SB,I) = Cn(SB \cup \{do(a)_t \mid (a,t) \in I\}).$$

We provide an example for weak beliefs using our running example.

Example 6.4 (Running example (Ctd.))**.** *Suppose the set SB contains strong beliefs describing the tree T of Figure 5.2. Some of the formulas in SB are:*

- $\Diamond_0(do(food)_0 \wedge do(cook)_1)$
- $\Diamond_0 do(equip)_1$
- $\neg\Diamond_0 do(food)_0 \wedge do(equip)_1\}$.

Let $I = \{(food,0),(cook,1)\}$. Some examples of weak beliefs $WB(SB,I)$ are:

- $do(food)_0 \wedge do(cook)_1$
- $\neg do(equip)_1$
- $post(food)_1 \wedge post(cook)_2$.

Note the model (T,π') from Figure 5.2 is a model of $WB(SB,I)$.

Note the difference between Example 6.3 and Example 6.4. Strong beliefs are true in a tree, while weak beliefs *depend on a path*. In this way, weak beliefs are contingent on the action executed on the actual path. We can thus understand adding intentions to strong beliefs semantically by *choosing a path* in a tree.

Remark 4. *Note that since weak beliefs contain strong beliefs with intentions, and everything following from that, they also contain postconditions of actions. For instance, if $I = \{(a,t)\}$ and $SB = \emptyset$, then $post(a)_t \in WB(SB,I)$. However, it does not mean that preconditions of intended actions are believed as well, i.e. $pre(a)_t \notin WB(SB,I)$. This is what is ensured by Axiom A9 (see also remark 3).*

6.2.2 The coherence condition on beliefs and intentions

The fact that postconditions of actions always hold on a path, but that preconditions may not, is a direct implementation of our proposal that preconditions, unlike postconditions, need not be believed when an action is intended. We might therefore think of our belief model as, in some sense, one of "optimistic" or "imaginary" beliefs. On the other hand, we do add a slightly weaker requirement, namely that it is considered possible to perform all intended action. We thus require that the joint preconditions of all intended actions not be disbelieved by the agent. This is our notion of coherence.

Definition 6.6 (Coherence)**.** *Given an intention database $I = \{(b_{t_1},t_1),\ldots,(b_{t_n},t_n)\}$ with $t_1 < \ldots < t_n$, let*

$$Cohere(I) = \Diamond_0 \bigvee_{\substack{a_k \in Act : k \notin \{t_1,\ldots,t_n\} \\ a_k = b_k : k \in \{t_1,\ldots,t_n\}}} pre(a_{t_1}, a_{t_1+1}, \ldots, a_{t_n})_{t_1}.$$

- *For a given msb set MSB (definition 6.3), we say that* (MSB, I) *is* coherent *iff there exists some* $m \in MSB$ *with* $m \models Cohere(I)$.
- *For a given belief-intention database* (SB, I)*, we say that it is* coherent *iff SB is consistent with* $Cohere(I)$*, i.e.,* $SB \nvdash \neg Cohere(I)$.
- *A pair* (ψ, I) *consisting of a strong belief formula* $\psi \in \mathbb{SB}$ *and an intention database I is coherent iff* ψ *is consistent with I, i.e.* $\psi \nvdash \neg Cohere(I)$.[1]

Let us explain this definition with a simple example.

Example 6.5. *Let* $\mathsf{Act} = \{a, b\}$ *and* $I = \{(a,1),(b,3)\}$*. Then,*

$$Cohere(I) = \Diamond_0 \bigvee_{x \in \mathsf{Act}} pre(a,x,b)_1 = \Diamond_0 (pre(a,a,b)_1 \vee pre(a,b,b)_1).^2$$

Intuitively, intentions cohere with beliefs if the agent considers it possible to jointly carry out all of the intended actions. This is a kind of minimal requirement on *rational balance* between the two mental states.

In the next section we will consider the revision of belief-intention database. We will require that a belief-intention database is coherent after revision.

Remark 5. *A word is in order concerning this choice of coherence conditions. Consider our example of Bobby intending to buy food at time 0. As we pointed out, it is not actually necessary that Bobby believes it has sufficient money; only that it does not believe it does not have sufficient money.*

Anticipating our treatment of contingent beliefs, we can also ask, what can be Bobby's working assumptions about the future, upon adopting this intention? In so far as Bobby is committing himself to this action, we may assume that it will buy food at time 0. If we then consider the paths in our belief models on which this action is taken at time 0, the postconditions will hold along all of them. However, to allow that the preconditions may not yet be believed, we admit paths on which the preconditions do not hold. We only require that they hold on some path in the set, so that Bobby cannot stray too far from reality.

Indeed, this is arguably closer to how we reason about future actions. We often commit to actions without explicitly considering the path that will lead us there. Eventually this decision will have to be made, but there is nothing incoherent about glossing over these details at the current moment. Bobby should assume it will have bought food at time 1 and can continue making plans about what it will do with the food after this. But it should not assume the preconditions will hold until it has made further, specific plans for bringing them about. And at the current time, Bobby may not even bother worrying about it.

The next proposition shows a direct correspondence between a coherent belief-intention database and the semantic counterpart.

[1] We will use this formulation in the next section when we represent a set of strong beliefs *SB* by a single formula ψ.

[2] Our construction of preconditions over action sequences may lead to a coherence condition involving a big disjunction. This is a drawback in terms of computational complexity. Alternatively, one may explicitly denote the time of each precondition, e.g. $pre(a,b)_{(t_1,t_2)}$. We chose the former since it is closer syntax of the other propositions, but the latter can be implemented straightforwardly.

Proposition 6.2. *For a given belief-intention database (SB,I), (SB,I) is coherent iff $(Mod(SB),I)$ is coherent.*

We now apply the coherence condition to our running example.

Example 6.6 (Running example)**.** *Let (SB,I) be such that the strong beliefs are represented by the tree in Figure 6.1*[3]*. We consider different choices for I:*

- *Let $I = \{(food,0),(cook,1)\}$. In this case, $(Mod(SB),I)$ is coherent, since there is some model $m \in Mod(SB)$ with $m \models Cohere(I)$, i.e. $m \models \Diamond_0 pre(food,cook)_0$. In fact, from $pre(food,cook) \in v(s_0)$, it follows that $T,\overline{m} \models pre(food,cook)_0$ holds for each model $(T,\overline{m})$. From Proposition 6.2 it follows that (SB,I) is coherent as well.*

- *Let $I = \{(food,0),(equip,1)\}$. In this case $(Mod(SB),I)$ is not coherent, since there is not $m \in Mod(SB)$ with $m \models \Diamond_0 pre(food,cook)_0$. Again by Proposition 6.2 we obtain that (SB,I) is not coherent either.*

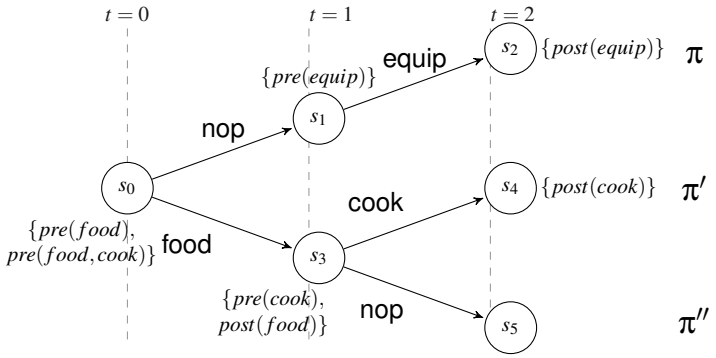


Figure 6.1: The tree T of Figure 5.2 reprinted.

Naturally, if a set of intentions is coherent with a set of strong beliefs, then its subset is coherent as well. This follows directly from the next lemma.

Lemma 6.1. *if $I' \subseteq I$, then $Cohere(I) \vdash Cohere(I')$.*

Next we show that a coherent belief-intention database implies joint consistency of beliefs and intentions.

Proposition 6.3. *Given some belief-intention database (SB,I), if (SB,I) is coherent, then $WB(SB,I)$ is consistent.*

Note the reverse direction of Proposition 6.3 does not hold. This is because of the nonparallel we have drawn between believing in preconditions and believing in postconditions (see Remark 5).

[3] In other words, the trees in each model in the msb set of *SB* is equivalent with the tree in Figure 6.1.

6.3 Revision of beliefs and intentions

Everything we did up until now has been to develop a static representation of beliefs (previous chapter), intentions and their joint consistency (Section 6.2). In this section we turn to the dynamic part of our databases by studying the revision of belief and intention. We provide and motivate a set of revision postulates on a belief-intention database (SB, I) in subsection 6.3.2, and we prove our main representation theorem in subsection 6.3.3.

The difficulty of obtaining our result is two-fold:

- When revising a belief database that is bounded up to some time t with a strong belief, we have to ensure that the resulting belief database is also bounded up to t,
- When revising a belief database we also have to ensure the new belief database remains a strong belief.

Our solution is to bound both the syntax of PAL and the revision operator up to some time t in the first subsection. In the second subsection we do the same for the semantics.

6.3.1 AGM belief revision

The AGM postulates [AGM85] formulate properties that should be satisfied by any (rational) revision operators defined on deductively closed sets of propositional formulas. [KM91] represent a belief set B as a propositional formula ψ such that $B = \{\varphi \mid \psi \vdash \varphi\}$. They define the following six postulates for revision on ψ and show that these are equivalent to the eight AGM postulates:

(R1) $\psi \circ_t \varphi$ implies φ

(R2) If $\psi \wedge \varphi$ is satisfiable, then $\psi \circ_t \varphi \equiv \psi \wedge \varphi$

(R3) If φ is satisfiable, then $\psi \circ_t \varphi$ is also satisfiable

(R4) If $\psi \equiv \psi'$ and $\varphi \equiv \varphi'$, then $\psi \circ_t \varphi \equiv \psi' \circ_t \varphi'$

(R5) $(\psi \circ_t \varphi) \wedge \varphi'$ implies $\psi \circ_t (\varphi \wedge \varphi')$

(R6) If $(\psi \circ_t \varphi) \wedge \varphi'$ is satisfiable, then $\psi \circ_t (\varphi \wedge \varphi')$ implies $(\psi \circ_t \varphi) \wedge \varphi'$

Given a set $\mathbb{I}$ of all interpretations over some propositional language, they define a faithful assignment as a function that assigns each ψ to a pre-order $\leq_\psi$ on models satisfying the following three conditions:

1. If $I, I' \in Mod(\psi)$, then $I <_\psi I'$ does not hold.
2. If $I \in Mod(\psi)$ and $I' \notin Mod(\psi)$, then $I <_\psi I'$ holds.
3. If $\psi \equiv \phi$, then $\leq_\psi = \leq_\phi$.

They show in a representation theorem that a revision operator $\circ$ satisfies postulates (R1)-(R6) iff there exists a faithful assignment that maps each formula ψ to a total pre-order $\leq_\psi$ such that

$$Mod(\psi \circ \varphi) = \min(Mod(\varphi), \leq_\psi).$$

6.3.2 Revision postulates

Recall from Section 6.1 that we aim to prove a representation theorem comparable to that of Katsuno and Mendelzon [KM91]. Therefore, we follow their convention to fix a way of representing a belief set *SB* consisting of strong beliefs by a propositional formula ψ such that $SB = \{\varphi \mid \psi \vdash \varphi\}$. One of the main difficulties in this respect is that time in PAL is infinite in the future, so it is generally not possible to represent *SB* closed under consequence by a single formula ψ, since this may potentially lead to an infinite conjunction. Therefore, we cannot prove the Katsuno and Mendelzon representation theorem directly. In this section, we define a *bounded* revision function and we restrict the syntax of PAL up to a specific time point. *Restriction* of a set of formulas up to a certain time point is defined as follows.

Definition 6.7 (t-restriction)**.** *Suppose some $t \in \mathbb{N}$. Let $max\_t(\varphi)$ denote the maximal time point occurring in φ. Let $Form_t = \{\varphi \in \mathscr{L} \mid max\_t(\varphi) \leq t\}$ and $Bel_t = \{Cl(S) \mid S \subseteq Form_t\}$.*

A set of strong beliefs bounded up to *t, denoted $SB^{|t}$, has the following two properties:*

- *$SB^{|t} \subseteq Bel_t$,*
- *$SB^{|t}$ is a set of strong beliefs.*

We next define some notation that we use in the rest of this section.

Definition 6.8. *We use the following conventions:*

- *By slight abuse of terminology, a pair (ψ, I) consisting of a strong belief formula ψ and an intention database I is also called a* belief-intention database*,*
- *$\mathbb{BI}$ denotes the set of all belief-intention databases,*
- *$\mathbb{I}$ denotes the set of all intentions,*
- *$\mathbb{ID}$ denotes the set of all intention databases.*
- *ε is the special "empty" intention.*

Recall $\mathbb{SB}$ is the set of all strong beliefs (Definition 6.1). We denote $\mathbb{BI}, \mathbb{SB}, \mathbb{I}$, and $\mathbb{ID}$ bounded up to t with respectively $\mathbb{BI}^{|t}, \mathbb{SB}^{|t}, \mathbb{I}^{|t}$, and $\mathbb{ID}^{|t}$. However, if the restriction is clear from context, we may omit the superscript notation.

In the next definition we define a bounded revision function $*_t$ revising a belief-intention database (ψ, I) with a tuple (φ, i) consisting of a strong belief φ and an intention i, denoted $(\psi, I) *_t (\varphi, i)$, where t is the maximal time point occurring in ψ, I, φ, and i.

Remark 6. *We will define a single revision function, revising by a pair* (φ, i)*. Our approach allows one to define separate revision as well:*

- *Revising by* (φ, ε) *mirrors revising by a strong belief* φ *and no intention,*
- *Revising by* $(\top, i)$ *mirrors revising by no belief and an intention i.*

We will show in example 6.8 that there are situations in which it is more desirable to revise with a pair (φ, i) *instead of first revising with* (φ, ε) *and then with* $(\top, i)$*, since it avoids throwing away intentions unnecessarily.*

Definition 6.9 (Belief-intention revision function)**.** *A* belief-intention revision function $*_t : \mathbb{BI} \times (\mathbb{SB} \times \mathbb{I}) \to \mathbb{BI}$ *maps a belief-intention database, a strong belief formula, and an intention— all bounded up to t— to a belief-intention database bounded up to t such that if,*
$(\psi, I) *_t (\varphi, i) = (\psi', I')$,
$(\psi_2, I_2) *_t (\varphi_2, i_2) = (\psi_2', I_2')$,
then following postulates hold:
$(P1)$ ψ' *implies* φ.
$(P2)$ *If* $\psi \wedge \varphi$ *is satisfiable, then* $\psi' \equiv \psi \wedge \varphi$.
$(P3)$ *If* φ *is satisfiable, then* ψ' *is also satisfiable.*
$(P4)$ *If* $\psi \equiv \psi_2$ *and* $\varphi \equiv \varphi_2$ *then* $\psi' \equiv \psi_2'$.
$(P5)$ *If* $\psi \equiv \psi_2$ *and* $\varphi_2 \equiv \varphi \wedge \varphi'$ *then* $\psi' \wedge \varphi'$ *implies* ψ_2'.
$(P6)$ *If* $\psi \equiv \psi_2$, $\varphi_2 \equiv \varphi \wedge \varphi'$, *and* $\psi' \wedge \varphi'$ *is satisfiable, then* ψ_2' *implies* $\psi' \wedge \varphi'$.
$(P7)$ *If* ψ *is consistent,* (ψ', I') *is coherent.*
$(P8)$ *If* $(\psi', \{i\})$ *is coherent, then* $i \in I'$.
$(P9)$ *If* $(\psi', I \cup \{i\})$ *is coherent, then* $I \cup \{i\} \subseteq I'$.
$(P10)$ $I' \subseteq I \cup \{i\}$.
$(P11)$ *If* $I = I_2$, $i = i_2$, *and* $\psi' \equiv \psi_2'$, *then* $I' = I_2'$.
$(P12)$ *For all* I'' *with* $I' \subset I'' \subseteq I \cup \{i\}$*:*$(\psi', I'')$ *is not coherent.*

Postulates (P1)-(P6) are the [KM91] postulates in our setting, which are equivalent to the AGM postulates. They also state that the revision of strong beliefs does not depend on the intentions. This can also be witnessed by the fact that all postulates involving intentions, i.e. (P7)-(P12), only use the revised strong belief formula ψ'. Therefore, revising strong beliefs does not depend on which intentions an agent had, or which intention it revises with. However, revising intentions *does* have an effect on the weak beliefs (see the last paragraph of example 6.7). In other words, the procedures runs as follows:

1. Revise strong beliefs (P1)-(P6),
2. Revise intentions (P7)-(P12), possibly revising weak beliefs as well.

Postulate (P7) states that the outcome of a revision should be coherent. Postulate (P8) states that the new intention i take precedence over all other current intentions; if possible, it should be added, even if all current intentions have to be discarded. It is thus comparable to the success postulate in AGM, albeit slightly weaker. Postulate (P9) and

(P10) together state that if it is possible to simply add the intention, then this is the only change that is made. These two postulates are comparable to inclusion and vacuity of AGM. Postulate (P11) states that if we revise with the same i but with a different belief, and we end up with the same belief in both cases, then we also end up with the same intentions. This postulates plays a crucial role in the interplay between beliefs and intentions, which we show in more detail in example 6.9. Finally, (P12) states that we do not discard intentions unnecessarily. This last postulate is a kind of maximality requirement, and is comparable to the *parsimony requirement* introduced by [GKPW10].

We now discuss our revision function with several example.

Example 6.7 (Running example (adding an intention))**.** *Suppose a belief-intention database* (ψ, I) *such that all models in* $Mod(\psi)$ *are the same as the partial model in Figure 6.1 up to* $t = 2$ *and suppose that* $I = \{(food, 0), (cook, 1)\}$*. That is, Bobby has the intention to buy food at time 0 and then to cook at time 1. Suppose now Bobby changes its intention to buy cleaning equipment at time 1. Formally:*

$$(\psi, I) *_1 (\top, (equip, 1)) = (\psi, I').$$

First note $(\psi, I \cup \{(equip, 1)\})$ *is not coherent because no two intentions can occur at the same time moment. Moreover, since* $(\psi, (equip, 1))$ *is coherent, from (P8) and (P9) we obtain* $(equip, 1) \in I'$*. Furthermore, from (P10) we have that* $I' \subseteq \{(food, 0), (cook, 1), (equip, 1)\}$*. Finally,* $(\psi, \{(food, 0), (equip, 1)\})$ *is not coherent either, since the agent does not believe the preconditions of buying food and buying equipment are true along a single path. Combining this gives* $I' = \{(equip, 1)\}$ *as the only coherent outcome. Thus, Bobby no longer intends to buy food and to cook, but to buy cleaning equipment instead.*

Note that, although the strong beliefs didn't change after revising with the new intention, the weak beliefs did change. For example, $post(food)_1 \in WB(\psi, I) \setminus WB(\psi, I')$ *and* $post(equip)_2 \in WB(\psi, I') \setminus WB(\psi, I)$*.*

The revision function $*_t$ takes a tuple (φ, i) as input, and the postulates (P1)-(P7) ensure that revision of strong beliefs occurs prior to the revision of intentions. Therefore, it may seem plausible that revising with (φ, i) is the same as first revising with (φ, ε) and then with $(\top, \imath)$. In other words, the following postulate seems to follow:

$$\begin{aligned} &\text{If } (\psi, I) *_t (\varphi, i) = (\psi', I') \\ &\text{and } ((\psi, I) *_t (\varphi, \varepsilon)) *_t (\top, i) = (\psi'', I''), \\ &\quad \text{then } \psi' \equiv \psi'' \text{ and } I' = I''. \end{aligned} \tag{P13*}$$

However, this property does not follow from (P1)-(P12), and we show in the following example that adding the postulate would in fact conflict with the maximality postulate (P12).

Example 6.8 (Joint vs separate revision)**.** *Suppose some belief-intention database* (ψ, I) *with beliefs up to* $t = 2$ *corresponding to the model on the left of Figure 6.2. It is possible to go to the dentist (dentist) or to stay at work (work), and after that to go eating (eating) or go to the movies (movies).*

Before revision, the intentions are $I = \{(dentist, 0), (eat, 1)\}$ *(left image of Figure 6.2, intentions shown as bold lines).*

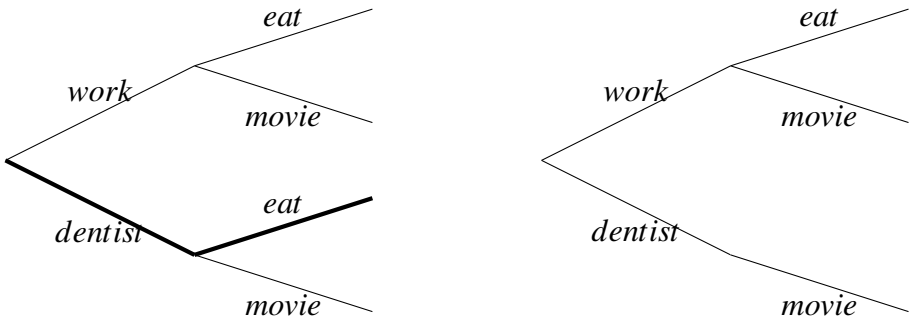


Figure 6.2: Left: Partial model of strong beliefs ψ of agent (ψ, I) with $I = \{(dentist, 0), (eat, 1)\}$ (bold lines). Right: Revised strong beliefs of agent after learning it is not possible to eat (*eat*) after the dentist (*dentist*).

Suppose now the beliefs are revised with the fact that it is not possible to go eating after going to the dentist (φ) and with the intention to go to the movie at time 1 ($i = (movie, 1)$). The resulting strong beliefs after revising with φ are shown on the right of Figure 6.2.

Let us analyze two ways of revising this information:

- *Suppose (ψ, I) is revised with both the new belief and the new intention. That is,*

$$(\psi, I) *_2 (\varphi, i) = (\psi', I').$$

  Both $(\psi', \{(dentist, 0), (movie, 1)\})$ and $(\psi', \{(movie, 1)\})$ are coherent, so by the maximality postulate (P12), $I' = \{(dentist, 0), (movie, 1)\}$. Hence, the new intentions are to go to the dentist and then to go to the movie.

- *Suppose beliefs are revised prior to intentions. That is,*

$$(\psi, I) *_2 (\varphi, \varepsilon) = (\psi', \bar{I})$$
$$(\psi', \bar{I}) *_2 (\top, i) = (\psi', \bar{I}').$$

  Now, since $(\psi', \{(dentist, 0)\})$ and $(\psi', \{(eating, 1)\})$ are both coherent, we either have $\bar{I} = \{(dentist, 0)\}$ or $\bar{I} = \{(eating, 1)\}$. Suppose that $\bar{I} = \{(eating, 1)\}$. In that case, since $(\psi', \{(eating, 1), (movie, 1)\}$ is incoherent, we obtain $\bar{I}' = \{(movie, 1)\}$ by the postulates (P8) and (P10).

Thus, (P13) doesn't hold. We see that revising separately allows a choice between the intention to go eating or to go to the dentist after revising beliefs. When choosing to go eating, the intention again has to be discarded because it is conflicting with the new intention to go to the movie. In joint revision, this is not the case since the choice between eating or the dentist can be made in light of the new incoming intention, and the maximal set can be chosen.*

We use the next example to explain postulate P11.

Example 6.9. *Suppose*

$$(\psi, I) *_t (\varphi, i) = (\psi', I')$$
$$(\psi_2, I) *_t (\varphi_2, i) = (\psi', I'_2),$$

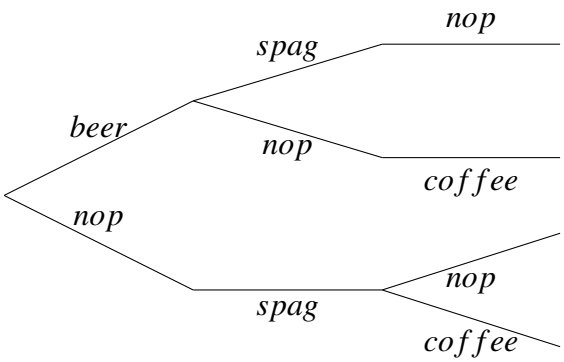


Figure 6.3: Tree representing Bobby's strong beliefs ψ' up to time $t = 2$ (example 6.9)

then postulate P11 ensures $I' = I'_2$. In words, this means that whenever we revise an intention database I by an intention i and we obtain the same strong beliefs ψ', then the new intention database should also be the same.

Suppose for instance our household robot Bobby has the intention database $I = \{(beer, 0), (spag, 1)\}$, i.e. it intends to buy beer at time 0 and to buy spaghetti at time 1. Suppose now Bobby learns it only has sufficient money to either buy beer and spaghetti, or beer and coffee. Bobby's beliefs ψ' are represented up to time 2 by the tree in Figure 6.3. If Bobby also adopts the intention to buy coffee at time 3, i.e. $i = (coffee, 3)$, then it should either drop $(beer, 0)$ or $(spag, 1)$ in order to incorporate i. Postulate P11 ensures that this choice is independent of its beliefs prior to revising them.

6.3.3 Representation theorem

In this subsection we present the main technical result of this thesis. We characterize all revision schemes satisfying (P1)-(P12) in terms of minimal change with respect to an ordering among interpretations and a selection function accommodating new intentions while restoring coherence.

In the previous subsection we bounded various sets of formulas up to some time point t. We now do the same for models. We bound models up to t, which means that all the paths in the model are "cut off" at t.

Definition 6.10 (t-bounded model)**.** *Suppose some model $m = (T, \pi)$.*

- *A t-*bounded path *$\pi^{|t}$ is defined from a path π in T as $\pi^{|t} = (\pi_0, \ldots, \pi_t)$.*
- *A t-*bounded model *$m^{|t} = (T^{|t}, \pi^{|t})$ is a pair where $T^{|t} = \{\pi^{|t} \mid \pi \in T\}$.*

We denote the set of all t-bounded models with $\mathbb{M}^{|t}$.

Recall we defined a set of models of strong beliefs *MSB* as an *msb set* (Definition 6.3). A belief database *SB* consists of a set of strong beliefs, and we showed in Proposition 6.1 that the set of models of *SB* is an msb set, i.e. $Mod(SB) \in \mathbb{MSB}$, where $\mathbb{MSB}$ is the set containing all msb sets.

In order to represent revision semantically, we define a t-bounded version of msb sets as well.

Definition 6.11 (t-bounded msb set)**.** *Given an msb set MSB (Definition 6.3), the* t-bounded msb set *contains all* t*-bounded models of MSB, i.e.*

$$MSB^{|t} = \{m^{|t} \mid m \in MSB\}.$$

Given an intention database I, we define a selection function γ_I^t that tries to accommodate a new intention based on strong beliefs. The selection function specifies preferences on which intention an agent would like to keep in the presence of the new beliefs.

Definition 6.12 (Selection Function)**.** *Given an intention database I, a* selection function $\gamma_I^t : \mathbb{MSB} \times \mathbb{I} \to \mathbb{ID}$ *maps an msb set (definition 6.3) and an intention to an updated intention database—all bounded up to t— such that if* $\gamma_I^t(MSB^{|t}, \{i\}) = I'$*, then:*

1. $(MSB^{|t}, I')$ *is coherent.*
2. *If* $(MSB^{|t}, \{i\})$ *is coherent, then* $i \in I'$.
3. *If* $(MSB^{|t}, I \cup \{i\})$ *is coherent, then* $I \cup \{i\} \subseteq I'$.
4. $I' \subseteq I \cup \{i\}$.
5. *For all* I'' *with* $I' \subset I'' \subseteq I \cup \{i\}$*:*$(MSB^{|t}, I'')$ *is not coherent.*

The five conditions on the selection function are in direct correspondence with postulates (P7)-(P10), (P12) of the agent revision function $*_t$. Postulate (P11) doesn't have a corresponding condition in the definition above, but it is represented in the representation theorem below (Theorem 6.1). In Condition 2 of the representation theorem, the selection function takes $Mod(\psi')$ as its first argument, which are the revised beliefs. This ensures the outcome of the intention revision process only depends on the new beliefs, the old intention database, and the incoming intention. In other words, it does not depend on the previous beliefs, or the belief that is used in revision. This is exactly what postulate (P11) states as well.

Remark 7. *We will show in Corollary 6.1 below that it is possible to represent each set of strong beliefs SB (Definition 6.1) by a formula* ψ *such that* $Cn(SB) = Cn(\psi)$*. Using this lemma, we adapt the definition of a Katsuno and Mendelzon faithful assignment below.*

Katsuno and Mendelzon[KM91] define a faithful assignment from a belief formula to a pre-order over models. Since we are also considering intentions, we extend this definition such that it also maps intentions databases to selection functions.

Definition 6.13 (Faithful assignment)**.** *A* faithful assignment *is a function that assigns to each strong belief formula* $\psi \in \mathbb{SB}^{|t}$ *a total pre-order* $\leq_\psi^t$ *over* $\mathbb{M}$ *and to each intention database* $I \in \mathbb{D}^{|t}$ *a selection function* γ_I^t *and satisfies the following conditions:*

1. *If* $m_1, m_2 \in Mod(\psi)$*, then* $m_1 \leq_\psi^t m_2$ *and* $m_2 \leq_\psi^t m_1$.
2. *If* $m_1 \in Mod(\psi)$ *and* $m_2 \notin Mod(\psi)$*, then* $m_1 < m_2$.
3. *If* $\psi \equiv \phi$*, then* $\leq_\psi^t = \leq_\phi^t$.

4. *If* $T^{|t} = T_2^{|t}$, *then* $(T,\pi) \leq^t_\psi (T_2,\pi_2)$ *and* $(T_2,\pi_2) \leq^t_\psi (T,\pi)$.

Conditions 1 to 3 on the faithful assignment are the same as the conditions that Katsuno and Mendelzon put on a faithful assignment. Condition 4 ensures the two difficulties we pointed out in the beginning of this subsection are handled correctly:

- It ensures we do not distinguish between models in the total pre-order $\leq^t_\psi$ whose trees are the same up to time t. This is essentially what is represented in the revision function by bounding the all input of the revision function $*_t$ up to t.
- Moreover, $\leq^t_\psi$ does not distinguish between models obtained by selecting two different paths from the same tree. In other words, it ensures that msb sets (sets of models of a strong belief) remain in the same ordering. This corresponds to the fact that we are using strong belief formulas in the revision, which do not distinguish between different paths in the same tree as well.

We are now ready to state our main theorem.

Theorem 6.1 (Representation Theorem)**.** *A belief-intention revision operator* $*_t$ *satisfies postulates (P1)-(P12) iff there exists a faithful assignment that maps each* ψ *to a total pre-order* $\leq^t_\psi$ *and each I to a selection function* γ^t_I *such that if* $(\psi,I) *_t (\varphi,i) = (\psi',I')$, *then:*

1. $Mod(\psi') = \min(Mod(\varphi), \leq^t_\psi)$
2. $I' = \gamma^t_I(Mod(\psi'),i)$

We will use the remainder of this subsection to prove the representation theorem above. We first show the number of t-bounded models is finite.

Lemma 6.2. *For each* $t \in \mathbb{N}, \mathbb{M}^{|t}$ *is finite.*

Proof. Suppose some $t \in \mathbb{N}$. Since actions are deterministic and there are finitely many actions in our logic, each state has a finite number of successor states. Moreover, since there are finitely many propositions in our language, the number of possible valuations of the states is finite as well. Therefore, the number of models in $\mathbb{M}^{|t}$ is finite. □

The following lemma obtains a correspondence between semantic consequence of two models equivalent up to t. The proof is by induction on the depth of the formula.

Lemma 6.3. *For each* $\varphi \in \mathscr{L}^{|t}$ *and models* $m_1, m_2' \in \mathbb{M}$, *if* $m_1^{|t} = m_2^{|t}$, *then* $m_1 \models \varphi$ *iff* $m_2 \models \varphi$.

Let $Ext(MSB^{|t})$ be the set of all possible extensions of a t-bounded msb set $MSB^{|t}$ to models, i.e.

$$Ext(MSB^{|t}) = \{m \in \mathbb{M} \mid m^{|t} \in MSB^{|t}\}.$$

We next show that we can represent $MSB^{|t}$ by a single strong belief formula using $Ext(MSB^{|t})$.

Lemma 6.4. *Given a t-bounded msb set* $MSB^{|t}$, *there exists a strong belief formula* $form(MSB^{|t}) \in \mathbb{SB}$ *such that* $Mod(form(MSB^{|t})) = Ext(MSB^{|t})$.

The following corollary shows we can represent a belief database consisting of strong beliefs up to some time t with a single formula.

Corollary 6.1. *Given a t-bounded set of strong beliefs $SB^{|t}$, there exists a strong belief formula $\psi \in \mathbb{SB}$ such that $SB^{|t} = \{\varphi \mid \psi \vdash \varphi\}$.*

6.4 Iterated revision

Until now we only considered single-step revision of beliefs and intentions. In future work, we aim to develop a full account of iterated revision of belief and intention. In this section, we already give a preliminary account of iterated revision for strong beliefs.

In order to model iterated revision of strong beliefs we follow the approach of Darwiche and Pearl [DP97]. They observe that the AGM postulates are too permissive to enforce plausible iterated revision. In order to remedy this, they suggest the following changes:

- Instead of performing revision on a propositional formula, perform revision on an abstract object called an *epistemic state* Ψ, which contains the entire information needed for coherent reasoning.
- Postulate (R4) is weakened as follows:

  (R*4) If $\Psi = \Psi'$ and $\varphi \equiv \varphi'$, then $\Psi \circ_t \varphi \equiv \Psi' \circ_t \varphi'$
- The following four desirable postulates are added for iterated revision:

  (C1) If $\varphi \models \varphi'$, then $(\Psi \circ \varphi') \circ \varphi \equiv \Psi \circ \varphi$.

  (C2) If $\varphi \models \neg\varphi'$, then $(\Psi \circ \varphi') \circ \varphi \equiv \Psi \circ \varphi$.

  (C3) If $\Psi \circ \varphi \models \varphi'$, then $(\Psi \circ \varphi') \circ \varphi \models \varphi'$.

  (C4) If $\Psi \circ \varphi \not\models \neg\varphi'$, then $(\Psi \circ \varphi') \circ \varphi \not\models \neg\varphi'$

We refer to the Darwiche and Pearl postulates defined on epistemic states as (R*1)-(R*6). When switching from belief revision on a belief state to belief revision on an epistemic state the definition of a faithful assignment should be adopted accordingly. We will do this now for our setting. In our setting, epistemic states contain strong belief sets. Note that Ψ stands for $Bel(\Psi)$ whenever it is embedded in a propositional formula, and that $Bel(\Psi)$ is thus a set of strong beliefs.

Definition 6.14 (Faithful assignment on epistemic states)**.** *A t-bounded faithful assignment on epistemic states maps each epistemic state Ψ with $Bel(\Psi)$ from $\mathbb{SB}$, to a total pre-order $\leq^t_\Psi$ on all models such that:*

1. *If $m_1, m_2 \models \Psi$, then $m_1 \leq^t_\Psi m_2$ and $m_2 \leq^t_\Psi m_1$*
2. *If $m_1 \models \Psi$ and $m_1 \not\models \Psi$, then $m_1 <^t_\Psi m_2$*
3. *If $\Psi = \Phi$, then $\leq_\Psi = \leq_\Phi$*
4. *If $T^{|t} = T_2^{|t}$, then $(T,\pi) \leq^t_\Psi (T_2,\pi_2)$ and $(T_2,\pi_2) \leq^t_\Psi (T,\pi)$.*

The next representation theorem is similar to Theorem 6.1, but it uses the Darwiche and Pearl postulates on epistemic states and the t-bounded faithful assignment on epistemic states that we defined above. The proof is a straightforward modification of the proof of Theorem 6.1.

Theorem 6.2 (Representation Theorem)**.** *A t-bounded revision operator $\circ_t$ satisfies postulates (R*1)-(R*6) precisely when there exists a t-bounded faithful assignment on epistemic states that maps each epistemic state Ψ to a total pre-order $\leq^t_\Psi$ such that*

$$Mod(\Psi \circ_t \varphi) = \min(Mod(\varphi), \leq^t_\Psi)$$

Theorem 6.3. *Suppose that a t-bounded revision operator on epistemic states satisfies postulates (R*1)-(R*6). The operator satisfies postulates (C1)-(C4) iff the operator and its corresponding faithful assignment satisfy:*

CR1 If $m_1 \models \varphi$ and $m_2 \models \varphi$, then $m_1 \leq^t_\Psi m_2$ iff $m_1 \leq^t_{\Psi\circ\varphi} m_2$.

CR2 If $m_1 \not\models \varphi$ and $m_2 \not\models \varphi$, then $m_1 \leq^t_\Psi m_2$ iff $m_1 \leq^t_{\Psi\circ\varphi} m_2$.

CR3 If $m_1 \models \varphi$, $m_2 \not\models \varphi$ and $m_1 <^t_\Psi m_2$, then $m_1 <^t_{\Psi\circ\varphi} m_2$.

CR4 If $m_1 \models \varphi$, $m_2 \not\models \varphi$ and $m_1 \leq^t_\Psi m_2$, then $m_1 \leq^t_{\Psi\circ\varphi} m_2$.

We provide a proof sketch here, and the full proof can be found in Appendix D.

Proof Sketch. ($\Leftarrow$): Identical to Darwiche and Pearl.
($\Rightarrow$): We prove $(C1) \Rightarrow (CR1)$, the proofs for the other postulates are similar. Suppose that (C1) holds and $m, m' \models \varphi$. Let $\alpha = form(M^{|t}, M^{|t\prime})$. By Lemma Lemma **??**, $Mod(\alpha) = Ext(M^{|t}) \cup Ext(M^{|t\prime})$. If $m'' \in Ext(M^{|t})$, then $M^{|t} = M^{|t\prime\prime}$, so by Lemma Lemma 6.3 we obtain $m'' \models \varphi$. Similarly, if $m'' \in Ext(M^{|t\prime})$, then $m'' \models \varphi$, so $\alpha \models \varphi$. By (C1), $(\Psi \circ_t \varphi) \circ_t \alpha \equiv \Psi \circ_t \alpha$. In other words, $\min(Mod(\alpha), \leq^t_{\Psi\circ_t\varphi}) = \min(Mod(\alpha), \leq^t_\Psi)$. Consequently, $m \leq^t_{\Psi\circ_t\varphi} m'$ iff $m \leq^t_\Psi m'$. □

6.5 Discussion

6.5.1 Related work

Philosophy of mind

Before the early 1990s most work on intention was done by philosophers. The dominant view prior to the 1980s was that fundamental mental states included belief-like attitudes such as belief and knowledge on the one hand, and desire-like attitudes such as desires and preferences on the other [Dav63]. These capture the "two directions of fit" between world and mind [Sea83]. It was generally believed that intentions could be reduced to these more basic mental states. For instance, an intention can be thought of as a complex kind of belief: an intention to φ at some time t might be a belief that the agent will φ at t, perhaps by virtue of this very belief [Hol08]. This *belief-desire* model of the mind was also common in decision theory in the spirit of Savage [Sav72] (see also Chapter

2, Section 2.5.2), as well as in artificial intelligence. The basic idea underlying all this is that rational action amounts to what leads to the most *desirable* outcome, given the agent's *beliefs* about the world and *preferences* about how it ought to be.

In the mid 1980s Michael Bratman wrote the book "Intentions, plans, and practical reason" [Bra87], which would turn out to be very influential in both philosophy and in artificial intelligence. In the book, he argued convincingly that intentions cannot be reduced to any "more basic" mental states, but that intentions must be taken as basic themselves. His argument is complex, and difficult to understand for non-philosophers, but the fundamental ideas are not, and are in fact very simple. Intentions are subject to their own set of norms, which cannot be reduced to those for beliefs or desires. For instance, some norms of intention consistency definitely do not apply to desires: if someone knows that φ and ψ are mutually inconsistent, it is irrational to intend φ and to ψ simultaneously, but it would not be irrational to desire both φ and ψ.[4]

Another example of a norm of intentions, and one which has been very influential in computer science, is the idea that intentions constrain further practical reasoning in a characteristic way. Bratman (and others) have argued that if someone intends to φ, that gives her a *pro tanto* reason to φ, even in the face of evidence that she ought not φ. In other words, intentions have some amount of *inertia* that beliefs and desires do not have. An agent is more deeply *committed* to its intentions that to its beliefs. If we think of intentions as making up a plan, then reconsidering an intention potentially requires revising an entire plan, which can be computationally expensive and problematic in real time. A consequence of this *relative resistance to revision* is that intentions can be very useful for resource-bounded agents who have to select appropriate actions at different times. Both by adopting appropriate *policies* that can apply at different times to similar decision problems, and by devising complex plans which can easily be followed when computation time is limited, an agent with intentions is able to avoid performing complex computations every time a new decision problem arises. Observations such as those above are common knowledge in artificial intelligence by now, but it is important to realize there is a well explored philosophical foundation underlying them.

Artificial Intelligence

In the 1990s there was a crucial moment of cross-pollination between philosophy of mind and artificial intelligence. Inspired by the philosophical literature, research in artificial intelligence began to explore intentions. One of the first attempts of what was later known as the *belief-desire-intention architecture* was in a paper by Bratman, Israel, and Pollack [BIP88], which systematized many aspects of the view laid out in Bratman's book [Bra87], including flowcharts and some suggestions for implementations. Most importantly, it served as a call for other researcher to investigate these matters more systematically and formally.

One of the most well-known and first attempts of a formalization of intention revision was by Cohen and Levesque [CL90a]. They developed a formal logical language, including modal operators for mental state such as "belief" and "having a goal", for temporal expressions, and for descriptions of events such as "A happens" or "x did action

[4] Some recent philosophers in fact argue that this norm on intentions can be derived from a norm on consistent of beliefs, but this remains a point of debate (see for instance Harman [Har76]).

a". The main point of the paper is to define a useful notion of intention. To this end they first define a P-GOAL, or *persistent goal*, which is a goal p that one beliefs now not to hold, and which will cease to be a goal as soon as either the agent believes p will never hold, or the agent comes to believe p is true. Intending to take an action a is then defined in terms of a P-GOAL for agent x [CL90a, p.245]:

$$\text{INTEND}(x,a) := (\text{P-GOAL}x[\text{DONE}x(\text{BEL}x(\text{HAPPENS}a))?;a]).$$

Unpacking this, agent x intends to a just in case x has a persistent goal to ensure that x will believe a will be carried out, up until a is in fact carried out. They also define "intending to bring about some state of affairs", and show their approaches solves the "Little Nell" problem and avoid the "Dentist Problem" [Bra87].

While Cohen and Levesque's approach is much cited, it is fair to say that it is rather complicated. Some early criticisms of technical details can be found in [Sin92b]. In Shoham and Leyton-Brown's textbook the approach is called "the road to hell" [SLB08]. Due to the complexity of the logic, mathematical properties such as axiomatizability, decidability and complexity of fragments were never investigated. None of the BDI logics that were introduced subsequently adapted Cohen and Levesque's four steps definition of intention and instead considered intentions to be primitive. Moreover, while Cohen and Levesque's provide some criteria for the abandonment of intentions through the notion of rational balance (forbidding to intend something that is true or believed to be impossible to achieve), it does not further analyze the 'other reasons' for which a persistent goal is abandoned. More details on these critiques can be found in a recent articles by Herzig *et al.* [HLPX16].

Some of Cohen and Levesque's shortcomings were corrected to an extent in subsequent work. For instance, Rao and Georgeff's approach [RG91] offers an alternative, and arguably simpler, formalism based on CTL. Shoham [Sho09] argues that one of the most fundamental flaws in all such approaches is not being sufficiently grounded in a computational setting, and as a result living in the no-man-land between philosophy and computer science. Shoham further developed these ideas with Jacob Banks, one of his PhD students, and behavioral economist Dan Ariely in the intelligent calendar application Timeful, which attracted over $6.8 million in funding and was acquired by Google in 2015[5], who aim to integrate it into their Calendar applications. As Shoham [Sho16] says himself: "The point of the story is there is a direct link between the original journal paper and the ultimate success of the company." (p.47) Thus, it seems clear that his philosophical proposal has lead to some success on the practical side. In this paper, we investigate whether his proposal can lead to interesting theoretical insights as well.

There exists previous work on intention dynamics, that is, how intentions change over time. van der Hoek, Jamroga, and Wooldridge [vdHJW07] (see also [Woo00] and [WP98]) explore a system similar to Rao and Georgeff's with representations of beliefs, desires, and intentions separately, including a specific module for practical reasoning; they propose how to revise intentions together with beliefs. Grant *et al.* [GKPW10] continue this line of work, offering postulates for intention revision (similar to our contribution). Lorini and Herzig [LH08] introduce a logic of intention with modal operators for attempt, tacking the question of when an agent's intentions translate into an actual

[5] http://venturebeat.com/2015/05/04/google-acquires-scheduling-app-timeful-and-plans-to-integrate-it-into-google-apps/

attempted action. Finally, Shapiro *et al.* [SSTC12] also explore intention revision, working in a setting with complex hierarchical plans.

6.5.2 Open issues

There are a large number of directions for future research. Shoham [Sho09] give some pointers as well, some of which we have conducted some preliminary research already. We next discuss various directions and we explain what we have done for it so far.

Going beyond atomic intentions

In our approach, we model revision of atomic intentions, putting more structured intentions to the side. However, recently Herzig *et al.* [HLPX16] argued convincingly that an important element of intentions, namely the refinement of intentions, has not received much attention in the literature. Subsequently Herzig *et al.* [HPXZ16] develop a formal framework for the refinement of intentions, which is based on Shoham's database perspective as well. It thus seems that our approach is a good fit with their, and it would be interesting to see whether we could obtain a similar result for revision in their framework as we did in ours.

Developing a quantitative ("probabilistic") theory of intention

Most of the work described above, including our own contributions, is anchored in logic. Indeed, intention like some of its cousins, belief, desire, knowledge, etc. naturally lends itself to logical treatment. However, at least in the case of belief, logical models have long been superseded in popularity by quantitative models, and in particular Bayesian methods for reasoning about partial belief are currently dominant in both philosophy and computer science (not to mention psychology, economics, and other social sciences).

We have conducted some preliminary research in this direction [vZI15], which we briefly summarize here. Our work build on earlier work in the BDI literature by Kinny and Georgeff [SWP04]. We present a theoretical framework for the intention reconsideration problem in Markov Decision Processes, in the same spirit as much work on metareasoning. This involves the construction of a meta-level Markov Decision Process in which the two actions are 'think' or 'act'.

Theoretical Framework We formalize intention reconsideration as a metareasoning problem. At each time step, the agent faces a choice between two meta-level actions: *acting* (i.e., executing the optimal action for the current decision problem, based on the current plan) or *deliberating* (i.e., recomputing a new plan). We assume that the agent's environment is inherently dynamic, potentially changing at each time step. As a result, some plan that may be optimal at a certain time may no longer be optimal, or worse, may not be executable at a later time moment. We formalize the sequential decision problem as an MDP (S,A,T,R), where S is a set of states, A is a set of actions, $T : S \times A \times S \to [0,1]$ is a transition function, and $R : S \times A \times S \to \mathbb{R}$ is a reward function. An agent's view on the world is captured by a *scenario* $\sigma = (S,A,T,R,\lambda)$, where (S,A,T,R) is an MDP, and $\lambda \in S$ is the agent's location in the MDP. At any given time

the agent also maintains a policy, or plan, $\pi : S' \to A'$ for some set of states S' and set of actions A', which may or may not equal S and A. Thus, the domain and range of the agent's policy may not even coincide with the current set of states and actions.

We also assume an agent might have a memory store μ, which in the most general case simply consists of all previous scenario/plan pairs: $\mu = \langle\langle\sigma_1,\pi_1\rangle,\ldots,\langle\sigma_{n-1},\pi_{n-1}\rangle\rangle$. (We will typically be interested in agents with significantly less memory capacity.) Summarizing, an agent's overall state (σ,π,μ) consists of a scenario σ, a plan π, and a memory μ.

Meta-Level Actions: Think or Act If the environment were static, then there would be no reason to revise a perfectly good plan.[6] However, environments are of course rarely static. States may become unreachable, new states may appear, and both utilities and probabilities may change. This raises the question of plan reconsideration. We assume that at each time moment, an agent has a choice between two meta-level actions, namely whether to act or to think (deliberate). When the agent decides to act, it will attempt the optimal action according to the current plan. When the agent decides to think, it will recompute a new plan based on the current MDP. The cost of deliberation can either be charged directly, or can be captured indirectly by opportunity cost (missing out on potentially rewarding actions).

The Dynamics of the Environment An environment specifies how a state $s = \langle\sigma,\pi,\mu\rangle$, and a choice of meta-decision $\alpha \in \{\text{think},\text{act}\}$, determine (in general stochastically, according to P_d and P_a) a new state $s' = \langle\sigma',\pi',\mu'\rangle$:

$$\langle\sigma,\pi,\mu\rangle \xrightarrow{\alpha} \langle\sigma',\pi',\mu'\rangle$$

- $\mu' = \langle\langle\sigma_1,\pi_1\rangle,\ldots,\langle\sigma_{n-1},\pi_{n-1}\rangle,\langle\sigma,\pi\rangle\rangle$;
- if $\alpha = \text{think}$:
  - σ' is some perturbation of σ: $\sigma' \sim P_d(\cdot \mid \sigma)$.
  - π' is a new policy for σ.
- if $\alpha = \text{act}$:
  - σ' is a noisy result of taking action $a = \pi(\lambda)$: $\sigma' \sim P_a(\cdot \mid \sigma)$.
  - $\pi' = \pi$.

Let $\mathcal{S}$ be the set of all possible environment states, which are the scenarios that we introduced in the first subsection, and let $\mathcal{A}$ be the set of all possible actions. Let us assume we have specified concrete perturbation functions P_d and P_a for $a \in \mathcal{A}$. We can lift these to a general transition function $\mathcal{T} : \mathcal{S} \times \{\text{think},\text{act}\} \times \mathcal{S} \to [0,1]$, so that

$$\mathcal{T}(s,\alpha,s') = \begin{cases} P_d(\sigma' \mid \sigma) & \text{if } \alpha = \text{think and } \pi' \text{ is the revised plan for } \sigma \\ P_{\pi(\lambda)}(\sigma' \mid \sigma) & \text{if } \alpha = \text{act and } \pi' = \pi \\ 0 & \text{otherwise} \end{cases}$$

[6] Of course, there still might be a question of whether further thought might lead to a better plan in case the current plan was itself selected heuristically or sub-optimally.

We can also lift the reward functions R over S to reward functions $\mathcal{R}$ over $\mathcal{S}$:

$$\mathcal{R}(s,\alpha,s') = \begin{cases} R(\lambda,a,\lambda') & \text{if } \alpha = \mathsf{act} \\ 0 & \text{if } \alpha = \mathsf{think}, \end{cases}$$

where λ' is the agent's location in scenario σ'. This defines a new meta-level MDP as follows:

$$\langle \mathcal{S}, \{\mathsf{think}, \mathsf{act}\}, \mathcal{T}, \mathcal{R} \rangle$$

Thus, once the set $\mathcal{S}$ and the function $\mathcal{T}$ are specified, we have a well defined MDP, whose space of policies can be investigated just like any other MDP.

Experiments Computing an optimal policy for the meta-level MDP is difficult in general. We have implemented the general framework from the previous section in Java. The source code is available on the Github page of this thesis:

https://github.com/marcvanzee/RationalArchitecture

In the folder "Ch6 mdp intention revision". An example MDP visualization is depicted in Figure E.1 of Appendix E.

Kinny and Georgeff present the Tileworld as a 2-dimensional grid on which the time between two subsequent hole appearances is characterized by a gestation period g, and holes have a life-expectancy l, both taken from a uniform distribution. Planning cost p is operationalized as a time delay. The ratio of clock rates between the agent's action capabilities and changes in the environment is set by a *rate of world change* parameter γ. This parameter determines the *dynamism* of the world. When an agent plans, it selects the plan that maximizes hole score divided by distance (an approximation to computing an optimal policy in this setting). The performance of an agent is characterized by its *effectiveness* ε, which is its score divided by the maximum possible score it could have achieved. The setup is easily seen as a specific case of our meta-decision problem (see Fig. E.2).

Kinny and Georgeff propose two families of intention reconsideration strategies: bold agents, who inflexibly replan after a fixed number of steps, and reactive agents, who respond to specific events in the environment. For us, a *bold* agent only reconsiders its intentions when it has reached the target hole; and a *reactive* agent is a bold agent that also replans when a hole closer than its current target appears, or when its target disappears.

In addition, we consider an *angelic* agent, who approximates the value of computation calculations that would allow always selecting think or act in an optimal way. It does so by recursively running a large number of simulations for the meta-level actions from a given state, approximating the expected value of both, and choosing the better. Because we are interested in the theoretically best policy, the angelic agent is not charged for any of this computation: time stops, and the agent can spend as much time as it needs to determine the best meta-level action (hence the term 'angelic').

Results Graphs of the results can be found in Appendix E. In Figure E.3 we compare the bold agent with the angelic planner with the same parameter settings as Kinny and Georgeff and a planning time of 2. Unsurprisingly, the angelic planner outperforms the bold agent. In Figure E.4, we increase the planning time to 4, which increases the difference in performance between the angelic planner and the bold agent, while the reactive planner does equally well. However, in Figure E.5, we see that when we change the parameters settings such that the world is significantly smaller and holes appear as quickly as they come, the angelic planner outperforms the reactive agent as well. Finally, in Figure E.6 we consider a highly dynamic domain in which holes appear and disappear very fast. Here the bold agent outperforms the reactive strategy, and does nearly as well as the angelic agent. In such an environment, agents that replan too often never have a chance to make it toward their goals.

Intriguingly, even these very simple agents—bold agents and rudimentary reactive agents—come very close to ideal in certain environments. This suggests that if we fix a given environment, near-optimal intention/plan reconsideration can actually be done quite tractably. However, since these optimal meta-level strategies differ from environment to environment, this seems to be a natural setting in which meta-meta-level reasoning can be useful. One would like a method for determining which of a family of meta-level strategies one ought to use, given some (statistical or other) information about the current environment, its dynamics and the relative (opportunity) cost of planning.

While these results concern a rather specific case of the intention revision problem—in the Tileworld, which is not necessarily representative of other domains—the general framework concerns any sequential decision problem in a dynamic environment. Thus, in addition to exploring the possibility of meta-meta-level strategies for this particular domain, we are also currently exploring other settings, e.g., where states themselves may appear and disappear and probabilities may change. We would like as comprehensive an understanding of the general relation between these rational meta-level strategies and environmental parameters as possible, and we believe the results here mark a good first step.

Multi-agent extension

The importance of "joint intentions" has been recognized for some time. There is still much disagreement in the philosophical literature about whether joint intentions can be *reduced* to more basic intentions of the member of the group, or whether the joint intentions are perhaps irreducible. Again, focusing on a database may allow us to sidestep these debates and center our study on what promises to be most useful. Clearly, the intentions of an agent may depend on intentions of another agent, so it is important that their intentions are properly *coordinated*, which is an important reason why much of this literature has focused on *communication*.

In order to tackle this problem, we have done some preliminary research in which we propose to extend the database perspective with a single auxiliary database in the multi-agent system that we coin the "Collective Intention Database". The collective intention database contains a set of collective intentions. A collective intention is formalized as a set of action-agent pairs and a time point. In this way, the collective intention database captures two basic notions. Firstly, it serves as a coordination mechanism for a set of agents by specifying what actions of the agents are carried out simultaneously,

and in this way represents dependencies between the intentions of the individual agents. Secondly, it allows agents to reason about each others intentions, in the sense that each agent believes that the other agents will carry out their part in a collective intention. We extend our definition of coherence in a natural way: An agent's beliefs cohere with his intentions and the collective intentions if 1) the agent considers it possible to carry out all of his intended actions (this is equivalent to our definition), and 2) the agent considers it possible that all intentions by other agents in collective intentions in which the agent itself also participates are carried out by the corresponding agents. A visualization of this view is shown in Figure 6.4.

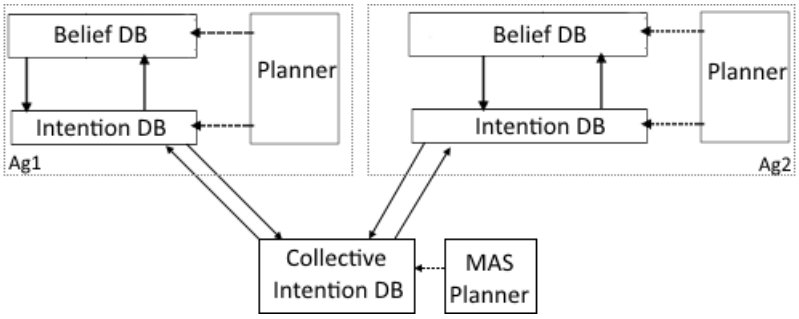


Figure 6.4: The Multi-agent database perspective

6.5.3 Conclusion

We develop a logical theory for reasoning about temporal beliefs and intentions based on Shoham's database perspective. We propose postulates for revision of strong beliefs and intentions, and prove a representation theorem relating the postulates to our formal model.

Our approach, contrary to much of the work emanating from Cohen and Levesque, has been to study the *temporal* dimension of commitments, as opposed to much existing research which often focuses on the *teleological* dimension of commitments. As a result, our theory is conceptually considerably simpler than other formalism attempting to incorporate a large amount of mental attitudes. However, as we have seen, even in this simple setting already nontrivial complications arise, and interesting technical results can be obtained.

We have focused on the temporal dimension by using Shoham's database perspective, which means that we separate the planning capabilities of an agent from the storing of commitments. Beliefs and commitments are stored in a database, while the planner operates independently of these databases. By focusing on the temporal dimension of commitments and separating planning from the storing of commitments, we make the link between artificial intelligence planning and philosophical theories of intention more explicit.

7

Conclusion

The overall aim of this thesis is to study enterprise architect reasoning, in order to support enterprise architects in their daily activities. In Part 1, we focus on understanding the characteristics of enterprise architecture better. We first interview enterprise architects in order to gain a general understanding about the domain, and we obtain a list of characteristics (Chapter 2). Then we formalize an existing framework for enterprise architecture decision rationalization and notice various shortcomings (Chapter 3). In Part 2, we focus on goals. We develop the RationalGRL logical framework, which combines an approach for goal modeling with formal argumentation (Chapter 4). In Part 3, we focus on planning and scheduling. We develop the BDI logic Parameterized-time Action Logic for beliefs and commitments, axiomatize it and proof it is sound and strongly complete (Chapter 5). We characterize the dynamics of beliefs and commitments and prove a variation of the Katsuno and Mendelzon, as well as the Darwiche and Pearl representation theorems (Chapter 6).

The introductory chapter has put forward a number of research questions aimed at understanding enterprise architect reasoning about high-level decisions. Those questions have been answered in the following way:

RQ 1. Which aspects of enterprise architect reasoning about high-level decisions can be supported by decision support systems?

We provide a list of eight characteristics in Chapter 2, which we derive from interviews with enterprise architects. These are the following:

1. Translating strategic goals into an IT strategy
2. Communicating plans of action
3. Explaining decisions instead of making them
4. Qualitative before quantitative data
5. Stronger business focus than other disciplines
6. Politics, emotions, and soft skills play a bigger role than in other disciplines
7. Large number of stakeholders with conflicting views
8. Highly uncertain plans in a changing environment

Of these characteristics, we focus primarily on goals (characteristic 1, Part 2), and planning (characteristic 2 and 8, Part 3). These characteristics indicate that approaches based

on *resource bounded* reasoning may be appropriate, as well as *qualitative* theories as opposed to those that are strictly *quantitative*.

RQ 2 Which aspects of enterprise architecture reasoning and goal modeling can be supported by decision support systems?

Reasoning about the refinement of goals is an important part of enterprise architecture, and various specific modeling languages have been developed to address this need. However, although these languages have been developed to support communication between the enterprise architect and stakeholders, many of the elements of the discussions cannot be captured clearly. Artificial intelligence research has a solid foundation of research on argumentation. Applying such theoretical results to a practical field is challenging and runs into the risk of becoming too complicated to be of use to practitioners. We show that goal modeling and argumentation are a very good fit, in particular when using argument schemes and critical questions.

RQ 3. Which aspects of enterprise architecture planning will be considered in this thesis?

We develop a belief-desire-intention logics in order to formalize a specific part of enterprise architecture dynamics, namely the storing of future commitments with explicit time points. In this way, we separate the storing of commitments from planning, and our system can in this sense be seen as an *intelligent calendar*. This calendar can be used to store commitments (which can be tasks, appointments, or anything else) and beliefs about these commitments. The intelligence of the calendar comes from the fact that it can reason about the consistency of commitments and beliefs.

Returning to the main theme of this thesis, we should ask ourselves: what does our resulting logical theory contribute to enterprise architecture? Let us briefly zoom out and reconsider our methodology. The last part of this thesis was concerned with the last "brick" of the bridge from enterprise architecture to artificial intelligence. Our journey has led us from interviews with enterprise architects to a technical and abstract logical framework for reasoning about the dynamics of abstract notions such as "commitments" and "beliefs". How can such a system be of use to enterprise architects? One simple answer may be as follows: The resulting logic should now be extended with more characteristics from enterprise architecture. To continue our bridge metaphor: the next part of this journey goes into the other direction. We start with our logical theory for the dynamics of commitments, and while we walk back over the bridge, we add characteristics and techniques from our various contributions. For the second brick, we may add the possibility to refine intentions into more specific ones, and a multi-agent perspective to argue about intentions. For the first brick, we may add very domain-specific elements we found in the EA Anamnesis framework, such as enterprise architecture layers, types of relationship between various commitments, and so on.

These are all very exciting possibilities, and I hope my thesis has inspired my successors to such an extent that they consider working on them.

A

UCI Design Workshop Prompt

Design Prompt: Traffic Signal Simulator

Problem Description

For the next two hours, you will be tasked with designing a traffic flow simulation program. Your client for this project is Professor E, who teaches civil engineering at UCI. One of the courses she teaches has a section on traffic signal timing, and according to her, this is a particularly challenging subject for her students. In short, traffic signal timing involves determining the amount of time that each of an inter- section's traffic lights spend being green, yellow, and red, in order to allow cars in to flow through the intersection from each direction in a fluid manner. In the ideal case, the amount of time that people spend waiting is minimized by the chosen settings for a given intersection's traffic lights. This can be a very subtle matter: changing the timing at a single intersection by a couple of seconds can have far- reaching effects on the traffic in the surrounding areas. There is a great deal of theory on this subject, but Professor E. has found that her students find the topic quite abstract. She wants to provide them with some software that they can use to "play" with different traffic signal timing schemes, in different scenarios. She anticipates that this will allow her students to learn from practice, by seeing first-hand some of the patterns that govern the subject.

Requirements

The following broad requirements should be followed when designing this system:

1. Students must be able to create a visual map of an area, laying out roads in a pattern of their choosing. The resulting map need not be complex, but should allow for roads of varying length to be placed, and different arrangements of intersections to be created. Your approach should readily accommodate at least six intersections, if not more.

2. Students must be able to describe the behavior of the traffic lights at each of the intersections. It is up to you to determine what the exact interaction will be, but a variety of sequences and timing schemes should be allowed. Your approach should also be able to accommodate left-hand turns protected by left-hand green arrow lights. In addition:

   (a) Combinations of individual signals that would result in crashes should not be allowed.

(b) Every intersection on the map must have traffic lights (there are not any stop signs, over- passes, or other variations). All intersections will be 4-way: there are no "T" intersections, nor one-way roads.

(c) Students must be able to design each intersection with or without the option to have sensors that detect whether any cars are present in a given lane. The intersection's lights' behavior should be able to change based on the input from these sensors, though the exact behavior of this feature is up to you.

3. Based on the map created, and the intersection timing schemes, the students must be able to simulate traffic flows on the map. The traffic levels should be conveyed visually to the user in a real-time manner, as they emerge in the simulation. The current state of the intersections' traffic lights should also be depicted visually, and updated when they change. It is up to you how to present this information to the students using your program. For example, you may choose to depict individual cars, or to use a more abstract representation.

4. Students should be able to change the traffic density that enters the map on a given road. For ex- ample, it should be possible to create a busy road, or a seldom used one, and any variation in between. How exactly this is declared by the user and depicted by the system is up to you. Broadly, the tool should be easy to use, and should encourage students to explore multiple alternative approaches. Stu- dents should be able to observe any problems with their map's timing scheme, alter it, and see the results of their changes on the traffic patterns. This program is not meant to be an exact, scientific simulation, but aims to simply illustrate the basic effect that traffic signal timing has on traffic. If you wish, you may assume that you will be able to reuse an existing software package that provides relevant mathematical functionality such as statistical distributions, random number generators, and queuing theory.

You may add additional features and details to the simulation, if you think that they would support these goals.

Your design will primarily be evaluated based on its elegance and clarity both in its overall solution and envisioned implementation structure.

Desired Outcomes

Your work on this design should focus on two main issues:

1. You must design the interaction that the students will have with the system. You should design the basic appearance of the program, as well as the means by which the user creates a map, sets traffic timing schemes, and views traffic simulations.

2. You must design the basic structure of the code that will be used to implement this system. You should focus on the important design decisions that form the foundation of the implementation, and work those out to the depth you believe is needed.

The result of this session should be: *the ability to present your design to a team of software developers who will be tasked with actually implementing it.* The level of competency you can expect is that of students who just completed a basic computer science or software engineering undergraduate degree. You do not need to create a complete, final diagram to be handed off to an implementation team. But you should have an understanding that is sufficient to explain how to implement the system to competent developers, without requiring them to make many high-level design decisions on their own.

To simulate this hand-off, you will be asked to briefly explain the above two aspects of your design after the design session is over.

Timeline

- 1 hour and 50 minutes: Design session
- 10 minutes: Break / collect thoughts
- 10 minutes: Explanation of your design
- 10 minutes: Exit questionnaire

B

Transcripts Excerpts

Respondent	Text	Annotation
0:17:39.5 (P1)	And in that process there are activities like create a visual map, create a road	**[14 task (AS2)]** Student has task "Create road"
0:24:36.0 (P3)	And, well interaction. Visualization sorry. Or interaction, I don't know. So create a visual map would have laying out roads and a pattern of their choosing. So this would be first, would be choose a pattern.	**[31 critical question CQ?? for 14]** Is Task "Create road" clear? **[32 answer to 31]** no, according to the specification the student should choose a pattern.
0:24:55.4 (P1)	How do you mean, choose a pattern	**[32a REPLACE]** "Create road" becomes "Choose a pattern"
0:24:57.5 (P3)	Students must be able to create a visual map of an area, laying out roads in a pattern of their choosing	
0:25:07.5 (P1)	Yeah I'm not sure if they mean that. I don't know what they mean by pattern in this case. I thought you could just pick roads, varying sizes and like, broads of roads.	**[33 critical question CQ?? for 32a]** Is "Choose a pattern" clear? **[34 answer to 33]** No, not sure what they mean by a pattern.
0:25:26.0 (P3)	No yeah exactly, but you would have them provide, it's a pattern, it's a different type of road but essentially you would select- how would you call them, selecting a-	**[34a REPLACE]** "Choose a pattern" becomes "Choose a pattern preference"
0:25:36.3 (P1)	Yeah, selecting a- I don't know	
0:25:38.0 (P3)	Pattern preference maybe? As in, maybe we can explain this in the documentation	
0:25:43.9 (P1)	What kind of patterns though. Would you be able to select	**[35 critical question CQ?? for 34a]** Is "Choose a pattern preference" clear? **[36 answer to 35]** no, what kind of pattern?
0:25:47.4 (P3)	Maybe, I don't know it's-	**[36a rename]** "Choose a pattern preference" becomes "Choose a road pattern"
0:25:48.5 (P1)	[inaudible] a road pattern	

Table B.1: Clarifying the name of a task (transcript t_3)

Respondent	**Text**	**Annotation**
0:15:11.2 (P1)	And then, we have a set of actions. Save map, open map, add and remove intersection, roads	**[20 task (AS2)]** Student has tasks "save map", "open map", "add intersection", "remove intersection", "add road", "add traffic light"
0:15:34.7 (P2)	Yeah, road. Intersection, add traffic lights	
0:15:42.3 (P1)	Well, all intersection should have traffic lights so it's	**[21 critical question CQ12 for 20]** Is the task "Add traffic light" useful/relevant? **[22 answer to 22]** Not useful, because according to the specification all intersections have traffic lights.
0:15:44.9 (P2)	Yeah	
0:15:45.2 (P1)	It's, you don't have to specifically add a traffic light because if you have	
0:15:51.4 (P2)	They need-	

Table B.2: Adding tasks, disabling useless task "Add traffic light" (transcript t_1)

Respondent	Text	Annotation
0:18:55.7 (P1)	Yeah. And then two processes, static, dynamic and they belong to the goal simulate.	**[17 goal (AS3)]** Actor "System" has goal "Simulate" **[18 task (AS2)]** Actor "System" has task "Static simulation" **[19 task (AS2)]** Actor "System has task "Dynamic simulation" **[20 decomposition (AS?)]** Goal "Simulation" AND-decomposes into "Static simulation" and "Dynamic simulation"
0:30:10.3 (P1)	Yeah. But this is- is this an OR or an AND?	**[26 critical question CQ10b for 20]** Is the decomposition type of "simulate" correct? **[27 answer to 26]** No, it should be an OR decomposition.
0:30:12.6 (P2)	That's and OR	
0:30:14.3 (P3)	I think it's an OR	
0:30:15.4 (P1)	It's for the data, it's an OR	
0:30:18.1 (P3)	Yep	

Table B.3: Incorrect decomposition type for goal *Simulate* (transcript t_3)

Respondent	Text	Annotation
0:10:55.2 (P1)	Maybe developers	**[4 actor (AS0)]** Development team
0:11:00.8 (P2)	Development team, I don't know. Because that's- in this context it looks like she's gonna make the software	**[5 critical question CQ0 for 4]** Is actor "development team" relevant? **[6 answer to 5]** No, it looks like the professor will develop the software.
0:18:13.4 (P2)	I think we can still do developers here. To the system	**[16 counter argument for 6]** According to the specification the professor doesn't actually develop the software.
0:18:18.2 (P1)	Yeah?	
0:18:19.8 (P2)	Yeah, it isn't mentioned but, the professor does-	
0:18:22.9 (P1)	Yeah, when the system gets stuck they also have to be [inaudible] ok. So development team	

Table B.4: Discussion about the relevance of an actor (transcript t_3)

C

GRL Specification

This is the specification of the GRL model in Figure 4.2 using the atomic language developed in Chapter 4, Section 4.5.

```
%% GRL Elements
actors([1,24,43]).
ies([2...17, 25...34, 44, 45]).
links([18...23, 35...42, 47, 48, 49]).

%%%%% Actor student %%%%%%
name(1, student).

% IE types of actor student
softgoal(2).
goal(3).
tasks(4). task(5). ... task(17).

% Containments of actor student
has(1, 2). has(1, 3). ... has(1, 17).

% IE names of actor student
name(2, learn_queuing_theory_from_practice).
name(3, use_simulator).
name(4, map_design).
name(5, open_map).
name(6, add_road).
name(7, add_sensor_to_traffic_light).
name(8, control_grid_size).
name(9, add_intersection).
name(10, run_simulation).
name(11, save_map).
name(12, control_simulation).
name(13, control_car_influx_per_road).
name(14, adjust_car_spawing_rate).
name(15, adjust_timing_schemes_of \
            sensorless intersections).
name(16, remove_intersection).
name(17, add_traffic_light).

% Links of actor student
contr(18, 3, 2, pos).
contr(19, 4, 3, pos).
contr(20, 11, 3, pos).
decomp(21, 4, [5...11], and).
decomp(22, 4, [5...11, 13], and).
decomp(23, 12, [13,14,15], and).

% Disabled elements of actor student
disabled(16). disabled(17). disabled(22).

%%%% Actor Traffic Tycoon %%%%%%%/
name(24, traffic_tycoon).

% IE types of actor Traffic Tycoon
softgoal(25). softgoal(26).
goal(27). goal(28).
task(29). ... task(32).
resource(33). resource(34).

% Containments of actor Traffic Tycoon
has(24, 25). ... has(24,34).

% IE names of actor Traffic Tycoon
name(25, dynamic_simulation).
name(26, realistic_simulation).
name(27, show_simulation).
name(28, generate_cars).
name(29, keep same_cars).
name(30, create_new_cars).
name(31, show_map_editor).
name(32, store_load_map).
name(33, external_library).
name(34, storage).

% Links of actor Traffic Tycoon
contr(35, 29, 25, neg).
contr(36, 29, 26, pos).
contr(37, 30, 25, pos).
contr(38, 30, 26, neg).
decomp(39, 28, [29, 30], xor).
decomp(40, 27, [28, 33], and).
decomp(41, 31, [32], and).
decomp(42, 32, [34], and).

%%%% Actor Teacher  %%%%%%%
name(43, teacher).

% IE types of actor Teacher
softgoal(44).
task(45).

% Containments of actor Teacher
has(43, 44). has(43,45).

% IE names of actor Teacher
name(44, students_learn_from_practice).
name(45, pass_students_if_simulation_is_correct).

% Disabled elements of actor Teacher
disabled(45).

%%%%%% Dependencies  %%%%%
goal(46).
name(46, value_has_changed).

dep(47, 32, 46).
dep(48, 46, 14).
dep(49, 32, 11).

%%%%%% Rules for containment  %%%%%
has(Act,E1) :- has(Act, E2), decomposes(_,E2,X,_),
        member(E1,X).
has(Act,E1) :- has(Act,E1), contr(E2, E1,_).
```

D

Proofs

D.1 Completeness proofs

Theorem D.1 (Completeness Theorem)**.** *The logic PAL is sound and strongly complete, i.e.* $\Sigma \vdash \varphi$ *iff* $\Sigma \models \varphi$.

$T \vdash \varphi \Rightarrow T \models \varphi$ can be proven by standard techniques.

Strong completeness: $T \models \varphi \;\; \Rightarrow \;\; T \vdash \varphi$.

We prove strong completeness by constructing a canonical model, given a consistent set of formulas, but before this we introduce some concepts that we will need in different parts of the proof. These concepts will be largely familiar to most readers.

Definition D.1 (Maximally consistent set (mcs))**.** *Given the logic PAL, a set of formulas T is PAL-consistent if one cannot derive a contradiction from it, i.e. if $\bot$ cannot be inferred from it, in the proof system for PAL. A set of formulas T^* is a* maximally PAL-consistent set *(mcs) if it is PAL-consistent and for every formula φ, either φ belong to the set or $\neg\varphi$ does.*

We denote the part of a mcs up to and include time t with T^*_t, formally: $T^*_t = T^* \cap Past(t)$.

The proofs of the following two results, Deduction theorem and Lindenbaum's lemma, are standard.

Lemma D.1 (The Deduction Theorem)**.** $\Sigma \cup \{\varphi\} \vdash \psi \;\; \Leftrightarrow \;\; \Sigma \vdash \varphi \rightarrow \psi$.

Lemma D.2 (Lindenbaum's lemma)**.** *Every consistent set of formulas can be extended to a maximal consistent set of formulas.*

Now we define an equivalence relation on maximally consistent sets of formulas.

Definition D.2 (Mcs Equivalence Relation)**.** *Suppose some $t \in \mathbb{N}$ and two mcs's T^* and $\overline{T}^*$, we define the* equivalence relation *between T^* and $\overline{T}^*$, denoted by $T^* \equiv_t \overline{T}^*$ as follows: $T^* \equiv_t \overline{T}^*$ iff $T^* \cap Past(t) = \overline{T}^* \cap Past(t)$.*

Definition D.3 (Equivalence class)**.** *Let T^* be a mcs. $[T^*]_t$ is the set of all mcs's that are equivalent to T^* up and including time t, i.e. $[T^*]_t = \{\overline{T}^* \mid T^* \equiv_t \overline{T}^*\}$.*

The next step is to reduce truth of a formula in a maximal consistent set to membership of that set, which is the content of the truth lemma. We first present a lemma that we need in the proof of the valuation lemma, which follows after that.

Lemma D.3. *Let* $\Sigma = \{\varphi_1, \ldots, \varphi_n\}$ *be some set of PAL-formulas and abbreviate* $\{\Box_t\varphi_1, \ldots, \Box_t\varphi_n\}$ *with* $\Box_t\Sigma$*. If* $\Sigma \vdash \varphi$*, then* $\Box_t\Sigma \vdash \Box_t\varphi$*.*

Proof. Suppose $\{\varphi_1, \ldots, \varphi_n\} \vdash \varphi$. By the Deduction theorem, $\vdash (\varphi_1 \wedge \ldots \wedge \varphi_n) \rightarrow \varphi$. Applying necessitation gives $\vdash \Box_t((\varphi_1 \wedge \ldots \wedge \varphi_n) \rightarrow \varphi)$, and from the K-axiom it follows that $\vdash \Box_t(\varphi_1 \wedge \ldots \wedge \varphi_n) \rightarrow \Box_t\varphi$. Since $\Box_t(\varphi_1 \wedge \ldots \wedge \varphi_n) \equiv \Box_t\varphi_1 \wedge \ldots \Box_t\varphi_n$, we obtain $\vdash (\Box_t\varphi_1 \wedge \ldots \Box_t\varphi_n) \rightarrow \Box_t\varphi$. Finally, by Deduction theorem $\{\Box_t\varphi_1, \ldots, \Box_t\varphi_n\} \vdash \Box_t\varphi$. □

Lemma D.4. $T_t^* \vdash \Box_t T_t^*$.

Proof. It is sufficient to show that for all $\varphi \in \mathit{Past}(t)$ we have

$$\varphi \vdash \Box_t\varphi.$$

We prove that claim by induction on the length of the formula φ. We distinguish two base cases, one where φ is a proposition, and another where φ is an atomic "do" formula.

(Base case 1) Suppose $\varphi = \chi_{t'}$ with $\chi \in \mathsf{Prop}$ and $t' \leq t$. $\Box_{t'}\chi_{t'}$ follows by applying Axiom A1. Then, apply Axiom A5 repeatedly until $\Box_t\chi_{t'}$ follows.

(Base case 2) Suppose $\varphi = do(a)_{t'}$ with $t' < t$ (note that $do(a)_t$ does not occur in $\mathit{Past}(t)$). Using Axiom A3 and then repeatedly Axiom A5 we obtain $\Box_t do(a)_{t'}$.

(Conjunction) Suppose $\varphi = \psi \wedge \chi$. By the induction hypothesis $\psi \vdash \Box_t\psi$ and $\chi \vdash \Box_t\chi$, so therefore $\psi \wedge \chi \vdash \Box_t\psi \wedge \Box_t\chi$. Since $\Box_t\psi \wedge \Box_t\chi$ is equivalent to $\Box_t(\psi \wedge \chi)$, it follows directly that $\psi \wedge \chi \vdash \Box_t(\psi \wedge \chi)$.

(Box) Suppose $\varphi = \Box_{t'}\psi$. By transitivity (which is not an axiom of our logic, but it holds in KT5): $\Box_{t'}\psi \rightarrow \Box_{t'}\Box_{t'}\psi$. Next, apply Axiom A5 repeatedly to obtain $\Box_t\Box_{t'}\varphi$.

(Negation) Suppose $\varphi = \neg\psi$. We will make a case distinction on the form of negated formula ψ.

(Base case 1) Suppose $\varphi = \neg\chi_{t'}$ with $\chi \in \mathsf{Prop}$ and $t' \leq t$, then $\Box_t\neg\chi_t$ follows from Axiom A2 (we use the contraposition of A2) and A5.

(Base case 2) Suppose $\varphi = \neg do(a)_{t'}$ with $t' < t$. We apply Axiom A4 (we use the contraposition of A4) and A5 repeatedly until we have $\Box_t\neg do(a)_{t'}$.

(Disjunction) Suppose $\varphi = \neg(\psi \vee \chi)$, which is equivalent to $\neg\psi \wedge \neg\chi$. By the induction hypothesis, $\neg\psi \vdash \Box_t\neg\psi$ and $\neg\chi \vdash \Box_t\neg\chi$, thus $\neg\psi \wedge \neg\chi \vdash \Box_t\neg\psi \wedge \Box_t\neg\chi$. Consequently, $\neg\psi \wedge \neg\chi \vdash \Box_t(\neg\psi \wedge \neg\chi)$, or, equivalently, $\varphi \vdash \Box_t\varphi$.

(Box) Suppose $\varphi = \neg\Box_{t'}\psi$, which is equivalent to $\Diamond_{t'}\neg\psi$. From Axiom 5 we obtain $\Box_{t'}\Diamond_{t'}\neg\psi$, and by again applying Axiom A5 repeatedly we obtain $\Box_t\Diamond_{t'}\neg\psi$, which is equivalent to $\Box_t\neg\Box_{t'}\psi$, and this is what we had to show.

(Negation) Suppose $\varphi = \neg\neg\psi$. Since ψ is a subformula of φ, by the induction hypothesis we have $\psi \vdash \Box_t\psi$. But ψ is equivalent to φ, so we have $\varphi \vdash \Box_t\varphi$.

□

Lemma D.5 (Valuation lemma)**.** *For any maximal consistent set T^*, the following are true*

1. *T^* is deductively closed: $T^* \vdash \varphi$ implies that $\varphi \in T^*$;*
2. *$\varphi \in T^*$ iff $\neg\varphi \notin T^*$;*
3. *$\varphi \wedge \psi \in T^*$ iff $\varphi \in T^*$ and $\psi \in T^*$;*
4. *$\Box_t\varphi \in T^*$ iff for all $\overline{T}^*$ s.t. $T^* \equiv_t \overline{T}^* : \varphi \in \overline{T}^*$.*

Proof.

1. Because T^* is maximally consistent, either $\varphi \in T^*$ or $\neg\varphi \in T^*$. Suppose that $T^* \vdash \varphi$, and suppose for contradiction that $\neg\varphi \in T^*$. From this it follows that $T^* \vdash \neg\varphi$ and therefore $T^* \vdash \bot$, which would contradict consistency of T^*. Hence $\varphi \in T^*$.

2. Follows directly from the definition of a maximally consistent set.

3. Follows directly as well.

4. $\Rightarrow$: Suppose $\Box_t\varphi \in T^*$. Take arbitrary $\overline{T}^*$ with $T^* \equiv_t \overline{T}^*$. From Def. D.2 (equivalent mcs) it follows that $\Box_t\varphi \in \overline{T}^*$. Therefore, by Axiom T we obtain $\varphi \in \overline{T}^*$.

   $\Leftarrow$: We show this by contraposition. Therefore, suppose $\Box_t\varphi \notin T^*$. We will show that there exists some $\overline{T}^*$ with $T^* \equiv_t \overline{T}^*$ and $\varphi \notin \overline{T}^*$.

   Suppose for contradiction that $\neg\varphi$ is not consistent with T_t^*, i.e. $T_t^* \cup \{\neg\varphi\} \vdash \bot$, so by the Deduction Theorem (Lemma D.1), $T_t^* \vdash \varphi$ holds. By Lemma D.3, we have (1) $\Box_t T_t^* \vdash \Box_t\varphi$. From Lemma D.4 it follows that (2) $T_t^* \vdash \Box_t T_t^*$, so combining (1) and (2) gives $T_t^* \vdash \Box_t\varphi$. But this contradicts with our initial assumption that $\Box_t\varphi \notin T^*$. Thus, the assumption is invalid so $T_t^* \cup \{\neg\varphi\}$ is consistent.

   By Lindenbaum's lemma, $T_t^* \cup \{\neg\varphi\}$ can be extended to a mcs $\overline{T}^*$, and since $T_t^* \subseteq \overline{T}^*$, it follows directly that $\overline{T}^* \equiv_t T^*$. Therefore, there exists a mcs $\overline{T}^*$ with $\overline{T}^* \equiv_t T^*$ and $\varphi \notin \overline{T}^*$, and this is what we had to show.

□

We construct the canonical model by naming the states in our model as equivalence classes of mcs's, which are parameterized by a time point. For instance, the state $s = [T^*]_t$ is named as the set of mcs's equivalent to the mcs T^* up to and including time t. We then define accessibility relation between states named after equivalence classes up to and including subsequent time points of the same mcs. Finally, the valuation function assigns the set of propositions that are true in an equivalence class to the corresponding state.

Definition D.4 (Canonical Tree)**.** *Given a mcs T^*, we obtain a PAL-canonical tree $Tree_{T^*} = (S, R, v, act)$, where*

1. *$S = \bigcup_{t\in\mathbb{N}} S_t$ where $S_t = \{[\overline{T}^*]_t \mid \overline{T}^* \equiv_0 T^*\}$*
2. *sRs' iff $(\exists \overline{T}^*, t \in \mathbb{N}).(s = [\overline{T}^*]_t \wedge s' = [\overline{T}^*]_{t+1})$*
3. *$p \in v(s)$ iff $(\exists \overline{T}^*, t \in \mathbb{N}).(s = [\overline{T}^*]_t \wedge p_t \in \overline{T}^*)$.*
4. *if $\pi_t = s$ and $\pi_{t+1} = s'$, then $a = act(\pi_t)$ iff $(\exists \overline{T}^*).(s = [\overline{T}^*]_t \wedge s' = [\overline{T}^*]_{t+1} \wedge do(a)_t \in \overline{T}^*)$*

Note that the existential quantifier in (3) of Def. D.4 could equivalently be replaced by a universal quantifier, because of the definition of equivalence classes (Def. D.3): All

mcs's in $[\overline{T}^*]_t$ are equivalent up to time t, so if some timed proposition p_t is an element of some mcs in this set, then it is necessarily an element of any other mcs in this set as well.

Lemma D.6. *Given a mcs T^*, $Tree_{T^*}$ is a tree.*

Proof. Suppose some T^* and let $Tree_{T^*} = (S, R, v, act)$. We have to show that R is serial, linearly ordered in the past, and connected.

- *serial*: Suppose some $s \in S$ s.t. $s = [\overline{T}^*]t$. We have to show that there exists some $s' \in S$ such that sRs', i.e. there exists some $\overline{T'}^*$ s.t. $s = [\overline{T'}]_t$ and $s' = [\overline{T'}]_{t+1}$. This directly follows for $\overline{T'} = \overline{T}$ and by the fact that T^* is a mcs.

- *linearly ordered in the past:* Suppose some $s \in S$ s.t. $s = [[\overline{T}^*]t$ and $t > 0$ ($t = 0$ is the root of the tree). We show that there exists exactly one s' s.t. $s'Rs$. Suppose for contradiction that there exist distinct $s', s'' \in S$ such that $s'Rs$ and $s''Rs$. Then there are $\overline{T'}, \overline{T''}$ such that $s' = [\overline{T'}]_{t-1}$, $s'' = [\overline{T''}]_{t-1}$ and $s = [\overline{T'}]_t = [\overline{T''}]_t$. From $s' \neq s''$ we obtain $\overline{T'} \not\equiv_{t-1} \overline{T''}$. However, then $\overline{T'} \not\equiv_t \overline{T''}$ holds as well, but this contradicts with $s = [\overline{T'}]_t = [\overline{T''}]_t$. Thus, $s' \neq s''$ is not possible.

- *connected:* Suppose $s, s' \in S$ with $s = [\overline{T}^*]_t$ and $s' = [\overline{T'}^*]_{t'}$. We show that there exists some s'' such that $s''R^*s$ and $s''R^*s'$, where R^* is the transitive closure of R. This directly holds for $s'' = [T^*]_0$, since then $s'' \in \{[\overline{T}^*]_0 \mid \overline{T}^* \equiv_0 T^*\}$.

□

Given a mcs T^*, we construct a path $\pi_{T^*} = (s_0, s_1, \ldots)$ from it by letting $s_t = [T^*]_t$. So $p \in v_p(\pi_{T^*t})$ iff $p_t \in T^*$ and $a = act(\pi_{T^*t})$ iff $do(a)_t \in T^*$.

Given a path π in a canonical tree $Tree_{T^*}$ and a $t \in \mathbb{N}$, we denote

$$T_{\pi|t} := \overline{T}^* \cap Past(t),$$

where $\overline{T}^* \in \pi_t$. Note that the definition is correct: $T_{\pi|t}$ does not depend on the choice of the element from T_π by Definition 2 and 3. We construct the set T_π from a path π as follows:

$$T_\pi = \bigcup_{t \in \mathbb{N}} T_{\pi|t}.$$

The next two lemmas show that for each π in the canonical tree $Tree_{T^*}$, T_π is a mcs.

Lemma D.7. *Given a mcs T^*, For any path π in the canonical tree $Tree_{T^*}$: $T_{\pi|t} \subseteq T_{\pi|t+1}$.*

Proof. Suppose some mcs T^* and some arbitrary path π in the canonical tree $Tree_{T^*}$. From the construction of the canonical tree we have that $\pi_t R \pi_{t+1}$ iff there is some $\overline{T}^*$ with $\pi_t = [\overline{T}^*]_t$ and $\pi_{t+1} = [\overline{T}^*]_{t+1}$. Clearly, we have that $\overline{T}^*_t \subseteq \overline{T}^*_{t+1}$, so $T_{\pi|t} \subseteq T_{\pi|t+1}$. □

Lemma D.8. *Given a mcs T^*, for any path π in the canonical tree $Tree_{T^*}$: T_π is a mcs.*

Proof. (Consistent) From Lemma D.7 we have that $T_{\pi|0} \subseteq \ldots \subseteq T_{\pi|t}$. Moreover, $T_{\pi|t} \subseteq \overline{T}^*$ where $\overline{T}^*$ is a mcs, which is consistent by definition, so T_π is consistent as well.

(Maximal) Suppose an arbitrary PAL-formula φ. Then there is a maximal t that appears in φ, therefore, $\varphi \in \mathit{Past}(t)$. By definition, $\varphi \in T_{\pi|t}$ or $\neg\varphi \in T_{\pi|t}$. Since $T_{\pi|t} \subseteq T_\pi$, we have that $\varphi \in T_\pi$ or $\neg\varphi \in T_\pi$. Hence T_π is maximal. □

The following three lemmas are direct consequences of the construction of T_π and π_T.

Lemma D.9. *Given a mcs T^*, two paths π and π' in the canonical tree $Tree_{T^*}$ and some time point t: $\pi \sim_t \pi'$ iff. $T_\pi \equiv_t T_{\pi'}$.*

Lemma D.10. *Given two mcs T^* and $\overline{T}^*$ and some time point t, $T^* \equiv_t \overline{T}^*$ iff. $\pi_{T^*} \sim_t \pi_{\overline{T}^*}$.*

Lemma D.11. *Given a mcs T^*, in the canonical tree $Tree_{T^*}$,*

1. *For each π, $\pi_{(T_\pi)} = \pi$.*
2. *For each $\overline{T}^*$, $T_{(\pi_{\overline{T}^*})} = \overline{T}^*$.*

Note that by the previous lemma, for every path π in the canonical tree $Tree_{T^*}$, there exists a unique mcs $\overline{T^*}$ such that $\pi = \pi_{\overline{T^*}}$.

Lemma D.12. *Given a mcs T^*: $(Tree_{T^*}, \pi_{T^*})$ is a model.*

Proof. Suppose some T^*. From Lemma D.6 we have that $Tree_{T^*}$ is a tree. In order to show that $(Tree_{T^*}, \pi_{\overline{T}^*})$ is a model we prove the four conditions on a model. Recall that

$$\pi_{T^*} = (s_0, s_1, \ldots) \text{ where } s_t = [T^*]_t.$$

1. Suppose $\mathit{act}(s_t) = a$. By the truth definition and the construction of s_t, we have $do(a)_t \in T^*$. By Axiom A8 and the maximality of $\overline{T}^*$, $\mathit{post}(a)_{t+1} \in \overline{T}^*$, so $\mathit{post}(a) \in s_{t+1}$.
2. Suppose $\mathit{pre}(a) \in v(\pi_{T^*t})$. We need to show that there is some π' in $Tree_{T^*}$ with $\pi_{T^*} \sim_t \pi'$ and $v(\pi'_t) = a$. From $\mathit{pre}(a) \in v(\pi_{T^*t})$ we conclude $\mathit{pre}(a)_t \in T^*$. Then by Axiom A9 and the maximality of $\overline{T}^*$ we obtain $\Diamond_t do(a)_t \in T^*$. By Lemma D.5(4), there exist $\overline{T}^*$ s.t. $T^* \equiv_t \overline{T}^*$ and $do(a)_t \in \overline{T}^*$. Obviously, for $\pi' = \pi_{\overline{T}^*}$ we have $\pi_{T^*} \sim_t \pi'$ (from Lemma D.10) and $v(\pi'_t) = a$.

3. and 4. can be proved in a similar way, using the axioms A9–A12 and the maximality of $\overline{T}^*$.

□

Lemma D.13 (Truth lemma)**.** *Given a mcs T^*: for every maximally PAL-consistent set of formula $\overline{T^*}$ and for every formula φ:*

$$(Tree_{T^*}, \pi_{\overline{T}^*}) \models \varphi \textit{ iff. } \varphi \in \overline{T}^*$$

Proof. By induction on the depth of the proof.

(Base case) Suppose $\varphi = \chi_t$, for some atomic proposition $\chi \in \mathscr{L}$. From the truth definition we have $Tree_{T^*}, \pi_{\overline{T}^*} \models \chi_t$ iff. $\chi \in v(\pi_{\overline{T}^*t})$. From the construction of $\pi_{\overline{T}^*t}$ it follows then directly that $\chi_t \in \overline{T}^*t$. Suppose $\varphi = do(a)_t$. From the truth definition we have $Tree_{T^*}, \pi_{\overline{T}^*} \models do(a)_t$ iff $act(\pi_{\overline{T}^*t}) = a$. Again, from the construction of $\pi_{\overline{T}^*}$ we obtain $do(a)_t \in \overline{T}^*$.

(Negation) Suppose $\varphi = \neg\psi$. From the valuation lemma we know that $\neg\psi \in \overline{T}^*$ iff $\psi \notin \overline{T}^*$. By the induction hypothesis, $\psi \notin \overline{T}^*$ is equivalent to $Tree_{T^*}, \pi_{\overline{T}^*} \not\models \psi$. According to the truth definition that is equivalent to $Tree_{T^*}, \pi_{\overline{T}^*} \models \neg\psi$. Hence, $\neg\psi \in \overline{T}^*$ is equivalent to $Tree_{T^*}, \pi_{\overline{T}^*} \models \neg\psi$.

(Conjunction) Suppose $\varphi = \psi \wedge \chi$. From the valuation lemma we know that $\psi \wedge \chi \in \overline{T}^*$ iff $\psi \in \overline{T}^*$ and $\chi \in \overline{T}^*$. By the induction hypothesis, that is equivalent to $Tree_{T^*}, \pi_{\overline{T}^*} \models \psi$ and $Tree_{T^*}, \pi_{\overline{T}^*} \models \chi$, respectively. Lastly, applying the truth definition, this is equivalent to $Tree_{T^*}, \pi_{\overline{T}^*} \models \psi \wedge \chi$. Therefore, $\psi \wedge \chi \in \overline{T}^*$ iff $Tree_{T^*}, \pi_{\overline{T}^*} \models \psi \wedge \chi$.

(Necessity) Suppose $\varphi = \Box_t\psi$. We show both directions of the bi-implication separately.

$\Rightarrow$: Suppose that $Tree_{T^*}, \pi_{\overline{T}^*} \models \Box_t\psi$, i.e. for all π' with $\pi_{\overline{T}^*} \sim_t \pi' : Tree_{T^*}, \pi' \models \psi$. Pick such π' arbitrarily. From Lemma D.11 we have that there is a unique mcs $\overline{\overline{T}}^*$ such that $\pi' = \pi_{\overline{\overline{T}}^*}$. Thus, $Tree_{T^*}, \pi_{\overline{\overline{T}}^*} \models \psi$ holds, and by the induction hypothesis, $\psi \in \pi_{\overline{\overline{T}}^*}$ holds as well. Since $\pi_{\overline{T}^*} \sim_t \pi_{\overline{\overline{T}}^*}$, from Lemma D.10 we obtain $\overline{T}^* \equiv_t \overline{\overline{T}}^*$. Thus, by the valuation lemma we obtain $\Box_t\psi \in \overline{T}^*$.

$\Leftarrow$: Suppose that $\Box_t\psi \in \overline{T}^*$. By the valuation lemma, for all $\overline{\overline{T}}^*$ with $\overline{T}^* \equiv_t \overline{\overline{T}}^* : \psi \in \overline{\overline{T}}^*$. Take such $\overline{\overline{T}}^*$ arbitrarily. From the induction hypothesis we have that $Tree_{T^*}, \pi_{\overline{\overline{T}}^*} \models \psi$. Since $\overline{T}^* \equiv_t \overline{\overline{T}}^*$, it follows from Lemma D.10 that $\pi_{\overline{T}^*} \sim_t \pi_{\overline{\overline{T}}^*}$. Since $\overline{\overline{T}}^*$ was chosen arbitrarily, we have that for all π' with $\pi_{\overline{T}^*} \sim_t \pi'$ it holds that $Tree_{\overline{T}}, \pi' \models \psi$. Therefore, $Tree_{T^*}, \pi_{\overline{T}^*} \models \Box_t\psi$. □

We can now prove that the logic PAL is strongly complete:

Theorem 1, Completeness. We prove this by contraposition, showing that $T \nvdash \varphi$ implies $T \not\models \varphi$. If $T \nvdash \varphi$, then $T \cup \{\varphi\}$ is inconsistent, so there is a mcs $T^* \supset T$ containing $\neg\varphi$, as the Lindenbaum lemma shows. By the Truth lemma we have that $Tree_{T^*}, \pi_{\overline{T}^*} \models \neg\varphi$ iff. $\neg\varphi \in \overline{T}^*$. And thus $Tree_{T^*}, \pi_{\overline{T}^*} \models \neg\varphi$, since $\neg\varphi \in \overline{T}^*$. Hence there is a model, namely $Tree$, and a path, namely $\pi_{\overline{T}^*}$, where $T \cup \{\neg\varphi\}$ is true and $T \cup \{\varphi\}$ is false. Therefore, $T \models \neg\varphi$, and that is what we had to show. □

D.2 Coherence Condition Proofs

Lemma 6.1. *if $I' \subseteq I$, then $Cohere(I) \vdash Cohere(I')$.*

Proof. Suppose an intention database $I = \{(b_{t_1}, t_1), \ldots, (b_{t_n}, t_n)\}$ with $t_1 < \ldots < t_n$. Re-

call that

$$Cohere(I) = \Diamond_0 \bigvee_{\substack{a_k \in Act : k \notin \{t_1,\ldots,t_n\} \\ a_k = b_k : k \in \{t_1,\ldots,t_n\}}} pre(a_{t_1}, a_{t_1+1}, \ldots, a_{t_n})_{t_1}. \tag{D.1}$$

The case where $I' = I$ follows directly. Suppose $I' \subset I$. We consider three cases and show that for each case $Cohere(I) \vdash Cohere(I')$. We write $preform(I)$ to denote the precondition disjunction in the coherence condition, i.e. $Cohere(I) = \Diamond_0 preform(I)$. Moreover, if the subscript notation is omitted from a big disjunction, then it is equal to that of Eq. (D.1).

1. *Case 1:* $I' = I \backslash \{(b_{t_1}, t_1)\}$. Thus,

   $$Cohere(I') = \Diamond_0 \bigvee pre(a_{t_2}, a_{t_2+1}, \ldots, a_{t_n})_{t_2}.$$

   Using Axiom (A11), we can derive $pre(a_{t_1})_{t_1}$ for each disjunct in $preform(I)$. Thus, we have that $preform(I)$ infers $pre(a_{t_1})_{t_1}$. From $pre(a_{t_1})_{t_1}$ using Axiom (A8) we obtain $\Diamond_{t_1} do(a_{t_1})_{t_1}$.

   Using (A1) and $preform(I)$, we derive

   $$\bigvee_{\substack{a_k \in Act : k \notin \{t_1,\ldots,t_n\} \\ a_k = b_k : k \in \{t_1,\ldots,t_n\}}} \Box_{t_1} pre(a_{t_1}, \ldots, a_{t_n})_{t_1}. \tag{D.2}$$

   Note that $(\Box_{t_1}\varphi \vee \Box_{t_1}\psi) \rightarrow \Box_{t_1}(\varphi \vee \psi)$ is a theorem in our logic. Combining this with Eq. (D.2) gives $\Box_{t_1} preform(I)$. The K-axiom is equivalent to $(\Box_{t_1}\varphi \wedge \Diamond_{t_1}\psi) \rightarrow \Diamond_{t_1}(\varphi \wedge \psi)$. Therefore, from $\Diamond_{t_1} do(a_{t_1})_{t_1}$ and $\Box_{t_1} preform(I)$ we obtain $\Diamond_{t_1}(do(a_{t_1})_{t_1} \wedge preform(I))$. Let $\varphi = do(a_{t_1})_{t_1}$ and $preform(I) = \psi_1 \vee \ldots \vee \psi_n$. We can rewrite $\Diamond_{t_1}(\varphi \wedge (\psi_1 \vee \ldots \vee \psi_n)$ to $\Diamond_{t_1}(\varphi \wedge \psi_1) \vee \ldots \vee \Diamond_t(\varphi \wedge \psi_n)$. Using Axiom (A12), we obtain

   $$\Diamond_{t+1} \bigvee pre(a_{t_1+1}, \ldots, a_{t_n})_{t_1+1} \tag{D.3}$$

   In each of the disjuncts of Eq. (D.3). In case $t_1 + 1 < t_2$, we make another case distinction on a_{t_1+1}: for each case we apply the same procedure as before and we obtain $pre(a_{t_1+2}, \ldots, a_{t_n})_{t_1+2}$ in each case distinction. Thus we can derive this formula. We can repeat this procedure until $t_1 + i = t_2$. This means we have shown that

   $$preform(I) \rightarrow \Diamond_{t_2} preform(I'). \tag{D.4}$$

   We remains to show is that $\Diamond_0 preform(I) \rightarrow \Diamond_0 preform(I')$. Let $\varphi = preform(I)$ and $\psi = \Diamond_{t_2} preform(I')$. Using Necessitation and contraposition, we have $\Box_0(\neg\psi \rightarrow \neg\varphi)$. Using the K-axiom and contraposition again, we obtain $\Diamond_0\varphi \rightarrow \Diamond_0\psi$, i.e. $Cohere(I) \rightarrow \Diamond_0\Diamond_{t_2} preform(I')$. $\Diamond_0\Diamond_{t_2} preform(I')$ derives $\Diamond_0 preform(I')$, so $Cohere(I) \rightarrow Cohere(I')$ is a theorem. Hence, by the deduction theorem, $Cohere(I) \vdash Cohere(I')$.

2. *Case 2:* $I' = I \backslash \{(b_{t_n}, t_n)\}$. Using Axiom (A11).

3. *Case 3:* $I' = I \backslash \{(b_{t_i}, t_i)\}$ with $t_1 < i < t_n$. Note that $preform(I) \rightarrow preform(I')$ is a theorem of our logic. Thus, using the same technique as in case 1, we obtain $\vdash Cohere(I) \rightarrow Cohere(I')$, and again by the deduction theorem, $Cohere(I) \vdash Cohere(I')$.

□

Proposition 6.3. *Given some belief-intention database* (SB, I)*, if* (SB, I) *is coherent, then* $WB(SB, I)$ *is consistent.*

Proof. First we show that $pre(a_0, \ldots, a_m)_t \vdash \Diamond_t(do(a)_t \wedge do(a_1)_{t+1} \wedge \ldots \wedge do(a_m)_{t+m})$. Therefore, suppose $(1) pre(a_0, \ldots, a_m)_t$. Applying Axiom A11 m times gives $pre(a_0)_t$. By Axiom A8 we obtain (2) $\Diamond_t do(a_0)_t$. From (1) and (2) and by the fact that $pre(a_0, \ldots, a_m)_t \equiv \Diamond_t pre(a_0, \ldots, a_m)_t$, we derive (3) $\Diamond_t(pre(a_{t+1}, \ldots, a_m)_{t+1} \wedge do(a_0)_t)$. Applying the same procedure to $pre(a_{t+1}, \ldots, a_m)_{t+1}$ iteratively until there are not *pre* formulas left, we obtain $\Diamond_t(do(a_0)_t \wedge \Diamond_{t+1}(do(a_1)_{t+1} \wedge \ldots))$. By taking the contrapositive of Axiom A3 repeatedly we obtain $\Diamond_t(do(a_0)_t \wedge \Diamond_t(do(a_1)_{t+1} \wedge \ldots))$. Rewriting this and using transitivity (i.e. $\Diamond_t \Diamond_t \varphi \rightarrow \Diamond_t \varphi$, which isn't an axiom but can be derived from KT5) we obtain $\Diamond_t(do(a)_t \wedge do(a_1)_{t+1} \wedge \ldots \wedge do(a_m)_{t+m})$, i.e. $\Diamond_t \bigwedge_{k=0}^{m} do(a_k)_{t+k}$.

Next, let $I = \{(b_{t_1}, t_1), \ldots, (b_{t_n}, t_n)\}$ with $t_1 < \ldots < t_n$. By the fact that $pre(a_0, \ldots, a_m)_t \vdash \bigwedge_{k=0}^{m} do(a_k)_{t+k}$, $Cohere(I)$ (Def. 11) implies

$$\Diamond_0 \bigvee_{\substack{a_k \in Act : k \notin \{t_1, \ldots, t_k\} \\ a_k = b_k : k \in \{t_1, \ldots, t_n\}}} \Diamond_{t_1}(do(a_{t_1})_{t_1} \wedge do(a_{t_1+1})_{t_1+1} \wedge \ldots \wedge do(a_{t_n})_{t_n}$$

Consequently, $Cohere(I)$ implies $\Diamond_0 \Diamond_{t_1} \bigwedge_{k=1}^{n} do(b_{t_k})_{t_k}$, and by (A3) and transitivity this implies $\Diamond_0 \bigwedge_{(a,t) \in I} do(a)_t$. Therefore, if (B, I) is coherent, then the set $B \cup \{\Diamond_0 \bigwedge_{(a,t) \in I} do(a)_t\}$ is consistent. By the completeness theorem, this means that there exists a model $m = (T, \pi)$ such that $m \models B$ and $m \models \Diamond_0 \bigwedge_{(a,t) \in I} do(a)_t$. Since B is a strong belief set, it follows that for all $\pi' \in T : T, \pi' \models B$. Since $m \models \Diamond_0 \bigwedge_{(a,t) \in I} do(a)_t$, there exists some $\pi'' \in T$ with $T, \pi'' \models \bigwedge_{(a,t) \in I} do(a)_t$. But then also $T, \pi'' \models B$, so there exists some model (T, π'') with $T, \pi'' \models B$ and $T, \pi'' \models \bigwedge_{(a,t) \in I} do(a)_t$. By the completeness theorem we obtain that $B \cup \{\bigwedge_{(a,t) \in I} do(a)_t\}$ is consistent, hence $WB(B, I)$ is consistent. □

D.3 Representation theorems proofs

Recall that $Mod(\varphi)$ is the set of models of φ. Similarly, given some t-restricted PAL formula φ, we define $Mod^{|t}(\varphi)$ as the set of t-restricted models of φ. Recall from the paper that $Ext(MSB^{|t})$ is the set of all possible extensions of a set of bounded model of strong beliefs $MSB^{|t}$ to models, i.e.

$$Ext(MSB^{|t}) = \{m \in \mathbb{M} \mid m^{|t} \in MSB^{|t}\}.$$

Note that Ext is defined on the sets of bounded models of **strong beliefs** only. In order to simplify notation in the proof of the representation theorem, we define the following abbreviation:

Given a set of restricted models $\{m_1^{|t},\dots,m_n^{|t}\}$, with $m_i^{|t} = (T_i^{|t},\pi_i^{|t})$, we introduce $Ext(m_1^{|t},\dots,m_n^{|t})$ as:

$$Ext(m_1^{|t},\dots,m_n^{|t}) := Ext(\{(T^{|t},\pi^{|t}) \mid \bigvee_{k=1}^{n} T^{|t} = T_i^{|t}\}). \tag{D.5}$$

Lemma 6.4. *Given a t-bounded msb set $MSB^{|t}$, there exists a strong belief formula $form(MSB^{|t}) \in \mathbb{SB}$ such that $Mod(form(MSB^{|t})) = Ext(MSB^{|t})$.*

Proof. For a given $T^{|t}$ we define the strong belief formula

$$form(T^{|t}) = \bigwedge_{\pi^{|t\prime} \in T^{|t}} \Diamond_0 \alpha_{\pi^{|t\prime}} \wedge \bigwedge_{\pi^{|t\prime} \notin T^{|t}} \neg\Diamond_0 \alpha_{\pi^{|t\prime}},$$

where

$$\alpha_{\pi^{|t}} = \bigwedge_{n=0}^{t} \left(\bigwedge_{\chi \in v(\pi^{|t}{}_n)} \chi_n \wedge \bigwedge_{\chi \notin v(\pi^{|t}{}_n)} \neg\chi_n \wedge \bigwedge_{act(\pi^{|t}{}_n)=a} do(a)_n \right).$$

Intuitively, $form(T^{|t})$ is a strong belief formula describing all of the paths of T up to t. Each $\alpha_{\pi^{|t}}$ is a formula describing the path π up to t: It contains all propositions that are true and false at each time moment, and all actions that are executed. Note that Axiom A7 of PAL-P ensures that only one action can be executed per time moment.

Let T' be a tree. From the construction of the formula $form(T^{|t})$ it follows that if $T^{|t\prime} = T^{|t}$, then for every $\pi' \in T'$ we have $(T',\pi') \models form(T^{|t})$. On the other hand, if $T^{|t\prime} \neq T^{|t}$, then there is π such that either $\pi^{|t} \in T^{|t} \setminus T^{|t\prime}$ or $\pi^{|t} \in T^{|t\prime} \setminus T^{|t}$. Suppose that $\pi^{|t} \in T^{|t} \setminus T^{|t\prime}$. Then for any $\pi \in T'$ we have $(T',\pi') \not\models \Diamond_0 \alpha_{\pi^{|t}}$, so $(T',\pi') \not\models form(T^{|t})$. Similarly, if $\pi^{|t} \in T^{|t\prime} \setminus T^{|t}$, then $(T',\pi) \not\models \neg\Diamond_0 \alpha_{\pi^{|t}}$, so again we have $(T',\pi) \not\models form(T^{|t})$. Thus, we proved

$$Mod(form(T^{|t})) = Ext(\{(T^{|t},\pi^{|t}) \in \mathbb{M}^{|t} \mid \pi^{|t} \in T^{|t}\}).$$

Now we define

$$form(M^{|t}{}_{SB}) = \bigvee \{form(T^{|t}) \mid (T^{|t},\pi^{|t}) \in M^{|t}{}_{SB}\}.$$

Note that our set of propositional letters is finite, and that we have finitely many deterministic actions, so $M^{|t}{}_{SB}$ is a finite set. Consequently, the above disjunction is finite.

Finally, we have

$$\begin{aligned} & Mod(form(M^{|t}{}_{SB})) \\ = & \bigcup \{Mod(form(T^{|t})) \mid (T^{|t},\pi^{|t}) \in M^{|t}{}_{SB}\} \\ = & \bigcup Ext(\{(T^{|t},\pi^{|t}) \in M^{|t}{}_{SB} \mid \pi^{|t} \in T^{|t}\}) \\ = & Ext(M^{|t}{}_{SB}). \end{aligned}$$

□ □

Note that $form$ is defined on the set of bounded models of **strong beliefs** only. In order to simplify notation in the proof of the representation theorem, we define the following

abbreviation:

Given a set of models $\{m_1,\ldots,m_n\}$, with $m_i=(T_i,\pi_i)$, we introduce $form(m_1,\ldots,m_n)$ as:

$$form(m_1,\ldots,m_n) := form(\{(T^{|t},\pi^{|t}) \mid \bigvee_i^n T^{|t} = T_i^{|t}\}). \tag{D.6}$$

Lemma D.14. *If $\varphi \in B_t$ and $T_1^{|t} = T_2^{|t}$, then $(T_1,\pi_1) \models \varphi$ iff $(T_2,\pi_2) \models \varphi$.*

Proof. By induction on the complexity of φ. □

Corollary 6.1. *Given a t-bounded set of strong beliefs $SB^{|t}$, there exists a strong belief formula $\psi \in \mathbb{SB}$ such that $SB^{|t} = \{\varphi \mid \psi \vdash \varphi\}$.*

Proof. For a given belief set B, from Lemma D.14 follows that $Mod^{|t}(B)$ is a set of t-bounded models of a strong beliefs such that $Ext(Mod^{|t}(B)) = Mod(B)$. If $\psi = form(Mod^{|t}(B))$, then form Lemma 6.4 we obtain $Mod(\psi) = Mod(B)$, and by the completeness theorem, $B = Cl(\psi)$. □ □

Note that for a formula $\varphi \in B_t$, the satisfiability of the formula in a model m depends only on the paths in its restricted counterpart $m^{|t}$, for a set of intentions bounded up to t so we can write that

$$(M^{|t},I) \text{ is coherent iff } (\mathrm{M},\mathrm{I}) \text{ is coherent.} \tag{D.7}$$

Definition 6.13 (Faithful assignment)**.** *A* faithful assignment *is a function that assigns to each strong belief formula $\psi \in \mathbb{SB}^{|t}$ a total pre-order $\leq_\psi^t$ over $\mathbb{M}$ and to each intention database $I \in \mathbb{D}^{|t}$ a selection function γ_I^t and satisfies the following conditions:*

1. *If $m_1,m_2 \in Mod(\psi)$, then $m_1 \leq_\psi^t m_2$ and $m_2 \leq_\psi^t m_1$.*
2. *If $m_1 \in Mod(\psi)$ and $m_2 \notin Mod(\psi)$, then $m_1 < m_2$.*
3. *If $\psi \equiv \phi$, then $\leq_\psi^t = \leq_\phi^t$.*
4. *If $T^{|t} = T_2^{|t}$, then $(T,\pi) \leq_\psi^t (T_2,\pi_2)$ and $(T_2,\pi_2) \leq_\psi^t (T,\pi)$.*

Theorem 6.1 (Representation Theorem)**.** *A belief-intention revision operator $*_t$ satisfies postulates (P1)-(P12) iff there exists a faithful assignment that maps each ψ to a total pre-order $\leq_\psi^t$ and each I to a selection function γ_I^t such that if $(\psi,I) *_t (\varphi,i) = (\psi',I')$, then:*

1. $Mod(\psi') = \min(Mod(\varphi), \leq_\psi^t)$
2. $I' = \gamma_I^t(Mod(\psi'),i)$

Proof. " $\Rightarrow$": Suppose that some agent revision operator $*_t$ satisfies postulates (P1)-(P12). Given models m_1 and m_2, let $(\psi,\emptyset) *_t (form(m_1,m_2),\varepsilon) = (\psi',\emptyset)$ (note that we use the abbreviation (D.6) for $form$). We define $\leq^t_\psi$ by $m_1 \leq^t_\psi m_2$ iff $m_1 \models \psi$ or $m_1 \models \psi'$. We also define γ^t_I by $\gamma^t_I(MSB^{|t},i) = I'$, where $(form(MSB^{|t}),I) *_t (\top,i) = (\psi_2,I')$ (note that $\psi_2 \equiv form(MSB^{|t})$).

In order to prove that the assignment is faithful for $\leq^t_\psi$, we define $\psi \circ_t \varphi = \psi'$ when $(\psi,\emptyset) *_t (\varphi,\varepsilon) = (\psi',\emptyset)$. We can now prove that postulates (P1)-(P6) for $*_t$ imply the Katsuno and Mendelzon postulates (R1)-(R6).

(R1) $\psi \circ_t \varphi$ implies φ

(R2) If $\psi \wedge \varphi$ is satisfiable, then $\psi \circ_t \varphi \equiv \psi \wedge \varphi$

(R3) If φ is satisfiable, then $\psi \circ_t \varphi$ is also satisfiable

(R4) If $\psi \equiv \psi'$ and $\varphi \equiv \varphi'$, then $\psi \circ_t \varphi \equiv \psi' \circ_t \varphi'$

(R5) $(\psi \circ_t \varphi) \wedge \varphi'$ implies $\psi \circ_t (\varphi \wedge \varphi')$

(R6) If $(\psi \circ_t \varphi) \wedge \varphi'$ is satisfiable, then $\psi \circ_t (\varphi \wedge \varphi')$ implies $(\psi \circ_t \varphi) \wedge \varphi'$

We show that (R5) holds, i.e. $(\psi \circ_t \varphi) \wedge \varphi'$ implies $\psi \circ_t (\varphi \wedge \varphi')$ holds, the other cases are similar. Let us denote with ψ' the formula $\psi \circ_t \varphi$, with $\overline{\psi}'$ the formula $\psi \circ_t (\varphi \wedge \varphi')$, and with $\overline{\varphi}$ the formula $\varphi \wedge \varphi'$. Then we have $(\psi,\emptyset) *_t (\varphi,\varepsilon) = (\psi',\emptyset)$ and $(\psi,\emptyset) *_t (\overline{\varphi},\varepsilon) = (\overline{\psi}',\emptyset)$. Then by postulate (P5), $\psi' \wedge \varphi'$ implies $\overline{\psi}'$, or equivalently $(\psi \circ_t \varphi) \wedge \varphi'$ implies $\psi \circ_t (\varphi \wedge \varphi')$.

Modifying the proof technique of Katsuno and Mendelzon, we show that 1) $\leq^t_\psi$ is a total pre-order, 2) the assignment ψ to $\leq^t_\psi$ is faithful, 3) $Mod(\psi') = \min(Mod(\varphi), \leq^t_\psi)$. Then we show that 4) γ^t_I is a selection function and 5) $I' = \gamma^t_I(Mod^{|t}(\psi'),i)$.

1. To show: $\leq^t_\psi$ is a total pre-order.

   - To show: Totality and reflexivity. From (R1) and (R3): $Mod(\psi \circ_t form(m_1,m_2))$ is a nonempty subset of $Ext(m_1^{|t},m_2^{|t})$ (note that we use the abbreviation (D.5) for Ext). Therefore, for each $m \in Ext(m_1^{|t})$ and $m' \in Ext(m_2^{|t})$, we have that either $m \leq^t_\psi m'$ or $m' \leq^t_\psi m$. We now show, without loss of generality, that for each $m,m' \in Ext(m_1^{|t})$, both $m \leq^t_\psi m'$ and $m' \leq^t_\psi m$ hold. Therefore, let $m,m' \in Ext(m_1^{|t})$, so $m^{|t} = m^{|t\prime}$. By Lemma 3 of the paper, $Mod(form(m^{|t})) = Ext(m^{|t}) = Ext(m^{|t\prime}) = Mod(form(m^{|t\prime}))$. Hence, $form(m) \equiv form(m')$, so $form(m,m') \equiv form(m)$. By (R4): $Mod(\psi \circ_t form(m^{|t})) = Mod(\psi \circ_t form(m,m')))$. By (R1), $m \in Mod(\psi \circ_t form(m))$, so $m \in Mod(\psi \circ_t form(m,m'))$. Hence, by the definition of $\leq^t_\psi$: $m \leq^t_\psi m'$. We can prove $m' \leq^t_\psi m$ similarly. This proves that $\leq^t_\psi$ is total, which implies reflexivity.
   - To show: Transitivity. Assume $m_1 \leq^t_\psi m_2$ and $m_2 \leq^t_\psi m_3$. We show $m_1 \leq^t_\psi m_3$. There are three cases to consider:

     (a) $m_1 \in Mod(\psi)$. $m_1 \leq^t_\psi m_3$ follows from the definition of $\leq^t_\psi$.

(b) $m_1 \notin Mod(\psi)$ and $m_2 \in Mod(\psi)$. Since $Mod(\psi \wedge form(m_1, m_2)) = Ext(m_2^{|t})$ holds, $Mod(\psi \circ_t form(m_1, m_2)) = Ext(m_2^{|t})$ follows from (R2). Thus $m_1 \not\leq_\psi^t m_2$ follows from $m_1 \notin Mod(\psi)$. This contradicts $m_1 \leq_\psi^t m_2$, so this case is not possible.

(c) $m_1 \notin Mod(\psi)$ and $m_2 \notin Mod(\psi)$. By (R1) and (R3), $Mod(\psi \circ_t form(m_1, m_2, m_3))$ is a nonempty subset of $Ext(m_1^{|t}, m_2^{|t}, m_3^{|t})$. We now consider two subcases.

i. $Mod(\psi \circ_t form(m_1, m_2, m_3)) \cap Ext(m_1^{|t}, m_2^{|t}) = \emptyset$. In this case, $Mod(\psi \circ_t form(m_1, m_2, m_3)) = Ext(m_3^{|t})$ holds. If we regard φ and φ' as $form(m_1, m_2, m_3)$ and $form(m_2, m_3)$ respectively in Conditions (R5) and (R6), we obtain

$$Mod(\psi \circ_t form(m_1, m_2, m_3)) \cap Ext(m_2^{|t}, m_2^{|t}) = Mod(\psi \circ_t form(m_2, m_3)).$$

Hence, $Mod(\psi \circ_t form(m_2, m_3)) = Ext(m_3^{|t})$. This contradicts $m_2 \leq_\psi^t m_3$ and $m_2 \notin Mod(\psi)$. Thus, this subcase is not possible.

ii. $Mod(\psi \circ_t form(m_1, m_2, m_3)) \cap Ext(m_1^{|t}, m_2^{|t}) \neq \emptyset$. Since $m_1 \leq_\psi^t m_2$ and $m_1 \notin Mod(\psi)$, $m_1 \in Mod(\psi \circ_t form(m_1, m_2))$ holds. Hence, by regarding φ and φ' as $form(m_1, m_2, m_3)$ and $form(m_1, m_2)$ respectively in Conditions (R5) and (R6), we obtain

$$Mod(\psi \circ_t form(m_1, m_2, m_3)) \cap Ext(m_1^{|t}, m_2^{|t}) = Mod(\psi \circ_t form(m_1, m_2)).$$

Thus,

$$m_1 \in Mod(\psi \circ_t form(m_1, m_2, m_3)) \cap Ext(m_1^{|t}, m_2^{|t})$$

holds. By using conditions (R5) and (R6) again in a similar way, we can obtain $m_1 \in Mod(\psi \circ_t form(m_1, m_3))$. Therefore, $m_1 \leq_\psi^t m_3$ holds.

2. To show: The assignment mapping ψ to $\leq_\psi^t$ is faithful. We prove the four conditions separately

(a) The first condition follows from the definition of $\leq_\psi^t$.

(b) For the second condition, assume that $m \in Mod(\psi)$ and $m' \notin Mod(\psi)$. Then $Mod(\psi \circ_t form(m, m')) = Ext(m^{|t})$ follows from (R2). Therefore, $m <_\psi^t m'$ holds.

(c) The third condition follows from (R4).

(d) For the fourth condition, for $m_1 = (T_1, \pi_1)$ and $m_2 = (T_2, \pi_2)$ such that $T_1^{|t} = T_2^{|t}$, let ψ' be as above. Since $\psi, \psi' \in B_t$, by Lemma D.14 we obtain $m_1 \models \psi$ iff $m_2 \models \psi$ and $m_1 \models \psi'$ iff $m_2 \models \psi'$, so $m_1 \leq_\psi^t m_2$ and $m_2 \leq_\psi^t m_1$.

3. To show: $Mod(\psi') = \min(Mod(\varphi), \leq_\psi^t)$. Note that this can be equivalently rewritten as $Mod(\psi \circ_t \varphi) = \min(Mod(\varphi), \leq_\psi^t)$. If φ is unsatisfiable then both are empty. So we assume φ is satisfiable. We show both containments separately.

- To show: $Mod(\psi \circ_t \varphi) \subseteq \min(Mod(\varphi), \leq^t_\psi)$. Assume for contradiction that $m \in Mod(\psi \circ_t \varphi)$ and $m \notin \min(Mod(\varphi), \leq^t_\psi)$. By condition (R1), m is a model of φ. Hence, there is a model m' of φ such that $m' <^t_\psi m$. We consider two cases:
  (a) $m' \in Mod(\psi)$. Since $m' \in Mod(\varphi)$, $\psi \wedge \varphi$ is satisfiable. Hence, by condition (R2), $\psi \circ_t \varphi \equiv \psi \wedge \varphi$ holds. Thus, $m \in Mod(\psi)$ follows from $m \in Mod(\psi \circ_t \varphi)$. Therefore, $m \leq^t_\psi m'$ holds. This contradicts $m' <^t_\psi m$.
  (b) $Mod(\psi \circ_t form(m,m')) = Ext(m^{|t})$. Since both m and m' are models of φ, $\varphi \wedge form(m,m') \equiv form(m,m')$ holds. Thus,

$$Mod(\psi \circ_t \varphi) \cap Ext(m^{|t}, m^{|t\prime}) \subseteq Mod(\psi \circ_t form(m,m'))$$

  follows from condition (R5). Since we assume $Mod(\psi \circ_t form(m,m')) = Ext(m^{|t\prime})$, we obtain $m \notin Mod(\psi \circ_t \varphi)$. This is a contradiction.
- To prove: $\min(Mod(\varphi), \leq^t_\psi) \subseteq Mod(\psi \circ_t \varphi)$. Assume for contradiction that $m \in \min(Mod(\varphi), \leq^t_\psi)$ and $m \notin Mod(\psi \circ_t \varphi)$. Since we also assume that φ is satisfiable, it follows from condition (R3) that there is an interpretation m' such that $m' \in Mod(\psi \circ_t \varphi)$. Since both m and m' are models of φ, $form(m,m') \wedge \varphi \equiv form(m,m')$ holds. By using conditions (R5) and (R6), we obtain

$$Mod(\psi \circ_t \varphi) \cap Ext(m^{|t}, m^{|t\prime}) = Mod(\psi \circ_t form(m,m')).$$

  Since $m \notin Mod(\psi \circ_t \varphi)$, $Mod(\psi \circ_t form(m,m')) = Ext(m^{|t\prime})$ holds. Hence, $m' \leq^t_\psi m$ holds. On the other hand, since m is minimal in $Mod(\varphi)$ with respect to $\leq^t_\psi$, $m \leq^t_\psi m'$ holds. Since $Mod(\psi \circ_t form(m,m')) = Ext(m^{|t\prime})$, $m \in Mod(\psi)$ holds. Therefore, $m \in Mod(\psi \circ_t \varphi)$ follows from condition (R2). This is a contradiction.

4. To show: γ^t_I is a selection function. This is direct consequence of the completeness theorem and the postulates (P7)-(P10) and (P12), taking into account (D.3). For example, if (P7) holds, then ψ' is consistent with $Cohere(I')$, so by completeness there is a model of both ψ' and $Cohere(I')$. since $Mod(\psi') = \min(Mod(\varphi), \leq^t_\psi)$, we obtain that $(Mod(\psi'), I')$ is coherent.

5. To show: $I' = \gamma^t_I(Mod^{|t}(\psi'), i)$. By our definition of γ^t_I we have that $(\psi', I) *_t (\top, i) = (\overline{\psi}, \gamma^t_I(Mod(\psi'), i))$ (recall that $\psi' \equiv \psi_2$). Since $(\psi, I) *_t (\varphi, i) = (\psi', I')$, by (P11) we obtain that $I' = \gamma^t_I(Mod(\psi'), i)$.

"$\Leftarrow$": Assume that there is a faithful assignment that maps ψ to a total pre-order $\leq^t_\psi$ and I to a selection function γ^t_I. We define the t-bounded revision operator $*_t$ as follows:

$$(\psi, C) *_t (\overline{\varphi}, c) = (form(\min(Mod^{|t}(\varphi), \leq^t_\psi), \gamma^t_I(\min(Mod^{|t}(\varphi), \leq^t_\psi), i)).$$

First we show that the operator is well correctly defined, i.e. that $\min(Mod^{|t}(\varphi), \leq^t_\psi)$ is a set of t-bounded models of strong beliefs. Let $\overline{T} = \overline{T}'$. Since $\varphi \in \mathbb{B}^{|t}$, by Lemma D.14 we obtain that $(T, \pi) \models \varphi$ iff $(T', \pi') \models \varphi$. Now suppose that $(T, \pi) \in \min(Mod(\varphi), \leq^t_\psi)$. If $(T', \pi') \notin \min(Mod(\varphi), \leq^t_\psi)$, then $(T, \pi) < (T', \pi')$, which is impossible by the definition

of faithful assignment. Thus, $\min(Mod^{|t}(\varphi), \leq^t_\psi)$ is a set of t-bounded models of strong beliefs.

Let us now prove that $*_t$ satisfies the postulates (P1)-(P-12). In order to prove the first six postulates, we define the operator $\circ_t$ by $\psi \circ_t \varphi = form(\min(Mod^{|t}(\varphi), \leq^t_\psi))$. Let use show that $\circ_t$ satisfies conditions (R1)-(R6) of KM (see the $(\Rightarrow)$ part of the proof). It is obvious that condition (R1) follows from the definition of the revision operator $\circ_t$. It is also obvious that conditions (R3) and (R4) follow from the definition of the faithful assignment. What remains to show is condition (R2), (R5), and (R6).

- To prove: (R2). It suffices to show if $Mod(\psi \wedge \varphi)$ is not empty then $Mod(\psi \wedge \varphi) = \min(Mod(\varphi), \leq^t_\psi)$. $Mod(\psi \wedge \varphi) \subseteq \min(Mod(\varphi), \leq^t_\psi)$ follows from the conditions of the faithful assignment. To prove the other containment, we assume that $m \in \min(Mod(\varphi), \leq^t_\psi)$ and $m \notin Mod(\varphi \wedge \varphi)$. Since $Mod(\psi \wedge \varphi)$ is not empty, there is a model $m' \in Mod(\psi \wedge \varphi)$. Then $m \not\leq^t_\psi m'$ follows from the conditions of the faithful assignment. Moreover, $m' \leq^t_\psi m$ follows from the conditions of the faithful assignment. Hence, m is not minimal in $Mod(\varphi)$ with respect to $\leq^t_\psi$. This is a contradiction.

- To prove: (R5) and (R6). It is obvious that if $(\psi \circ_t \varphi) \wedge \varphi'$ is unsatisfiable then (R6) holds. Hence, it suffices to show that if $\min(Mod(\mu), \leq^t_\psi) \cap Mod(\varphi')$ is not empty then

  $$\min(Mod(\varphi), \leq^t_\psi) \cap Mod(\varphi') = \min(Mod(\varphi \wedge \varphi'), \leq^t_\psi)$$

  holds. Assume that $m \in \min(Mod(\varphi), \leq^t_\psi) \cap Mod(\varphi')$ and $m \notin \min(Mod(\varphi \wedge \varphi'), \leq^t_\psi)$. Then, since $m \in Mod(\varphi \wedge \varphi')$, there is an interpretation m' such that $m' \in Mod(\varphi \wedge \varphi')$ and $m' <_\psi m$. This contradicts $m \in \min(Mod(\varphi), \leq^t_\psi)$. Therefore, we obtain

  $$\min(Mod(\varphi), \leq^t_\psi) \cap Mod(\varphi') \subseteq \min(Mod(\varphi \wedge \varphi'), \leq^t_\psi).$$

  To prove the other containment, we assume that $m \notin \min(Mod(\varphi), \leq^t_\psi) \cap Mod(\varphi')$ and $m \in \min(Mod(\varphi \wedge \varphi'), \leq^t_\psi)$. Since $m \in Mod(\varphi'), m \notin \min(Mod(\varphi), \leq^t_\psi)$ holds. Since we assume that $\min(Mod(\varphi), \leq^t_\psi) \cap Mod(\varphi')$ is not empty, suppose that m' is an element of $\min(Mod(\varphi), \leq^t_\psi) \cap Mod(\varphi')$. Then $m' \in Mod(\varphi \wedge \varphi')$ holds. Since we assume that $m \in \min(Mod(\varphi \wedge \varphi'), \leq^t_\psi)$ and $\leq^t_\psi$ is total, $m \leq^t_\psi m'$ holds. Thus, $m \in \min(Mod(\varphi), \leq^t_\psi)$ follows from $m' \in \min(Mod(\varphi), \leq^t_\psi)$. This is a contradiction.

Note that the conditions (R1)-(R6) imply the conditions (P1)-(P6). For example, suppose (R3) and let $(\psi, I) *_t (\varphi, i) = (\psi', I')$. Then $\psi' = form(\min(Mod(\varphi), \leq^t_\psi) \cap Mod(\varphi')) = \psi \circ_t \varphi$ so if φ is satisfiable, ψ' is satisfiable as well. Thus, (P3) holds.

The postulates (P7)-(P10) follow directly (using the completeness theorem and taking into account (D.3)) from the conditions the conditions 1-4 of the definition of selection function, and (P12) follows from the fourth condition as well.

Finally, let us prove (P11). Let $(\psi, I) *_t (\varphi, i) = (\psi', I')$ and $(\overline{\psi}, \bar{I}) *_t (\overline{\varphi}, \bar{i}) = (\overline{\psi}', \bar{I}')$, and suppose that $I = \bar{I}$, $i = \bar{i}$, and $\psi' \equiv \overline{\psi}'$. Then $Mod(\psi') = Mod(\overline{\psi}')$, i.e. $\min(Mod(\varphi), \leq^t_\psi) = \min(Mod(\varphi'), \leq^t_{\psi'})$, so

$$I' = \gamma^t_I(\min(Mod(\varphi), \leq^t_\psi), i) = \gamma^t_I(\min(Mod(\varphi'), \leq^t_{\psi'}), i) = \bar{I}'.$$

□

Theorem 6.2 (Representation Theorem)**.** *A t-bounded revision operator $\circ_t$ satisfies postulates (R*1)-(R*6) precisely when there exists a t-bounded faithful assignment on epistemic states that maps each epistemic state Ψ to a total pre-order $\leq^t_\Psi$ such that*

$$Mod(\Psi \circ_t \varphi) = \min(Mod(\varphi), \leq^t_\Psi)$$

Proof. The proof is very similar to that of Darwiche and Pearl, where we apply the same modifications to the proof as we did in the proof of Katsuno and Mendelzon (Theorem 6.1). □

Theorem 6.3. *Suppose that a t-bounded revision operator on epistemic states satisfies postulates (R*1)-(R*6). The operator satisfies postulates (C1)-(C4) iff the operator and its corresponding faithful assignment satisfy:*

CR1 If $m_1 \models \varphi$ and $m_2 \models \varphi$, then $m_1 \leq^t_\Psi m_2$ iff $m_1 \leq^t_{\Psi\circ\varphi} m_2$.

CR2 If $m_1 \not\models \varphi$ and $m_2 \not\models \varphi$, then $m_1 \leq^t_\Psi m_2$ iff $m_1 \leq^t_{\Psi\circ\varphi} m_2$.

CR3 If $m_1 \models \varphi$, $m_2 \not\models \varphi$ and $m_1 <^t_\Psi m_2$, then $m_1 <^t_{\Psi\circ\varphi} m_2$.

CR4 If $m_1 \models \varphi$, $m_2 \not\models \varphi$ and $m_1 \leq^t_\Psi m_2$, then $m_1 \leq^t_{\Psi\circ\varphi} m_2$.

Proof. 1. Postulate (C1) is equivalent to (CR1).

(⇒) Suppose that (C1) holds and $m, m' \models \varphi$. Let $\alpha = form(M^{|t}, M^{|t\prime})$. By Lemma 3, $Mod(\alpha) = Ext(M^{|t}) \cup Ext(M^{|t\prime})$. If $m'' \in Ext(M^{|t})$, then $M^{|t} = M^{|t\prime\prime}$, so by Lemma 2 we obtain $m'' \models \varphi$. Similarly, if $m'' \in Ext(M^{|t\prime})$, then $m'' \models \varphi$, so $\alpha \models \varphi$. By (C1), $(\Psi \circ_t \varphi) \circ_t \alpha \equiv \Psi \circ_t \alpha$. In other words, $\min(Mod(\alpha), \leq^t_{\Psi\circ_t\varphi}) = \min(Mod(\alpha), \leq^t_\Psi)$. Consequently, $m \leq^t_{\Psi\circ_t\varphi} m'$ iff $m \leq^t_\Psi m'$.

(⇐) The proofs of this direction are identical to those of Darwiche and Pearl. We give this case as an illustration but omit the other three. Suppose that (CR1) holds and assume $\varphi \models \varphi'$. We want to show that $\Psi \circ_t \varphi \equiv (\Psi \circ_t \varphi') \circ_t \varphi$. Condition (CR1) implies that $\leq_\Psi$ and $\leq_{\Psi\circ_t\mu}$ are equivalent for all $m_1, m_2 \in Mod(\varphi)$ since $\varphi \models \varphi'$. Hence $Mod(\Psi \circ_t \varphi) = \min(\varphi, \leq^t_\Psi) = \min(\varphi, \leq^t_{\Psi\circ_t\varphi'}) = Mod((\Psi \circ_t \varphi') \circ_t \varphi)$ and $\Psi \circ_t \varphi \equiv (\Psi \circ_t \varphi') \circ_t \varphi$.

2. Postulate (C2) is equivalent to (CR2): Symmetric to (C1) and (CR1).

3. Postulate (C3) is equivalent to (CR3).

(⇒) Suppose that (C3) holds, $m \models \varphi$, $m' \not\models \varphi$, and $m <_\Psi m'$. We want to show that $m <_{\Psi\circ_t\varphi} m'$. Let $\alpha = form(M^{|t}, M^{|t\prime})$. By Lemma 3: $Mod(\alpha) = Ext(M^{|t}) \cup Ext(M^{|t\prime})$. From $Ext(M^{|t}) \models \alpha \wedge \varphi$, $m <^t_\Psi m'$, and $Mod(\alpha \wedge \neg\varphi) = Ext(M^{|t\prime})$, we obtain $\Psi \circ_t \alpha \models \varphi$. By (C3): $(\Psi \circ_t \varphi) \circ_t \alpha \models \varphi$. Therefore, since $Mod(\alpha \wedge \varphi) = Ext(M^{|t})$ and $Mod(\alpha \wedge neg\varphi) = Ext(M^{|t\prime})$, we have $m_1 <^t_{\Psi\circ_t\varphi} m_2$ for each $m_1 \in Ext(M^{|t}{}_1)$ and $m_2 \in Ext(M^{|t}{}_2)$. In particular, since $m \in Ext(M^{|t}{}_1)$ and $m' \in Ext(M^{|t}{}_2)$, $m <^t_{\Psi\circ_t\varphi} m'$ holds.

($\Leftarrow$) Identical to the DP proof.

4. Postulate (C4) is equivalent to (CR4).

   $\Rightarrow$ Suppose (C4) holds, $m \models \varphi$ and $m' \not\models \varphi$. We prove this by taking the contrapositive. Therefore, suppose $m' <^t_{\Psi \circ_t \varphi} m$. We want to show that $m' <^t_{\Psi} m$. Let $\alpha = form(M^{|t}, M^{|t\prime})$. Since $Ext(M^{|t\prime}) \models \alpha \wedge \neg\varphi$, $m_2 <_{\Psi \circ_t \varphi} m_1$, and $Mod(\alpha \wedge \varphi) = Ext(M^{|t})$, we have $(\Psi \circ_t \varphi) \circ_t \alpha \models \neg\varphi$. From (C4): $\Psi \circ_t \alpha \models \neg\varphi$. So, from $Mod(\alpha \wedge \neg\varphi) = Ext(M^{|t\prime})$ and $Mod(\alpha \wedge \varphi) = Ext(M^{|t})$, we obtain $m_1 \leq^t_{\Psi} m_2$ for every $m_1 \in Ext(M^{|t})$ and $m_2 \in Ext(M^{|t\prime})$. Finally, since $m \in Ext(M^{|t})$ and $m' \in Ext(M^{|t\prime})$, we obtain $m_1 \leq^t_{\Psi} m_2$.

   $\Leftarrow$ Identical to the DP proof.

□

E

Tileworld Experiments

In this Appendix we present some illustrations of our simulation environments, and present graphs from some of our simulation results. The corresponding text can be found in Chapter 6, Section 6.5.2.

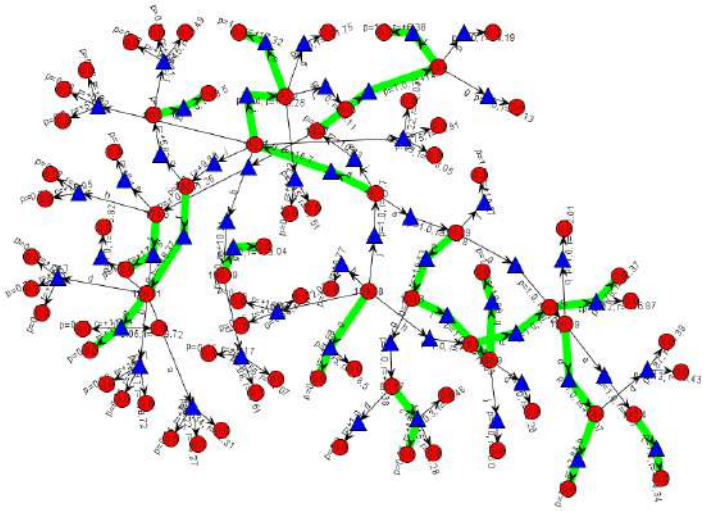


Figure E.1: A simulated Markov Decision Process in our software. Red circles denote MDP states, blues triangles denote Q-states, and green arrows denote the optimal policy computed using value iteration. Rewards and probabilities are denotes respectively next to the states and arcs.

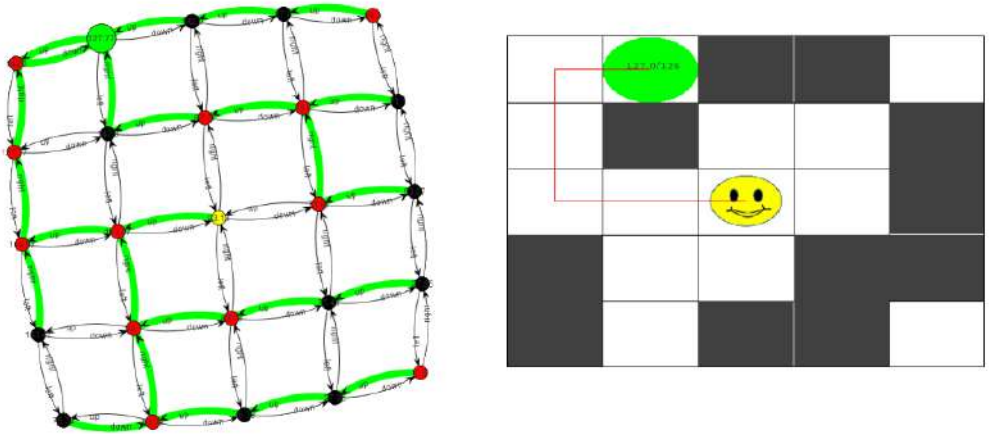


Figure E.2: Tileworld representation in our software as an MDP (left), and in the more familiar Tileworld format (right), omitting Q-states (since all probabilities are 1).

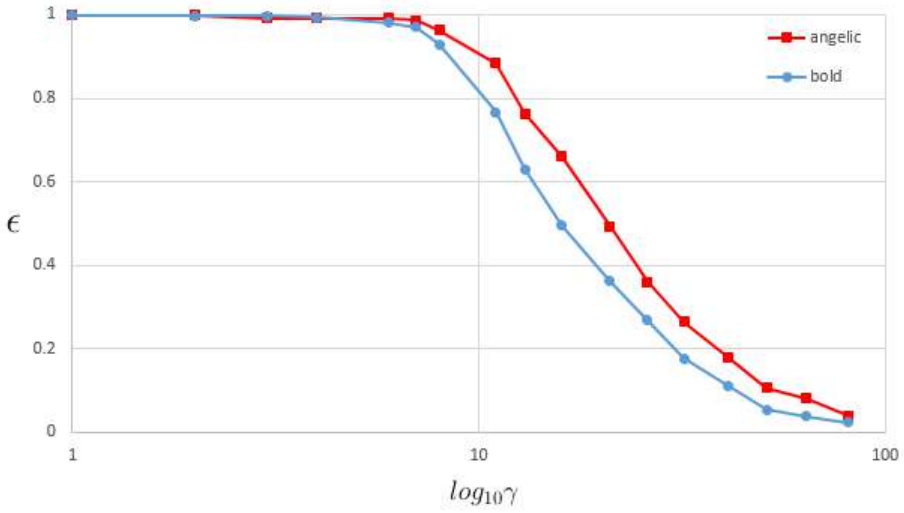


Figure E.3: Angelic planner vs Bold agent ($p = 2$). Following Kinny and Georgeff, we plot the rate of the world change γ against the agent's effectiveness ε, and we plot values of γ in log_{10}.

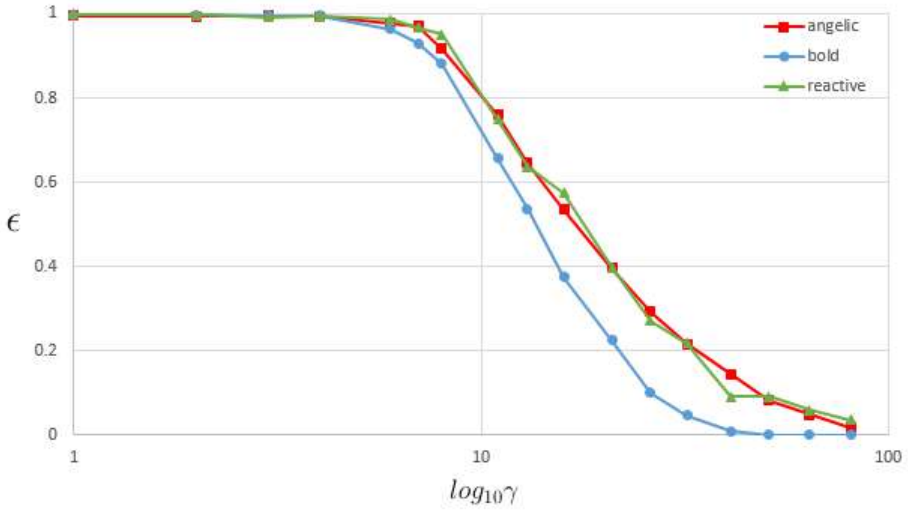


Figure E.4: Angelic planner vs Bold agent vs Reactive agent ($p = 4$). The rate of the world change γ is plotted against the agent's effectiveness ε.

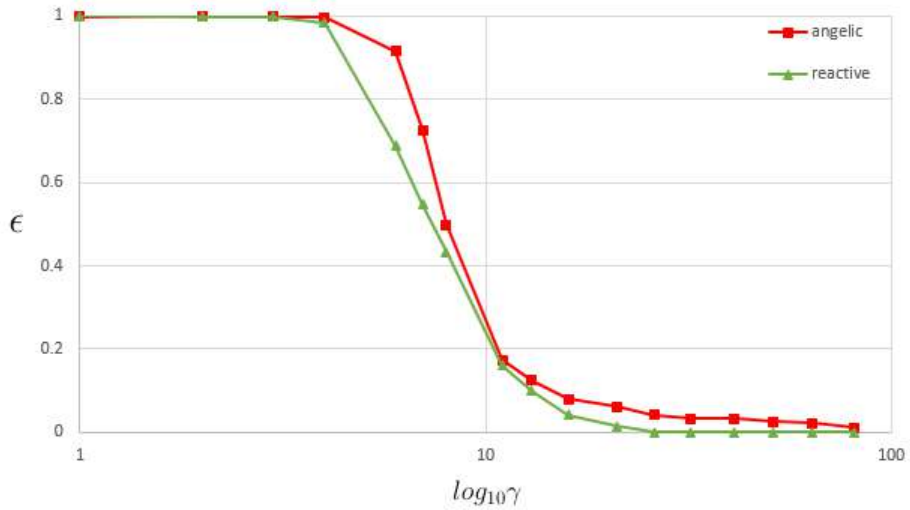


Figure E.5: Angelic planner vs Reactive agent ($p = 2, w = 5 \times 5, g = [10,20], l = [10,20]$). The rate of the world change γ is plotted against the agent's effectiveness ϵ.

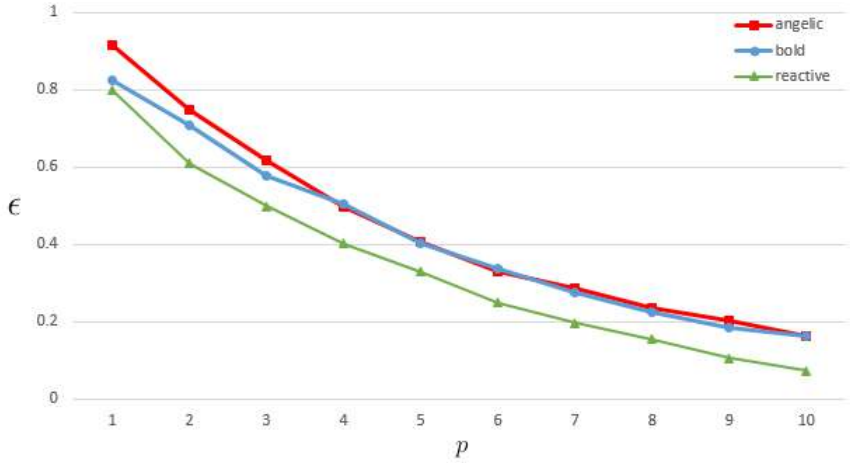


Figure E.6: Angelic planner vs Bold agent vs Reactive agent ($p = 2, w = 5 \times 5, g = [3,5], l = [5,8]$). The planning time p is plotted against the agent's effectiveness ϵ.

Bibliography

[ABC05] Katie Atkinson and Trevor Bench-Capon. Legal case-based reasoning as practical reasoning. *Artificial Intelligence and Law*, 13(1):93–131, 2005.

[ABC07] Katie Atkinson and Trevor Bench-Capon. Practical reasoning as presumptive argumentation using action based alternating transition systems. *Artificial Intelligence*, 171(10):855–874, 2007.

[AC02] Leila Amgoud and Claudette Cayrol. A reasoning model based on the production of acceptable arguments. *Annals of Mathematics and Artificial Intelligence*, 34(1-3):197–215, 2002.

[AD94] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[AFH96] Rajeev Alur, Tomás Feder, and Thomas A. Henzinger. The benefits of relaxing punctuality. *J. ACM*, 43(1):116–146, January 1996.

[AGH+10] Daniel Amyot, Sepideh Ghanavati, Jennifer Horkoff, Gunter Mussbacher, Liam Peyton, and Eric S. K. Yu. Evaluating goal models within the goal-oriented requirement language. *International Journal of Intelligent Systems*, 25:841–877, August 2010.

[AGM85] C. E. Alchourrön, P. Gärdenfors, and D. Makinson. On the logic of theory change: Partial meet contractions and revision functions. *Journal of Symbolic Logic*, pages 510–530, 1985.

[AH90] Rajeev Alur and Thomas A. Henzinger. Real-time logics: Complexity and expressiveness. *Information and Computation*, 104:390–401, 1990.

[AHK98] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. In *Revised Lectures from the International Symposium on Compositionality: The Significant Difference*, COMPOS'97, pages 23–60, London, UK, UK, 1998. Springer-Verlag.

[BCK12] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice*. Addison-Wesley Professional, 3rd edition, 2012.

[BD94] Mark S. Boddy and Thomas L. Dean. Deliberation scheduling for problem solving in time-constrained environments. *Artif. Intell.*, 67(2):245–285, 1994.

[Ber08] Brian Berenbach. The other skills of the software architect. In *1st Int. Workshop on Leadership and Management in Software Architecture*, pages 7–12. ACM, 2008.

[BG13] Nils Bulling and Valentin Goranko. How to be both rich and happy: Combining quantitative and qualitative strategic reasoning about multi-player games (extended abstract). In Fabio Mogavero, Aniello Murano, and Moshe Y. Vardi, editors, *SR*, volume 112 of *EPTCS*, pages 33–41, 2013.

[BH95] Jonathan P Bowen and Michael G Hinchey. Seven more myths of formal methods. *IEEE software*, 12(4):34–41, July 1995.

[BIP88] Michael E Bratman, David J Israel, and Martha E Pollack. Plans and resource-bounded practical reasoning. *Computational intelligence*, 4(3):349–355, 1988.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.

[BMN00] P. Bellini, R. Mattolini, and P. Nesi. Temporal logics for real-time system specification. *ACM Comput. Surv.*, 32(1):12–42, March 2000.

[BNS03] P. Bernus, L. Nemes, and G. Schmidt, editors. *Handbook on Enterprise Architecture*. International Handbooks on Information Systems. Springer, Berlin, Germany, 2003.

[Bra87] Michael E. Bratman. *Intention, plans, and practical reason*. Harvard University Press, Cambridge, MA, 1987.

[Bro99] Rodney Allen Brooks. *Cambrian intelligence: the early history of the new AI*, volume 1. MIT press Cambridge, MA, 1999.

[BS02] Valerie Belton and Theodor Stewart. *Multiple criteria decision analysis: an integrated approach*. Springer Science & Business Media, 2002.

[Byl94] Tom Bylander. The computational complexity of propositional strips planning. *Artificial Intelligence*, 69:165–204, 1994.

[CA09] Dan Cartwright and Katie Atkinson. Using computational argumentation to support e-participation. *IEEE Intelligent Systems*, 24(5):42–52, 2009.

[CD06] J.D. Cummins and N.A. Doherty. The economics of insurance intermediaries. *The Journal of Risk and Insurance*, 73(3):359–396, 2006.

[Cha87] David Chapman. Planning for conjunctive goals. *Artificial intelligence*, 32(3):333–377, 1987.

[CKI88] Bill Curtis, Herb Krasner, and Neil Iscoe. A field study of the software design process for large systems. *Communications of the ACM*, 31(11):1268–1287, 1988.

[CKM02] Jaelson Castro, Manuel Kolp, and John Mylopoulos. Towards requirements-driven information systems engineering: the tropos project. *Information systems*, 27(6):365–389, 2002.

[CL90a] Philip R Cohen and Hector J Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(2-3):213–261, 1990.

[CL90b] Philip R. Cohen and Hector J. Levesque. Intention is choice with commitment. *Artif. Intell.*, 42(2–3):213–261, 1990.

[CNYM12] Lawrence Chung, Brian A Nixon, Eric Yu, and John Mylopoulos. *Non-functional requirements in software engineering*, volume 5. Springer Science & Business Media, 2012.

[Dan09] P Dankova. Main aspects of enterprise architecture concept. *Economic Alternatives Journal*, 1(1):102–114, 2009.

[Dav63] Donald Davidson. Actions, reasons, and causes. *Journal of Philosophy*, 60(23):685–700, 1963.

[Die08] J.L.G. Dietz. *Architecture – Building strategy into design*. Netherlands Architecture Forum, Academic Service – SDU, The Hague, The Netherlands, 2008.

[DKKN95] Thomas Dean, Leslie Pack Kaelbling, Jak Kirman, and Ann Nicholson. Planning under time constraints in stochastic domains. *Artificial Intelligence*, 76(1):35–74, 1995.

[Don04] Paolo Donzelli. A goal-driven and agent-based requirements engineering framework. *Requirements Engineering*, 9(1):16–39, 2004.

[Doy88] J. Doyle. Artificial intelligence and rational self-government, 1988.

[DP95] Didier Dubois and Henri Prade. Possibility theory as a basis for qualitative decision theory. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pages 1924–1930, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[DP97] Adnan Darwiche and Judea Pearl. On the logic of iterated belief revision. *Artificial Intelligence*, 89(12):1 – 29, 1997.

[DPvdMM14] Jacobus Du Preez, Alta van der Merwe, and Machdel Matthee. Enterprise architecture schools of thought: An exploratory study. In *EDOCW 2014*, pages 3–12. IEEE, 2014.

[DT99] Jon Doyle and Richmond H. Thomason. Background to qualitative decision theory. *AI MAGAZINE*, 20, 1999.

[Dun95] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial intelligence*, 77(2):321–357, 1995.

[DvL96] Robert Darimont and Axel van Lamsweerde. Formal refinement patterns for goal-driven requirements elaboration. In *Proceedings of the 4th ACM SIGSOFT Symposium on Foundations of Software Engineering*, SIGSOFT '96, pages 179–190, New York, NY, USA, 1996. ACM.

[DVLF93] Anne Dardenne, Axel Van Lamsweerde, and Stephen Fickas. Goal-directed requirements acquisition. *Science of computer programming*, 20(1):3–50, 1993.

[Eco17] The Economist. Automation and anxiety - Will smarter machines cause mass unemployment?, February 2017. February 4.

[ET99] E. Allen Emerson and Richard J. Trefler. Parametric quantitative temporal reasoning. In *LICS*, pages 336–343. IEEE Computer Society, 1999.

[FO17] Carl Benedikt Frey and Michael A Osborne. The future of employment: how susceptible are jobs to computerisation? *Technological Forecasting and Social Change*, 114:254–280, 2017.

[GFE05] Salvatore Greco, J Figueira, and M Ehrgott. Multiple criteria decision analysis. *Springer's International series*, 2005.

[Gha13] Sepideh Ghanavati. *Legal-urn framework for legal compliance of business processes*. PhD thesis, University of Ottawa, 2013.

[Gia10] R.E. Giachetti. *Design of Enterprise Systems: Theory, Architecture, and Methods*. CRC Press, 2010.

[GKPW10] John Grant, Sarit Kraus, Donald Perlis, and Michael Wooldridge. Postulates for revising BDI structures. *Synthese*, 175(1):39–62, 2010.

[GMS05] Paolo Giorgini, John Mylopoulos, and Roberto Sebastiani. Goal-oriented requirements analysis and reasoning in the tropos methodology. *Engineering Applications of Artificial Intelligence*, 18(2):159–171, 2005.

[Got13] Joachim Gotze. The changing role of the enterprise architect. In *EDOCW 2013*, pages 319–326. IEEE, 2013.

[GP11] Danny Greefhorst and Erik Proper. *Architecture Principles: The Cornerstones of Enterprise Architecture*. Springer Publishing Company, Incorporated, 1st edition, 2011.

[GS02] Gerd Gigerenzer and Reinhard Selten. *Bounded rationality: The adaptive toolbox*. MIT press, 2002.

[GT14] Matthias Galster and Dan Tofan. Exploring web advertising to attract industry professionals for software engineering surveys. In *Proceedings of the 2nd International Workshop on Conducting Empirical Studies in Industry*, pages 5–8. ACM, 2014.

[Hal90] Anthony Hall. Seven myths of formal methods. *Software, IEEE*, 7(5):11–19, 1990.

[Har76] Gilbert Harman. Practical reasoning. *The Review of Metaphysics*, pages 431–463, 1976.

[HBJ+14] Jennifer Horkoff, Daniele Barone, Lei Jiang, Eric Yu, Daniel Amyot, Alex Borgida, and John Mylopoulos. Strategic business modeling: representation and reasoning. *Software & Systems Modeling*, 13(3):1015–1041, 2014.

[HL04] Andreas Herzig and Dominique Longin. C&L Intention Revisited. *KR*, 04:527–535, 2004.

[HLPX16] Andreas Herzig, Emiliano Lorini, Laurent Perrussel, and Zhanhao Xiao. Bdi logics for bdi architectures: Old problems, new perspectives. *KI - Künstliche Intelligenz*, pages 1–11, 2016.

[HMLN05] Charles B Haley, Jonathan D Moffett, Robin Laney, and Bashar Nuseibeh. Arguing security: Validating security requirements using structured argumentation. In *in Proc. of the Third Symposium on RE for Information Security (SREIS'05)*, 2005.

[Hol08] Richard Holton. Partial belief, partial intention. *Mind*, pages 1–26, 2008.

[Hoo09] J.A.P. Hoogervorst. *Enterprise Governance and Enterprise Engineering*. Springer, Berlin, Germany, 2009.

[HPXZ16] Andreas Herzig, Laurent Perrussel, Zhanhao Xiao, and Dongmo Zhang. Refinement of intentions. In *European Conference on Logics in Artificial Intelligence*, pages 558–563. Springer, 2016.

[IJLP09] M.-E. Iacob, H. Jonkers, M.M. Lankhorst, and H.A. Proper. *ArchiMate 1.0 Specification*. The Open Group, 2009.

[IJLP12] M.-E. Iacob, H. Jonkers, M.M. Lankhorst, and H.A. Proper. *ArchiMate 2.0 Specification*. The Open Group, 2012.

[IPS10] Thomas F. Icard, Eric Pacuit, and Yoav Shoham. Joint revision of beliefs and intention. In *KR*, 2010.

[ITU08] ITU-T. Recommendation Z.151 (11/08): User Requirements Notation (URN) – Language Definition. `http://www.itu.int/rec/T-REC-Z.151/en`, 2008.

[JB05] Anton Jansen and Jan Bosch. Software architecture as a set of architectural design decisions. In *Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture*, WICSA '05, pages 109–120, Washington, DC, USA, 2005. IEEE Computer Society.

[JBQ12] H. Jonkers, I. Band, and D. Quartel. The ArchiSurance Case Study. White Paper Y121, The Open Group, Spring 2012.

[JFS08] I.J. Jureta, S. Faulkner, and P.Y. Schobbens. Clear justification of modeling decisions for goal-oriented requirements engineering. *Requirements Engineering*, 13(2):87–115, May 2008.

[KAV05] Stephen H Kaisler, Frank Armour, and Michael Valivullah. Enterprise architecting: Critical problems. In *System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on*, pages 224b–224b. IEEE, 2005.

[KL05] Evangelia Kavakli and Pericles Loucopoulos. Goal modeling in requirements engineering: Analysis and critique of current methods. In John Krogstie, Terry A. Halpin, and Keng Siau, editors, *Information Modeling Methods and Methodologies*, pages 102–124. Idea Group, 2005.

[KLvV06] Philippe Kruchten, Patricia Lago, and Hans van Vliet. Building up and reasoning about architectural knowledge. In *Proceedings of the Second International Conference on Quality of Software Architectures*, QoSA'06, pages 43–58, Berlin, Heidelberg, 2006. Springer-Verlag.

[KM91] Hirofumi Katsuno and Alberto O. Mendelzon. Propositional knowledge base revision and minimal change. *Artificial Intelligence*, 52(3):263–294, dec 1991.

[Koy90] Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Syst.*, 2(4):255–299, October 1990.

[KR70] Werner Kunz and Horst Rittel. Issues as elements of information systems. Working Paper 131, Institute of Urban and Regional Development, University of California, Berkeley, California, 1970.

[Kru08] Philippe Kruchten. What do software architects really do? *Journal of Systems and Software*, 81(12):2413–2416, 2008.

[KS93] Don N. Kleinmuntz and David A. Schkade. Information displays and decision processes. *Psychological Science*, 4:221 – 227, 1993.

[KTZT12] Evangelos Kalampokis, Efthimios Tambouris, Maria Zotou, and Konstantinos Tarabanis. The enterprise architecture competence framework. *Int. J. of Learning Technology*, 7(1):79–94, 2012.

[Kva05] Jonas Kvarnström. *TALplanner and other extensions to Temporal Action Logic*. PhD thesis, Linköpings universitet, 2005.

[Lan05a] M. Lankhorst. *Enterprise architecture at work: modelling, communication, and analysis*. Springer, 2005.

[Lan05b] M.M. Lankhorst, editor. *Enterprise Architecture at Work: Modelling, Communication and Analysis*. Springer, Berlin, Germany, 2005.

[Lee91] Jintae Lee. Extending the potts and bruns model for recording design rationale. In *ICSE*, pages 114–125, 1991.

[LH08] Emiliano Lorini and Andreas Herzig. A logic of intention and attempt. *Synthese*, 163(1):45–77, 2008.

[LY04] Lin Liu and Eric Yu. Designing information systems in social context: a goal and scenario modelling approach. *Information systems*, 29(2):187–203, 2004.

[MA09] Gunter Mussbacher and Daniel Amyot. Goal and scenario modeling, analysis, and transformation with jUCMNav. In *ICSE Companion*, pages 431–432, 2009.

[Mar78] James G March. Bounded rationality, ambiguity, and the engineering of choice. *The Bell Journal of Economics*, pages 587–608, 1978.

[Mas10] Mark Mason. Sample size and saturation in PhD studies using qualitative interviews. In *Forum Qualitative Sozialforschung/Qualitative Social Research*, volume 11, 2010.

[McB07] Matthew R McBride. The software architect. *Comm. of the ACM*, 50(5):75–81, 2007.

[MGABCM13] Rolando Medellin-Gasque, Katie Atkinson, Trevor Bench-Capon, and Peter McBurney. Strategies for question selection in argumentative dialogues about plans. *Argument & Computation*, 4(2):151–179, 2013.

[MKTS15a] Pradeep K Murukannaiah, Anup K Kalia, Pankaj R Telangy, and Munindar P Singh. Resolving goal conflicts via argumentation-based analysis of competing hypotheses. In *Requirements Engineering Conference (RE), 2015 IEEE 23rd International*, pages 156–165. IEEE, 2015.

[MKTS15b] Pradeep K Murukannaiah, Anup K Kalia, Pankaj R Telangy, and Munindar P Singh. Resolving goal conflicts via argumentation-based analysis of competing hypotheses. In *23rd Int. Requirements Engineering Conf.*, pages 156–165. IEEE, 2015.

[MKvdM12] Jan Mentz, Paula Kotzé, and Alta van der Merwe. A comparison of practitioner and researcher definitions of enterprise architecture using an interpretation method. *Advances in Enterprise Information Systems II*, pages 11–26, 2012.

[Mod09] Sanjay Modgil. Reasoning about preferences in argumentation frameworks. *Artificial Intelligence*, 173(9):901–934, 2009.

[Mue10] Erik T Mueller. *Commonsense reasoning*. Morgan Kaufmann, 2010.

[MvdHvL99] J.-J.Ch. Meyer, W. van der Hoek, and B. van Linder. A logical approach to the dynamics of commitments. *Artificial Intelligence*, 113(1-2):1–40, September 1999.

[MvZG16] Diana Marosin, Marc van Zee, and Sepideh Ghanavati. Formalizing and modeling enterprise architecture (ea) principles with goal-oriented

requirements language (grl). In *Proceedings of the 28th International Conference on Advanced Information System Engineering (CAiSE16)*, June 2016.

[Nie07] Eetu Niemi. Enterprise architecture stakeholders–a holistic view. *AMCIS 2007 Proceedings*, 2007.

[NMD14] Lea Nedomová, Milos Maryska, and Petr Doucek. The enterprise architect role–and its mission in corporate information and communication technology–a czech study. *Journal of Applied Economic Sciences*, pages 88–100, 2014.

[Obj12] Object Management Group. Unified Profile for DoDAF and MODAF (UPDM). Technical Report formal/2012-01-03, OMG, June 2012.

[OP07] M. Op 't Land and Erik Proper. Impact of principles on enterprise engineering. In Hubert sterle, Joachim Schelp, and Robert Winter, editors, *ECIS*, pages 1965–1976. University of St. Gallen, 2007.

[OPW+08] M. Op 't Land, H.A. Proper, M. Waage, J. Cloo, and C. Steghuis. *Enterprise Architecture – Creating Value by Informed Governance*. Enterprise Engineering Series. Springer, Berlin, Germany, 2008.

[Ost89] J.S. Ostroff. *Temporal logic for real-time systems*. Advanced software development series. Research Studies Press, 1989.

[Pat10] Theodore Patkos. *A formal theory for reasoning about action, knowledge and time*. PhD thesis, University of Crete-Heraklion, 2010.

[PdKP13a] Georgios Plataniotis, Sybren de Kinderen, and Henderik A Proper. Capturing decision making strategies in enterprise architecture–a viewpoint. In *Enterprise, business-process and information systems modeling*, pages 339–353. Springer Berlin Heidelberg, 2013.

[PdKP13b] Georgios Plataniotis, Sybren de Kinderen, and Henderik A Proper. Capturing decision making strategies in enterprise architecture–a viewpoint. In *Enterprise, business-process and information systems modeling*, pages 339–353. Springer Berlin Heidelberg, 2013.

[PdKP13c] Georgios Plataniotis, Sybren de Kinderen, and Henderik A Proper. Relating decisions in enterprise architecture using decision design graphs. In *Enterprise Distributed Object Computing Conference (EDOC), 2013 17th IEEE International*, pages 139–146. IEEE, 2013.

[PDKP14a] Georgios Plataniotis, Sybren De Kinderen, and Henderik A Proper. Ea anamnesis: An approach for decision making analysis in enterprise architecture. *International Journal of Information System Modeling and Design (IJISMD)*, 5(3):75–95, 2014.

[PDKP14b] Georgios Plataniotis, Sybren De Kinderen, and Henderik A Proper. Ea anamnesis: An approach for decision making analysis in enterprise architecture. *International Journal of Information System Modeling and Design (IJISMD)*, 5(3):75–95, 2014.

[Pea88] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[PH13] Marian Petre and Andre Van Der Hoek. *Software Designers in Action: A Human-Centric Look at Design Work*. Chapman & Hall/CRC, 1st edition, 2013.

[PO10] H.A. Proper and M. Op 't Land. Lines in the Water – The Line of Reasoning in an Enterprise Engineering Case Study from the Public Sector. In *Proceedings of the Practice-Driven Research on Enterprise Transformation*, pages 193–216, 2010.

[POtL10] H.A. Proper and M Op 't Land. Lines in the Water: The Line of Reasoning in an Enterprise Engineering Case Study from the Public Sector. In *Proceedings of the 2nd Working Conference on Practice-driven Research on Enterprise Transformation (PRET), Delft, The Netherlands*, volume 69 of *Lecture Notes in Business Information Processing*, pages 193–216. Springer, 2010.

[PP03] Nikos Papadakis and Dimitris Plexousakis. Actions with duration and constraints: The ramification problem in temporal databases. *International Journal on Artificial Intelligence Tools*, 12(3):315–353, 2003.

[Raz78] Joseph Raz. *Practical Reasoning*. Oxford University Press, 1978.

[Rey02] M. Reynolds. An axiomatization of full computation tree logic. *Journal of Symbolic Logic*, 66(3):1011–1057, 2002.

[RG91] AS Rao and MP Georgeff. Modeling rational agents within a BDI-architecture. *KR*, 1991.

[Riz16] Rizkiyanto. Better Design Rationale to Improve Software Design Quality. Master's thesis, Utrecht University, the Netherlands, 2016.

[RK96] Alexander Ran and Juha Kuusela. Design decision trees. In *Proceedings of the 8th International Workshop on Software Specification and Design*, IWSSD '96, pages 172–, Washington, DC, USA, 1996. IEEE Computer Society.

[RW91] S. Russell and E. Wefald. *Do the right thing. Studies in limited rationality*. MIT Press, 1991.

[RWR06] J.W. Ross, P. Weill, and D.C. Robertson. *Enterprise architecture as strategy: creating a foundation for business execution*. Harvard Business School Press, Boston, Massachusetts, 2006.

[Sav72] Leonard J Savage. *The foundations of statistics*. Courier Corporation, 1972.

[Sch16] Courtney Schriek. How a Simple Card Game Influences Design Reasoning: a Reflective Method. Master's thesis, Utrecht University, the Netherlands, 2016.

[Sea83] J.R. Searle. *Intentionality. An Essay in the Philosophy of Mind*. Cambridge University Press, 1983.

[SH93] Steven H Spewak and Steven C Hill. *Enterprise architecture planning: developing a blueprint for data, applications and technology*. QED Information Sciences, Inc., 1993.

[SH15] Sofia Sherman and Irit Hadar. Toward defining the role of the software architect: An examination of the soft aspects of this role. In *8th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE 2015)*, 2015.

[Sho09] Y. Shoham. Logical theories of intention and the database perspective. *Journal of Philosophical Logic*, 38:633–647, 2009.

[Sho16] Yoav Shoham. Why knowledge representation matters. *Commun. ACM*, 59(1):47–49, January 2016.

[Sim98] Herbert A Simon. What we know about learning. *Journal of Engineering Education*, 87(4):343, 1998.

[Sin92a] Munindar P Singh. A critical examination of the cohen-levesque theory of intentions. In *Proceedings of the tenth european conference on artificial intelligence (ECAI-92)*, pages 364–368, 1992.

[Sin92b] Munindar P. Singh. A critical examination of the cohen-levesque theory of intentions. In *Proceedings of the 10th European Conference on Artificial Intelligence*, ECAI '92, pages 364–368, New York, NY, USA, 1992. John Wiley & Sons, Inc.

[SLB08] Yoav Shoham and Kevin Leyton-Brown. *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press, 2008.

[SP08] Claudia Steghuis and Erik Proper. Competencies and responsibilities of enterprise architects. In JanL.G. Dietz, Antonia Albani, and Joseph Barjis, editors, *Advances in Enterprise Engineering I*, volume 10, pages 93–107. Springer, 2008.

[SR07] Carolyn Strano and Qamar Rehmani. The role of the enterprise architect. *Information Systems and e-Business Management*, 5(4):379–396, 2007.

[SSS+06] S. J. Buckingham Shum, A. M. Selvin, M. Sierhuis, J. Conklin, C. B. Haley, and B. Nuseibeh. Hypermedia support for argumentation-based rationale. In *Rationale management in software engineering*, pages 111–132. Springer, 2006.

[SSTC12] Steven Shapiro, Sebastian Sardina, John Thangarajah, and Lawrence Cavedon. Revising conflicting intention sets in bdi agents. In *Proceedings of the 11th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2012)*, pages 1081–1088, Valencia, Spain, June 2012.

[SUS14] Sofia Sherman and Naomi Unkelos-Shpigel. What do software architects think they (should) do? In Lazaros Iliadis, Michael Papazoglou, and Klaus Pohl, editors, *Advanced Information Systems Engineering Workshops*, volume 178, pages 219–225. Springer, 2014.

[SW02] M.C. Schut and M. Wooldridge. The control of reasoning in resource-bounded agents. *Knowledge Engineering Review*, 16:215–240, 2002. KER.

[SWP04] Martijn C. Schut, Michael Wooldridge, and Simon Parsons. The theory and practice of intention reconsideration. *J. Exp. Theor. Artif. Intell.*, 16(4):261–293, 2004.

[TA05] Jeff Tyree and Art Akerman. Architecture decisions: demystifying architecture. *IEEE Software*, 22:19–27, 2005.

[TGA13] Dan Tofan, Matthias Galster, and Paris Avgeriou. Difficulty of architectural decisions–a survey with professional architects. In *Software Arch.*, pages 192–199. Springer, 2013.

[THA94] Austin Tate, James Hendler, and James Allen. *Readings in planning*. Morgan Kaufmann Publishers Inc., 1994.

[The00] The Architecture Working Group of the Software Engineering Committee. Recommended Practice for Architectural Description of Software Intensive Systems. Technical Report IEEE P1471:2000, ISO/IEC 42010:2007, Standards Department, IEEE, Piscataway, New Jersey, September 2000.

[The09a] The Open Group. *The Open Group – TOGAF Version 9*. Van Haren Publishing, Zaltbommel, The Netherlands, 2009.

[The09b] The Open Group. *TOGAF Version 9*. Van Haren Publishing, Zaltbommel, The Netherlands, 2009.

[Thi01] Michael Thielscher. The concurrent, continuous fluent calculus. *Studia Logica*, 67(3):315–331, 2001.

[TJH07] Antony Tang, Yan Jin, and Jun Han. A rationale-based architecture model for design traceability and reasoning. *J. Syst. Softw.*, 80(6):918–934, June 2007.

[TMA+12] Pancho Tolchinsky, Sanjay Modgil, Katie Atkinson, Peter McBurney, and Ulises Cortés. Deliberation dialogues for reasoning about safety critical actions. *Autonomous Agents and Multi-Agent Systems*, 25(2):209–259, 2012.

[TP94] Sek-Wah Tan and Judea Pearl. Qualitative decision theory. In Barbara Hayes-Roth and Richard E. Korf, editors, *Proceedings of the 12th National Conference on Artificial Intelligence, Seattle, WA, USA, July 31 - August 4, 1994, Volume 2.*, pages 928–933. AAAI Press / The MIT Press, 1994.

[TSSR11] Toomas Tamm, Peter B Seddon, Graeme Shanks, and Peter Reynolds. How does enterprise architecture add value to organisations. *Comm. of the AIS*, 28(1):141–168, 2011.

[UCI] UCI. Design Prompt: Traffic Signal Simulator. `http://www.ics.uci.edu/design-workshop/files/UCI_Design_Workshop_Prompt.pdf`. Accessed: 2016-12-27.

[Van01] A. Van Lamsweerde. Goal-Oriented Requirements Engineering: A Guided Tour. In *Proc. RE'01: 5th Intl. Symp. Req. Eng.*, pages 249–262, 2001.

[vdHJW07] Wiebe van der Hoek, Wojciech Jamroga, and Michael Wooldridge. Towards a theory of intention revision. *Synthese*, 155(2):265–290, 2007.

[VdHW03] Wiebe Van der Hoek and Michael Wooldridge. Towards a logic of rational agency. *Logic Journal of IGPL*, 11(2):135–159, 2003.

[vdLvZ15] Dirk van der Linden and Marc van Zee. Insights from a Study on Decision Making in Enterprise Architecture. In *PoEM (Short Papers)*, volume 1497 of *CEUR Workshop Proceedings*, pages 21–30, 2015.

[vGvD14] Bas van Gils and Sven van Dijk. *The practice of enterprise architecture: experiences, techniques, and best practices*. BiZZdesign Academy, 2014.

[VHP04] G.E. Veldhuijzen van Zanten, S.J.B.A. Hoppenbrouwers, and H.A. Proper. System Development as a Rational Communicative Process. *Journal of Systemics, Cybernetics and Informatics*, 2(4):47–51, 2004.

[VL01] Axel Van Lamsweerde. Goal-oriented requirements engineering: A guided tour. In *Proc. 5th IEEE Int. Symposium on RE*, pages 249–262, 2001.

[vL08] Axel van Lamsweerde. Requirements engineering: from craft to discipline. In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, pages 238–249. ACM, 2008.

[VNM07] John Von Neumann and Oskar Morgenstern. *Theory of games and economic behavior*. Princeton university press, 2007.

[vZ15] Marc van Zee. Rational Architecture = Architecture from a Recommender Perspective. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2015.

[vZD16] Marc van Zee and Dragan Doder. Agm-style revision of beliefs and intentions. In *Proceedings of the 22nd European Conference on Artificial Intelligence (ECAI'16)*, September 2016.

[vZDDvdT15a] Marc van Zee, Mehdi Dastani, Dragan Doder, and Leendert van der Torre. Consistency Conditions for Beliefs and Intentions. In *Twelfth International Symposium on Logical Formalizations of Commonsense Reasoning*, 2015.

[vZDDvdT15b] Marc van Zee, Dragan Doder, Mehdi Dastani, and Leendert van der Torre. AGM Revision of Beliefs about Action and Time. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2015.

[vZDSvdT14] Marc van Zee, Mehdi Dastani, Yoav Shoham, and Leendert van der Torre. Collective intention revision from a database perspective. In *Collective Intentionality Conference*, July 2014.

[vZG14] Marc van Zee and Sepideh Ghanavati. Capturing Evidence and Rationales with Requirements Engineering and Argumentation-Based Techniques. In *Proc. of the 26th Benelux Conf. on Artificial Intelligence (BNAIC2014)*, November 2014.

[vZI15] Marc van Zee and Thomas Icard. Intention reconsideration as metareasoning. In *Bounded Optimality and Rational Metareasoning NIPS 2015 Workshop*, December 2015.

[vZMBG16] Marc van Zee, Diana Marosin, Floris Bex, and Sepideh Ghanavati. The rationalgrl toolset for goal models and argument diagrams. In *Proceedings of the 6th International Conference on Computational Models of Argument (COMMA'16), Demo abstract*, September 2016.

[vZMGB16] Marc van Zee, Diana Marosin, Sepideh Ghanavati, and Floris Bex. Rationalgrl: A framework for rationalizing goal models using argument diagrams. In *Proceedings of the 35th International Conference on Conceptual Modeling (ER'2016), Short paper*, November 2016.

[vZPvdLM14] Marc van Zee, Georgios Plataniotis, Dirk van der Linden, and Diana Marosin. Formalizing enterprise architecture decision models using integrity constraints. In *CBI 2014*, volume 1, pages 143–150. IEEE, 2014.

[Wal90] Douglas N Walton. *Practical reasoning: goal-driven, knowledge-based, action-guiding argumentation*, volume 2. Rowman & Littlefield, 1990.

[Wij13] Commission Wijffels. Naar een dienstbaar en stabiel bankwezen. http://www.rijksoverheid.nl/bestanden/documenten-en-publicaties/kamerstukken/2013/06/28/rapport-naar-een-dienstbaar-en-stabiel-bankwezen/rapport-naar-een-dienstbaar-en-stabiel-bankwezen.pdf, 2013. Accessed: 2015-04-15.

[Woo00] M.J. Wooldridge. *Reasoning about Rational Agents*. MIT Press, 2000.

[WP98] Michael Wooldridge and Simon Parsons. Intention reconsideration reconsidered. In Jörg P. Müller, Munindar P. Singh, and Anand S. Rao, editors, *ATAL*, volume 1555 of *Lecture Notes in Computer Science*, pages 63–79. Springer, 1998.

[WRM08] Douglas Walton, Christopher Reed, and Fabrizio Macagno. *Argumentation schemes*. Cambridge University Press, 2008.

[WS10] C. Wilson and J. Short. Magic Quadrant for Enterprise Architecture Tools. Technical Report ID Number: G00207406, Gartner, October 2010.

[Yu97a] Eric S. K. Yu. Towards modeling and reasoning support for early-phase requirements engineering. In *Proceedings of the 3rd IEEE International Symposium on Requirements Engineering*, RE '97, pages 226–, Washington, DC, USA, 1997. IEEE Computer Society.

[Yu97b] Eric SK Yu. Towards modelling and reasoning support for early-phase requirements engineering. In *Proc. of the 3rd IEEE Int. Symposium on RE*, pages 226–235, 1997.

[Zac87] J.A. Zachman. A framework for information systems architecture. *IBM Systems Journal*, 26(3), 1987.

[ZKL+09] Olaf Zimmermann, Jana Koehler, Frank Leymann, Ronny Polley, and Nelly Schuster. Managing architectural decision models with dependency relations, integrity constraints, and production rules. *Journal of Systems and Software*, 82(8):1249–1267, 2009.

[ZR93] Shlomo Zilberstein and Stuart J. Russell. Anytime sensing planning and action: A practical model for robot control. In Ruzena Bajcsy, editor, *Proceedings of the 13th International Joint Conference on Artificial Intelligence. Chambéry, France, August 28 - September 3, 1993*, pages 1402–1407. Morgan Kaufmann, 1993.

Publications

[ARGAIP2013] Diego Agustin Ambrossio, Alessio Antonini, Yehia Elrakaiby, Dov Gabbay, and Marc van Zee. Argument revival in annotated argumentation networks. In *Second workshop on Argumentation in Artificial Intelligence and Philosophy: computational and philosophical perspectives (ARGAIP-13)*, December 2013.

[AAAI2013] Natasha Alechina, Tristan Behrens, Mehdi Dastani, Koen Hindriks, Koen Hubner, Fred Jomi, Brian Logan, Hai H. Nguyen, and Marc van Zee. Multi-cycle query caching in agent programming. In *Twenty-Seventh AAAI Conference on Artificial Intelligence (AAAI-13)*, July 2013.

[EMAS2013] Mehdi Dastani and Marc van Zee. Belief caching in 2apl. In *The workshop on Engineering Multi-Agent Systems (EMAS)*, June 2013.

[CAISE2016] Diana Marosin, Marc van Zee, and Sepideh Ghanavati. Formalizing and modeling enterprise architecture (ea) principles with goal-oriented requirements language (grl). In *Proceedings of the 28th International Conference on Advanced Information System Engineering (CAiSE16)*, June 2016.

[SOCINFO2014] Silvano Colombo Tosatto and Marc van Zee. Bridging social network analysis and judgment aggregation. In *Proceedings of the 6th International Conference on Social Informatics.*, December 2014.

[AAMAS2013] Silvano Colombo Tosatto and Marc van Zee. Social network analysis for judgment aggregation. In *13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS2014)*, 2014.

[CBI2014B] Dirk van der Linden and Marc van Zee. On the semantic feature structure of modeling concepts: an empirical study. In *16h IEEE Conference on Business Informatics (CBI)*, May 2014.

[POEM2015] Dirk van der Linden and Marc van Zee. Insights from a study on decision making in enterprise architecture. In *Proceedings of the 8th IFIP WG 8.1 Working Conference on the Practice of Enterprise Modeling (PoEM)*, September 2015.

[IJCAI2015B] Marc van Zee. Rational architecture = architecture from a recommender perspective. In *Proceedings of the International Joint Conference on Artificial Intelligence*, July 2015.

[RE2015] Marc van Zee, Floris Bex, and Sepideh Ghanavati. Rationalization of goal models in grl using formal argumentation. In *Proceedings of the Requirements Engineering Conference 2015 (RE'15), RE: Next! track*, August 2015.

[ECAI2016] Marc van Zee and Dragan Doder. Agm-style revision of beliefs and intentions. In *Proceedings of the 22nd European Conference on Artificial Intelligence (ECAI'16)*, September 2016.

[NMR2016] Marc van Zee and Dragan Doder. Agm-style revision of beliefs and intentions from a database perspective (preliminary version). In *Proceedings of the 16th International Workshop on Non-Monotonic Reasoning (NMR'16)*, April 2016.

[IJCAI2015A] Marc van Zee, Mehdi Dastani, Dragan Doder, and Leendert van der Torre. Agm revision of beliefs about action and time. In *Proceedings of the International Joint Conference on Artificial Intelligence*, July 2015.

[COMMONSENSE2015] Marc van Zee, Mehdi Dastani, Dragan Doder, and Leendert van der Torre. Consistency conditions for beliefs and intentions. In *Twelfth International Symposium on Logical Formalizations of Commonsense Reasoning*, March 2015.

[DARE2014] Marc van Zee, Patrick Doherty, and John-Jules Meyer. Encoding definitional fragments of temporal action logic into logic programming. In *International Workshop on Defeasible and Ampliative Reasoning (DARe)*, June 2014.

[CI2014] Marc van Zee, Mehdi Dastani, Yoav Shoham, and Leendert van der Torre. Collective intention revision from a database perspective. In *Collective Intentionality Conference*, July 2014.

[BNAIC2014] Marc van Zee and Sepideh Ghanavati. Capturing evidence and rationales with requirements engineering and argumentation-based techniques. In *Proceedings of the 26th Benelux Conference on Artificial Intelligence (BNAIC2014)*, November 2014.

[BORM2015] Marc van Zee and Thomas Icard. Intention reconsideration as metareasoning. In *Bounded Optimality and Rational Metareasoning NIPS 2015 Workshop*, December 2015.

[COMMA2016] Marc van Zee, Diana Marosin, Floris Bex, and Sepideh Ghanavati. The rationalgrl toolset for goal models and argument diagrams. In *Proceedings of the 6th International Conference on Computational Models of Argument (COMMA'16), Demo abstract*, September 2016.

[ER2016] Marc van Zee, Diana Marosin, Sepideh Ghanavati, and Floris Bex. Rationalgrl: A framework for rationalizing goal models using argument diagrams. In *Proceedings of the 35th International Conference on Conceptual Modeling (ER'2016)*, November 2016.

[CBI2014A] Marc van Zee, Georgios Plataniotis, Diana Marosin, and Dirk van der Linden. Formalizing enterprise architecture decision models using integrity constraints. In *16h IEEE Conference on Business Informatics (CBI)*, May 2014.

[BNAIC2015] Marc van Zee and Dirk van der Linden. Armed: Argumentation mining and reasoning about enterprise architecture decisions. In *Proceedings of the 27th Benelux Conference on Artificial Intelligence (BNAIC2015)*, November 2015.

[AAAI2014] Pouyan Ziafati, Yehia Elrakaiby, Marc van Zee, Leendert van der Torre, Holger Voos, Mehdi Dastani, and John-Jules Meyer. Reasoning on robot knowledge from discrete and asynchronous observations. In *Knowledge Representation and Reasoning in Robotics*, March 2014.

Curriculum Vitae

2017 – ...	Google Research Zurich, Switzerland.
2013 – 2017	Ph.D. student in Computer Science, University of Luxembourg, Luxembourg.
2011 – 2013	Master of Science in Technical Artificial Intelligence, Utrecht University, The Netherlands.
2005 – 2009	Bachelor of Science in Industrial Design, Eindhoven University of Technology, The Netherlands.
1998 – 2004	Secondary education, Sint Willibrord College Goes, The Netherlands.

Born on October 28, 1985, Vlissingen, The Netherlands.