Exploring the practical benefits of argumentation in multi-agent deliberation

This research was supported by the Netherlands Organisation for Scientific Research (NWO) under project number 612.066.823.

SIKS Dissertation Series No. 2013-16
The research reported in this thesis has been carried out under auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

Cover photo by Gino Santa Maria
Cover graphics by Ricardo Moreira and Dave Tappy
Lay-out and cover design by Eric M. Kok
Printed by Wöhrman Print Service

ISBN 978-90-393-8997-3

Exploring the practical benefits of argumentation in multi-agent deliberation

Een verkenning naar de praktische voordelen van argumentatie in multiagentberaadslaging

(met een samenvatting in het Nederlands)

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de Universiteit Utrecht op gezag van de rector magnificus, prof.dr. G.J. van der Zwaan, ingevolge het besluit van het college voor promoties in het openbaar te verdedigen op woensdag 21 mei 2013 des middags te 14.30 uur

door

Eric Michiel Kok

geboren op 3 april 1982 te Leiderdorp

Promotoren:	Prof. dr. J-J.Ch. Meyer
	Prof. dr. mr. H. Prakken
Co-promotor:	Dr. G.A.W. Vreeswijk

Contents

List of Symbols

$\mathcal{A}$	Agents
$\mathcal{B}$	Knowledge revision function
$\mathcal{C}$	Unique chaining function
$\mathcal{DK}$	Dialogue context
$\mathcal{G}$	Move generation function
$\mathcal{H}$	Attitude assignment function
$\mathcal{I}$	Attack point identification function
$\mathcal{L}$	Set of locutions
$\mathcal{O}$	Dialogue outcome
$\mathcal{P}$	Dialogue protocol
$\mathcal{R}$	Roles
$\mathcal{SK}$	Scenario generation context
$\mathcal{S}$	Deliberation strategy
$\mathcal{U}$	Combined goal utility function
$\mathcal{X}$	Experiment run
agent	Returns the agent with some identifier
build	Build strategy
cf	Contrariness function
conceal	Degree of concealment metric
conc	Returns the top conclusion of an argument
content	Returns the move content
destroy	Destroy strategy
dsub	Returns defeasible sub-arguments of an argument
id	Returns the move identifier
indifferent	Indifferent strategy
movecount	Move count metric

player	Returns the agent that played the move
prem	Returns all premises used in an argument
proposal	Returns the proposal move of this tree
relevance	Degree of relevance metric
rules	Returns all applied rules of an argument
sub	Returns sub-arguments of an argument
target	Returns the move's targeted move
utility	Combined utility metric
ϱ	Typical notation for a rule name
A	Typical notation for an argument
a	Typical notation for an agent
AF	Dung-style argumentation framework
AR	All possible rules
AS	Argumentation system
AT	Argumentation theory
B	Typical notation for a set of beliefs
C	Rule chain
D	All dialogues
d	Typical notation for a dialogue
G	Typical notation for a set of goals
g	Typical notation for an element (goal) in L_g
H	All strategy attitudes
K	Knowledge base
L	Logical language
L_b	Beliefs language
L_c	Communication language
L_g	Goals language
L_o	Options language
L_t	Topic language
M	All moves
m	Typical notation for a move
O	Typical notation for a set of options
o	Typical notation for an element (option) in L_o
P	Typical notation for a proposal tree
p	Typical notation for an element (belief) in L_b
Q	Set of proposals
R	Set of strict and defeasible rules

r	Typical notation for a role
S	Relation between locutions
T	Turntaking function
U	Utility
X	Termination function

Chapter 1

Introduction

With ubiquitous internet access, recommender engines, localization and social networks, it seems that settling on a restaurant to eat with friends should be quick and straightforward. In reality though, at least in my experience, it takes a discussion on food styles, atmosphere, prices, the variety of beers they serve or even the source of the meat to settle on a single place. As it turns out, even a wealth of information and clear personal preferences do not make these ever-present collaborative tasks trivial. Instead, we use the power of persuasion, inquiry, means-end reasoning and other forms of argument to come to a decision. It should therefore not come as a surprise that argumentation is a topic that has been studied in many fields of science, from philosophy and linguistics to law and computer science.

It is not only for scientific curiosity, though, that argumentation has received such wide interest, in its many forms and through many alternative views. Rather, argumentation has potential to improve the expressibility of scientific models and to better the applications built on these models. It is in the field of artificial intelligence that so many potential uses of argumentation come together, as the software systems in AI are frequently modelled after human ways already, such as communication, reasoning and planning. This begs the question how argumentation can help in the field of artificial intelligence, or rather, what the benefits are of using argumentation in models for AI.

1.1 Argumentation in AI

The study of argumentation has roots in a wide array of scientific fields and can be traced back to the theory of rhetoric arguments of Aristotle. In the last century, the theory of argumentation was developed in various fields, such as law, philosophy, linguistics and, eventually, artificial intelligence. Interestingly, independent of the area of scientific interest, argumentation is used in several very distinct ways. First, argumentation can relate to a style of dialogue in which participants try to find justification for a claim or alternative. Second, argumentation concerns the evaluation of claims, based on some premises, to derive justified conclusions. Arguably there is a third distinct function of argumentation, which is the reasoning process of an (artificial) agent through construction and validation of claims to infer what to do.

Within artificial intelligence the most interest has been on the development of logics for argumentation, as a non-monotonic way of deriving justified conclusions from a body of knowledge (Pollock, 1987). Using world beliefs and assumptions as premises and combining them with inference rules, it is possible to construct arguments that can be compared by considering how they attack and defeat each other. Most modern argumentation logics are (partial) implementations or extensions of the abstract framework for argumentation-based inference of Dung (1995). Chapter 2 will introduce the concepts behind argumentation logics and how they can be used in AI.

One of the most influential areas of artificial intelligence is that of agent technology, where software systems are designed as autonomous and pro-active entities. Commonly, agents are modelled with a BDI architecture, where the core of an agent consists of a reasoning mechanism that derives from beliefs and goals a plan to act in the world (Rao and Georgeff, 1991). Historically the reasoning is performed using logics from classic first-order logic to temporal and modal logics. Non-monotonic logics, such as argumentation logics, have not as much been applied to agent reasoning.

Argumentation as a type of dialogue has found a lot of attention in the field of multi-agent systems. As communication is such a vital part of multi-agent design, many models have been proposed in the literature that employ argumentation in the communicative design of agents. They are grounded in the theory of speech acts (Searle, 1976), where communication is modelled as actions performed by an agent, as well as in the idea of dialogue games (Carlson, 1983). Dialogue frameworks have been proposed for multi-agent persuasion,

negotiation, deliberation and other styles of argumentation-based dialogues.

1.1.1 Potential benefits of argumentation

With argumentation present in all its forms in multi-agent systems, its potential benefits can also be found in different parts of agent design. Indeed, argumentation has been proposed to be beneficial for defeasible logics (Pollock, 1987; Prakken and Vreeswijk, 2002) and internal reasoning to form plans (Parsons et al., 1998). Nevertheless, the strongest claims that using argumentation is beneficial for agents come from the introduction of argumentation-based models for multi-agent dialogue. Here, argumentation is not only proposed as a way of modelling and understanding the design, but argumentation is claimed to make agent interactions more efficient and more effective (Rahwan, 2005).

Many studies into argumentation-based dialogue systems leave the proclaimed benefits unspecified, but there are still some concrete reasons found in the literature on the source of these benefits. First, argumentation can focus a dialogue. If an argument cannot be contested, there is no reason to study the particular case further (Rahwan et al., 2004). If everybody agrees that a certain restaurant is too expensive, then clearly nobody will further ask for or provide reasons for going here, shortening the discussion. Second, arguments can provide a more general insight in the situation. If someone refused to eat at a steak house because he or she is vegetarian, this knowledge can be used to make better proposals (Sierra et al., 1997), such as proposing an Indian restaurant with lots of vegetarian dishes. This potentially increases the quality of the overall outcome. Third, arguments explicitly couple premises and conclusion, by which a statement can be checked on whether it brings insight into the discussion. It can therefore serve as a way of checking for unnecessary statements (Prakken, 2005). Finally, by providing arguments, agents do not yet have to reveal all their information (Dijkstra et al., 2005). Only when an argument is questioned further information needs to be supplied. Withholding information may be beneficial for the efficiency of a dialogue. If nobody doubts that some restaurant serves the best steaks in town, there is no reason to explain why this would be the case.

In conclusion, claims of benefits of argumentation for agents are most abundant in relation to dialogues. Interactions are promised to be more efficient and more effective. Intriguingly, these claims have rarely been validated. Only a few examples exist where the aforementioned benefits are studied. Granted, the

introduction of a lot of frameworks for argumentation-based dialogues has been accompanied by proofs of various formal properties, such as that argumentation-based dialogues will perform at least as good as without argumentation (Rahwan et al., 2007) or that claims derived from debate align with the underlying argumentation system (Prakken, 2005). However, only few studies have looked at the practical benefits that argumentation is supposed to bring (Pasquier et al., 2010; Karunatillake et al., 2009).

1.1.2 Deliberation

This thesis will present how the practical benefits of argumentation can be studied in deliberation style dialogues. Before continuing, a short note is justified on the English word *deliberation*. As it will appear in this thesis, starting with its title, it pertains to the dialogue type where agents mutually have to decide on some course of action. A decision has to be made and as such the agents will use their knowledge to present alternatives, provide arguments why the proposal of an alternative is justified, discuss the underlying premises, etc. Contrary to the dialogue style, the word deliberation has also been used for the process of reasoning from means to end. In this way, deliberation is an internal process to consider several alternatives. Although this style of reasoning is used in this thesis, as part of the agent behaviour, the word deliberation will be exclusively used for the dialogue type.

1.1.3 Running deliberation example

Throughout this thesis a running example is used to illustrate how multi-agent deliberation can be modelled and measured. Several agents will need to mutually decide on a place for dinner. Every agent has its own knowledge of alternatives, relevant background information as well as personal goals it would like to achieve. The examples of this thesis will model and analyse the proposal of two restaurants and the subsequent discussion thereof by two agents on why they should (not) be selected as final outcome.

a	I suggest we go to the bistro.
b	Why should we go there? We could go to the pizzeria instead.
a	In the bistro we will enjoy our dinner, for we will get served the best steak. So I prefer to go to the bistro.
b	Why would we get the best steak? The pizzeria and bistro are equally preferable to me.
a	These are the best steaks, since they are made from wagyu cattle and therefore wagyu steaks, which are considered the best. Moreover, in the bistro we will enjoy our dinner, for we can drink tasty beer.
b	I admit that the steaks are made from wagyu cattle. But in this case the steaks are not the best, since they are improperly handled.

Agent a seemingly wants to go to the bistro, while agent b originally knew about a pizzeria. Throughout the dialogue they ask for and provide arguments why these options should be considered. On the one hand there is an argument that connects the mutual goal of enjoying dinner to the proposed action to go to the bistro. In contrast to that, the argument that steaks from wagyu cattle are considered the best is purely based on background knowledge. In any case, the agents form their arguments, can reason with them internally and then decide to submit them in the dialogue, in order to win over the other agents to their own preferred place to eat.

1.2 Research questions

As discussed above, there are many potential benefits of argumentation for multi-agent systems. However, it still has to be shown which benefits actually materialize and under which circumstances. The central research question for this thesis can therefore be asked:

Research question What are the benefits of using argumentation in multi-agent deliberation dialogues?

The focus of the research is to uncover the benefits of argumentation for the whole multi-agent system. It might very well be possible that there is also a potential gain for individual agents to use argumentation. Agents, especially

those that are BDI-based, typically are at least partially self-interested. It is therefore very interesting to see if individual agents can have uses for argumentation as well, either in dialogue, as reasoning method to construct plans or to validate knowledge. Unfortunately, the scope of this thesis project did not allow for a thorough investigation into benefits for individual agents. Instead, the benefits are studied for the multi-agent system as a whole.

1.2.1 Desirable properties

As the research into benefits of argumentation in multi-agent dialogues is limited, there is a proportional lack of development on how to test for these benefits. This starts with the question on what exactly are these claimed benefits. Several informal examples have been presented above, but it is not yet formalized which properties of multi-agent dialogues are in fact desirable.

When desirable properties have been established, it is still to be modelled how they can be measured. Given a fully played out deliberation dialogue, how can we measure to what degree the desirable properties hold? In order to compare dialogues, the measurements need to be quantifiable. Hence, for each desirable property, a metric needs to be defined that explains the performance of a dialogue through a numeric measurement. This leads to the research subquestion:

Research sub-question 1 Which properties of deliberation dialogues are desirable and how can we measure these properties?

In the above example, the agents proposed two options, explicitly stated their preferences and gave various arguments. Say that they now settle on going to the bistro, then how did this dialogue perform with regards to the desirable properties of this dialogue? On a first impression the dialogue seems fairly short and concise. Choosing the bistro also looks like a good choice, as they did not state to prefer some other option. Still, to compare this dialogue to another these intuitions need to be formalized and measurable.

1.2.2 Deliberation framework

It is straightforward to find benefits of argumentation by comparing agents use argumentation to those that do not. Unfortunately, neither the modelling of argumentation-enabled deliberation dialogues nor the method for comparing

deliberating agents is standardized in the literature. Part of this thesis will therefore concern the development of a suitable framework which models deliberation style dialogues and allows for comparison of different deliberating agents:

Research sub-question 2 What is a suitable framework for specifying deliberation dialogues and agent behaviours and how can the performance of deliberating agents be compared?

Coming up with a performance indicator for a single dialogue does not yet help to test if arguing was beneficial. To know if arguing was beneficial for the agents in the running example, they need to be compared to agents that do not use argumentation. Furthermore, the dialogue with a pizzeria and a bistro is not necessarily representative for any deliberation situation. Instead, the framework needs to support an analysis on a wide array of different deliberation situations to draw general conclusions on the benefits of using argumentation.

1.2.3 Argumentation-based reasoning

To study the performance of dialogues means to study the performance of agents, for the simple fact that dialogues are merely the resulting structures of agent behaviour. Or rather, to compare dialogues with argumentation to dialogues without it, an expressive model is required that allows for the specification of arguing and non-arguing agents.

As already hinted upon, it is not only in the dialogue that arguments are used, but agents reason with them internally as well. The agent behaviour framework has to cater for this type of reasoning. It allows agents to use personal goals and background knowledge to determine the preferred options and connects these to the generation of dialogue moves.

Research sub-question 3 How can agents use arguments to reason about options and generate appropriate moves?

An expressive model of agent behaviour allows for the comparison of specific arguing and non-arguing elements in agents to see their direct effect on the benefits of argumentations. If agent a in the running example wants to further defend the case for going to the bistro, it has several options. Perhaps it has additional knowledge on the handling of the wagyu steaks to debunk agent

b's argument on improper handling of the meat. Alternatively it might have additional reasons why the bistro will satisfy their mutual goal, such as that the atmosphere in the place is great. In any way, the argumentation internal to an agent will directly influence the course and therefore the performance of a dialogue.

1.3 Thesis overview

Above, a case was made for the validation of claims that argumentation in dialogue is beneficial for a multi-agent system. To answer the questions that come with this target, this thesis will introduce a platform for deliberation dialogues and use that platform to compare arguing and non-arguing agents. It builds upon the state of the art argumentation logics and models for argumentation-based dialogues, which are discussed in Chapter 2.

In Chapter 3 a framework for deliberation dialogues will be proposed, which models how agents can interact, how they can be restricted and how eventually a decision can be made by selecting one of the proposed options in the dialogue. Chapter 4 introduces the idea of deliberation scenarios, which define the situation that agents are confronted with before engaging in a dialogue. They form a vital part of the method used in this thesis to compare arguing and non-arguing agent behaviours. This is achieved by generating situations that reflect typical deliberation problems in the knowledge of individual agents. This generation method is validated in Chapter 5, which leads to the possibility of experimentally comparing arguing and non-arguing agents.

In Chapter 6 the behaviour of agents is modelled, split up in the different steps that agents use to reason about a scenario and to generate moves accordingly. Several variations of arguing and non-arguing agents are proposed. Chapter 7 settles which are the desirable properties of deliberation dialogues and, importantly, how to measure these properties. This allows for a performance comparison of the earlier introduced agents in Chapter 8. Here, the intuitions behind the benefits of using argumentation are tested and it is shown why these benefits (do not) present themselves. Finally, Chapter 9 discusses how the method and platform developed in this thesis can be used for future research in argumentation dialogues.

Chapter 2

Background

The modern research field of computational argumentation is, roughly speaking, concerned with two distinct types of arguing: the activity of discussing some topic to find justification for a claim or action, and the evaluation of arguments for a claim or action to derive a justified conclusion. The use of arguments in these is different and consequently the research into models for these is distinct. Being in discourse concerns multiple agents, or participants, each trying to convince the others or at least searching for a mutual agreement on a claim or proposed action. On the other hand, the evaluation of arguments is a process of reasoning with knowledge to derive a justified conclusion, much like how a classic logic is used to find if a certain claim follows from some body of knowledge.

The goal of this thesis is to study the use of argumentation in dialogues, but this can not be seen in isolation from argumentation-based inference. It makes use of an argumentation system that can evaluate arguments. We will thus need both uses of argumentation. Firstly, Section 2.1 introduces the argumentation system that is used in this thesis when arguments are constructed or evaluated. Secondly, Section 2.2 describes the theory behind multi-agent communication and the role of argumentation-based dialogue systems.

2.1 Argumentation

The primary application of arguments in computer science is that of an argument as reasoning construct. Arguments are structures that allow to draw a conclusion based on a set of premises, which are said to provide the justification for the argument's conclusion. While it is natural for a human to use arguments

in reasoning, for a computer to do so, a suitable framework is needed in which arguments can be expressed and formally evaluated.

Arguments often pertain to the justification of some belief. For example, 'these are the best steaks, since wagyu steaks are typically considered the best' is a purely epistemic argument. On the other hand, there are arguments that combine practical reasoning about actions with the reasoning about beliefs. For example, 'we will enjoy our dinner, for we will get served the best steak, as we will go to the bistro' connects an epistemic belief of the bistro serving the best steaks with the proposal for action to go to the bistro and the goal to enjoy dinner. Section 2.1.3 discusses the modelling of practical reasoning.

Whether an argument is purely epistemic or contains aspects of practical reasoning, inferences are made to connect the premises to the conclusion of the argument. Some of these inferences are strict, because their conclusion holds without exception if the premises hold. Argumentation systems do not allow these inferences to be disputed. For example, it makes no sense to question the inference in 'these are wagyu steaks, since they are steaks and they are made from wagyu cattle'. Additionally, arguments can contain inferences that typically hold, but to which there may be exceptions. For example, 'these are the best steaks, since wagyu steaks are typically considered the best' as a rule might be true, but many exceptions can be thought of, such as that the beef was not sufficiently aged or that it was not properly handled. Argumentation logics allow for defeasible inferences to be questioned, by providing other arguments that attack the original argument in a specific manner. Attack between arguments is formalized in Section 2.1.2.

In our approach, both epistemic and practical arguments are derived from a single knowledge base, contained in a single argumentation system. To find an acceptable claim, attack between arguments is evaluated through three ways in which arguments can conflict, as discussed in Section 2.1.2. However, when it comes to stating arguments in a dialogue, evaluating arguments as a single argumentation system is no longer appropriate. Individual agents have different background knowledge and different preferences and hence have their own distinct argumentation system. To still find justification for a claim or proposal for action the agents will use an argumentation-based dialogue model, as is outlined in Section 2.2.1. Still, individual agents can reason defeasibly with their knowledge using their own argumentation systems. As is shown from Chapter 6 onwards, this allows them, for example, to reason whether to agree with a statement in a dialogue and construct a counterargument where applicable.

2.1.1 Argumentation framework

The base argumentation framework used in this thesis is that of Prakken (2010), called ASPIC+. This framework for argumentation with structured arguments is a generalization and extension of the framework that was developed in the European ASPIC project (Amgoud et al., 2006). The framework allows for the construction and evaluation of structured arguments with strict and defeasible rules, while being general enough to support both epistemic and practical reasoning. One interesting feature is the distinction between three different ways of attack between arguments, which, for example, is utilized in the scenario generation of Chapter 4.

Several features of the original framework are not needed for the purpose of this thesis, or are only needed in a limited sense. This does not mean that these features are not compatible with the work presented in the next chapters, but rather that they are omitted or a specific instantiation is used. The resulting framework will be called ASPIC−. Omissions or restrictions of features are discussed at their appropriate location throughout the introduction of the framework.

The central notion behind the ASPIC− framework is that of an argumentation system. It works as a classic proof system, but makes a distinction between strict and defeasible inference rules and defines a preference order on the set of defeasible rules.

Definition 2.1 Let AR be the set of all possible rules in an ASPIC− framework. An *argumentation system* is a tuple $AS = \langle L, R, \mathsf{cf}, \leq \rangle$ such that

- L is a logical language, closed under classical negation,
- $R_s \cup R_d$ is a set of strict inference rules and $R_d \cup R_d$ a set of defeasible inference rules, such that $R = R_s \cup R_d$ and $R_s \cap R_d = \emptyset$,
- cf is a contrariness function $\mathsf{cf} : L \longrightarrow Pow(L)$ such that for every $p \in L$ it holds that $\mathsf{cf}(p) = \{\neg p\}$ and $\mathsf{cf}(\neg p) = \{p\}$,
- $\leq$ is a partial order on R_d.

There are few restrictions on the used logical language. Typical elements in the logical language will be written as p, q, g, o, etc. There is, however, the requirement that the language is closed under classical negation. Prakken does not make this requirement, instead allowing for any type of contrariness relation.

However, for the purpose of this study the classic logical negation suffices. The contrariness function is therefore implemented in a straightforward fashion.

Inference rules in an argumentation system are either strict or defeasible.

Definition 2.2 Given the formulas $p_1, \ldots, p_n, p \in L$

- a *strict inference* rule is of the form $p_1, \ldots, p_n \xrightarrow{\varrho} p$
- a *defeasible inference* rule is of the form $p_1, \ldots, p_n \overset{\varrho}{\Rightarrow} p$

The elements $p_1, \ldots, p_n$ are called the antecedents of the rule and p the consequent. The name of the rule is denoted ϱ, which for the purpose of convenience is added to the original rule notation. Rule names are themselves formulas in the object language and as such can be used in rules. As object they represent the statement that a certain rule is applicable. The negation of a rule name object then means that the application of a certain rule is not applicable. This makes it possible to write rules that state an exception to some defeasible inference. For example, there might be a defeasible rule wagyuSteak $\overset{\varrho_1}{\Rightarrow}$ bestSteak to which another rule states the exception improperlyHandled $\overset{\varrho_2}{\Rightarrow} \neg\varrho_1$.

The justification for a conclusion is based on beliefs that are taken from a body of background knowledge called the knowledge base.

Definition 2.3 Given an argumentation system $AS = \langle L, R, \leq, \mathsf{cf} \rangle$, a *knowledge base* is a set K such that $K \subseteq L$.

Knowledge consists of formulas expressed in the system's logical language. This thesis will not only allow expressions on beliefs, as represented by propositional logic formulas. It also includes goals and proposals for action as elements of the logical language and thus in the knowledge from which arguments are constructed. Chapter 2.1.3 discusses their relationship.

Prakken's original definition allows for four different kinds of premises in the knowledge base. Throughout this thesis only so-called ordinairy premises are used, which can be contested by counterarguments (unlike necessary axioms), need an acceptable counterargument to be contested successfully (unlike assumptions) and do not need a backup argument to accept the conclusion (unlike issues). Consequently, the partition of the knowledge base into different premise types is omitted from the framework as used in this thesis.

Arguments can now be constructed using the knowledge base and the set of inference rules in the system. They are structured as inference trees of rule

application, an approach adopted from Vreeswijk (1997). Note that arguments can be compound, where the conclusion of one argument is directly used as premise in another.

Several functions are defined on arguments. conc returns the (top) conclusion of an argument. prem returns the premises used in the arguments, being formulas in K. The function rules returns the set of all strict and defeasible rules that were used in the argument. Finally, the dsub function is used to return all the sub arguments that used a defeasible inference rule.

Definition 2.4 Given an argumentation system $AS = \langle L, R, \leq, \mathsf{cf} \rangle$, an *argument* A in AS, constructed on the basis of K, is either

- p if $p \in K$, where
  - $\mathsf{conc}(A) = p$
  - $\mathsf{prem}(A) = \{p\}$
  - $\mathsf{rules}(A) = \emptyset$
  - $\mathsf{dsub}(A) = \emptyset$
- $A_1, \ldots, A_n \xrightarrow{\varrho} p$ if $A_1, \ldots, A_n$ are arguments such that there is a strict rule $\mathsf{conc}(A_1), \ldots, \mathsf{conc}(A_n) \xrightarrow{\varrho} p \in R_s$, where
  - $\mathsf{conc}(A) = p$
  - $\mathsf{prem}(A) = \mathsf{prem}(A_1) \cup \ldots \cup \mathsf{prem}(A_n)$
  - $\mathsf{rules}(A) = \{\mathsf{conc}(A_1), \ldots, \mathsf{conc}(A_n) \xrightarrow{\varrho} p\} \cup \mathsf{rules}(A_1) \cup \ldots \cup \mathsf{rules}(A_n)$
  - $\mathsf{dsub}(A) = \mathsf{dsub}(A_1) \cup \ldots \cup \mathsf{dsub}(A_n)$
- $A_1, \ldots, A_n \overset{\varrho}{\Rightarrow} p$ if $A_1, \ldots, A_n$ are arguments such that there is a defeasible rule $\mathsf{conc}(A_1), \ldots, \mathsf{conc}(A_n) \overset{\varrho}{\Rightarrow} p \in R_d$, where
  - $\mathsf{conc}(A) = p$
  - $\mathsf{prem}(A) = \mathsf{prem}(A_1) \cup \ldots \cup \mathsf{prem}(A_n)$
  - $\mathsf{rules}(A) = \{\mathsf{conc}(A_1), \ldots, \mathsf{conc}(A_n) \overset{\varrho}{\Rightarrow} p\} \cup \mathsf{rules}(A_1) \cup \ldots \cup \mathsf{rules}(A_n)$
  - $\mathsf{dsub}(A) = A \cup \mathsf{dsub}(A_1) \cup \ldots \cup \mathsf{dsub}(A_n)$

The construction of arguments is an iterative process. The formulas in the knowledge base can first be used to construct atomic premises, from which their conclusions can be used to make compound arguments and so to eventually arrive at the desired conclusion.

Example 2.1 Consider, for instance, an argumentation system with

- $K = \{\text{steak}; \text{wagyuCattle}; \text{improperlyHandled}; \neg\text{improperlyHandled}\}$
- $R_d = \{\text{wagyuSteak} \overset{\varrho_1}{\Rightarrow} \text{bestSteak}; \text{improperlyHandled} \overset{\varrho_2}{\Rightarrow} \neg\varrho_1\}$
- $R_s = \{\text{steak}, \text{wagyuCattle} \overset{\varrho_3}{\longrightarrow} \text{wagyuSteak}\}$

Several arguments can be constructed with this knowledge base, which can graphically be represented. Single lined inferences are strict while double lined inferences are defeasible.

$$A' = \dfrac{\dfrac{\text{steak} \qquad \text{wagyuCattle}}{\text{wagyuSteak}}\,\varrho_3}{\text{bestSteak}}\,\varrho_1$$

$$B = \dfrac{\text{improperlyHandled}}{\neg\varrho_1}\,\varrho_2$$

$$C = \neg\text{improperlyHandled}$$

Note that these are not all possible arguments that can be constructed using K. For example, the premises of A and B as well as sub-argument A' can themselves be derived as separate arguments. Given example arguments A, A', B and C, the argument functions return

$\mathsf{conc}(A) = \text{bestSteak}$
$\mathsf{prem}(A) = \{\text{steak}, \text{wagyuCattle}\}$
$\mathsf{rules}(A) =$
$\{\text{steak}, \text{wagyuCattle} \xrightarrow{\varrho_3} \text{wagyuSteak};$
$\text{wagyuSteak} \overset{\varrho_1}{\Longrightarrow} \text{bestSteak}\}$
$\mathsf{dsub}(A) = \{A\}$

$\mathsf{conc}(A') = \text{wagyuSteak}$
$\mathsf{prem}(A') = \{\text{steak}, \text{wagyuCattle}\}$
$\mathsf{rules}(A') =$
$\{\text{steak}, \text{wagyuCattle} \xrightarrow{\varrho_3} \text{wagyuSteak}\}$
$\mathsf{dsub}(A) = \emptyset$

$\mathsf{conc}(B) = \neg\varrho_1$
$\mathsf{prem}(B) = \{\text{improperlyHandled}\}$
$\mathsf{rules}(B) =$
$\{\text{improperlyHandled} \overset{\varrho_2}{\Longrightarrow} \neg\varrho_1\}$
$\mathsf{dsub}(B) = \{B\}$

$\mathsf{conc}(C) = \neg\text{improperlyHandled}$
$\mathsf{prem}(C) = \{\neg\text{improperlyHandled}\}$
$\mathsf{rules}(C) = \emptyset$
$\mathsf{dsub}(C) = \emptyset$

Note that A' is not part of $\mathsf{dsub}(A)$ or $\mathsf{dsub}(A')$ since it uses a strict inference rule.

A shorthand notation is available to indicate that from a certain set of background knowledge we can construct an argument with a certain conclusion.

Definition 2.5 Given an argument A that can be constructed from a set $S \subseteq L$ it holds that $S \mid\!\sim p$ iff $\mathsf{conc}(A) = p$.

The argumentation system is left implicit in this notation. Also, Prakken originally differentiated between $\vdash$ and $\mid\!\sim$ as strict and defeasible arguments respectively, while the above definition uses $\mid\!\sim$ for any argument, as the distinction with this notation is not used further in this thesis. The above defined example argument A with $\mathsf{conc}(A) = \text{bestSteak}$ is constructed using the knowledge base K and it is therefore possible to write $K \mid\!\sim \text{bestSteak}$.

Finally, the combination of an argumentation system, a knowledge base and some preference ordering over all arguments forms a so called argumentation theory.

Definition 2.6 An *argumentation theory* is a tuple $AT = \langle AS, K, \preceq \rangle$, where AS is an argumentation theory, K is a knowledge base and $\preceq$ is a partial preference order on the set of all arguments that can be constructed from K in AS.

For the purpose of this thesis the preference over arguments is not made explicit and unless otherwise noted the arguments are assumed to be of equal strength.

2.1.2 Attack and defeat

Arguments are individually advocating some conclusion, but other arguments can exist in the argumentation theory that conflict with the argument. Indeed the power of an argumentation framework is that it can evaluate the justification for a claim considering a set of arguments that conflict in a specific manner.

Conflict is modelled by providing an attack relation between constructed arguments in an argumentation theory. The framework of Prakken defines three ways in which arguments can attack each other: rebutting, undercutting and undermining. The attack between two arguments models the intuitive notion of conflict between two arguments, but this does not yet entail that the conclusion of the attacked argument is no longer justified. Rather, it represents that the attacked argument is contested. An additional step, which uses argument preferences, is needed to decide which arguments are successfully contested, which is called defeated.

The first type of attack between arguments is rebuttal. A defeasible argument is rebutted if the argument is attacked on its conclusion or one of the conclusions of its defeasible sub arguments. Hence, it states that the conclusion (to one of the sub arguments) does not hold since the opposite is true.

Definition 2.7 Argument A *rebuts* argument B on B' iff $\mathsf{conc}(A) = \mathsf{cf}(\mathsf{conc}(B'))$ such that $B' \in \mathsf{dsub}(B)$.

The second attack type is undercutting. This is the form of attack where one argument states an exception to some defeasible inference in the attacked argument. It is modelled as an argument that has as conclusion the contrary of a rule that was used in another argument. The definition makes use of the newly introduced rule naming convention as introduced above.

Definition 2.8 Argument A *undercuts* argument B iff $\mathsf{conc}(A) = \mathsf{cf}(\varrho)$ such that $\mathsf{conc}(A_1), \ldots, \mathsf{conc}(A_n) \overset{\varrho}{\Rightarrow} p \in \mathsf{rules}(B)$.

The final way to attack an argument is to undermine it. Undermining an argument is done by attacking one of the premises of the attacked argument. Any argument premise can be attacked in this way as, contrary to Prakken's necessary premises, only attackable ordinary premises are allowed in the knowledge base.

Definition 2.9 Argument A *undermines* argument B on p iff $\mathsf{conc}(A) = \mathsf{cf}(p)$ such that $p \in \mathsf{prem}(B)$.

Example 2.2 Recall the example arguments A, B and C as introduced in 2.1.

$$A' = \frac{\text{steak} \qquad \text{wagyuCattle}}{A = \dfrac{\text{wagyuSteak}}{\text{bestSteak}}\ \varrho_1}\ \varrho_3$$

$$B = \frac{\text{improperlyHandled}}{\neg\varrho_1}\ \varrho_2$$

$$C = \neg\text{improperlyHandled}$$

Argument A is undercut by argument B (on A) since the conclusion $\neg\varrho_1$ is contrary to the defeasible rule application of ϱ_1. Argument C undermines B, since C''s conclusion is contrary to the premise improperlyHandled of argument B. That same premise improperlyHandled can itself be instantiated as an argument B' = improperlyHandled, which rebuts C, while C would in turn rebut B' since their conclusions are contrary. Note that it is impossible to undercut (on) A', since the application of rule ϱ_3 is a strict inference, to which attacks are not allowed.

After the attack relations have been established, it is possible to determine defeat between arguments. This is done by considering the preferences that were defined in the argumentation theory, as introduced in Definition 2.6. Prakken uses some subtleties in the way the three different types of attack result in defeat. In particular, a difference between contrary or contradictory conflict between arguments is used, as made explicit by the contrariness function cf. However, this thesis strictly uses the contrary variant that is the classical negation. Moreover, special care has to be taken for arguments that contain issues (premises that require backup), but as this thesis allows only ordinary premises, this complexity does not apply here. For these reasons it is possible to directly specify defeat from the attack relation and preference ordering over arguments.

Definition 2.10 Argument A defeats argument B iff one of the following holds:

- A undercuts B,
- if A rebuts B on B' and $A \not\prec B'$, or
- if A undermines B on p and $A \not\prec p$.

It must be noted that throughout the next chapters, although the framework allows for it, the preference ordering over arguments remains mostly unused, or rather, no explicit preferences will be given to individual arguments. On the other hand, adopting a more complex way of specifying these preferences is still supported in the models of this thesis.

At this point it has become possible to evaluate which arguments, and consequently their conclusion, can be accepted in the given argumentation theory. A large body or research has studied how to derive acceptable conclusions given a set of arguments and their defeat relations. Most notably is the work done by Dung (1995) who was able to generalize argumentation logics by considering arguments as abstract nodes in a graph, where the nodes are connected in a directed fashion using the defeat relationship. Using the notion of conflict free sets and an array of acceptability semantics, there are various more or less intuitive methods to select the final justified claims, if any.

Definition 2.11 An *argumentation framework* is defined by a tuple $AF = \langle Args, Defeat \rangle$ where *Args* is the set of arguments described by Definition 2.6 and *Defeat* is the binary relation on *Args* described by Definition 2.10.

Before several (acceptability) properties are defined on argumentation frameworks, the notions of a conflict free set and a defending set are introduced. Dung defines that a set of arguments is conflict free if it does not contain an argument that defeats an argument in the set. A set is said to defend an argument if it defeats all arguments that defeat it.

Definition 2.12 Given an argumentation framework $AF = \langle Args, Defeat \rangle$ with some set of arguments $S \subseteq Args$, then

- S is *conflict free* iff there is no $A, B \in S$ such that $(A, B) \in Defeat$
- S defends an argument $A \in Args$ iff for each argument $B \in Args$ such that $(B, A) \in Defeat$ there exists an argument $C \in S$ such that $(C, A) \in Defeat$.

Various acceptability properties can now be defined on some set of arguments in a Dung-style argumentation framework. Only those relevant to this thesis are listed. Many more variations on acceptability of arguments given certain extensions exist. See (Baroni et al., 2011) for an up to date account on extensions and acceptability semantics in argumentation frameworks.

Definition 2.13 Given a conflict free set of arguments S and a function $\mathcal{F} : Pow(Args) \longrightarrow Pow(Args)$ such that $\mathcal{F}(S) = \{A | S \text{ defends } A\}$, then

- S is *admissible* iff $S \subseteq \mathcal{F}(S)$,
- S is a *complete extension* iff $S = \mathcal{F}(S)$,
- S is a *grounded extension* iff it is the smallest (w. r. t. set inclusion) complete extension,
- S is a *preferred extension* iff it is a maximal (w. r. t. set inclusion) complete extension,
- S is a *stable extension* iff it is a preferred extension that defeats all arguments $Args \setminus S$.

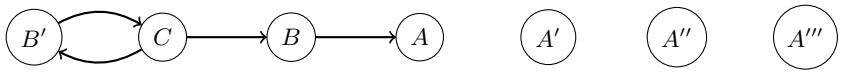
Example 2.3 The knowledge base of Example 2.1 gave rise to arguments A, B and C, of which it was identified in Example 2.2 that B undercuts A and C undermines B. But various other arguments can be constructed with this knowledge.

$$A' = \frac{\text{steak} \qquad \text{wagyuCattle}}{A = \dfrac{\text{wagyuSteak}}{\text{bestSteak}}\,\varrho_1}\,\varrho_3 \qquad A'' = \text{steak}$$

$$B = \frac{\text{improperlyHandled}}{\neg\varrho_1}\,\varrho_2 \qquad A''' = \text{wagyuCattle}$$

$$C = \neg\text{improperlyHandled} \qquad B' = \text{improperlyHandled}$$

The abstract Dung-style argumentation theory behind this framework can be visualized as a directed graph. The nodes represent arguments, where the edges represent the defeat relation between arguments.



The resulting argumentation theory has a unique grounded extension $S = \{A', A'', A'''\}$ and two preferred extensions, $S = \{A', A'', A''', C, A\}$ and $S = \{A', A'', A''', B', B\}$, which are also the stable extensions.

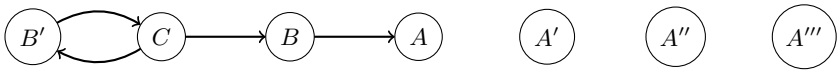
Finally, it is possible to assign an acceptability status to formulas. Acceptability is based on the evaluation of defeat relations in the argumentation framework, given some acceptability semantics. Two types of acceptable formulas will be used in this thesis: sceptically and credulously acceptable conclusions.

Definition 2.14 Given an argumentation theory $AT = \langle AS, K, \leq \rangle$, the argumentation framework $AF = \langle Args, Defeat \rangle$ on the basis of AT and a semantic S, a formula $p \in L$ is

- *sceptically S-acceptable* iff there exists an $A \in Args$ where $\mathsf{conc}(A) = p$ and A is in all S-extensions of AT
- *credulously S-acceptable* iff there exists an $A \in Args$ where $\mathsf{conc}(A) = p$ and A is in at least one S-extension of AT

Sceptical acceptability requires that there is no way to reason such that the formula could possibly not be acceptable. In contrast, credulous acceptability requires that there exists an argument that is part of a set that can defend against any attack, but allows for the possibility that another conclusion can be drawn within the same argumentation theory. Any semantic may be used, for example, a sceptically grounded-acceptable or credulously preferred-acceptable formula.

Example 2.4 Recall the argumentation framework of Example 2.3.



While there is an argument such that $K \mid\sim$ bestSteak, the conclusion bestSteak can not be sceptically or credulously drawn with grounded semantics. The only argument that claims bestSteak, argument A, is not in any grounded extension. On the other hand, when preferred semantics is used to evaluate the acceptability of a claim, bestSteak is credulously acceptable, since argument A is in extension $S = \{A', A'', A''', C, A\}$. bestSteak is not sceptically acceptable under preferred semantics, since argument A is not in the preferred extension $S = \{A', A'', A''', B', B\}$.

2.1.3 Practical reasoning

In the informal introduction on arguments there was a reference to a difference between epistemic and practical reasoning, but as so far this difference has not yet been made explicit in the way arguments are constructed or evaluated. Recall that an epistemic argument merely draws conclusions about beliefs, like 'these are the best steaks, since wagyu steaks are typically considered the best'. On the other hand, practical reasoning draws conclusions that a certain action should be performed, taking into account some goal that needs to be achieved. Therefore, practical arguments represent means-end reasoning, connecting a goal to a single statement that some action should be done. For example, 'if we go to the bistro, we will get served the best steak and therefore enjoy our dinner'

There have been various approaches in the literature to enhance argumentation logics with practical reasoning. Atkinson et al. (2005b) have introduced an argument scheme for practical reasoning. This scheme encodes connecting a goal, the underlying value, a (proposal for) action and the circumstances in a single inference step.

> In the current circumstances R
> Action A should be performed
> To bring about new circumstances S
> Which will realise goal G
> And promote value V

An extensive list is provided of ways in which the application of this inference scheme can be attacked. For example, a counterargument might state that the current circumstances do not hold or that the performing of the action does not lead to the claimed realization of the goal. Application of the scheme leads to an argument, which has essentially a single inference step, combining the value, goal, circumstances and action. For that reason, every variation in one of the elements in the scheme leads to a new argument. The attack and defeat between these arguments is evaluated using a value-based argument framework, which is an extension of Dung-style argumentation frameworks. (Bench-Capon, 2003)

Every argument in Atkinson et al. (2005b,a) is an instantiation of the argument scheme for practical reasoning. As a consequence, there is no way to combine epistemic and practical arguments. Moreover, arguments are limited

to the single inference step that is the instantiation of the scheme and do not allow for structured sub-arguments.

Rahwan and Amgoud (2006) as well as Bench-Capon and Prakken (2006) have introduced logics that allow for both epistemic and practical arguments to be combined. In both cases desires (goals) are introduced as separate language entities, which are used to build structured arguments. These arguments form a Dung-style argumentation graph, which is evaluated with a custom acceptability semantic. The two papers differ in the way that desires in arguments are used to draw conclusions and how attack among arguments is defined.

Bench-Capon and Prakken (2006) take inspiration from the abductive nature of means-end reasoning. Desires are introduced as premise in an argument, denoted with a desire modality D. A new inference rule type can be used to draw (sub-)conclusions about the desirability of beliefs. For example, given some desire to DenjoyDinner and the defeasible inference rules goToBistro $\overset{\varrho 4}{\Rightarrow}$ bestSteak and bestSteak $\overset{\varrho 5}{\Rightarrow}$ enjoyDinner an argument can be constructed that concludes that bestSteak is now also desirable.

$$A' = \dfrac{\dfrac{D\text{enjoyDinner} \qquad \text{bestSteak} \overset{\varrho 5}{\Rightarrow} \text{enjoyDinner}}{D\text{bestSteak}} \qquad \text{goToBistro} \overset{\varrho 4}{\Rightarrow} \text{bestSteak}}{D\text{goToBistro}} \quad A$$

Attack between practical arguments is modelled through the introduction of a new way in which arguments can attack each other and a custom acceptability semantic called *e-p-semantics*. A desire-based argument attacks another if the conclusion is a desire that is a *sufficient alternative* to the desire in the other argument. An argument is an alternative of another argument if their conclusions are two different desires or if they are derived using accrual of arguments and in that share the same desire. A sufficient alternative is an argument that is an alternative while there is no other strictly preferred argument. For example, two atomic arguments with conclusions DenjoyDinner and DspendLittle are alternatives and without further preference ordering they are both sufficient alternatives of each other.

In the above example, sub-argument A' of A is itself also an argument that can be constructed. Unfortunately, the conclusion of A' conflicts with that of A, as they are sufficient alternatives of each other, resulting in two arguments

attacking each other. Since there are no explicit preferences, which would be unrealistic to have for all possible conclusions, neither of these arguments is now sceptically acceptable without further relevant arguments to resolve this mutual attack. However, intuitively there seems no reason why going to the bistro is not a reasonable action, even when reasoning sceptically.

The problem with practical compound arguments leading to mutual attack with sub-arguments was first established in the experimental work done which will be presented in Chapter 8. In the knowledge bases that are used there, the problem turned out especially prevalent, as practical arguments are used by agents to justify proposals for action. As a result, an explosion of equally acceptable alternatives would arise, without these alternatives being intuitively conflicting.

Rahwan and Amgoud (2006) define attack and defeat in their structured practical arguments in a different way, which does not result in the unnatural problem with conclusions of sub-arguments defeating the top conclusion. Practical arguments follow the same modus ponens-based reasoning style as epistemic arguments. Specific desire-related attack is realized though the classification of belief-undercutting and desire-undercutting. Belief-undercuts are the type of attack that ASPIC+ calls an underminer, attacking a premise. Desire-undercuts also undermine a premise, but this premise is a desire instead of a belief. That means that for desires to be undercut, they must be used as antecedent in a defeasible inference rule. The above example on where to enjoy dinner therefore has different rules. Given some knowledge bestSteak, claiming that the bistro serves the best steaks, and defeasible rules bestSteak $\overset{\varrho 6}{\Rightarrow}$ enjoyDinner and enjoyDinner $\overset{\varrho 7}{\Rightarrow}$ goToBistro the conclusion goToBistro can be drawn.

$$A' = \frac{\text{bestSteak}}{\text{enjoyDinner}}\,\varrho_6$$
$$A = \frac{\text{enjoyDinner}}{\text{goToBistro}}\,\varrho_7$$

However, the required inference rules arguably have an unintuitive structure, such as the consequence to go to the bistro based on the desire to enjoy dinner. After all, going to the bistro is justified in that we can expect the best steak there, not on a desire to enjoy dinner. Hence, the enjoyDinner predicate should be read not as a general desire, but rather a conditional desire to enjoy dinner by eating the best steak.

Three approaches to model practical, structured argumentation have been discussed: single inference step instantiations of the argumentation scheme for practical reasoning, abductive reasoning-inspired allotment of desires and the distinction of belief-undercutting and desire-undercutting. To overcome the problems that have been identified with each of these three approaches, this thesis will adopt another way of modelling practical arguments. It will allow for the mixing of practical and epistemic arguments, which are evaluated as a normal Dung-style framework with preferred semantics for argument acceptability. The approach is to have defeasible rules that directly use goals and actions, without an explicit desire modality, and use the standard epistemic way of evaluating argument attack, defeat and acceptability.

Definition 2.15 Let

- L_b contain beliefs,
- L_o contain options,
- L_g contain goals

For any argumentation system $AS = \langle L, R, \mathsf{cf}, \leq \rangle$ it holds that $L = L_t$ and $L_t = L_b \cup L_o \cup L_g$ such that $L_b \cap L_o \cap L_g = \emptyset$.

Any argumentation system in this thesis will use topic language L_t. The beliefs of L_b are the normal language elements as would be used in a strictly epistemic topic language. L_o contains options, which are statements that some action should be done. The term option is used to indicate that arguments are about the possible executing of some action, not about an agent's capability, planning or execution of the action. Goals in L_g are the desires that agents have. The three element types together form the topic language that any argumentation system in this thesis will use.

Now that beliefs, options and goals are elements of the same logical language, structured practical arguments can be constructed in a similar fashion to purely epistemic arguments. These practical arguments can be compared to epistemic arguments, do not have a problematic attack relation with sub-arguments and use defeasible inference rules that follow a natural flow.

Example 2.5 Consider again the example phrase 'we will enjoy our dinner, for we will get served the best steak, as we will go to the bistro'. Relevant formulas

in this example include a belief bestSteak $\in L_b$, an option goToBistro $\in L_o$ and a goal enjoyDinner $\in L_g$. Given knowledge base $K = \{\text{goToBistro}\}$ and the set of defeasible rules $R_d = \{\text{goToBistro} \overset{\varrho_4}{\Rightarrow} \text{bestSteak}; \text{bestSteak} \overset{\varrho_5}{\Rightarrow} \text{enjoyDinner}\}$ a practical argument can be constructed that concludes enjoyDinner.

$$A' = \frac{\text{goToBistro}}{\text{bestSteak}}\,\varrho_4$$
$$A = \frac{\text{bestSteak}}{\text{enjoyDinner}}\,\varrho_5$$

Argument A' reads 'we will get served the best steak, as we will go to the bistro', while argument A reads 'we will enjoy our dinner, for we will get served the best steak'.

2.1.4 Software implementation

Although argumentation logics have been subject of extensive formal study throughout several decades, there exist few software implementations of it, especially of those supporting structured argumentation. Moreover, for a logic implementation to be of use in the context of this thesis, it needs to comply with the ASPIC− framework.

Very recently a new implementation, called TOAST, was proposed by Snaith and Reed (2012b). It strictly adheres to the ASPIC+ framework, supporting all premise types, strict and defeasible rules, rule preferences, explicit premise contrariness and various acceptability semantics. It is implemented in Java and offers a graphical web interface as well as a web service. At the moment, TOAST is one of the most complete and state of the art implementations. Unfortunately, it was not yet available during the research project of this thesis.

In the master thesis of Visser (2008) another implementation of the ASPIC+ framework was proposed. Implemented in Java, it provides a graphical desktop interface and can be used as module. Although it is limited to a propositional logical language, it does support most of the ASPIC+ features, such as strict and defeasible rules, the three types of argument attack and various acceptability semantics. Interestingly, it has support for Bench-Capon and Prakken (2006)-style practical reasoning through e-p-semantics. Indeed, it was through experimentation with this software implementation that the above described issue with conflicting sub-arguments was identified.

The final ASPIC+-compatible implementation available was created by South and Vreeswijk (2009). Originally built as proof of concept in Ruby, it was later

developed as a full implementation of the original ASPIC (Amgoud et al., 2006) framework as a reusable Java module. A knowledge base is defined, which may consist of rule and belief, which are encoded as a set of body-less rules. Facts and ordinary premises, as well as strict and defeasible rules, are supported through the assignment of a *degree of belief*. This DOB is a real number in range $(0, 1]$, where a higher DOB results in a stronger inference (and accordingly a stronger argument). Since the above presented model does not allow such graduation, rules are either implemented with DOB 0.5 (i. e. defeasible) or 1.0 (i. e. strict).

Example 2.6 Consider an example situation with knowledge on where to go in order to enjoy dinner. Two defeasible rules are available to construct a practical argument how going to the bistro leads to enjoying dinner. An epistemic rule is available that can undercut the inference that going to the bistro will lead to eating the best steak.

```
[r4] bestSteak <- goToBistro 0.5.
[r5] enjoyDinner <- bestSteak 0.5.
goToBistro.
~r4 <- notServingSteak 0.5.
notServingSteak 0.5.
```

All of the rules are defeasible, as their degree of belief is set to 0.5. Querying this knowledge base on enjoyDinner with grounded semantics will return `no` while the same query for preferred semantics will yield `yes`.

The (South and Vreeswijk, 2009) implementation of the ASPIC+ framework makes a distinction between two types of attack rather than the three available in the formal model presented above. Undermining is not separately implemented, but is still present in the form of rebuttal of atomic arguments. This exploits the fact that every premise used in an argument first needs to be actualized as an atomic argument before it can be used in a compound argument. As a result, undermining becomes a specific form of rebutting and the attack (and defeat) rules for rebutting apply.

There is an issue with the South and Vreeswijk implementation of the undercutter in the implementation. Any rule definition in an implemented knowledge base will automatically add the rule name as fact, causing any undercutter on the rule application to be rebutted. For example, if there is an argument that applies rule $p \overset{\varrho}{\Rightarrow} q$, and some argument attacks it with an argument with conclusion $\mathsf{conc}\neg\varrho$, it would automatically be rebutted with an argument stating

$\mathsf{conc}\varrho$. This behaviour deviates from the definition of undercutting in the ASPIC+ framework. Unfortunately, the software built in this thesis has to work around this issue by designing agents that simply do not play such arguments in a dialogue, which is explained further in Chapter 8.

2.2 Agent communication

With the advancement of the agent paradigm in artificial intelligence, an array of sub fields have been established, of which each pertain to a distinct characteristic of agent design. One of the critical features that has received significant attention is that of communication.

Typically, communication between agents is designed by means of a communication language that is used by the agents to perform speech acts. (Searle, 1976) Rooted in linguistics and the philosophy of speech, a speech act basically is an action that is performed by an agent, by which a message is sent to another agent that will then parse and interpret it. To ensure that the agents understand each other, all agents in a system are required to use the same transport mechanism for messages and adopt the same language.

Early work on agent communication tried to define a single definitive agent communication language for all types of dialogues. This was attempted by coming up with a fixed set of locution types, which were supposed to provide all types of statements that an agent would need to make. (FIPA, 2000) Every locution type is then associated with some content as expressed in a topic language, being a logical language that agents agree upon. The locution and content together is called a performative. For example, an agent could inform another agent of some status to be normal by sending a message such as inform(status(normal)). The interpretation of such a message was at first left to the implementation of a specific agent, but in later research the content of messages was bound to a specific ontology and every performative was bound to certain preconditions, for an agent to be able to state it, and postconditions, enforcing a change in the other agent's beliefs.

2.2.1 Argumentation in dialogue

While in human communication there exist simple statements like providing or requesting information, it is often the case that statements are actually part of a dialogue process to find a mutual belief or mutually reach some decision. The

content of messages is thus not isolated, but rather part of a dialogical argument. The same holds for agents, in that agents are autonomous and therefore need communication to align beliefs and coordinate their actions, including to negotiate on available resources and to mutually make decisions.

Early work that adopted argumentation in multi-agent systems was done by Parsons and Jennings (1996); Amgoud et al. (2000); McBurney and Parsons (2002). These studies introduced dialogue frameworks in which a tailored communication language is used to allow expression of arguments between agents to agree on some initial claims or negotiate on scarce resources. By allowing arguments rather than merely informational statements, it was possible to evaluate the status of the initial claim or proposed resource division. Importantly, the rationale for this work was not only in the ability to allow agents to argue in dialogues, but that the use of argumentation would allow the agents to either improve the outcome of the dialogue situation or improve on the way that an outcome comes about. Some of these benefits were proved formally, while other benefits still need to be defined and tested, being one of the reasons for the research behind this thesis.

2.2.2 Typology of argumentation-based dialogues

Frameworks for argumentation-based dialogues have been proposed for a variety of dialogue types where some form of argumentation is appropriate. The most influential work in distinguishing types of dialogues is that of Walton and Krabbe (1995). They introduce a typology of various types, grounded in the philosophy of human argument. (Walton and Krabbe were interested in fallacies in human dialogue by studying how an utterance leads to commitment. Although this was later covered in agent communication research as well, most frameworks for argumentation dialogues merely use the typology to delineate their dialogue scope.)

Although Walton and Krabbe claim to not provide an exhaustive or even fully developed typology, and allow for mixing of types within a single dialogue, their distinction between 6 general types has been widely adopted. Most clearly defined and commonly studied of these are persuasion, negotiation, inquiry, deliberation and information seeking. Eristic dialogues have received lesser attention, as their do not (yet) translate to agent theory. Table 2.1 list the dialogue types and their characteristics regarding the dialogue goal and the goal of its participants.

Table 2.1: Dialogue typology as taken from Walton and Krabbe (1995)

type	initial situation	main goal	participant's aims	side benefits
Persuasion	Conflicting points of view	Resolution of such conflicts by verbal means	Persuade the others	Develop & Reveal positions, Build up confidence, Influence onlookers, Add to prestige
Negotiation	Conflict of interest & Need for cooperation	Making a deal	Get the best out of it for oneself	Agreement, Build up confidence, Reveal positions, Influence onlookers, Add to prestige
Inquiry	General ignorance	Growth of knowledge & Agreement	Find a proof or destroy one	Add to prestige, Gain experience, Raise funds
Deliberation	Need for action	Reach a decision	Influence outcome	Agreement, Develop & Reveal positions, Add to prestige, Vent emotions
Information seeking	Personal ignorance	Spreading knowledge & Revealing positions	Gain, pass on, show or hide personal knowledge	Agreement, Develop position, Influence onlookers, Add to prestige, Vent emotions
Eristics	Conflict & Antagonism	Reaching a (provisional) accommoda-tion in a relationship	Strike the other party & Win in the eyes of onlookers	Develop & Revela positions, Add to prestige, Gain experience, Amusement, Vent emotions

By far the most commonly studied are frameworks for *persuasion* dialogues, also called dispute (McBurney and Parsons, 2002; Prakken, 2005). Agents discuss the truth status of a claim in order to convince others. Persuasion is almost exclusively a competitive game, as even the mutual goal of resolution is usually strongly subordinate to the personal goals. Persuasion has received major attention from fields such as law, linguistics and multi-agent theory and as such a significant amount of formal models for it exists.

Research on *negotiation* has also resulted in a rich body of formal frameworks. Negotiation is used by agents if (usually scarce) resources are available in a system, which agents need to work with. The earlier argumentation-based dialogues models were for negotiation (Parsons and Jennings, 1996) and early experimental work on argumentation dialogues also used negotiation to investigate the use of argumentation in multi-agent dialogues (Pasquier et al., 2010). This interest is perhaps due to the easy to see uses of arguments to reveal resource division proposals that would otherwise not be possible. Negotiation is predominantly competitive, but mutual seeking into alternatives can increase the realm of possible outcomes. Moreover, since failure to agree is often a highly undesirable situation there can be pressure to cooperate on finding the best deal.

Inquiry revolves around finding the truth status of some claim or just generally acquiring more knowledge, much like the scientific investigation process itself. It has received a relative modest amount of attention from the multi-agent research community, Black and Hunter (2007) being one exception, maybe due to the almost exclusively cooperative nature. Still, the process is very well suited to be supported by an argumentation logic, to constantly evaluate newly acquired information through the development of acceptably arguments in the background knowledge. Information seeking dialogues are highly similar, although the dispersion of knowledge is more important as agents seek for information from a more personal motivation, rather than a general one.

Finally, deliberation is a style in which agents acknowledge the need for action, which they mutually have to decide on. Deliberation dialogues contain an interesting balance between cooperation, since agents have a mutual goal to agree on an action, and competition, with agents having both compatible and incompatible goals. Moreover, deliberation combines practical and epistemic reasoning and typically concern more than two agents.

2.2.3 Elements of dialogue frameworks

A framework for any of the argumentation-based dialogue types will always include a communication language, a dialogue structure, a move evaluation method and a dialogue protocol. These elements will now be briefly introduced.

As described above, agents communicate by using speech acts. These contain not only the content of the communicative act, but also a locutionary type. While the content is constructed in the topic language that is agreed upon, the locution of a speech act is one of the types that is defined by the communication language. Moreover, the communication language in a dialogue framework typically enforces the type of content of every locution, possibly even restricting the content to include or exclude certain statements. A communication language may include locutions such as inform, argue and prefer.

Individual communicative acts by the agents, also called moves, are related and evaluated in a dialogue framework though the specification of a relationship between the acts and describing the effect of a move. First of all, a framework allows agents to play by explicitly giving turns to agents. Turn taking can be defined in three ways: direct replying to a previous move, resulting in a linear structure, requiring a move target, giving rise to an explicit tree-like reply structure, or by a direct playing of arguments, directly forming an argumentation system with attacking (and defeating) arguments.

Through taking of turns and playing of moves, the agents construct a dialogue. This dialogue, being a linear structure, a tree or a dialectical structure of arguments, can in turn be evaluated to draw a conclusion. Based on the structure and the content of the moves, the outcome of a dialogue is determined using an outcome function that is specific to the type of dialogue. Outcome selection can be direct, following from the dialogue structure, as well as indirect, for example by constructing an argumentation system with the moved arguments.

Finally, the agents are restricted in playing moves through a dialogue protocol. Moves allowed by a protocol are referred to as legal moves at a certain state in a dialogue. Protocol rules may include disallowing replies to ones own moves, restricting the use of certain locutions in reply to a specific locution type or enforcing an agent to be consistent regarding earlier moves.

The first step towards the evaluation of the benefits of argumentation in multi-agent dialogues is the introduction of a suitable framework for argumentation-based dialogues. The next chapter will introduce the model for deliberation style dialogues that is used in this thesis.

Chapter 3

Deliberation dialogue framework

The previous chapter introduced the general concepts behind argumentation-based dialogues and briefly introduced how such dialogues can be modelled using a structured framework. This chapter will present a framework for deliberation style dialogues between multiple agents. The design of this framework is driven by the characteristics of the deliberation dialogue type. The initial situation, the dialogical goal of reaching a decision and the personal interests of agents will each be modelled explicitly and concepts are introduced that allow for the playing of moves, including proposals and arguments. Agents are restricted in which moves they can play using a deliberation protocol. In the end, it is shown how a dialogue can be evaluated and how a proposal can be selected as dialogue outcome.

3.1 Existing deliberation frameworks

While many frameworks for argumentation-based dialogues exist in the literature, not many pertain to deliberation. Two notable exceptions are the studies by McBurney et al. (2007) and Black and Atkinson (2010).

McBurney et al. (2007) have proposed a framework for deliberation dialogues that allows multiple agents. The paper's main goal is to provide a protocol that allows for the open nature of deliberation, giving the agents much freedom for establishing goals, constraints, perspectives, facts, actions and evaluations. Goals, actions and facts correspond to the goals, options and beliefs, respectively, as

defined in Section 2.1.3. Constraints and perspectives are general expressions, much like expressing a belief but with the explicit intention of discussing the constraint, perspective or evaluation. For example, an agent can propose the evaluation of an action, which signals that other agents should express their preferences, ask for more information or reject the action.

A deliberation dialogue moves through several stages. Agents explicitly join and leave the dialogue. Via inform and propose stages, agents are asked to evaluate and decide on actions in the consideration, revision, recommendation and confirmation stages. Stages are entered and left based on the statements by the agents and stages in turn restrict the legal moves. The internal structure of move content is left mostly unspecified. Notably, there is no specific argumentation logic to give arguments or counterarguments.

The framework puts few hard constraints on the course of the dialogue and, importantly, does not try to establish a formal way for determining the outcome of a dialogue based on the played moves. By contrast, the framework proposed in this chapter will provide a way to evaluate the status of proposed actions based on the played moves and their explicit reply structure. Moreover, the dialogue structure will be used to further restrict agents in which moves they can play. Several protocol rules will be discussed that are useful in supporting a more structured evaluation and lead to more coherent dialogues than is possible with the liberal protocol of McBurney et al..

Black and Atkinson (2010) have also proposed a framework for deliberation dialogues. Special attention is given to the way the agents' moves lead to a dialogue outcome in the form of a selected action. Except for opening and closing a dialogue, every move is either a single practical argument or a statement of agreeing on one of the earlier played arguments. Every practical argument is an instantiation of the argument scheme for practical reasoning, discussed in Section 2.1.3, and these arguments together form a value-based argumentation framework that can be evaluated following the earlier discussed work by Atkinson et al. (2005b). Apart from requiring an action to be played in an argument before agreeing to it, there are no restrictions on which moves agents can play.

Importantly, it is shown that, when agents adhere to the dialogue protocol, any agreed upon dialogue outcome is also acceptable to both agents. That means that, although no outcome is enforced, for every outcome the agents have an acceptable argument for it. Arguably this is good, since agents are not forced to go along with an action that they disapprove of.

Arguments in this deliberation framework consist solely of a single inference

step, that is, the instantiation of the argument scheme for practical reasoning. This allows for the comparison of proposed actions in a Dung-style fashion. However, as already described in the previous chapter, this logic does not allow for arguments with further structure or purely epistemic arguments. Hence, the framework does not allow the agents to discuss factual information surrounding proposed options. As an extension, the dialogue protocol merely supports proposing and accepting of practical arguments, where the practical arguments (potentially) conflict directly. It does not support asking for justification or providing of counterarguments.

Another, very recent addition was proposed by Atkinson et al. (2012), which makes concrete the interplay between persuasion and explorative deliberation. Moving between these dialogue types is actuated by the strength of user preferences and the appropriate use of a different set of speech acts. Unfortunately, no explicit method is defined on how to derive the outcome from the moves that the agents made. Instead it seems that this is assumed that agents, in a separate dialogue phase, explicitly accept some option or otherwise make a decision. While the interplay between persuasion and deliberation is elegantly modelled in this paper, the time of its publication was after the development and description of the model in this thesis.

3.2 Languages and dialogue structure

A deliberation framework will now be introduced based on the framework for persuasion dialogues of Prakken (2005). It is adjusted for use with deliberation dialogues, which includes allowing for more than two agents, the introduction of proposals, preferences over proposals and an action as dialogue outcome, and the reworking of turn taking, move evaluation and protocol rules. Definitions 3.2, 3.3, 3.4, 3.11 and 3.12 are taken from Prakken (2005), while the other definitions in this chapter are new or reworked for use with deliberation.

3.2.1 Context

Before agents can discuss anything, a context needs to be set. In frameworks for agent communication, this context specifies the languages that agents use to communicate and to describe the topic that they are discussing. The topic language is the logical language for practical reasoning, which consists of options, goals and beliefs. Arguments are expressed and evaluated using the ASPIC- ar-

gumentation system described in Section 2.1. Every agent is assumed to know about the used language and argumentation logic.

Definition 3.1 A deliberation dialogue context $\mathcal{DK} = \langle AS, L_t, L_c, \mathcal{P}, \mathcal{A}, g_d \rangle$ consists of:

- An ASPIC− *argumentation system* $AS = \langle L_t, R, \mathsf{cf}, \leq \rangle$, where L_t is the topic language of Definition 2.15
- A communication language L_c
- A protocol $\mathcal{P}$
- A sequence $\mathcal{A} = \langle a_1, \ldots, a_i, \ldots, a_n \rangle$ of agents, where $\mathsf{agent}(i) = a_i$
- A mutual goal $g_d \in L_g$

Recall that the topic language L contains options L_o, goals L_g and beliefs L_b, which are disjoint sets of language elements. Typical elements are denoted $o \in L_o$, $g \in L_g$, $p \in L_b$ and $a \in \mathcal{A}$. $\mathsf{agent}(i)$ is used to refer to the ith agent in the sequence of all agents $\mathcal{A}$, where i is said to be the identifier of the agent.

When a deliberation dialogues commences, agents realise, or acknowledge, the need for action. That is, the agents commit themselves to finding the best course of action for some problem. While some models for deliberation dialogue in the literature include the process of establishing this dialogical goal in the deliberation framework, such as that of McBurney et al. (2007), here the goal is made explicit as part of dialogue context. Every agent is assumed to adopt this mutual dialogue goal and to work towards finding an action that supposedly realises it. Note that later chapters will discuss an agent's individual goals, and how these may coincide or conflict with the mutual goal in the dialogue.

Every dialogue context $\mathcal{DK}$ is associated with a mutual goal g_d that defines the objective that the agents agreed upon. The dialogue purpose is to reach a decision on a single course of action that fulfils the mutual goal. The final course of action is determined by some outcome selection function, as defined, based on the proposals made by the agents in the dialogue.

3.2.2 Communication language

As explained in Section 2.2.3, every dialogue system contains a communication language. This consists of several types of locutions specific for a deliberation

style dialogue. Every locution is associated with some content, which has a specific type. For example, the locution to make proposals only accepts options and the locution to state an argument only accepts arguments expressed following the ASPIC- framework. The medium of communication is left unspecified, but the agents are assumed to be able to receive and process all utterances, without distortion or ambiguity. Finally, the communication language here presented defines for each speech act an attacking or surrendering relationship with other speech acts.

Definition 3.2 , The *communication language* L_c in a dialogue context $\mathcal{DK} = \langle AS, L_t, L_c, \mathcal{P}, \mathcal{A}, g_d \rangle$ is a set of locutions $\mathcal{L}$ and two binary relations S_a and S_s of attacking and surrendering reply on $\mathcal{L}$. Every $s \in \mathcal{L}$ is of the form $p(l)$ where p is a performative and $l \in L_t$ or l is an argument expressed in the ASPIC− framework. S_a and S_s are disjunct and irreflexive. Locutions cannot attack one locution and surrender to another. Every surrendering locution has an *attacking counterpart*, which is an attacking locution in L_c.

A statement by one of the agents in a dialogue, using one of the available speech acts, is called a move. A move is not only associated with the speech act and some move content, but is also assigned a unique number, a reference to the player that made the move and a move target. The target of a move is either (the identifier of) another move or, in case the move did not target another move, 0.

Definition 3.3 The set of all possible *moves* M defined as $\mathbb{N} \times \mathcal{A} \times L_c \times \mathbb{N}$ where each element of a move m respectively is denoted by:

- $\mathsf{id}(m)$, the move identifier,
- $\mathsf{player}(m)$, the identifier of the agent that played the move,
- $\mathsf{content}(m)$, the speech act of the move,
- $\mathsf{target}(m)$, the move target.

The locution types available in communication language L_c are itemized in Table 3.1, each with the appropriate attacking and surrendering replies. The attacking counterpart for each surrendering locution is displayed in the same row. The locutions that deal with proposals (propose, inform, why-propose, prefer and prefer-equal) are taken from McBurney et al. (2007) while the ones dealing with

Table 3.1: The available speech acts in the communication language L_c, where every $o, o' \in L_o$ and $p, q \in L_b$ and A, B are arguments in AS

speech act	attacks	surrenders
propose(o)	why-propose(o) reject(o)	
why-propose(o)	argue(A) where $o \in$ prem(A)	
reject(o)		
prefer(o, o')		
prefer-equal(o, o')		
skip		
inform(p)		
argue(A)	argue(B) where B defeats A	concede(p) where $p =$ conc(A)
	why(q) where $q \in$ prem(A)	concede(q) where $q \in$ prem(A)
why(p)	argue(A) where $p =$ conc(A)	retract(p)
concede(p)		
retract(p)		

persuasion (argue, why, retract, concede) are adopted from Prakken (2005)'s framework. Below the term *proposal move* is used when the content(m) = propose(o), *argue move* is used when the content(m) = argue(A), etc.

Proposals, forwarding an option to be considered as dialogue outcome, are made using the propose speech act, which takes an option $o \in L_o$. Proposal moves can be questioned using the why-propose speech act, which explicitly targets the original proposal. Alternatively, the reject move can be used if agents do not argue, but limit themselves to proposing and rejecting options. Agents state preferences over already proposed options using prefer and prefer-equal, which both take two options $o, o' \in L_o$. skip is a content-less speech act with special meaning to end a turn, as specified below.

Agents can play arguments using the argue speech act. It takes a well-formed argument expressed in the context's AS as content and is used to provide backup to earlier statements, for options as well as beliefs, or to counter-attack another argument. Arguments supporting a proposal are required to use the proposed option as premise. This is further formalized as a dialogue protocol in Definition 3.14 below. Arguments supporting a specific claim, some belief

$p \in L_b$, are required to conclude with the supported claim, that is, such an $\mathsf{argue}(A)$ move supporting p has $p = \mathsf{conc}(A)$. Arguments played in response to a previous argue move must defeat the attacked argument, as defined by the defeat relation of AS.

Inform, concede and retract moves all hold a well-formed formula $p \in B$. Informing has a purely informative meaning, as they can not be attacked, similar to the inform speech acts in agent communication languages such as FIPA (2000) and the assert statement in the model of McBurney et al. (2007). Conceding and retracting are surrendering moves used to state that the agent no longer considers some earlier claim as significant or does not wish to further pursue discussing it. The exact meaning of surrendering to claims and how it affect the meaning of moves is formalised below.

Note that for every move m where content = propose, inform, skip, prefer or prefer-equal it holds that $\mathsf{target}(m) = 0$, because of the absence of an attacking or surrendering reply relation in L_c. All other locutions always have an explicit target move, such that target $\neq 0$. The communication language is extendible as long as the attack relation of speech acts is specified.

Example 3.1 In Section 1.1.3, an example dialogue was introduced in which two agents who want to go to dinner together share a mutual goal to enjoy dinner, that is, the dialogue context's mutual goal g_d = enjoyDinner. The moves of their dialogue are presented in Table 3.2, where the natural language is represented in speech acts on the right. Note that skip moves are numbered but not listed.

Table 3.2: An example dialogue showing the mapping of natural language to speech acts, where moves are marked with their numeric identifier and the agent (a or b) that made the move

move	statement	logical form
1a	I suggest we go to the bistro.	propose(goToBistro)
3b	Why should we go there?	why-propose(goToBistro)
4b	We could go to the pizzeria instead.	propose(goToPizzeria)
6a	We will enjoy our dinner, for we will get served the best steak, as we will go to the bistro.	argue(goToBistro; goToBistro $\overset{\varrho_4}{\Rightarrow}$ bestSteak; bestSteak $\overset{\varrho_5}{\Rightarrow}$ enjoyDinner $\mid\sim$ enjoyDinner)
7a	So I prefer to go to the bistro.	prefer(goToBistro, goToPizzeria)
9b	Why would we get the best steak?	why(bestSteak)
10b	The pizzeria and bistro are equally preferable to me.	prefer-equal(goToBistro, goToPizzeria)
12a	These are the best steaks, since wagyu steaks are typically considered the best and these are wagyu steaks since they are steaks made from wagyu cattle.	argue(steak; wagyuCattle; steak, wagyuCattle $\xrightarrow{\varrho_3}$ wagyuSteak; wagyuSteak $\overset{\varrho_1}{\Rightarrow}$ bestSteak $\mid\sim$ bestSteak)
13a	We will enjoy our dinner, for we can drink tasty beer, as we will go to the bistro.	argue(goToBistro; goToBistro $\overset{\varrho_6}{\Rightarrow}$ tastyBeer; tastyBeer $\overset{\varrho_7}{\Rightarrow}$ enjoyDinner $\mid\sim$ enjoyDinner)
15b	I admit that it's made from wagyu cattle.	concede(wagyuCattle)
16b	But in this case the steaks are not the best, since they are improperly handled.	argue(improperlyHandled; improperlyHandled $\overset{\varrho_2}{\Rightarrow}$ $\neg\varrho_1$ $\mid\sim$ $\neg\varrho_1$)

3.2.3 Dialogue tree structure

The individual moves by the agents combined form the deliberation dialogue structure.

Definition 3.4 The set of *dialogues* $M^{\leq\infty}$ is the set of all sequences $m_1, \ldots, m_i, \ldots$ from M, where each i^{th} element in the sequence has identifier i and for each m_i in the sequence it holds if $\mathsf{target}(m_i) \neq 0$ then $\mathsf{target}(m_i) = j$ for some m_j preceding m_i in d. The set of finite dialogues $M^{<\infty}$ is the set of all those dialogues that are finite, where one such dialogue is denoted by d.

Although moves are played in sequence, the resulting dialogue is not linear in structure, as moves are not merely a direct reply to the previous move. Instead, moves can explicitly target an earlier move and moves can be targeted by multiple other moves. In particular, a $\mathsf{propose}(o)$ move, which has no target, will be the root of a proposal tree, further formed by the directed replies.

Definition 3.5 For each proposal move m_i in dialogue d a *proposal tree* P is defined as follows:

1. The root of P is m_i.
2. For each move m_j that is a node in P, its children are all moves m_k in d such that $\mathsf{target}(m_k) = m_j$.

For any move m in proposal tree P we write $\mathsf{proposal}(m) = m_i$.

This is a tree since every move in d has at most a single target. Note that skip, inform, prefer and prefer-equal moves are not part of any dialogue tree. Instead, they have a special function, as explained below. Figure 3.1 shows the proposal trees that can be constructed from the example dialogue of Table 3.2.

Every path from the top of a proposal tree to one of the leafs represent a separate line of dispute. Each line forms another perspective in the discussion of the proposed option. As formalized below, it does not make sense to play the same move multiple times within a line of dispute, as it does not bring new information to the discussion.

Definition 3.6 A *line of dispute* in a dialogue d is a sequence of moves $\langle m_0, \ldots, m_i, \ldots, m_n \rangle$ in d where $\mathsf{content}(m_0) = \mathsf{propose}(o)$ and for every move in the sequence it holds that $\mathsf{target}(m_i) = \mathsf{id}(m_{i-1})$.

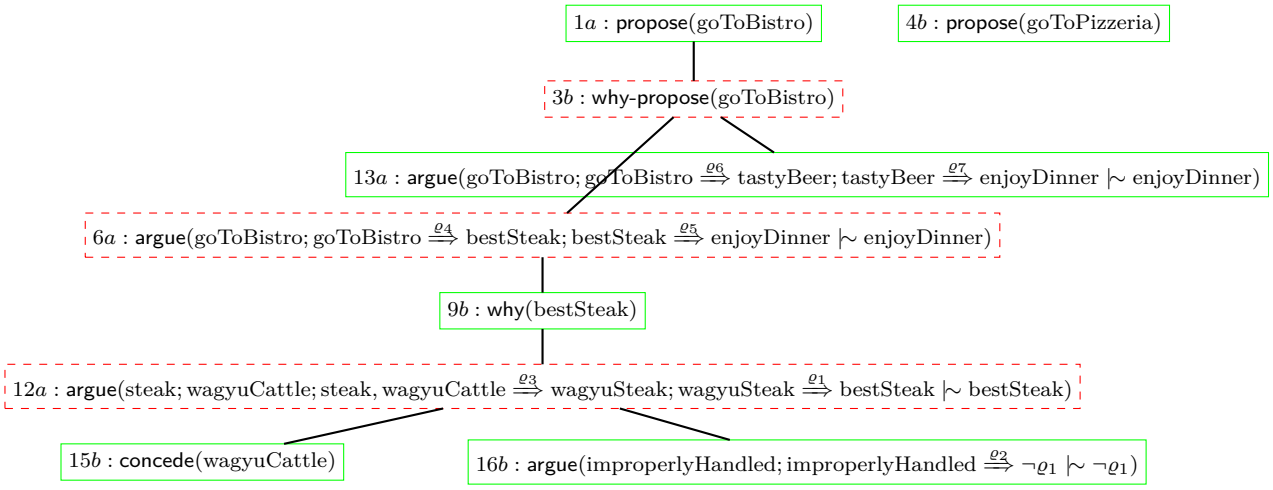


Figure 3.1: Proposal trees of the example dialogue of Table 3.2

3.2.4 Turn taking

Agents do not randomly make moves in a dialogue. Instead, turns are given in sequence to allow one agent to make moves at a time. A single turn may consist of multiple moves by an agent. Which agent currently is supposed to play a new move is determined by the turn taking function. Exactly one agent is identified as the new turn taker, based in the existing moves in a dialogue.

Definition 3.7 A *turn taking function* $T : D \longrightarrow \mathcal{A}$ maps a deliberation dialogue to a single agent.

An agent has to explicitly state that it wants to end its turn by playing a skip move. Otherwise the next player to move will always be the player of the last move in the dialogue.

Definition 3.8 For a dialogue $d = \langle m_1, \ldots, m_n \rangle$ the turn taker $T(d) = \mathsf{agent}(\mathsf{player}(m_n))$ unless $\mathsf{content}(m_n) = \mathsf{skip}$ in which case $T(d) = \mathsf{agent}(i)$ where $i = \mathsf{player}(m_n) + 1$ if $\mathsf{agent}(i) \in \mathcal{A}$ or else $i = 1$.

Note, however, that additional rules in the system may prevent an agent to keep on making moves by which it would prevent other agents from contributing. It is shown below how the dialogue protocol can restrict the possible moves an agent can make, even forcing an agent to skip turn.

3.3 Dialogical status of a move

The formal framework for persuasion dialogues of Prakken (2005), on which this deliberation framework is based, was introduced to study how the dialogue structure can be utilized to give a dialogical status to the moves the agents make. It was shown that, without maintaining an explicit underlying argumentation system, the status of the initial claim can be evaluated by assigning a status to every move, based on the move's replies. It will now be explained how this idea is extended for use with deliberation dialogues, in order to assign a dialogical status to proposals. Intuitively, this status represents whether a proposal seems justified, based on the attacking arguments and counter-attacks.

Every move in a proposal tree is always either *in* or *out*, drawing upon the explicit move targets and the difference between attacking and surrendering replies.

Definition 3.9 The *move status* of a move m in a proposal tree P is *in* in dialogue d iff:

1. m is surrendered in d by every agent $a \in \mathcal{A}$ where $a \neq \mathsf{player}(m)$; or else,
2. m has no attacking replies in d that are *in*.

Otherwise it is *out*.

From the definition it follows that, for a move to be surrendered, every agent that is not the original player needs to surrender to the move. Surrendering is done explicitly, by moving a surrendering reply.

Definition 3.10 A move m is *surrendered* in a dialogue d by some agent a iff:

1. m is an argue move A and a has made a reply $m' = \mathsf{concede}(\mathsf{conc}(p))$; or else
2. m is no argue move and a has made a surrendering reply to m in d.

Example 3.2 The box colour and shape of the moves in Figure 3.1 signify the move status. Move 1, why-propose(goToBistro), played by b is *out*, because it has an attacker that is *in*. The surrendering reply 15 is not enough to make move 12 *out*, as it is not the conclusion of the argue move 12 that was conceded. However, the argue move 16, played by agent b, does make move 12 *out*, since that now has an attacker that is *in*.

At first it might seem peculiar that other agents can make a move *in* by surrendering to it, and that all agents need to surrender to a move before it becomes *in*. However, even if the agent that originally made some statement may (no longer) agree with it, other agents might agree and assigning an *in* status to the move would force those agents to make the move again, possibly causing a loop.

Every move is assigned a status, but that of propose is of particular interest. It signifies whether the proposed option is still a viable alternative. Since the status of this proposal move is directly influenced by the other moves in the tree, it is also possible to identify whether a new to be played move is relevant with relation to the status of the proposal move.

Definition 3.11 An attacking move m in a dialogue d is *relevant* iff it changes the move status of $\mathsf{proposal}(m)$. A surrendering move is relevant iff its attacking counterpart is.

Depending on the domain, a different notion of surrendered move or relevance may be useful. Prakken describes a notion of weak relevance, as opposed to the normal, or *strong*, relevance, that may be adopted. It is weaker in the sense that an agent can contribute multiple ways to change the proposal tree root and still be relevant. This is achieved by only requiring a move to create an additional way to influence the status of a proposal.

Definition 3.12 An attacking move m in a dialogue d is *weakly relevant* iff it creates a new or removes an existing *winning part* in the proposal tree P associated with $\mathsf{proposal}(m)$ in d. A surrendering move is weakly relevant iff its attacking counterpart is. If the $\mathsf{proposal}(m)$ is *in*, a winning part w^P for this tree P is defined as follows:

1. First include the root of P;
2. For each m of odd depth, if m is surrendered by every agent $a \in \mathcal{A}$, include all its surrendering replies, otherwise include all its attacking replies;
3. For each m of even depth, include one attacking reply m' that is *in* in d;

The idea of a winning part is that it is 'a reason' why the proposal is *in* at that moment. Since this is not unique, there may be alternative attacking replies, a move is already weakly relevant if it succeeds to create an additional winning part or removes a winning part.

Example 3.3 In the dialogue of Example 3.1, move 13 by player a, providing an extra argument why going to the bistro leads to enjoying dinner, was not strongly relevant. It did not flip the dialogical status of the $\mathsf{propose}$(goToBistro) move at the state of the dialogue after move 12. It did not flip the status of the proposal since argue move 6 was still *in* at that moment. On the other hand, move 13 was weakly relevant, as it did create a new winning part.

3.4 Protocol rules

Agents, even if it is their turn, can not freely make moves. Rather, agents are bound by a protocol that dictates which moves can be played, based on the existing moves in a dialogue. Those moves are called the legal moves.

Definition 3.13 For any deliberation dialogue context $\mathcal{DK} = \langle AS, L_t, L_c, \mathcal{P}, \mathcal{A}, g_d \rangle$, protocol $\mathcal{P}$ on M is a function on a non-empty set of *legal finite dialogues* $D \subseteq M^{<\infty}$ and mutual goal g_d such that $\mathcal{P} : D \times L_t \longrightarrow Pow(M)$. The elements of $\mathcal{P}(d)$ are called the *legal moves* after d. $\mathcal{P}$ must satisfy the condition that for all legal finite dialogues d and moves m it holds that $d \in D$ and $m \in \mathcal{P}(d)$ iff $d \cup \langle m \rangle \in D$.

Although a protocol filters the set of moves that an agent can make, it is not yet a strategy. Chapter 6 details how agents can further use heuristics and concrete strategies to select the moves to play in a turn. Which protocol rules exactly should be adopted is depending on the specific deliberation context. There are, however, several rules that apply to every deliberation dialogue situation.

Definition 3.14 A protocol $\mathcal{P}$ is a *minimal deliberation protocol* iff for all moves m and all dialogues d with associated mutual goal g_d such that $m \in \mathcal{P}(d)$ it is satisfied that:

R1. $\mathsf{player}(m) = T(d)$,

R2. for every proposal move $m' \in d$ it holds that if $\mathsf{content}(m) = \mathsf{content}(m')$ then $m = m'$,

R3. for every $m' \in d$ where $m \neq m'$, and m and m' are in a single line of dispute, it holds that $\mathsf{content}(m) \neq \mathsf{content}(m')$,

R4. if $\mathsf{content}(m) = \mathsf{prefer}(o, o')$ or $\mathsf{prefer\text{-}equal}(o, o')$ it holds that the resulting option ordering maintains transitivity and antisymmetry,

R5. if $\mathsf{target}(m) = m'$ and $\mathsf{content}(m') = \mathsf{why\text{-}propose}(o)$ then $\mathsf{content}(m) = \mathsf{argue}(A)$ where $\mathsf{conc}(A) = g_d$ and $q \in \mathsf{prem}(A)$.

The rules of a minimal protocol ensure that the dialogue structure is kept consistent and sensible. Rule R1 states that agents only make a move when

they have the turn. Rule R2 ensures that proposals are unique in a dialogue. Rule R3 forbids agents to repeat moves in a single line of dispute. Rule R4 constrains agents to respect the option ordering that is updated by prefer and prefer-equal moves, as specified later in Definition 3.21. Rule R5 enforces that arguments supporting a proposal show how the mutual goal is achieved. This rule is important in demanding the agents contribute to finding an option that could bring about the mutual goal.

There are some additional rules that may be enforced in a deliberation dialogue. Each of them further restrict the freedom of the agents, but can also make the dialogue more focussed. Chapter 9 further explores the effect of protocols on deliberation dialogues.

Definition 3.15 A protocol $\mathcal{P}$ is a *confined deliberation protocol* iff it is a minimal deliberation protocol and for all moves m and all dialogues d such that $m \in \mathcal{P}(d)$ it is satisfied that:

R6. if m is an attacking move it holds that $player(m) \neq player(\mathsf{target}(m))$,

R7. if m is a proposal move then for every proposal move $m' \in d$ where $\mathsf{player}(m) = \mathsf{player}(m')$ there is a move m'' where $\mathsf{id}(m) > \mathsf{id}(m'') > \mathsf{id}(m')$ and $\mathsf{player}(m'') \neq \mathsf{player}(m)$,

R8. m is relevant w.r.t. respectively Definition 3.11 or Definition 3.12 or $\mathsf{content}(m) \in \{\mathsf{skip}, \mathsf{prefer}(o, o'), \mathsf{prefer\text{-}equal}(o, o')\}$.

Rule R6 disallows agents to attack their own moves. Rule R7 restricts agents to make at most one proposal per turn. Finally, rule R8 is used if agents should only play strongly or weakly relevant moves in the dialogue, not considering skip, prefer and prefer-equal moves.

3.5 Dialogue outcome

Every dialogue is supposed to come to an end. A deliberation dialogue terminates if all agents no longer make other moves than directly skipping. This is enforced formally by making the set of legal moves, defined by the protocol, empty.

Definition 3.16 If in a dialogue $d = \langle m_0, \ldots, m_{n-|\mathcal{A}|+1}, \ldots, m_n \rangle$ every $m \in \langle m_{n-|\mathcal{A}|+1}, \ldots, m_n \rangle$ $\mathsf{content}(m) = \mathsf{skip}$ then $\mathcal{P}(d, g_d) = \emptyset$.

The example dialogue of Table 3.2 ended after three consecutive explicit skip moves. The rationale behind the termination rule is that each agent should have the opportunity to make new moves when it still wants to. However, to prevent agents from endlessly skipping until some other agent makes a beneficial move or even a mistake, the number of skip moves is limited.

At any moment in time throughout the dialogue, it is possible to identify the list of proposed options. These are simply those options that have been proposed by the agents with propose moves.

Definition 3.17 The set of *proposed options* for a dialogue d is the set of options $Q_d = \{o | \mathsf{propose}(o) \in d\}$.

In the explanation of move relevance it was already shown that the status of proposal moves is of special interest. It defines whether the option, as contained in a propose move, can be justified or should be doubted because it is either attacked by some argument or it was not discussed at all.

Definition 3.18 An option $o \in O_d$ in any dialogue d is *justifiable* iff $move(o)$ is *in* in d; otherwise it is *disputed*.

Example 3.4 In the dialogue of Example 3.1, the set of proposed options $Q_d = \{\text{goToBistro}, \text{goToPizzeria}\}$. Since for both options their proposing moves are *in*, both options are justifiable.

The actual selection of a dialogue outcome is a complicated matter by itself. In any case, the outcome, which is a statement that some action should be done is one of the options that were proposed in the dialogue.

Definition 3.19 Given a a deliberation dialogue context $\mathcal{DK} = \langle AS, L_t, L_c, \mathcal{P}, \mathcal{A}, g_d \rangle$ and a dialogue d, the dialogue outcome function is a selection function $\mathcal{O} : D \times L_g \longrightarrow L_o$ mapping dialogue d and mutual goal g_d to a single proposed option $o \in Q_d$.

Agents can explicitly state preferences for certain proposed options during the dialogue. They do this by playing prefer and prefer-equal moves, which state that they prefer one option over another or that they have no preference between two options. The explicitly stated preferences can be used to construct a preference ordering for each agent using an option preference relation. Options can also be ordered based on the status of the moves in which they were

proposed. Proposals that are justified might be preferred as dialogue outcome over disputed proposals.

An option preference relation $\preceq$ is a partial order on a set of options $Q \in L_o$, such that $o \preceq o'$ reads that o' is preferred over o. When one option is strictly preferred over another it will be written $o \prec o'$, while $o \approx o'$ means that two options are equally preferred. Using the status of the moves in which the options where proposed, a preliminary ordering can be made.

Definition 3.20 Given a dialogue d and the set of proposed options Q_d, the set of *justifiable options* is denoted $Q_j = \{o | o \in Q_d$ where o is justifiable $\}$. The *preliminarily ordered set of proposed options* $Q_d^{\preceq} \subseteq Q_d \times Q_d$ is defined such that

- for every two options $o, o' \in Q_j$ it holds that $o \approx o' \in Q_d^{\preceq}$,
- for every $o \in Q_d \setminus Q_j$ and $o' \in Q_q$ it holds that $o \prec o' \in Q_d^{\preceq}$.

Although justifiable options are in principle preferred over disputed options, justifiable options should not always directly be selected as winner. The preferences as moved by the agents using prefer and prefer-equal moves should be taken into account as well.

Definition 3.21 Given a dialogue d and the set of proposed options Q_d, every agent $a \in \mathcal{A}$ has a *personally ordered set of proposed options* $Q_a^{\preceq} \subseteq Q_d \times Q_d$ such that for any two options $o, o' \in Q_d$

- $o \prec o' \in Q_a^{\preceq}$ if there is a move $m \in d$ where $\mathsf{content}(m) = \mathsf{prefer}(o, o')$ and $\mathsf{agent}(\mathsf{player}(m)) = A$,
- $o \approx o' \in Q_a^{\preceq}$ if there is a move $m \in d$ where $\mathsf{content}(m) = \mathsf{prefer\text{-}equal}(o_j, o_i)$ and $\mathsf{agent}(\mathsf{player}(m)) = a$.

Note that, since the agents are not forced to state their preferences over all options, this ordering is not necessarily made over the full set of proposed options. However, the minimal deliberation protocol does force an agent to be consistent in its preference utterances with relation to the strict ordering of options.

Example 3.5 In the recurring Example 3.1 both proposed options were justified, so goToBistro $\approx$ goToPizzeria $\in Q_d^{\preceq}$. Agent a explicitly stated in the dialogue that it prefers to go to the bistro, using the prefer(goToBistro, goToPizzeria)

move, hence, goToBistro $\prec$ goToPizzeria $\in Q_a^{\preceq}$. Agent b explicitly stated to not prefer going to the pizzeria over the bistro or vice versa, hence goToBistro $\approx$ goToPizzeria $\in Q_b^{\preceq}$.

When the dialogue terminates, the deliberation dialogue outcome should be selected from the set of options. How this final selection is achieved is totally dependent on the domain and the purpose of the system. For example, there may be an agent authority that chooses the winning option, an additional phase may be introduced in which agents vote on the outcome or a function may be used to aggregate all (preliminary and agent-specific) preference orderings. In any case we need to leave open the option for *mutual disagreement* (McBurney et al., 2007).

Preference aggregation is extensively studied in the field of social choice theory (Pini et al., 2008) and is out of the scope of this work on deliberation dialogues. It is interesting to note, though, that when maximum social welfare is desirable it may be good to incorporate the notion of our option status in the winner selection. The valuable information obtained during the deliberation dialogue can be used with a public calculus. This would decide on the outcome in a way similar to the use of public semantics and would not need to rely on agents considering these notions in their voting strategies. For single agents, this is already studied by Amgoud and Prade (2009). How to make use of this is left as future research.

In the remainder of this thesis we will use a simple dialogue outcome function $\mathcal{O}(d, g_d) = o$ such that o is an arbitrary option $o \in Q_j$. In other words, it selects one of the proposed options that were not disputed.

3.6 Basic fairness and efficiency

A framework for deliberation dialogues serves as a structure for agents to deliberate in some multi-agent system. To properly support deliberation it should not only allow for the playing of moves and picking of some dialogue outcome, but it is supposed to help the agents in doing so. In that perspective, the framework itself should allow for fair, efficient and effective dialogues. Establishing how fair, efficient or effective a dialogue is, is certainly not straightforward. Chapter 7 studies these desirable properties. However, it is already possible to establish some basic fairness, efficiency and effectiveness properties based on the dialogue framework itself and the protocol defined therein. McBurney et al.

(2002) have proposed a set of 13 desiderata for argumentation-based dialogue frameworks. These are criteria which dialogue frameworks and their protocols need to adhere to to achieve basic fairness, efficiency and effectiveness. Checking our deliberation framework against these desiderata already gives an insight into the capability of the framework in supporting multi-agent deliberation. The 13 desiderata are now listed by their reference name and an explanation is given how the framework of this chapter adheres to the specific desideratum.

1. **Stated Dialogue Purpose** The protocol is explicitly designed to decide on a course of action.

2. **Diversity of individual purpose** Agents are allowed to have personal goals that possibly conflict with the stated mutual goal.

3. **Inclusiveness** Many agents can join the deliberation dialogue and no roles are enforced upon them.

4. **Transparency** The rules of our framework are fully explained, but it is up to an implementation to make sure every agents knows these rules and knows how to play the game.

5. **Fairness** Every agent has equal rights in the dialogue and the framework allows for fair winner selection methods. Since an agent may always choose not to move (any more) at all, it is never forced to adopt or drop some belief or goal. Turn taking order doesn't influence evaluation and selection of options as outcome.

6. **Clarity of Argumentation Theory** The reply structure and notion of relevance in our framework are not hidden implicitly in a protocol but made explicit. Moreover, the structure of arguments is formalised in an explicitly defined argumentation logic and topic language.

7. **Separation of Syntax and Semantics** The communication language is separately defined from the protocol. Also, dialogues in the framework are independent of the agent specification while their public behaviour can still be monitored.

8. **Rule Consistency** We have not studied the rule consistency in detail, but the protocol will never lead to deadlocks; agents can always skip their turn or make a new proposal and within a proposal tree there is always a way to make a new contribution.

9. **Encouragement of Resolution** Agents are encouraged to stay focussed on the dialogue topic through the notion of relevance. If agents still have something to say, there is always the opportunity to do so.

10. **Discouragement of Disruption** Disruption is discouraged through the definition of legal speech acts, which are separated in attacks and replies. This restricts the available moves, for example agents cannot attack their own moves. However, it is still possible for aggressive agents to question everything that is claimed and no agent is compelled to accept any claim.

11. **Enablement of Self-Transformation** Agents are allowed to adjust their beliefs or goals depending on the arguments that are moved and preferences that are expressed. Moreover, they are allowed to drop proposals and to retract or concede claims.

12. **System Simplicity** Simplicity of the system is hard to prove or disprove. However, it is highly modular; communication and topic languages are separated and various alternative protocol rules may be adopted or dropped. The winner function is left unspecified, but this may range from a dictator agent to a social welfare-based function.

13. **Computational Simplicity** This will be further studied in Chapter 8.

Following the desiderata, the presented dialogue framework is compatible with some basic requirements for a deliberation system. However, the dialogue framework is only one part of a full deliberation system, as it models merely the structure of the dialogues and how these dialogues can be evaluated. To test argumentation in the deliberation system, actual deliberation situations need to be supplied, as well as agents that can play these.

Chapter 4

Scenario generation

In the past chapter, deliberation was considered as a type of dialogue. The proposed model allows agents to put forward and discuss proposals, so as to mutually deliberate which action to pursue. This chapter will look at the source of proposals, of arguments that are moved and the reasons for agents to play them, represented by knowledge about options that can be proposed, beliefs about the world and personal and shared goals. This knowledge, available at the start of the dialogue, is contained in a scenario.

Scenarios form a vital part in testing the role of argumentation in deliberation dialogues. The knowledge associated with a scenario is all that the agents have to reason over and deliberate with. Therefore, the scenario has a major influence on the course of the dialogue and as an extension on the results of an experiment with argumentation-based deliberation dialogues. As a result, a proper analysis of deliberation scenarios is required before the benefits of argumentation can be experimentally tested in Chapter 8 onwards. In particular, a way is required to generate the scenarios, so agents have meaningful knowledge to deliberate with.

Generating scenarios starts with a proper analysis of the structure of knowledge behind a deliberation situation. Section 4.1 investigates the characteristics of deliberation dialogues. Based on that analysis, a structured and meaningful method to generate deliberation scenarios is introduced. Through the concept of roles and the idea of rule chaining, in Section 4.2, scenarios are generated, through which rules, beliefs, goals and options are allocated to agents in Section 4.3. The resulting scenarios should reflect the characteristics of real world deliberation issues, in order to be useful for experiments with realistic deliberating agent strategies.

4.1 Characteristics of deliberation

The source of knowledge behind a deliberation dialogue is rooted in the meta-goal of deliberation as dialogical construct. As discussed in the dialogue typology in Section 2.2.2, deliberation has the dialogical goal to mutually reach a decision on a course of action. Arguably, a formal dialogue model cannot intrinsically have a goal, as it is merely a structure that serves the participants. However, the deliberation dialogue does have a goal from the multi-agent system perspective. Indeed, the model introduced in the previous chapter presupposed that the agents agreed on pursuing mutual agreement in a proactive manner. The dialogue goal of reaching agreement is therefore encoded in the agents themselves as the mutual goal to which their proposals should adhere.

The mutual goal is not the only thing that agents share in a deliberation dialogue. Typically the agents share a common body of knowledge as well, including world beliefs and possibilities for action. As argued by Walton (2007) as well as McBurney et al. (2007), goals, beliefs and knowledge of actions is dispersed among the agents in both an overlapping, complementary and contradictory way. In fact, the origins of knowledge in deliberation issues can be split even further, namely in knowledge from the agent system, from the role of an agent and personal knowledge. A method to generate realistic deliberation issues will thus have to provide knowledge from all three of these sources. Consider an agent with a role as engineer in a car company. The company goal to make profit is augmented with a goal to produce a reliable car from its role, while a personal goal might be to want to respect the environment. The role will also come with unique world beliefs, such as how an engine works, and possible actions that other agents might not know of, since they have other roles.

While knowledge is dispersed over dialogue participants, the possible actions, goals and beliefs that a single participant has are typically quite coherent. That is, an agent does not have random knowledge about options or unrelated inference rules. Rather, there is a coherence between an agent's goals and the options that it knows about. The marketing analyst agent's goal to make profit is linked to beliefs that support this goal, possibly even tied to a specific action. For example, it knowns that producing a small luxury car results in high margins, which promotes the goal to make profit.

Although an agent's knowledge has a clear cohesion, this does not mean that the agent has complete knowledge. In fact, it is often the case that small but essential bits of knowledge are missing. For instance, though the market

analysis agent knows about small luxury cars having potential for profit, it may miss the link between those small cars also being environmentally friendly. This illustrates that agent will have omissions in their knowledge regarding the complete deliberation issue. A scenario generation process has to reflect this aspect.

In conclusion, to generate deliberation scenarios in a structured way, the following characteristics need to be respected:

- Shared deliberation goal
- Unequal roles between agents
- Both shared and personal goals
- Dispersion of knowledge
- Incomplete information

4.2 Rule chains and conflicts

The main concept behind the generation scenario process is that of rule chaining. The idea is that agents reason about options by taking into consideration which goals can be achieved. The promise that executing a certain option realizes an agent's goal is connected through a line of practical reasoning. This approach is similar to that of Walton (2007), who has studied the connection between practical reasoning and deliberation as dialogical issue from a philosophical standpoint. This connection was discussed further in Section 2.1.3.

4.2.1 Rule chaining

For the purpose of generating scenarios, lines of reasoning between options and goals are modelled as chains of defeasible inference rules. Given an option, inference rules are created so that the conclusion of the final rule is some goal. The set of created inference rules is called a rule chain, as the rules form a chain that binds an option to some goal.

Definition 4.1 Given a goal g, an option o, a set of beliefs $S = \{p_1, \ldots, p_n\}$ and a chain length l, a *rule chain* is a set of rules $C^S_{g,o}$ such that

- if $l = 1$ then $C^S_{g,o} = \{o \overset{o}{\Rightarrow} g\}$

- if $l > 1$ then $C^S_{g,o} = \{o \overset{\varrho 1}{\Rightarrow} p_1,\ \ldots,\ p_i \overset{\varrho i}{\Rightarrow} p_j,\ \ldots,\ p_n \overset{\varrho l}{\Rightarrow} g\}$ where $n = l-1$

A chain starts with a rule that has an option as antecedent and ends with a rule that has a goal as consequence. The consequence to all but the last rule, if there are more, is an atom from a set of beliefs and this atom in turn is the antecedent for the follow-up rule. When the length of a chain is one, then the only possible rule that can be created is by linking the option and goal directly, that is, $o \overset{\varrho}{\Rightarrow} g$. For longer chains, different intermediate atoms are used using the beliefs in S. For readability, the set of beliefs on the basis of a chain will below be omitted in the notation if there is no danger of confusion, for example $C_{g,o}$.

Example 4.1 Consider an option o and two goals g_1 and g_2. Various rule chains can be constructed that link the option to one of the two goals. Given a rule length $l = 3$ and belief sets $S_1, S_2, S_3 \subset \{p_1, p_2, p_3, p_4, p_5, p_6\}$, possible rule chains include

$$C^{S_1}_{o,g_1} = \{o \overset{\varrho 1}{\Rightarrow} p_3,\ p_3 \overset{\varrho 2}{\Rightarrow} p_2,\ p_2 \overset{\varrho 3}{\Rightarrow} g_1\}$$

$$C^{S_2}_{o,g_2} = \{o \overset{\varrho 4}{\Rightarrow} p_5,\ p_5 \overset{\varrho 5}{\Rightarrow} p_1,\ p_1 \overset{\varrho 6}{\Rightarrow} g_2\}$$

$$C^{S_3}_{o,g_2} = \{o \overset{\varrho 4}{\Rightarrow} p_5,\ p_5 \overset{\varrho 7}{\Rightarrow} p_2,\ p_2 \overset{\varrho 8}{\Rightarrow} g_2\}$$

Given an ASPIC− argumentation system AS, as introduced in Chapter 2, with $R_d = C^{S_1}_{o,g_1} \cup C^{S_2}_{o,g_2} \cup C^{S_3}_{o,g_2}$ and knowledge base $K = \{o\}$, arguments that can be constructed include

$$A'' = \frac{o}{p_3}\,\varrho_1 \qquad B'' = \frac{o}{p_5}\,\varrho_4$$
$$A' = \frac{p_2}{p_2}\,\varrho_2 \qquad B' = \frac{p_5}{p_1}\,\varrho_5$$
$$A = \frac{p_1}{g_1}\,\varrho_3 \qquad B = \frac{p_1}{g_2}\,\varrho_6$$

$$C'' = \frac{o}{p_5}\,\varrho_4 \qquad D'' = \frac{o}{p_5}\,\varrho_4$$
$$C' = \frac{p_5}{p_2}\,\varrho_7 \qquad D' = \frac{p_5}{p_2}\,\varrho_7$$
$$C = \frac{p_2}{g_2}\,\varrho_8 \qquad D = \frac{p_2}{g_1}\,\varrho_3$$

Arguments A, B and C are directly related to the three rule chains that were used as set of defeasible rules. On the other hand, the rules can give rise to different arguments as well, that still connect an option to some goal. This is shown by argument D, which exploits rules from $C^{S_1}_{o,g_1}$ and $C^{S_3}_{o,g_2}$ to construct an argument for g_1.

The rules that are constructed to create chains might seem simplistic. Only one antecedent per rule is used, and that is never negated. However, these chains are already sufficiently complex to end up with interesting scenarios. This will be shown by means of a software experiment in Chapter 5.

4.2.2 Conflict generation

Rule chains provide a way to generate coherent sets of rules that tie an option to some goal. Intuitively, an agent can use these sets to draw conclusions on being able to reach a goal, given that some action will be performed. In contrast, an agent might know why the execution of an action might not lead to the realization of a goal. That information is said to conflict with the rule chain.

Conflicts are caused by negating a rule in a chain or negating an atom that appears in a rule. The reason for this is that an argument with such a negation as conclusion would attack an argument constructed using the rule chain.

Example 4.2 For example, consider again the example rule chain $C^{S_1}_{o,g_1} = \{o \overset{\varrho 1}{\Rightarrow} p_3, p_3 \overset{\varrho 2}{\Rightarrow} p_2, p_2 \overset{\varrho 3}{\Rightarrow} g_1\}$, with which potentially an argument can be constructed such that

$$\begin{aligned} A'' &= \dfrac{o}{p_3}\,\varrho_1 \\ A' &= \dfrac{p_3}{p_2}\,\varrho_2 \\ A &= \dfrac{p_2}{g_1}\,\varrho_3 \end{aligned}$$

To attack this argument A, a counterargument B is needed that undercuts one of the rule applications, undermines a premise or rebuts a (sub-)conclusion. That is, the counterargument B should have conclusion $\mathsf{conc}(B) = \neg o$, $\mathsf{conc}(B) = \neg\varrho_1$, $\mathsf{conc}(B) = \neg\varrho_2$, $\mathsf{conc}(B) = \neg\varrho_3$, $\mathsf{conc}(B) = \neg p_3$, $\mathsf{conc}(B) = \neg p_2$ or $\mathsf{conc}(B) = \neg g_1$. The structure of such a counterargument can be anything from a trivial atomic argument to a complex compound argument with various premises and rule applications. Two example counterarguments to A are compound argument B and atomic argument C such that

$$\begin{aligned} B'' &= \dfrac{p_5}{p_3}\,\varrho_4 \\ B' &= \dfrac{p_3}{p_6}\,\varrho_5 \qquad C = \neg\varrho_1 \\ B &= \dfrac{p_6}{\neg\varrho_1}\,\varrho_6 \end{aligned}$$

Note that while C merely requires a fact $\neg\varrho_1$ in the knowledge base for an agent to construct this argument, for argument B an agent needs to know about a fact p_5 as well as rules ϱ_4, ϱ_5 and ϱ_6 to draw conclusion $\neg\varrho_1$.

To allow agents to construct arguments with relevant negated conclusions, they will need appropriate knowledge in the form of beliefs and rules. This is realized through the concept of possible conflicts. Every rule in a rule chain has a set of associated conflicts, which are subsequently used to assign relevant knowledge to construct arguments with negated conclusions. Intuitively, the possible conflicts are those atoms that need to be negated for a possible conflict to arise.

Definition 4.2 A rule chain $C_{o,g}$, linking some goal g and option o, has a set of *possible conflicts* $\bar{C}_{o,g}$, being the smallest possible set containing for every rule $p \overset{\varrho}{\Rightarrow} q \in C_{o,g}$

- an atom $\neg\varrho$
- an atom $\neg p$ if $p \in L_b$
- an atom $\neg q$ if $q \in L_b$

Specifically not part of a set of possible conflicts are the negated option and goal for which a chain was generated. The negation of an option does not translate into an intuitive conflict. It states that a proposal for action is not the case, such as in ¬goToBistro. Although this is a legal statement, this direct negation without evidential backup is not part of a typical deliberation issue and is therefore omitted from the set of conflicts. Similarly, negated goals are omitted as well. Negated goals, such as ¬enjoyDinner, without factual backup, are not useful in deliberation dialogues. This would state that there is no such goal. In case of the mutual deliberation goal, which every agent will have, the goal is always present. Alternatively, it is equally not useful to play the negation of some agent's personal goal, as there is no ultimate truth status to a goal. In this thesis, agents will not argue with their personal goals and negating them is therefore not useful.

Note that the atoms in a set of possible conflicts define merely which atoms would possibly provide a way to counter attack an argument associated with a rule chain. It is not yet defined how agents are assigned appropriate knowledge to be capable of constructing arguments with these possible conflicts as conclusion. Two methods to do so will be explained in Section 4.3.3.

4.3 Knowledge allocation

A model for the generation of deliberation scenarios is now introduced. It uses the notion of rule chaining to construct scenarios that reflect the characteristics of deliberation dialogues as studied above. Specifically, scenario generation is the assignment of rules, beliefs, goals and options to agents in a structured way.

4.3.1 Context

Generation of scenarios happens in a specific context. Most importantly, the context determines the language domain of the deliberation scenario. Agents do not have a random knowledge of formulas from the topic language. Instead, the domain is restricted to a specific set of rules, beliefs, goals and options. It is from this restricted domain that knowledge will be further allocated to roles and, subsequently, to agents. Although knowledge is dispersed amongst agents, the restricted domain ensures a level of cohesion that is important in ending up with interesting dialogues.

The context also contains the mutual deliberation goal. That goal, as typical in deliberation, is shared amongst all the agents. It is used in the structured generation of rule chains and will be allocated directly to the agents, to ensure that they aspire to contribute in a constructive manner to the deliberation process.

Definition 4.3 Given the deliberation topic language $L_t = L_b \cup L_o \cup L_g$ of Definition 2.15, a *scenario generation context* is a tuple $\mathcal{SK} = \langle R_{\mathcal{SK}}, B_{\mathcal{SK}}, G_{\mathcal{SK}}, O_{\mathcal{SK}}, g_d, \mathcal{C}, \vec{\mathcal{C}} \rangle$ where

- $R_{\mathcal{SK}}$ is the set of all possible defeasible rules in an ASPIC− framework given logical language L,
- $B_{\mathcal{SK}} \subseteq L_b$ is a seed set of beliefs,
- $G_{\mathcal{SK}} \subseteq L_g$ is a seed set of goals,
- $O_{\mathcal{SK}} \subseteq L_o$ is a seed set of options,
- g_d is a mutual deliberation goal, such that $g_d \in L_g$,
- $\mathcal{C}$ is a chaining function $\mathcal{C} : Pow(L_g) \times Pow(L_b) \times L_o \longrightarrow Pow(R)$ mapping a set of goals G, a set of beliefs B and an option o to a specific rule chain $C^S_{g,o}$ such that $g \in G$ and $S \subseteq B$,

- $\bar{\mathcal{C}}$ is a conflict chaining function $\vec{\mathcal{C}} : Pow(L_b) \times L_b \longrightarrow Pow(R)$ mapping a set of beliefs B and a belief c to a specific rule chain $\vec{C}^S_{p,c}$ such that $p \in B$, $S \subseteq B$ and $p \notin S$.

The unique chaining and unique conflict chaining functions are used to construct unique rule chains on the basis of a set of beliefs, goals and options. Their use is further explained below, where rules and beliefs are allocated to roles.

4.3.2 Goal and option allocation

In typical deliberation situations, agents have knowledge from both the role that they play as well as personal knowledge. This is formalized using an explicit set of roles, from which agents are later assigned one. Roles are modelled as simple structures that represent some feature or duty that is part of a multi-agent system, an approach frequently advocated, such as by Wooldridge et al. (2000). Associated with a role are a set of defeasible rules, a set of beliefs, a set of goals and a set of options. Beliefs and rules will be generated using rule chaining, but first the goals and options are assigned from the scenario generation context.

Definition 4.4 Given a scenario generation context $\mathcal{SK} = \langle R_{\mathcal{SK}}, B_{\mathcal{SK}}, G_{\mathcal{SK}}, O_{\mathcal{SK}}, g_d, \mathcal{C}, \bar{\mathcal{C}} \rangle$, the set of *roles* $\mathcal{R}$ is defined such that every role $r \in \mathcal{R}$ is a tuple $r = \langle R_r, B_r, G_r, O_r \rangle$, such that

- $R_r \subseteq R_{\mathcal{SK}}$ is a set of defeasible rules
- $B_r \subseteq B_{\mathcal{SK}}$ is a set of beliefs
- $G_r \subseteq G_{\mathcal{SK}}$ is a set of goals
- $O_r \subseteq O_{\mathcal{SK}}$ is a set of options

Goals and options allocated to a role are taken directly from the context seed sets. Rules and beliefs, on the other hand, are allocated in a structural fashion as defined below.

Example 4.3 Consider an example scenario with three agents, which are assigned one of two roles. Goals and options are taken from the knowledge available in the scenario generation context.

$\mathcal{A}$	a_1, a_2, a_3
$\mathcal{R}$	r_1, r_2
$B_{\mathcal{SK}}$	$p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}$
$G_{\mathcal{SK}}$	g_1, g_2, g_3, g_4, g_5
$O_{\mathcal{SK}}$	o_1, o_2, o_3
G_{r_1}	g_1, g_d
G_{r_2}	g_3, g_4
O_{r_1}	o_1, o_2
O_{r_2}	o_2, o_3

For the purpose of generating deliberation scenarios, agents are abstract entities that hold knowledge in the form of rules, beliefs, goals and options. The basic idea is that agents are assigned a role and inherit the role's options and goals. If agents share roles, knowledge will be shared amongst them as a result. On top of this, the agents are assigned several personal goals, called non-role originating goals. This further underlines the dispersion of knowledge between agents, even amongst those sharing a role.

Definition 4.5 Given scenario generation context $\mathcal{SK} = \langle R_{\mathcal{SK}}, B_{\mathcal{SK}}, G_{\mathcal{SK}}, O_{\mathcal{SK}}, g_d \rangle$ and the set of roles $\mathcal{R}$, the set of *agents* $\mathcal{A}$ is defined such that every agent $a \in \mathcal{A}$ is a tuple $a = \langle r, R_a, B_a, G_a, O_a \rangle$, such that

- $r \in \mathcal{R}$
- $R_a \subseteq R_{\mathcal{SK}}$
- $B_a \subseteq B_r$
- $O_a = O_r$
- $G_a^{\bar{r}} \subseteq G_{\mathcal{SK}} \setminus G_r$
- $G_a = G_r \cup G_a^{\bar{r}} \cup \{g_d\}$

The rule and beliefs are, as with roles, allocated in a structural fashion, as will be explained below. Options are inherited directly from the role, as are the role's goals G_r, which are therefore also called the role-originating goals. Additionally, there is a set of non-role originating goals $G_a^{\bar{r}}$, which are not inherited from the role of the agent. Finally, the mutual goal g_d is always assigned to an agent.

4.3.3 Belief allocation

The final step in the generation process is the allocation of rules and beliefs to roles and subsequently to agents. Rule and belief allocation to a role is done by generating rule chains for each of the known options to some of the role's goals. The construction of such rule chains is done by the context's unique chaining function $\mathcal{C}$. Recall that this function constructs a rule chain on the basis of a set of goals, from which one will be picked, a set of beliefs and a specific option. Note that the function does not yet make concrete which goal is selected from the input set of goals or which beliefs from the input set are used to construct the rule chain. A unique chaining function implementation that fully specifies the chain construction behaviour is given in the next chapter.

Chains can now be constructed to form the role's rule knowledge. For each of the options allocated to a role, a chain is generated to one of the role's personal goals. Additionally, a rule chain is constructed for every option to the mutual deliberation goal g_d. The rules of these two chains form the set of so called role-option rules.

Definition 4.6 Given a scenario generation context with mutual goal g_d, every pair of a role $r \in \mathcal{R}$ and an option $o \in O_r$ is associated with, on the basis of $C_{o,g} = \mathcal{C}(G_r, B_{\mathcal{SK}}, o)$ and $C_{o,g_d} = \mathcal{C}(\{g_d\}, B_{\mathcal{SK}}, o)$, a set of *role-option rules*

$$R_r^o = C_{o,g} \cup C_{o,g_d}$$

Not every option in the context is assigned to a role. While for options assigned to a role the chain is allocated, for options not assigned to a role the role will be allocated possible conflicts. As described above, conflicts are used to introduce the ability for agents to construct counterarguments. The set of possible conflicts associated with a certain rule chain contains negated atoms. For an agent to construct arguments with one of the negated atoms as conclusion, it needs to be allocated appropriate knowledge. Two methods are used in this thesis to provide this knowledge. First, a trivial method is introduced that allocates the various possible conflict atoms directly as facts. Second, a rule-chain-based method is provided, which allocates facts and rules by constructing chains with a negated atom as consequence.

The simplest way to allocate appropriate knowledge for possible conflicts is by directly assigning the negated atoms as facts. This might seem simplistic at first, but it does already allow for scenarios where practical arguments, as

constructed with rule chains, can be attacked by a direct counterargument. An example will be given at the end of this chapter, while Chapter 5 experimentally evaluates how effective this type of allocation of possible conflicts is.

Definition 4.7 Every pair of a role $r \in \mathcal{R}$ and an option $o \in O_{\mathcal{SK}} \setminus O_r$ is, on the basis of the set of possible conflicts $\bar{C}_{o,g}$ associated with rule chain $C_{o,g} = \mathcal{C}(G_r, B_{\mathcal{SK}}, o)$, assigned a set of *direct role-option beliefs*

$$B_r^{\bar{o}} = \bar{C}_{o,g}$$

The alternative to direct allocation of the possible conflicts as facts is to instead generate a rule chain for every possible conflict. In contrast with the rule chains used above, which start with an option and end with a goal, the negated atom representing a possible conflict is used as a rule chain's consequence, while a normal belief is used as starting point for that chain.

Definition 4.8 Given a possible conflict $c \in \bar{C}_{o,g}$ associated with a rule chain $C_{o,g}$, a set of beliefs S, a *starting belief* $p \in S$ and a chain length l, a *conflict chain* is a set of rules $\vec{C}_{p,c}$ such that

- if $l = 1$ then $\vec{C}_{p,c} = \{p \overset{o}{\Rightarrow} c\}$
- if $l > 1$ then $\vec{C}_{p,c} = \{p \overset{o1}{\Rightarrow} p_1,\ \ldots,\ p_i \overset{oi}{\Rightarrow} p_j,\ \ldots,\ p_n \overset{ol}{\Rightarrow} c\}$ where $n = l - 1$ and $\{p_1, \ldots, p_n\} = S$

Note that the possible conflict c used in this definition is a negated atom and that starting belief p is a normal belief taken from the seed set of beliefs in the scenario generation context. Construction of conflict chains is achieved through the above defined conflict chaining function $\vec{\mathcal{C}}$, which maps a set of beliefs and a conflict to a unique conflict rule chain. A conflict chain is assigned to those options that were not assigned to the role.

Definition 4.9 Every triple of a role $r \in \mathcal{R}$, an option $o \in O_{\mathcal{SK}} \setminus O_r$ and a conflict $c \in \bar{C}_{o,g}$ associated with rule chain $C_{o,g} = \mathcal{C}(G_r, B_{\mathcal{SK}}, o)$ is, on the basis of conflict chain $\vec{C}_{p,c} = \vec{\mathcal{C}}(B_{\mathcal{SK}}, c)$, assigned a set of *chained role-option-conflict rules and beliefs*

$$R_r^{\vec{o},c} = \vec{\mathcal{C}}_{p,c} \qquad\qquad B_r^{\vec{o},c} = \{p\}$$

such that the set of *chained role-option rules and beliefs* are defined

$$R_r^{\vec{o}} = \bigcup_{c \in \bar{C}_{o,g}} R_r^{\vec{o},c} \qquad\qquad B_r^{\vec{o}} = \bigcup_{c \in \bar{C}_{o,g}} B_r^{\vec{o},c}$$

In conclusion, roles have direct (rules) or indirect (beliefs) knowledge about options. For every option allocated to a role, a set of defeasible rules is assigned by generating rule chains for the mutual goal as well as a personal goal. For every option not allocated to a role, two approaches are given to allocate knowledge relevant to the possible conflict associated with the rule chain for an option. In case the direct assignment method is used, negated facts are allocated to the agent. With the chained assignment method, a conflict chain of defeasible rules is constructed for every possible conflict associated with the rule chain for the option. Hence, roles are always assigned some relevant knowledge for an option, which helps to ensure knowledge cohesion in the scenario.

Example 4.4 Consider again the example scenario of 4.3. Role-option rules and beliefs are constructed for every of the role's options by providing either the two rule chains (for g_d and an arbitrary other goal g) or a set of beliefs and rules associated with possible conflicts. Every rule chain has length 3, that is, three defeasible rules per chain are used.

	r_1
$R^{o_1}_{r_1}$	$o_1 \overset{\varrho 1}{\Longrightarrow} p_5,\ p_5 \overset{\varrho 2}{\Longrightarrow} p_2,\ p_2 \overset{\varrho 3}{\Longrightarrow} g_2,$
	$o_1 \overset{\varrho 4}{\Longrightarrow} p_6,\ p_6 \overset{\varrho 5}{\Longrightarrow} p_4,\ p_4 \overset{\varrho 6}{\Longrightarrow} g_d$
$R^{o_2}_{r_1}$	$o_2 \overset{\varrho 7}{\Longrightarrow} p_5,\ p_5 \overset{\varrho 2}{\Longrightarrow} p_2,\ p_2 \overset{\varrho 8}{\Longrightarrow} g_1,$
	$o_2 \overset{\varrho 9}{\Longrightarrow} p_9,\ p_9 \overset{\varrho 10}{\Longrightarrow} p_1,\ p_1 \overset{\varrho 11}{\Longrightarrow} g_d$
$B^{\bar{o_3}}_{r_1}$	$\neg\varrho_{17}, \neg p_3$
$R^{\vec{o_3}}_{r_1}$	$p_9 \overset{\varrho 23}{\Longrightarrow} p_{12},\ p_{12} \overset{\varrho 24}{\Longrightarrow} p_7,\ p_7 \overset{\varrho 25}{\Longrightarrow} \neg\varrho_{17},$
	$p_{12} \overset{\varrho 26}{\Longrightarrow} p_1,\ p_1 \overset{\varrho 27}{\Longrightarrow} p_{11},\ p_{11} \overset{\varrho 12}{\Longrightarrow} \neg p_3$
$B^{\vec{o_3}}_{r_1}$	p_9, p_{12}
	r_2
$R^{o_2}_{r_2}$	$o_2 \overset{\varrho 9}{\Longrightarrow} p_9,\ p_9 \overset{\varrho 12}{\Longrightarrow} p_8,\ p_8 \overset{\varrho 13}{\Longrightarrow} g_4,$
	$o_2 \overset{\varrho 14}{\Longrightarrow} p_1,\ p_1 \overset{\varrho 15}{\Longrightarrow} p_9,\ p_9 \overset{\varrho 16}{\Longrightarrow} g_d$
$R^{o_3}_{r_2}$	$o_3 \overset{\varrho 17}{\Longrightarrow} p_7,\ p_7 \overset{\varrho 18}{\Longrightarrow} p_3,\ p_3 \overset{\varrho 19}{\Longrightarrow} g_4,$
	$o_3 \overset{\varrho 17}{\Longrightarrow} p_7,\ p_7 \overset{\varrho 21}{\Longrightarrow} p_8,\ p_8 \overset{\varrho 22}{\Longrightarrow} g_d$
$B^{\bar{o_1}}_{r_2}$	$\neg p_2, \neg\varrho_3$
$R^{\vec{o_1}}_{r_2}$	$p_{12} \overset{\varrho 26}{\Longrightarrow} p_7,\ p_7 \overset{\varrho 21}{\Longrightarrow} p_8,\ p_8 \overset{\varrho 27}{\Longrightarrow} \neg p_2,$
	$p_{10} \overset{\varrho 28}{\Longrightarrow} p_6,\ p_6 \overset{\varrho 5}{\Longrightarrow} p_4,\ p_4 \overset{\varrho 29}{\Longrightarrow} \neg\varrho_3$
$B^{\vec{o_1}}_{r_2}$	p_{12}, p_{10}

Finally, the agents' rules and beliefs can be allocated. As with the goal allocation, rules and beliefs are dispersed among both roles and individual agents. This is modelled by the allocation of role and non-role-originating rules and beliefs to the agents.

Whether it be role or non-role-originating rules and beliefs, agents do not have complete knowledge. In fact, agents will miss, sometimes essential, knowledge, which is a characteristic of deliberation scenarios. This is true for both rules that come from constructed rule chains and knowledge regarding possible conflicts. For this reason, the number of rules and beliefs that are allocated to the agent is defined as a subset of the role's rules and beliefs.

Definition 4.10 An agent $a = \langle r, R_a, B_a, G_a, O_a \rangle$ with some role r is assigned a set of *role-originating rules and beliefs*

$$R_a^r \subseteq \bigcup_{o \in O_{S\mathcal{K}}} R_r^o \qquad\qquad B_a^r \subseteq \bigcup_{o \in O_{S\mathcal{K}}} B_r^{\bar{o}}$$

if the direct possible conflict allocation is used, or

$$R_a^r \subseteq \bigcup_{o \in O_{S\mathcal{K}}} R_r^o \cup R_r^{\vec{o}} \qquad\qquad B_a^r \subseteq \bigcup_{o \in O_{S\mathcal{K}}} B_r^{\vec{o}}$$

if the chained conflict allocation method is used.

The body of non-role-originating rules and beliefs, which is unique for a single agent, is typically incomplete as well. For every option assigned to the agent, a new chain is generated with defeasible rules and associated set of possible conflicts. A subset of those rules and beliefs is then selected.

Definition 4.11 An agent $a = \langle r, R_a, B_a, G_a, O_a \rangle$ is for every option $o \in O_a$, on the basis of $C_{o,g} = \mathcal{C}(G_a, B_{S\mathcal{K}}, o)$, assigned a set of *non-role-originating rules*

$$R_a^{\bar{r}} \subseteq \bigcup_{o \in O_a} C_{o,g}$$

For those options that were not assigned to the agent, the agent is instead given knowledge in the form of some negated fact or a conflict chain, in the same fashion as belief allocation to roles. As many negated facts or conflict chains are used here as there are options not allocated to the agent.

Definition 4.12 Every pair of an agent $a = \langle r, R_a, B_a, G_a, O_a \rangle$ and an option $o \in O_K \setminus O_a$ is, on the basis of a set of beliefs

$$S \subseteq B_{\mathcal{SK}} \text{ such that } |S| = |O_{\mathcal{SK}} \setminus O_a|$$

assigned a set of *non-role-originating option rules and beliefs*

$$R_a^{\bar{r}} = \emptyset \qquad\qquad B_a^{\bar{r}} \subseteq \bigcup_{p \in S} \neg p$$

if the direct possible conflict allocation is used, or, for every $p \in S$ and associated conflict chain $\vec{C}_{q,\neg p} = \vec{\mathcal{C}}(B_{\mathcal{SK}}, \neg p)$

$$R_a^{\bar{r},p} = \vec{C}_{q,\neg p} \qquad\qquad B_a^{\bar{r},p} = q$$

such that

$$R_a^{\bar{r}} \subseteq \bigcup_{p \in S} R_a^{\bar{r},p} \qquad\qquad B_a^{\bar{r}} \subseteq \bigcup_{p \in S} B_a^{\bar{r},p}$$

if the chained conflict allocation method is used.

As a result, the agents are allocated rules and beliefs from their role, but are also assigned personal knowledge, as is characteristic in deliberation situations. Note that the personal knowledge might be incomplete, as subsets of the generated rule chains and possible conflicts are used. To wrap up, the role and non-role-originating rules and beliefs are combined to complete the agents' knowledge allocation.

Definition 4.13 An agent $a = \langle r, R_a, B_a, G_a, O_a \rangle$ is assigned a set of *agent rules* and a set of *agent beliefs*

$$R_a = R_a^r \cup R_a^{\bar{r}} \qquad\qquad B_a = B_a^r \cup B_a^{\bar{r}}$$

Example 4.5 Consider again the example scenario between three agents. Agents a_1 and a_2 have role r_1, while agent a_3 has role r_2. Both personal and role knowledge is assigned. Recall that an agent is assigned its role's goals, personal goals and the mutual goal g_d. Options are directly inherited from the role. Both role and non-role-originating rules and beliefs are allocated, using the direct possible conflict assignment method. Single the agents have incomplete knowledge, some rules from the chains are missing, as well as some possible conflicts.

	a_1	
O_{a_1}		o_1, o_2
G_{a_1}		g_d, g_1, g_2, g_4
R_{a_1}	$R^{r_1}_{a_1}$	$o_1 \overset{\varrho 1}{\Longrightarrow} p_5,\ p_5 \overset{\varrho 2}{\Longrightarrow} p_2,\ p_2 \overset{\varrho 3}{\Longrightarrow} g_2,$
		$o_1 \overset{\varrho 4}{\Longrightarrow} p_6,\ p_6 \overset{\varrho 5}{\Longrightarrow} p_4,\ p_4 \overset{\varrho 6}{\Longrightarrow} g_d,$
		$o_2 \overset{\varrho 7}{\Longrightarrow} p_5,\ p_5 \overset{\varrho 2}{\Longrightarrow} p_2,$
		$o_2 \overset{\varrho 9}{\Longrightarrow} p_9,\ p_9 \overset{\varrho 10}{\Longrightarrow} p_1,$
	$R^{\bar{r_1}}_{a_1}$	$o_1 \overset{\varrho 23}{\Longrightarrow} p_{10},\ p_8 \overset{\varrho 25}{\Longrightarrow} g_3$
		$o_2 \overset{\varrho 26}{\Longrightarrow} p_3,\ p_5 \overset{\varrho 27}{\Longrightarrow} g_2,$
B_{a_1}	$B^{r_1}_{a_1}$	$\neg\varrho_{17}, \neg p_3$
	$B^{\bar{r_1}}_{a_1}$	$\neg\varrho_{27}$
	a_2	
O_{a_2}		o_1, o_2
G_{a_2}		g_d, g_1, g_2, g_3
R_{a_2}	$R^{r_1}_{a_2}$	$p_5 \overset{\varrho 2}{\Longrightarrow} p_2,\ p_2 \overset{\varrho 3}{\Longrightarrow} g_2,$
		$o_1 \overset{\varrho 4}{\Longrightarrow} p_6,\ p_6 \overset{\varrho 5}{\Longrightarrow} p_4,\ p_4 \overset{\varrho 6}{\Longrightarrow} g_d,$
		$o_2 \overset{\varrho 7}{\Longrightarrow} p_5,\ p_5 \overset{\varrho 2}{\Longrightarrow} p_2,\ p_2 \overset{\varrho 8}{\Longrightarrow} g_1,$
		$o_2 \overset{\varrho 9}{\Longrightarrow} p_9,\ p_9 \overset{\varrho 10}{\Longrightarrow} p_1,\ p_1 \overset{\varrho 11}{\Longrightarrow} g_d,$
	$R^{\bar{r_1}}_{a_2}$	$o_1 \overset{\varrho 28}{\Longrightarrow} p_8,\ p_8 \overset{\varrho 29}{\Longrightarrow} p_4,\ p_4 \overset{\varrho 30}{\Longrightarrow} g_1,$
		$p_{10} \overset{\varrho 32}{\Longrightarrow} p_8,\ p_8 \overset{\varrho 33}{\Longrightarrow} g_3$
B_{a_2}	$B^{r_1}_{a_2}$	$\neg\varrho_{17}, \neg p_3$
	$B^{\bar{r_1}}_{a_2}$	$\neg p_7$
	a_3	
O_{a_3}		o_2, o_3
G_{a_3}		g_d, g_3, g_4, g_2
R_{a_3}	$R^{r_1}_{a_3}$	$o_2 \overset{\varrho 9}{\Longrightarrow} p_9,\ p_9 \overset{\varrho 12}{\Longrightarrow} p_8,\ p_8 \overset{\varrho 13}{\Longrightarrow} g_4,$
		$o_2 \overset{\varrho 14}{\Longrightarrow} p_1,\ p_1 \overset{\varrho 15}{\Longrightarrow} p_9,\ p_9 \overset{\varrho 16}{\Longrightarrow} g_d,$
		$o_3 \overset{\varrho 17}{\Longrightarrow} p_7,\ p_7 \overset{\varrho 18}{\Longrightarrow} p_3,$
		$o_3 \overset{\varrho 17}{\Longrightarrow} p_7,\ p_7 \overset{\varrho 21}{\Longrightarrow} p_8,\ p_8 \overset{\varrho 22}{\Longrightarrow} g_d,$
	$R^{\bar{r_2}}_{a_3}$	$p_8 \overset{\varrho 13}{\Longrightarrow} g_4,$
		$o_3 \overset{\varrho 17}{\Longrightarrow} p_7,\ p_7 \overset{\varrho 34}{\Longrightarrow} p_5,\ p_5 \overset{\varrho 27}{\Longrightarrow} g_2$
B_{a_3}	$B^{r_2}_{a_3}$	$\neg p_2,\ \neg\varrho_3$
	$B^{\bar{r_2}}_{a_3}$	$\neg p_4$

4.4 Normative validation

A framework has been presented for the structured generation of scenarios for multi-agent deliberation dialogues. It was grounded in the characteristics of typical deliberation situations. Walton (2007) has taken the deliberation characteristics and combined them with the argument scheme for practical reasoning and the critical questions attached to that scheme. This culminates in a set of 10 requirements that form the normative validation for systems supporting deliberation. In particular, it defines requirements for the way agents should (be able to) reason over actions and on what information they should ground this reasoning. Hence, it is interesting to evaluate the scenario generation framework presented in this chapter against Walton's normative requirements for deliberation systems. The 10 requirements are listed here by their reference name and a short explanation is given how each requirement is adhered to.

1. **Dialogues start with a governing question** The dialogue purpose of deciding on a course of action is already fixed in the deliberation dialogue framework; scenarios contain a fixed set of options.

2. **Agents have shared and personal goals** Agents are assigned both the mutual goal g_d as well as role and personal goals.

3. **Agents have shared and personal values** Values are not modelled explicitly in this thesis, but are assumed to underlie goals.

4. **Agents share beliefs** Agents are assigned shared facts and rules through their role; on top of this agents have personal beliefs.

5. **Beliefs are updated as the dialogue proceeds** This is part of the agent's strategy, not the scenario generation model.

6. **Expect agreement as well as disagreement** This is explicitly modelled through the concept of possible conflicts and it is shown how conflicts result in arguments and counterarguments.

7. **Proposals are based in goals, actions and practical reasoning** Rule chaining is used to provide agents with the connection between goals and options (actions), by which they can construct practical arguments.

8. **Proposals are supported through (1), (2) and (3)** The generated scenarios provide agents with knowledge to combine their goals and beliefs into support for some option.

9. **Criticizing proposals by (1) questioning premises, (2) undermining premises, (3) undercutting practical arguments, (4) rebutting practical arguments and (5) proposing alternatives** How proposals are criticized is not restricted, but rather supported by allowing the combination of practical and epistemic arguments in the ASPIC− framework. Agents are assigned appropriate knowledge to construct arguments that criticize proposals in all five ways.

10. **Agents will stop advocating defeated proposals** Since the strategy ultimately specifies agent behaviour, such as when to stop advocating a proposal, this property cannot be verified yet.

With adherence to Walton's requirements, or at least those that are relevant to this chapter, the scenario generation method satisfies some basic requirements for deliberation systems. However, it does not yet show to what degree the scenarios are indeed interesting to deliberate on. To understand to what extent the scenarios have the potential to result in interesting deliberation dialogues, a study is required into the practical use of the knowledge that is allocated to agents.

Chapter 5

Evaluating scenario interestingness

The previous chapter introduced a model for the structured generation of deliberation scenarios. It was shown how scenarios can be generated that reflect the typical characteristics of deliberation situations. One reason for the generation of scenarios was to understand the structure behind deliberation scenarios, which was validated against a set of requirements for deliberation systems. However, as was already hinted on, the main purpose for deliberation scenarios is to support experimentation with deliberating agents that use argumentation. This chapter shows how scenarios can support arguing agents in a deliberation dialogue. An experiment was performed to show how to maximize the potential use of the generation model.

5.1 Constructing arguments from allocated knowledge

To understand the importance of scenarios for experimentation with argumentation-based deliberation, it is useful to first explore how a scenario can be used by an agent to make, attack and defend proposals. The model presented in the last chapter generates scenarios by assigning goals, options, rules and beliefs to agents. The knowledge is allocated in such a way that agents can construct arguments for various options, as well as give counterarguments.

Looking back at the deliberation dialogue model of Chapter 3, agents were assumed to have a strategy for playing dialogue moves, but few constraints have been set on their strategies. Agents can only play legal attacking or surrendering

moves, as specified by the communication language, and the agents need to adhere to the deliberation protocol that is in place. One important rule in the minimal deliberation protocol of Definition 3.14 is that arguments supporting a proposal should show how the mutual goal is achieved. This rule is in place to ensure that agents show why the supported option is indeed beneficial in relation to the mutual goal. Hence, for any discussion in a deliberation dialogue to take place regarding the proposal of an option, agents need to be capable of constructing arguments for the mutual goal, with the specific proposed option as conclusion.

Example 5.1 Consider again the knowledge that was assigned to agent a_1 in the generated scenario of Example 4.5.

	a_1	
O_{a_1}		o_1, o_2
G_{a_1}		g_d, g_1, g_2, g_4
R_{a_1}	$R_{a_1}^{r_1}$	$o_1 \overset{\varrho 1}{\Longrightarrow} p_5$, $p_5 \overset{\varrho 2}{\Longrightarrow} p_2$, $p_2 \overset{\varrho 3}{\Longrightarrow} g_2$,
		$o_1 \overset{\varrho 4}{\Longrightarrow} p_6$, $p_6 \overset{\varrho 5}{\Longrightarrow} p_4$, $p_4 \overset{\varrho 6}{\Longrightarrow} g_d$,
		$o_2 \overset{\varrho 7}{\Longrightarrow} p_5$, $p_5 \overset{\varrho 2}{\Longrightarrow} p_2$,
		$o_2 \overset{\varrho 9}{\Longrightarrow} p_9$, $p_9 \overset{\varrho 10}{\Longrightarrow} p_1$,
	$R_{a_1}^{\bar{r_1}}$	$o_1 \overset{\varrho 23}{\Longrightarrow} p_{10}$, $p_8 \overset{\varrho 25}{\Longrightarrow} g_3$
		$o_2 \overset{\varrho 26}{\Longrightarrow} p_3$, $p_5 \overset{\varrho 27}{\Longrightarrow} g_2$,
B_{a_1}	$B_{a_1}^{r_1}$	$\neg\varrho_{17}, \neg p_3$,
	$B_{a_1}^{\bar{r_1}}$	$\neg\varrho_{27}$

With this knowledge, the agent will need to construct an argument that has the mutual goal g_d as conclusion, if it wants to provide support to any of its options o_1 or o_2. The agent uses a personal ASPIC− argumentation system with $R_d = R_{a_1}$ and $K = B_{a_1} \cup \{o_1, o_2\}$ to construct:

$$A'' = \frac{o_1}{p_6}\,\varrho_4 \qquad A' = \frac{p_6}{p_4}\,\varrho_5 \qquad A = \frac{p_4}{g_d}\,\varrho_6$$

This is the only argument that can be constructed for g_d by the agent. Hence, it has the possibility to support o_1 but not o_2.

In a similar fashion, an agent could try to construct an argument on the basis of its other goals. It is not the deliberation protocol which enforces this, but rather it can help the agent in finding which options that it knows about are actually beneficial for itself. How personal goals can lead to an option being supported or attacked further depends on the agent's strategy, as explained in Chapter 6. Here, it will be shown how an agent can find options that comply with its goals.

Example 5.2 With the ASPIC− argumentation system with $R_d = R_{a_1}$ and $K = B_{a_1} \cup \{o_1, o_2\}$ of Example 5.1, the agent can construct:

$$B'' = \frac{\frac{o_1}{p_5}\,\varrho_1}{} \qquad C'' = \frac{\frac{o_2}{p_5}\,\varrho_7}{}$$
$$B' = \frac{\frac{p_2}{g_2}\,\varrho_2}{} \qquad C' = \frac{\frac{p_2}{g_2}\,\varrho_2}{}$$
$$B = \frac{\frac{p_2}{g_2}\,\varrho_3}{} \qquad C = \frac{\frac{p_2}{g_2}\,\varrho_3}{}$$

$$D'' = \frac{\frac{o_1}{p_5}\,\varrho_1}{} \qquad E'' = \frac{\frac{o_2}{p_5}\,\varrho_7}{}$$
$$D = \frac{\frac{p_5}{g_2}\,\varrho_{27}}{} \qquad E = \frac{\frac{p_5}{g_2}\,\varrho_{27}}{}$$

The agent can construct four arguments that support his personal goals, of which two are from option o_1 and two from o_2. Hence, both options seem to lead to the realization of some of the agent's personal goals, specifically goal g_2. Intuitively, these arguments are reasons to propose and support both the allocated options.

If agent a_1 were to play arguments for options o_1 and o_2 in a dialogue, other agents will analyse the desirability of the advocated options. How agents can perform this analysis is made concrete in the chapter on strategies, but quite possibly agents see reasons to attack one of the arguments that agent a_1 played. Attack is possible, according to the communication language of the deliberation dialogue model, by playing an argue move with an argument that defeats the attacked argument.

Example 5.3 Consider again agent a_1's argument in Example 5.1 from option o_1 to the mutual goal g_d.

$$A'' = \frac{\frac{o_1}{p_6}\,\varrho_4}{}$$
$$A' = \frac{\frac{p_4}{g_d}\,\varrho_5}{}$$
$$A = \frac{\frac{p_4}{g_d}\,\varrho_6}{}$$

The other two agents in this example scenario will try to counter this argument using their knowledge. Importantly, it are the negated rules and beliefs that were assigned through conflict generation that allow agents to form counterarguments. Agent a_3, for example, was assigned various relevant negated beliefs in its knowledge base.

	a_3	
O_{a_3}		o_2, o_3
G_{a_3}		g_d, g_3, g_4, g_2
R_{a_3}	$R^{r_1}_{a_3}$	$o_2 \overset{\varrho 9}{\Longrightarrow} p_9,\ p_9 \overset{\varrho 12}{\Longrightarrow} p_8,\ p_8 \overset{\varrho 13}{\Longrightarrow} g_4,$
		$o_2 \overset{\varrho 14}{\Longrightarrow} p_1,\ p_1 \overset{\varrho 15}{\Longrightarrow} p_9,\ p_9 \overset{\varrho 16}{\Longrightarrow} g_d,$
		$o_3 \overset{\varrho 17}{\Longrightarrow} p_7,\ p_7 \overset{\varrho 18}{\Longrightarrow} p_3,$
		$o_3 \overset{\varrho 17}{\Longrightarrow} p_7,\ p_7 \overset{\varrho 21}{\Longrightarrow} p_8,\ p_8 \overset{\varrho 22}{\Longrightarrow} g_d,$
	$R^{\bar{r}_2}_{a_3}$	$p_8 \overset{\varrho 13}{\Longrightarrow} g_4,$
		$o_3 \overset{\varrho 17}{\Longrightarrow} p_7,\ p_7 \overset{\varrho 34}{\Longrightarrow} p_5,\ p_5 \overset{\varrho 27}{\Longrightarrow} g_2$
B_{a_3}	$B^{r_2}_{a_3}$	$\neg p_2,\ \neg \varrho_3$
	$B^{\bar{r}_2}_{a_3}$	$\neg p_4$

The agent can construct one counterargument with its ASPIC− argumentation system with $R_d = R_{a_3}$ and $K = B_{a_3}$ that defeats A.

$$\bar{A} = \neg p_4$$

Argument $\bar{A}$ rebuts argument a on A', as their conclusions are contrary. Although $\bar{A}$ is a very trivial argument, it is still a valid counterargument to A and shows how the direct possible conflict allocation during scenario generation can already lead to the possibility for counterarguments. In a similar fashion, the agent can construct counter arguments to agent a_1's arguments B and C.

$$\bar{B} = \neg p_2 \quad \bar{D} = \neg \varrho_3$$

$$\bar{C} = \neg p_2 \quad \bar{E} = \neg \varrho_3$$

$\bar{B}$ rebuts B on B', just as $\bar{C}$ rebuts C on C', both using the belief that p_2 is not true. Arguments $\bar{D}$ and $\bar{E}$ undercut B and C respectively, both on ϱ_3, by

stating that the application of rule is not applicable. Arguments D and E can not be countered by agent a_3.

If the scenario is generated using the chained conflict allocation method, the knowledge bases of the agents can lead to more complex counterarguments. Instead of the knowledge from Example 4.5, consider an alternative knowledge allocation for agent a_3.

	a_3	
O_{a_3}		o_2, o_3
G_{a_3}		g_d, g_3, g_4, g_2
R_{a_3}	$R^{r_1}_{a_3}$	$o_2 \overset{\varrho 9}{\Longrightarrow} p_9,\ p_9 \overset{\varrho 12}{\Longrightarrow} p_8,\ p_8 \overset{\varrho 13}{\Longrightarrow} g_4,$
		$o_2 \overset{\varrho 14}{\Longrightarrow} p_1,\ p_1 \overset{\varrho 15}{\Longrightarrow} p_9,\ p_9 \overset{\varrho 16}{\Longrightarrow} g_d,$
		$o_3 \overset{\varrho 17}{\Longrightarrow} p_7,\ p_7 \overset{\varrho 18}{\Longrightarrow} p_3,$
		$o_3 \overset{\varrho 17}{\Longrightarrow} p_7,\ p_7 \overset{\varrho 21}{\Longrightarrow} p_8,\ p_8 \overset{\varrho 22}{\Longrightarrow} g_d,$
		$p_{12} \overset{\varrho 26}{\Longrightarrow} p_7,\ p_7 \overset{\varrho 21}{\Longrightarrow} p_8,\ p_8 \overset{\varrho 27}{\Longrightarrow} \neg p_2,$
		$p_{10} \overset{\varrho 28}{\Longrightarrow} p_6,\ p_6 \overset{\varrho 5}{\Longrightarrow} p_4,\ p_4 \overset{\varrho 29}{\Longrightarrow} \neg \varrho_3,$
	$R^{\bar{r}_2}_{a_3}$	$p_8 \overset{\varrho 13}{\Longrightarrow} g_4,$
		$o_3 \overset{\varrho 17}{\Longrightarrow} p_7,\ p_7 \overset{\varrho 34}{\Longrightarrow} p_5,\ p_5 \overset{\varrho 27}{\Longrightarrow} g_2$
		$p_1 \overset{\varrho 5}{\Longrightarrow} p_{10},\ p_{10} \overset{\varrho 30}{\Longrightarrow} p_{11},\ p_{11} \overset{\varrho 31}{\Longrightarrow} \neg p_4$
B_{a_3}	$B^{r_2}_{a_3}$	p_{12}, p_{10}
	$B^{\bar{r}_2}_{a_3}$	$\emptyset$

Instead of the direct assignment of negated beliefs for p_2, ϱ_3 and p_4, rule chains were generated instead. Note that the conflict chains for p_2, ϱ_3 were already showed in Example 4.4. A new chain is generated connecting p_6 to $\neg p_4$, forming the non-role rules and beliefs, but note that not all rules and beliefs necessarily end up in the agent's knowledge, according to Definition 4.12, and indeed the belief p_1 is missing from the agent's knowledge base.

As above, the agent can now try to construct a counterargument for agent a_1's argument A that connected option o_1 and mutual goal g_d. Again the sub-conclusion p_4 can be attacked, but this time a more complex argument is required.

$$\vec{A}' = \frac{p_{10}}{\underline{\underline{p_{11}}}}\ \varrho_{30}$$
$$\vec{A} = \frac{}{\underline{\underline{\neg p_4}}}\ \varrho_{31}$$

Even though the belief p_1 is missing in the agent's knowledge, it already believes p_{10} is the case and hence it does not need to apply rule ϱ_5 to still be able to construct a counterargument. Agent a_1's arguments B and C, connecting options to the agent's private goals, can also be attacked. Just like above, those counterarguments attack B and C on either the subconclusion p_2 or the rule application of ϱ_3.

$$\vec{B}'' = \frac{\frac{p_{12}}{p_7}\,\varrho_{26}}{p_8}\,\varrho_{21} \qquad \vec{D}'' = \frac{\frac{p_{10}}{p_6}\,\varrho_{28}}{p_4}\,\varrho_5$$
$$\vec{B}' = \frac{\frac{p_8}{\neg p_2}\,\varrho_{27}}{\neg p_2} \qquad \vec{D}' = \frac{\frac{p_4}{\neg \varrho_3}\,\varrho_{29}}{\neg \varrho_3}$$
$$\vec{B} = \vec{B}' \qquad \vec{D} = \vec{D}'$$

$$\vec{C}'' = \frac{\frac{p_{12}}{p_7}\,\varrho_{26}}{p_8}\,\varrho_{21} \qquad \vec{E}'' = \frac{\frac{p_{10}}{p_6}\,\varrho_{28}}{p_4}\,\varrho_5$$
$$\vec{C}' = \frac{\frac{p_8}{\neg p_2}\,\varrho_{27}}{\neg p_2} \qquad \vec{E}' = \frac{\frac{p_4}{\neg \varrho_3}\,\varrho_{29}}{\neg \varrho_3}$$
$$\vec{C} = \vec{C}' \qquad \vec{E} = \vec{E}'$$

In conclusion, agents will try to construct arguments relating to the options they know about and concerning the mutual and their personal goals. Moreover, other agents are typically assigned relevant knowledge, either direct or using conflict chains, to construct counterarguments, which defeat the arguments that sported some option.

5.2 Interesting scenarios

Scenarios are meant to support experimentation with argumentation in deliberation dialogues. For that reason, scenarios need to be evaluated to what degree they underpin such experimentation. In other words, scenarios need to be interesting with respect to the potential they have for further use in the experiments that can demonstrate the benefits of argumentation in multi-agent deliberation dialogues.

What makes a deliberation scenario interesting? An experiment that tests the benefits of argumentation will compare a situation in which agents do not argue with a situation in which agent do argue. For a clear distinction between these, the dialogues in which agents argue will ideally have discussions on multiple options, which means arguments supporting the options as well as counterarguments to those. These types of dialogues are better representatives for the arguing agents' situation, for the comparison against non-arguing agents, than a dialogue with little discussion going on. Hence, a dialogue is interest-

ing if agents have arguments to back up the proposal of various options and, distinctively, if agents can defeat those arguments with their counterarguments.

As explained above, an agent can find support for a proposal through the construction of arguments from its knowledge base. Foremost, the agent will need an argument that shows how an option leads to the mutual dialogue goal, as this is required by the dialogue model for arguments supporting a proposal. For its personal gain, the agent will also want to construct arguments to its other personal goals. Although this type of reasoning is tied to a strategy, likely agents will propose and defend those options by which it believes its personal goals will be achieved. Conversely, dialogues will spur discussions with counterarguments only if agents are assigned knowledge by the scenario generation mechanism to construct relevant counterarguments. Consequently, how interesting a generated scenario is can be tested by measuring if, and to what extent, agents have appropriate knowledge.

5.3 Experiment

While the scenario generation method is grounded in the characteristics of deliberation dialogues, the practical interestingness of generated scenarios still needs to be evaluated. For this, an experiment was designed and performed that tests to what degree a scenario is interesting through four different metrics. An implementation of the scenario generation method of Chapter 4 is used to generate and test many scenarios. A statistical analysis is applied to the experimental data to identify how the scenario generation method can be configured to have the most potential for interesting dialogues, with many arguments and counterarguments.

5.3.1 Parameters of scenario generation

The model for the generation of deliberation scenarios presented in the last chapter provides a model that is not yet fully defined. The structure of rule changes and subsequent assignment of knowledge to agents is modelled, but in several places it is left open how large the sets of rules, beliefs, options and goals are. For example, the length of (conflict) rule chains, the number of role and non-role rules and beliefs or the number of roles and agents in the scenario.

The places where the scenario generation method needs to be made concrete will now be identified. For each of these places a decision on the size of some

set needs to be made. Since the sizes of these sets determine directly how the resulting generated dialogue will look like, they act as input parameters for the generation method. The settings for all input parameters together is called a configuration.

The context for the generation of deliberation scenarios is specified in Definition 4.3. The components of a context still need to be made concrete, except $R_{\mathcal{SK}}$ which contains all rules that can be specified in an ASPIC− framework with topic language L_t and fixed mutual goal g_d. The $|S|$ notation is used to mean the number of elements in a set S.

Definition 5.1 Given a scenario generation context $\mathcal{SK} = \langle R_{\mathcal{SK}}, B_{\mathcal{SK}}, G_{\mathcal{SK}}, O_{\mathcal{SK}}, g_d, \mathcal{C}, \bar{\mathcal{C}} \rangle$ and set sizes $n_{B_{\mathcal{SK}}}, n_{G_{\mathcal{SK}}}, n_{O_{\mathcal{SK}}}$ and l:

- $|B_{\mathcal{SK}}| = n_{B_{\mathcal{SK}}}$
- $|G_{\mathcal{SK}}| = n_{G_{\mathcal{SK}}}$
- $|O_{\mathcal{SK}}| = n_{O_{\mathcal{SK}}}$
- $\mathcal{C}(G, B, o) = \{o \overset{\varrho 1}{\Rightarrow} p_1, \ldots, p_i \overset{\varrho i}{\Rightarrow} p_j, \ldots, p_n \overset{\varrho l}{\Rightarrow} g\}$ such that $g \in G$ is picked arbitrarily and $|\mathcal{C}(G, B, o)| = l$
- $\bar{\mathcal{C}} = \{p \overset{\varrho 1}{\Rightarrow} p_1, \ldots, p_i \overset{\varrho i}{\Rightarrow} p_j, \ldots, p_n \overset{\varrho l}{\Rightarrow} c\}$ such that $g \in G$ is picked arbitrarily and $|\bar{\mathcal{C}}(G, B, o)| = l$

The variables $n_{B_{\mathcal{SK}}}$, $n_{G_{\mathcal{SK}}}$ and $n_{O_{\mathcal{SK}}}$ control the number of atoms that will further be used in the generation process. Moreover, all the rule chains that will be generated, to connect options and goals as well as conflict chains, have length l. Normal and conflict chains have the same length for reasons of simplicity, but a future experiment might drop this constraint.

Definitions 4.4 and 4.5 define the structure of roles and agents, including fixed sets of goals and options that are allocated to them.

Definition 5.2 Given a scenario generation context $\mathcal{SK}$, the sets of roles $\mathcal{R}$ and agents $\mathcal{A}$ and the set sizes $n_{\mathcal{R}}, n_{\mathcal{A}}, n_{O_r}$ and n_{G_r}:

- $|\mathcal{R}| = n_{\mathcal{R}}$
- for each $r \in \mathcal{R}$ such that $r = \langle R_r, B_r, G_r, O_r \rangle$:
  - $|O_r| = n_{O_r}$

Table 5.1: Input parameters used in the scenario generation process

		min	example	max
$n_{B_{S\mathcal{K}}}$	The beliefs seed set size	10	12	100
$n_{G_{S\mathcal{K}}}$	The goals seed set size	3	5	15
$n_{O_{S\mathcal{K}}}$	The options seed set size	3	3	15
$n_{\mathcal{A}}$	The number of agents	1	3	6
$n_{\mathcal{R}}$	The number of roles	1	2	6
l	The length of rule chains	1	3	9
n_{O_r}	Role r's options set size	2	2	5
n_{G_r}	Role r's goals set size	2	2	5
$n_{G_a^{\bar{r}}}$	Agent a's non-role originating goals set size	0	1	2
$n_{B_a^r}$	Agent a's role-originating beliefs set size	1	12	15
$n_{B_a^{\bar{r}}}$	Agent a's non-role originating beliefs set size	0	5	20

  - $|G_r| = n_{G_r}$

- $|\mathcal{A}| = n_{\mathcal{A}}$
- for each agent $a \in \mathit{agents}$ such that $a = \langle r, R_a, B_a, G_a, O_a \rangle$ and $G_a = G_r \cup G_a^{\bar{r}} \cup \{g_d\}$:
  - $|G_a^{\bar{r}}| = n_{G_a^{\bar{r}}}$

Finally, rules and beliefs are allocated to roles and thereafter to agents using role and non-role originating rules and beliefs. Agents can end up with incomplete knowledge by only assigning part of the role's rules and beliefs, specified in Definition 4.10, and assigning only part of the generated non-role originating rules and beliefs, specified in 4.11.

Definition 5.3 Given an agent $a = \langle r, R_a, B_a, G_a, O_a \rangle$ with some role r and set sizes $n_{B_a^r}$ and $n_{B_a^{\bar{r}}}$:

- $|R_a^r| + |B_a^r| = n_{B_a^r}$
- $|R_a^{\bar{r}}| + |B_a^{\bar{r}}| = n_{B_a^{\bar{r}}}$

A maximum of $n_{B_a^r}$ role-originating rules and beliefs will be allocated to agents, as well as a maximum of $n_{B_a^{\bar{r}}}$ non-role originating rules and beliefs. In total, there are 11 input parameters that influence the scenario generation process, which are listed in Table 5.1. The table also shows a minimum and maximum reasonable setting, their uses explained below, and the setting used in the running example generated scenario.

5.3.2 Scenario metrics

Scenarios will be measured by how interesting they are. As explained above, interesting scenarios are those that will spur an interesting dialogue through the supporting of proposals and playing of counterarguments. At the core of this idea is that an agent can propose options if it can construct an argument from it, with the mutual goal as conclusion. The other interesting case is when agents can construct arguments from an option to one of its personal goals. An option for which an argument can be constructed will be called defensible for the agent.

Definition 5.4 Given an agent $a = \langle r, R_a, B_a, G_a, O_a \rangle$, some option $o \in O_a$, some goal $g \in G_a$ and an ASPIC− argumentation framework AS with $R_d = R_a$ and $K = B_a \cup \{o\}$, o is an *a-g-defensible option* if an argument A can be constructed such that $\mathsf{conc}(A) = g$ and $o \in \mathsf{prem}(A)$.

The first metric for interesting dialogues indicates to which extent agents can construct arguments from their options to the mutual goal. The defensible options are counted with arguments concluding the mutual dialogue goal. Hence, the metric is a ratio of options that can potentially be proposed in the dialogue versus all options available to the agents.

Definition 5.5 A generated scenario with a set of agents $\mathcal{A}$ and mutual goal g_d has an *option potential ratio*

$$j_{\mathcal{A}}^{g_d} = \frac{|\bigcup_{a \in \mathcal{A}} \{o | o \in O_a \text{ where } o \text{ is an } a\text{-}g\text{-defensible option such that } g = g_d\}|}{n_{\mathcal{A}} \times n_{O_r}}$$

Example 5.4 Consider again the example scenario originally presented in Example 4.5. As already showed in Example 5.1, agent a_1 can construct one argument that connects an option to the mutual dialogue goal, that is, an argument from option o_1. Since agent a_1 cannot construct an argument connecting

o_2 and g_d, this option is not defensible and for this agent the ratio would be $\frac{1}{2}$. Agents a_2 and a_3 can also construct arguments that connect an allocated option to the mutual goal.

$$
\begin{array}{lc}
 & A'' = \dfrac{o_1}{\dfrac{}{p_6}}\,\varrho_4 \\
a_1 & A' = \dfrac{p_6}{\dfrac{}{p_4}}\,\varrho_5 \\
 & A = \dfrac{p_4}{\dfrac{}{g_d}}\,\varrho_6
\end{array}
$$

$$
\begin{array}{lccc}
 & A'' = \dfrac{o_1}{\dfrac{}{p_6}}\,\varrho_4 & F'' = \dfrac{o_2}{\dfrac{}{p_9}}\,\varrho_9 & G'' = \dfrac{o_1}{\dfrac{}{p_8}}\,\varrho_{28} \\
a_2 & A' = \dfrac{p_6}{\dfrac{}{p_4}}\,\varrho_5 & F' = \dfrac{p_9}{\dfrac{}{p_1}}\,\varrho_{10} & G' = \dfrac{p_8}{\dfrac{}{p_4}}\,\varrho_{29} \\
 & A = \dfrac{p_4}{\dfrac{}{g_d}}\,\varrho_6 & F = \dfrac{p_1}{\dfrac{}{g_d}}\,\varrho_{11} & G = \dfrac{p_4}{\dfrac{}{g_d}}\,\varrho_6
\end{array}
$$

$$
\begin{array}{lcccc}
 & L'' = \dfrac{o_2}{\dfrac{}{p_1}}\,\varrho_{14} & M'' = \dfrac{o_3}{\dfrac{}{p_7}}\,\varrho_{17} & N'' = \dfrac{o_2}{\dfrac{}{p_9}}\,\varrho_9 & O'' = \dfrac{o_2}{\dfrac{}{p_9}}\,\varrho_9 \\
a_3 & L' = \dfrac{p_9}{\dfrac{}{g_d}}\,\varrho_{15} & M' = \dfrac{p_8}{\dfrac{}{g_d}}\,\varrho_{21} & N = \dfrac{p_9}{\dfrac{}{g_d}}\,\varrho_{16} & O' = \dfrac{p_8}{\dfrac{}{g_d}}\,\varrho_{12} \\
 & L = \dfrac{p_9}{\dfrac{}{g_d}}\,\varrho_{16} & M = \dfrac{p_8}{\dfrac{}{g_d}}\,\varrho_{22} & & O = \dfrac{p_8}{\dfrac{}{g_d}}\,\varrho_{22}
\end{array}
$$

The actual number of arguments connecting one option to the mutual goal is not important, as only one is enough to be able to propose the option in the dialogue. Given this example scenario the option potential ratio $j_{\mathcal{A}}^{g_d} = \frac{5}{6}$.

The second metric for interesting scenarios defines to which extent agents can form arguments that couple their allocated options to some of their personal goals. The number of options for which such an argument can be constructed is compared to all available options for the agents to get the option defensibility ratio of a scenario.

Definition 5.6 A generated scenario with a set of agents $\mathcal{A}$ has an *option defensibility ratio*

$$j_{\mathcal{A}} = \frac{|\bigcup_{a \in \mathcal{A}} \{o | o \in O_a \text{ where } o \text{ is an } a\text{-}g\text{-defensible option such that } g \in G_a\}|}{n_{\mathcal{A}} \times n_{O_r}}$$

Example 5.5 As shown above in Example 5.2, agent a_1 can construct arguments from both its options o_1 and o_2, connecting them its personal goals.

Agents a_2 and a_3 can also construct such arguments.

$$
\begin{array}{lllll}
 & B'' = \dfrac{o_1}{p_5}\,\varrho_1 & C'' = \dfrac{o_2}{p_5}\,\varrho_7 & D'' = \dfrac{o_1}{p_5}\,\varrho_1 & E'' = \dfrac{o_2}{p_5}\,\varrho_7 \\
a_1 & B' = \dfrac{p_5}{p_2}\,\varrho_2 & C' = \dfrac{p_5}{p_2}\,\varrho_2 & D = \dfrac{p_5}{g_2}\,\varrho_{27} & E = \dfrac{p_5}{g_2}\,\varrho_{27} \\
 & B = \dfrac{p_2}{g_2}\,\varrho_3 & C = \dfrac{p_2}{g_2}\,\varrho_3 & &
\end{array}
$$

$$
\begin{array}{llllll}
 & H'' = \dfrac{o_1}{p_8}\,\varrho_{28} & I'' = \dfrac{o_2}{p_5}\,\varrho_7 & J' = \dfrac{o_1}{p_8}\,\varrho_{28} & K'' = \dfrac{o_2}{p_5}\,\varrho_7 \\
a_2 & H' = \dfrac{p_4}{p_2}\,\varrho_{29} & I' = \dfrac{p_2}{p_1}\,\varrho_2 & J = \dfrac{p_8}{g_3}\,\varrho_{33} & K' = \dfrac{p_2}{g_2}\,\varrho_3 \\
 & H = \dfrac{p_2}{g_1}\,\varrho_{30} & I = \dfrac{p_2}{g_1}\,\varrho_8 & &
\end{array}
$$

$$
\begin{array}{llll}
 & P'' = \dfrac{o_2}{p_9}\,\varrho_9 & Q'' = \dfrac{o_3}{p_7}\,\varrho_{17} & R'' = \dfrac{o_3}{p_7}\,\varrho_{17} \\
a_3 & P' = \dfrac{p_8}{p_4}\,\varrho_{12} & Q' = \dfrac{p_7}{p_5}\,\varrho_{34} & R' = \dfrac{p_8}{p_4}\,\varrho_{21} \\
 & P = \dfrac{p_4}{g_4}\,\varrho_{13} & Q = \dfrac{p_5}{g_2}\,\varrho_{27} & R = \dfrac{p_4}{g_4}\,\varrho_{13}
\end{array}
$$

Each of the agents can construct arguments for all of their options, hence the option defensibility ratio $j_{\mathcal{A}} = \frac{6}{6}$.

Interesting dialogues do not only allow agents to construct support for options, but also spur discussion through the playing of counterarguments. As demonstrated above, agents use the rules and beliefs that were directly assigned or generated through conflict chains to construct arguments that defeat another agent's argument.

Definition 5.7 An agent a's *a-g-*defensible option o, as supported by argument A with $\mathsf{conc}(A) = g$, is also an *a-a'-countered option* if some agent $a' = \langle r', R_{a'}, B_{a'}, G_{a'}, O_{a'} \rangle$, where $a \neq a'$, can construct a counterargument B that defeats A, given an ASPIC− argumentation framework AS with $R_d = R_{a'}$ and $K = B_{a'}$

The metrics related to potential for counterarguments are split for the two types of defensible options used above, that is, options with arguments supporting the mutual goal and options with arguments supporting some personal goal. Looking at their potential in a dialogue, the arguments supporting the mutual goal are used to support proposals and counterarguments are subsequently used, by other agents, to attack this support.

Definition 5.8 A generated scenario with a set of agents $\mathcal{A}$ and mutual goal g_d has a *countered option potential ratio*

$$\bar{j}_{\mathcal{A}}^{g_d} = \frac{|\bigcup_{a\in\mathcal{A}}\{o|o \in O_a \text{ where } o \text{ is an } a\text{-}a'\text{-countered option}\}|}{|\bigcup_{a\in\mathcal{A}}\{o|o \in O_a \text{ where } o \text{ is an } a\text{-}g\text{-defensible option where } g = g_d\}|}$$

The example will, for brevity sake, from here only use show the direct conflict assignment method for scenario generation, consistent with the knowledge allocation of Example 4.5. Of course, the metrics work exactly the same way when the conflict chaining method is used instead.

Example 5.6 Example 5.3 showed how agent a_1's argument A could be attacked by agent a_3, since agent a_3 can use its assigned rules and beliefs to construct an argument $\bar{A}$.

$$\bar{A} = \neg p_4$$

On the other hand, agent a_2 can not construct an argument that defeats A. For the two options that were allocated to agent a_1 the countered option potential would be $\frac{1}{2}$, but the ratio is, of course, defined over all agents. To complete the picture, agent a_2 and a_3's options are considered as well.

a_1	o_1	a_3	$\bar{A} = \neg p_4$				
a_2	o_1	a_3	$\bar{A} = \neg p_4$	a_3	$\bar{G} = \neg p_4$		
	o_2						
a_3	o_2						
	o_3	a_1	$\bar{M} = \neg \varrho_{17}$	a_2	$\bar{M}' = \neg \varrho_{17}$	a_2	$\bar{M}'' = \neg p_7$

Considering the defensible options for each agent, three times another agent was able to construct a counter argument, hence the countered option potential ratio $\bar{j}_{\mathcal{A}}^{g_d} = \frac{3}{5}$.

In contrast to the countered option potential ratio, the countered option defensibility ratio considers those arguments that agents have for their personal goals. This type of argument can be used by an agent in its strategy to see how an option leads to the realisation of personal goals. The countered option defensibility ratio determines to what extent these arguments can be countered by other agents, if they were to be played in a dialogue setting.

Definition 5.9 A generated scenario with a set of agents $\mathcal{A}$ has a *countered option defensibility ratio*

$$\bar{j}_{\mathcal{A}} = \frac{|\bigcup_{a\in\mathcal{A}}\{o|o \in O_a \text{ where } o \text{ is an } a\text{-}a'\text{-countered option}\}|}{|\bigcup_{a\in\mathcal{A}}\{o|o \in O_a \text{ where } o \text{ is an } a\text{-}g\text{-defensible option}\}|}$$

Example 5.7 Example 5.3 showed how agent a_1's four arguments B and C could be defeated by agent a_3's $\bar{B}$ and $\bar{D}$ as well as $\bar{C}$ and $\bar{E}$ respectively. For the countered option defensibility ratio, every agent's assigned options will be considered. The names of the counterarguments correspond to the arguments that they defeat.

a_1	o_1	a_3	$\bar{B} = \neg p_2$	a_3	$\bar{B}' = \neg \varrho_3$			
	o_2	a_3	$\bar{C} = \neg p_2$	a_3	$\bar{C}' = \neg \varrho_3$			
a_2	o_1	a_3	$\bar{H} = \neg p_4$					
	o_2	a_3	$\bar{I} = \neg p_2$	a_3	$\bar{K} = \neg p_2$	a_3	$\bar{K}' = \neg \varrho_3$	
a_3	o_2							
	o_3	a_1	$\bar{Q} = \neg \varrho_{17}$	a_1	$\bar{R} = \neg \varrho_{17}$	a_1	$\bar{Q}''' = \neg \varrho_{27}$	
		a_2	$\bar{Q}' = \neg \varrho_{17}$	a_2	$\bar{R}' = \neg \varrho_{17}$	a_2	$\bar{Q}'' = \neg p_7$	a_2 $\bar{R}'' = \neg p_7$

Only for agent a_3's option o_2 there was no attack possible, but all other combinations allowed for at least one counter argument. Hence, the countered option defensibility ratio $\bar{j}_{\mathcal{A}} = \frac{5}{6}$.

None of the four metrics explain individually how interesting a dialogue is. Rather, the four metrics combined form the indicator of interestingness of a single generated scenario.

Definition 5.10 A generated scenario with a set of agents $\mathcal{A}$ has a *measure of interestingness*, which is a 4-tuple $\langle j^{g_d}_{\mathcal{A}}, j_{\mathcal{A}}, \bar{j}^{g_d}_{\mathcal{A}}, \bar{j}_{\mathcal{A}}\rangle$.

The most interesting scenario is that with the highest values for all of the four metrics. Note that a claim about relative interestingness between two scenarios can only be made if all metrics are strictly higher or strictly lower at the same time.

Definition 5.11 A scenario with set of agents $\mathcal{A}$ is a *more interesting scenario* than a scenario with set of agents $\mathcal{A}'$ iff $j_{\mathcal{A}}^{g_d} > j_{\mathcal{A}'}^{g_d}$, $j_{\mathcal{A}} > j_{\mathcal{A}'}$, $\bar{j}_{\mathcal{A}}^{g_d} > \bar{j}_{\mathcal{A}'}^{g_d}$ and $\bar{j}_{\mathcal{A}} > \bar{j}_{\mathcal{A}'}$. A scenario with set of agents $\mathcal{A}$ is a *less interesting scenario* than a scenario with set of agents $\mathcal{A}'$ iff $j_{\mathcal{A}}^{g_d} < j_{\mathcal{A}'}^{g_d}$, $j_{\mathcal{A}} < j_{\mathcal{A}'}$, $\bar{j}_{\mathcal{A}}^{g_d} < \bar{j}_{\mathcal{A}'}^{g_d}$ and $\bar{j}_{\mathcal{A}} < \bar{j}_{\mathcal{A}'}$.

5.3.3 Implementation details

The scenario generation method was implemented as a Java application, consisting of a scenario generator, to be reused in Chapter 8, and a scenario experiment, that implements the metrics for interesting dialogues. It was coupled with the ASPIC Java Components of South and Vreeswijk (2009) to construct and evaluate argumentation systems. Several noteworthy decisions were made during the development.

Every agent in a scenario is assigned a single role, from the set of all roles. In the software implementation, roles are assigned in sequence. With a set of two roles $\mathcal{R} = \{r_1, r_2\}$, agent a_1 is assigned role r_1, agent a_2 is assigned role r_2, agent a_3 is assigned r_1 again, agent a_4 is assigned r_2, a_5 is assigned r_1, etc. In contrast, the assignment of subsets of goals, options, rules and beliefs is done by randomly picking elements from the superset, with a uniform distribution. For example, when a seed set of goals $G_{\mathcal{SK}} = \{g_1, g_2, g_3, g_4\}$ is used to assign goals to a role, the resulting set (where $n_{G_r} = 3$) might be $\{g_2, g_2, g_4\}$, $\{g_4, g_2, g_1\}$, etc.

A second point on the implementation of knowledge assignment is that of loop prevention. Rule chains are generated by randomly selecting elements from the seed set of beliefs $B_{\mathcal{SK}}$ and producing (uniquely numbered) rules that connect these. The resulting rules are allocated to roles and subsequently to agents. The resulting set of rules assigned to a single agent can quite possibly be combined in several ways beyond the original purpose in a single chain. This is actually a strength of the generation method and several examples have been shown before where agents can take advantage of this. However, this also means that loops of rules might end up being assigned to an agent. For example, if some agent a was assigned $\{p_8 \overset{\varrho 12}{\Longrightarrow} p_9, p_9 \overset{\varrho 12}{\Longrightarrow} p_8, p_8 \overset{\varrho 13}{\Longrightarrow} g_4\} \in R_a$, the construction of an argument with $\mathsf{conc}(g_4)$ causes a loop. If the ASPIC Java Components needs to construct such an argument, it will end up in an infinite loop. To work around this problem the generated scenarios are tested to make sure that no rule loops are present in the agent's knowledge.

The parameters of Table 5.1 are used as input in the scenario generation

process. However, not every configuration of input parameters is valid. For instance, there should be at least one agent, that is, $n_{\mathcal{A}} >= 1$. More subtly, several combinations of parameters are invalid. For example, with a number of role-originating rules and beliefs $n_{B_a^r} = 50$ there should at least be 50 rules and beliefs available to be assigned. This might not be the case when, for instance, l or n_{O_r} are too small. Consequently, not every configuration of scenario settings can be tested and such configurations are thus not considered during the experiment.

5.3.4 Method

An experiment was performed to measure the interestingness of scenarios that are generated with the method from Chapter 4. The Java implementation was used to generate a total of $N = 2000$ valid scenarios. Input parameters for the generation process were taken randomly, with a reasonable lower and upper bound and with a uniform distribution. The minimum and maximum bounds for each input parameter is listed in Table 5.1. Invalid configurations of parameters were ignored. Both the direct and chained conflict assignment methods were used, with an even distribution, constituting an extra parameter in the experiment.

Each generated scenario was tested using the four metrics defined in Section 5.3.2. The results were logged to a data file and analysed using the statistical software package R. Generating 2000 valid scenarios, with the above given parameter bounds and both conflict assignment methods, costed about half an hour, where checking for rule loops was the most time-consuming. The experimental data were studied to answer four questions:

1. Is the generation method capable of producing interesting dialogues?

2. How do individual input parameters influence the scenario interestingness?

3. Which input parameters are most influential in making a scenario interesting?

4. What is the optimal input parameter configuration, producing the most interesting scenarios?

Table 5.2: Means and standard deviations of all scenario interestingness metrics

	$j_{\mathcal{A}}^{g_d}$	$j_{\mathcal{A}}$	$\bar{j}_{\mathcal{A}}^{g_d}$	$\bar{j}_{\mathcal{A}}$
Direct conflicts	0.11 (0.21)	0.22 (0.30)	0.72 (0.40)	0.72 (0.39)
Conflict chaining	0.07 (0.19)	0.16 (0.28)	0.32 (0.37)	0.29 (0.36)
Combined	0.10 (0.20)	0.19 (0.29)	0.60 (0.43)	0.57 (0.43)

5.4 Results

Four questions were posed to understand the scenario generation process and its capability to product interesting dialogues. These questions will now be answered using the experimental data, before providing a conclusion regarding the use of the generation process.

5.4.1 Capability of producing interesting dialogues

Analysis of the experimental data showed that the generated scenarios had a strongly varied degree of interestingness, for each of the four metrics and for both conflict assignment methods. For example, many scenarios did not give rise to any defensible or potential option at all, while many scenarios were generated that resulted in high values for all four of the metrics.

The means and standard deviations for each of the scenario interestingness metrics are listed in Table 5.2. The values are given for both methods of conflict assignment, as well as the combined values. Both methods allow for agents to construct arguments justifying their options from both the mutual ($j_{\mathcal{A}}^{g_d} = 0.11$ and 0.08) and goals ($\bar{j}_{\mathcal{A}}^{g_d} = 0.73$ and 0.33) and other agents are often capable of constructing counterarguments to defensible options. Most strikingly is the difference between direct and chained conflict assignment, in that direct assignment of conflicts results in a much higher chance of other agents being able to construct counterarguments.

5.4.2 Individual parameter influence

It is interesting to see what the influence of a single parameter is on the scenario generation process in terms of scenario interestingness. This can confirm or contradict the intuition behind the various parameters. Not every combination

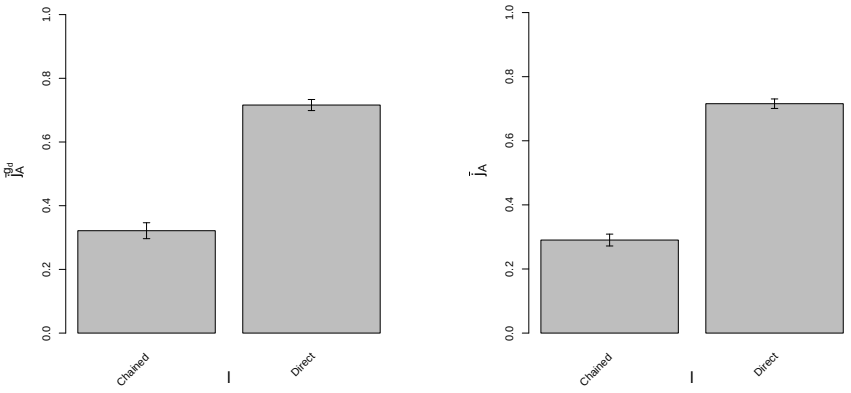


Figure 5.1: Average countered option potential and defensibility, $\bar{j}_{\mathcal{A}}^{g_d}$ and $\bar{j}_{\mathcal{A}}$, (with standard errors of the mean) for both conflict assignment methods

of parameter and metric will be discussed, as there are 48 of such combinations. Still, it is good to look in detail at four specific cases: the conflict assignment method, the number of assigned role-originating beliefs, the number of agents and the rule chain length. As explained below, they are especially interesting or indicative.

As we have already seen in the mean scenario interestingness for countered option potential and countered option defensibility, there is a clear difference between direct or chained conflict assignment. This warrants a closer look. Figure 5.1 shows the countered option potential and countered option defensibility ratios for both conflict assignment methods.

Considering the scenario generation model, there are several effects of the input parameters that one expects to show in the experimental data. Specifically, increasing or decreasing some input parameter is expected to have an incremental or decremental effect on the various metrics. To see if indeed these effects are reflected in the experimental data, two parameters are depicted, namely the number of assigned beliefs and the rule chain length.

Intuitively it is expected that if agents are assigned a larger number of rules

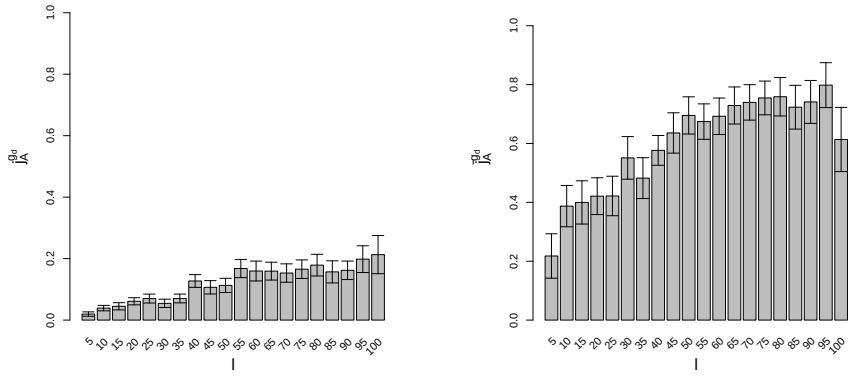
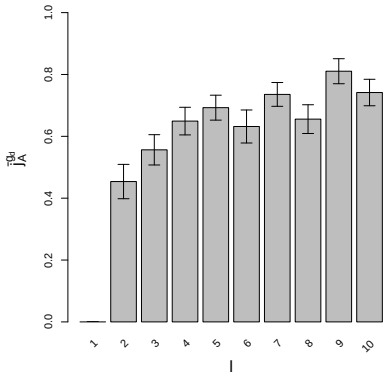


Figure 5.2: Average (countered) option defensibility ratios, $j_{\mathcal{A}}$ and $\bar{j}_{\mathcal{A}}$, (with standard errors of the mean) for $n_{B_a^r} \in \{5, \ldots, 100\}$

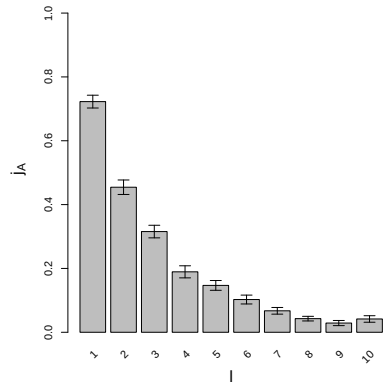
and beliefs, specified by $n_{B_a^r}$, that agents would more likely be able to construct arguments and counterarguments. The experiment showed that this was indeed the case, for arguments and counterarguments to the mutual goal and personal goals. Figure 5.2 shows that the option potential and countered option potential ratios are higher when agents are assigned more rules and beliefs.

Not every input parameter has a positive linear influence on the various metrics. The chain length l causes rule chains to be longer, in which case chains are more likely to broken, that is, that agents miss rules from the chains due to the restricted knowledge allocation to agents. This intuitive effect is indeed found in the experimental data, as shown in Figure 5.4.2.

The final parameter that is worth portraying is one that is expected to reveal meaningless parameter configurations. One of such parameters is the number of agents. The countered option potential and defensibility ratios are metrics that count counterarguments. Since counterarguments will only be constructed by other agents, the data should show that with only one agent in the game there should be no potential for counterarguments at all and therefore the countered option potential and defensibility should be 0. Figure 5.4.2 shows that the



(a) $\bar{j}_{\mathcal{A}}^{g_d}$ for $n_{\mathcal{A}} \in \{1, \ldots, 10\}$



(b) $j_{\mathcal{A}}$ for $l \in \{1, \ldots, 10\}$

Figure 5.3: Average countered option potential $\bar{j}_{\mathcal{A}}^{g_d}$ and average option defensibility $j_{\mathcal{A}}$ (with standard errors of the mean) for $n_{\mathcal{A}}$ and l

experimental results indeed show this behaviour.

5.4.3 Most influential parameters

It was established that individual parameters can influence the metrics in different ways, that is, positively, negatively or not at all. These effects are easily seen when an individual parameter is plotted for a certain interestingness metric. However, this does not provide a full insight into the dynamics between the parameters. For example, it might be that the size of the goals seed set has no effect unless the number of assigned goals to agents is high (or low). It is important to capture the interplay of parameters as so to find the parameters that are most influential to the various metrics. For a deliberating agents experiment that uses the scenario generation process, it is interesting to understand which parameters to adjust in order to create more or less interesting scenarios, for example when measuring the performance of deliberation strategies with a high (or low) availability of defensible options.

To assess what the optimal configuration of input parameter settings is, a multiple linear regression analysis was performed on the experimental data. This creates a model of the data and, in a stepwise fashion, determines the influence that parameters have on one of the interestingness metrics. Note that multiple linear regression analysis assumes the effect of parameters to have a linear effect, but a look at the individual parameters showed that this assumption can be safely made and no transformations were necessary.

Out of the 12 input parameters 8 have a statistically significant influence on the option potential ratio $j_{\mathcal{A}}^{g_d}$, with $F(12, 1987) = 77.77$, $P < 0.001$ and adjusted R-squared 0.32. This means that 32% of the variance in $j_{\mathcal{A}}^{g_d}$ can be explained from the 8 input parameters. For $j_{\mathcal{A}}$ 8 parameters have a statistically significant influence, with $F(12, 1987) = 157.5$, $P < 0.001$ and adjusted R-squared 0.48, that is, explaining 48% of the variation in $j_{\mathcal{A}}$. Table 5.3 shows the input parameters with significant influence ordered by the amount of influence (standard beta coefficient), t and P values (probability values, where non-significant parameters are labelled NP).

Out of the 12 input parameters 8 have a statistically significant influence on the option potential ratio $\bar{j}_{\mathcal{A}}^{g_d}$, with $F(12, 719) = 51.46$, $P < 0.001$ and adjusted R-squared 0.45. This means that 45% of the variance in $\bar{j}_{\mathcal{A}}^{g_d}$ can be explained from the 8 input parameters. For $\bar{j}_{\mathcal{A}}$ 9 parameters have a statistically significant influence, with $F(12, 1068) = 92.96$, $P < 0.001$ and adjusted R-squared

Table 5.3: Input parameters and their influence on $j_{\mathcal{A}}^{g_d}$ and $j_{\mathcal{A}}^{g_d}$

	option potential $j_{\mathcal{A}}^{g_d}$				option defensibility $j_{\mathcal{A}}$			
	β	t	P	*ideal*	β	t	P	*ideal*
Conflicts			< 0.001	D			< 0.001	C
$n_{\mathcal{A}}$	0.04	1.89	NS	10	0.03	1.70	NS	7
$n_{\mathcal{R}}$	−0.09	−4.57	< 0.001	6	−0.09	−5.60	< 0.001	8
$n_{B_{S\mathcal{K}}}$	−0.10	−5.17	< 0.001	20	−0.11	−6.50	< 0.001	40
$n_{O_{S\mathcal{K}}}$	−0.08	−4.24	< 0.001	30	−0.11	−6.78	< 0.001	34
$n_{G_{S\mathcal{K}}}$	−0.04	−1.99	< 0.05	14	−0.01	−0.43	NS	12
n_{O_r}	−0.04	−2.02	< 0.05	8	−0.13	−7.82	< 0.001	8
n_{G_r}	0.00	0.14	NS	4	0.02	1.02	NS	5
l	−0.48	−24.63	< 0.001	2	−0.61	−35.86	< 0.001	2
$n_{G_a^{\bar{r}}}$	−0.01	0.39	NS	6	−0.00	−0.03	NS	2
$n_{B_a^r}$	0.28	14.24	< 0.001	65	0.17	10.24	< 0.001	70
$n_{B_a^{\bar{r}}}$	−0.01	−0.28	NS	0	0.32	18.67	< 0.001	45

Table 5.4: Input parameters and their influence on $j_{\mathcal{A}}$ and $\bar{j}_{\mathcal{A}}$

	countered option potential $\bar{j}_{\mathcal{A}}^{g_d}$				countered option defensibility $\bar{j}_{\mathcal{A}}$			
	β	t	P	*ideal*	β	t	P	*ideal*
Conflicts			< 0.001	D			< 0.001	D
$n_{\mathcal{A}}$	0.40	14.52	< 0.001	10	0.43	19.60	< 0.001	10
$n_{\mathcal{R}}$	−0.02	−0.55	NS	10	−0.01	−0.36	NS	10
$n_{B_{S\mathcal{K}}}$	−0.13	−4.38	< 0.001	60	−0.12	−5.19	< 0.001	60
$n_{O_{S\mathcal{K}}}$	0.06	1.94	NS	40	0.10	4.47	< 0.001	40
$n_{G_{S\mathcal{K}}}$	0.01	0.48	NS	14	−0.04	−1.67	NS	14
n_{O_r}	−0.20	−6.94	< 0.001	3	−0.14	−6.05	< 0.001	3
n_{G_r}	−0.01	−0.46	NS	3	−0.02	−0.78	NS	3
l	−0.07	−2.09	< 0.05	3	−0.10	−4.03	< 0.001	3
$n_{G_a^{\bar{r}}}$	0.07	2.36	< 0.05	8	0.06	2.48	< 0.05	8
$n_{B_a^r}$	0.34	11.20	< 0.001	80	0.30	12.91	< 0.001	80
$n_{B_a^{\bar{r}}}$	0.06	2.16	< 0.05	15	0.08	3.14	< 0.01	15

0.51, that is, explaining 51% of the variation in $\bar{j}_{\mathcal{A}}$. Table 5.3 shows the input parameters with significant influence ordered by the amount of influence (standard beta coefficient), t and P values (probability values, where non-significant parameters are labelled NP). Note that the conflict method has a clear statistically significant influence on the countered option potential and defensibility ratios, reflecting Figure 5.1.

Different interesting results can be derived from the statistical analysis. Foremost, when experimenting with the deliberation system and scenario generation process of this paper the length of rule chains l is the first parameter that should be varied when a differing degree of option potential $j_{\mathcal{A}}^{g_d}$ or defensibility $j_{\mathcal{A}}$ is to be tested. The bigger l is, the smaller $j_{\mathcal{A}}$ will be, in line with the intuition that it increases belief disparity. On the other hand, if focus of an experiment is on countered option potential and defensibility the number of agents $n_{\mathcal{A}}$ should be varied instead, since that will have most influence on $\bar{j}_{\mathcal{A}}^{g_d}$ and $\bar{j}_{\mathcal{A}}$. Strikingly, yet not totally surprisingly, the number of non-role originating beliefs $n_{B_a^{\bar{r}}}$ has a particularly high influence on option defensibility and countered option defensibility, but not so much on the option potential and countered option potential. Find both arguments and counterarguments for personal rules and beliefs is influenced more by knowledge not shared with other agents in the role than is the case for constructing arguments for the mutual goal.

5.4.4 Combined parameter influence

The most influential parameters have been established, but not yet the configuration of parameters that gives the most interesting dialogues, that is, that maximizes $j_{\mathcal{A}}^{g_d}, j_{\mathcal{A}}, \bar{j}_{\mathcal{A}}^{g_d}, \bar{j}_{\mathcal{A}}^{g_d}$ and $\bar{j}_{\mathcal{A}}$. This will often be the starting point when experimenting since that configuration offers the agents scenarios with most chances of proposing options and countering those. The parameter configuration that maximizes one of the metrics will be called the optimal configuration and can be found using the linear regression model used above. As the model predicts precisely (with $P < 0.001$) the outcome of each metric, it can also be used to find the expected maximal value. A sufficiently large data set ($N = 2000$) is produced for all four metrics through extrapolation of the regression model, that is, predicating the outcome for every possible scenario configuration. In this new data set the optimal values are found for each input parameter, which are shown in Tables 5.3 and 5.4.

Fortunately, the optimal values for each of the four metrics are largely compatible. There do not seem to be parameters that have inverse optimal settings for one metric versus another. An experiment that uses the scenario generation model can set the parameters without having to make big trade-offs in expected values for all four metrics. Note that the regression model can find the optimal setting, while generalizing over all parameters. For example, although for the option potential it individually was best to have chains of length $l = 1$, the regression models shows that the optimal setting is actually $l = 2$. Chapter 8 utilizes the ideal setting for the running of experiments with deliberation dialogues, but for a note on the practical application of the ideal parameter settings see 8.1.2.

5.5 Employing interesting scenarios

Few examples exist in the literature in which scenarios are generated for multi-agent systems. Relevant to arguing agents, there are two studies that have to deal with the generation of scenarios. In both studies the goal is to experiment with agent strategies in negotiation dialogues. Karunatillake et al. (2009) generate scenarios for agents that negotiate about social influence, actions to perform and commitments. In their experiment, scenarios are task allocation problems. Agents are randomly allocated actions (with a certain utility) within some reasonable bounds which are controllable using parameters in roughly the same manner as the scenarios generation model of this thesis. Also similar is how roles are used to assign, in their case, social relationships and commitments to the agents, although there is no non-role-originating allocation of resources. The major difference between this thesis and the scenarios of Karunatillake et al. (2009) is that their scenarios have no allocation of beliefs and defeasible inference rules. Rather, the agents decide whether to support or attack a proposal based on the direct utility that they assign to options. Since the agents do not construct structured arguments, they also have no need to generate rules and beliefs, while, as shown above, this is the most complex part of generating scenarios.

The second relevant study is that of Pasquier et al. (2010). Their project also included negotiating agents, but focusses on finding agreement on a division of scarce resources rather than a set of actions bound to a social context. The approach in this study is to experimentally find whether providing arguments

to back up proposals is beneficial with respect to the likelihood of reaching an agreement. Scenarios, which are called domains in this paper, are generated by randomly allocating goals, resources and budgets to agents, controlled by various parameters. As with the work of Karunatillake et al. (2009), no beliefs and rules are allocated to agents, because their agent strategies do not construct structured arguments. Furthermore, neither of these two negotiation-based based studies provide a handle to find the optimal settings for their parameters or study the influence of individual parameters. Rather, they have to run the agent strategy experiments with all possible parameter configurations available, within a reasonable bound. Specifically, Pasquier et al. (2010) need to run 168200 negotiation dialogues to test one strategy, while still several parameters are fixed.

In contrast to these existing studies, this chapter has shown how agents can be generated that may use structured argumentation to find support for options. Moreover, it was discussed how an optimal configuration of input parameters can be established that ensure a maximal interestingness of scenarios possible with the presented scenario generation model. The model proved capable of generating scenarios that spur discussion of proposals through arguments and counterarguments. This provides a solid basis for the experimentation with different agent deliberation strategies.

Agreeably, the used scenarios are still relatively simple to the sophisticated and expressive deliberation situations that humans use. It would be interesting to further develop and improve scenario generation. One suggestion is to allow for rule chains that use more than one premise per rule and include strict rules. Such rules would allow for more complex arguments. Another suggestion is to have various specialized pools of beliefs, instead of the simple single set of beliefs used now in the context. Example pools are expert knowledge, common sense knowledge and an internet knowledge base. This change potentially makes the generated scenarios more realistic.

Chapter 6

Agent behaviour

With the dialogical model and a way to generate knowledge for agents, all that is still required for deliberation is a definition of agent behaviour. As will be shown, behaviour of agents in a deliberation dialogue consists of two parts: knowledge revision and the deliberation strategy. Knowledge revision is the process of analysing the ongoing dialogue and updating the personal rules, beliefs, goals and options accordingly. On the other hand, the deliberation strategy determines how an agent selects new moves to play in the dialogue.

This chapter introduces a high-level deliberation behaviour model for agents, based on the decision procedure of making and defending a proposal in a deliberation dialogue. Grounded in the typical design of BDI agents, several basic heuristics are discussed for simple yet rational agent behaviour.

6.1 Decision procedure

Deliberation is a complex decision making process, taking into account a dialogue situation, personal knowledge, a mutual goal and more, and deciding what move to make from many possibilities. To understand the structure behind this process, it is useful to take a look at how humans make strategic decisions in deliberation dialogues. Walton (2007) has studied the relation between formal argumentation, practical reasoning, agent theory and human deliberation in terms of how proposals are made, attacked and defended. He reasons that any deliberating agent should ground its behaviour in the making and evaluation of proposals. BDI agents, he argues, are at least partially self-interested and therefore evaluate how an option is beneficial using practical reasoning, making

and defending a proposal for those options that seem beneficial.

The behavioural model in this chapter will follow Walton's approach of treating the game-strategic part of agent behaviour like a decision procedure. First, the current situation is evaluated by analysing the laid out proposals and their arguments. Agents can evaluate proposals by considering if the arguments are indeed valid support to a proposal and establish those points in the dialogue where it has conflicting beliefs.

Second, from its own knowledge and those alternatives already proposed, it can decide which options are beneficial to the agent using practical reasoning. A self-interested BDI agent has personal goals which it tries to satisfy. In Section 5.1 it was shown how agents construct practical arguments that connect their personal goals to known options. Such practical arguments give reasons to propose or defend an option in the dialogue. Conversely, an agent might want to attack a proposal if it sees no practical benefits.

Third, the minimal protocol for deliberation dialogues, introduced in 3.4, shows how agents can be enforced to support proposals using practical arguments that connect the mutual goal to the proposed option. A deliberation strategy should therefore use the evaluation of the current dialogue situation to find practical arguments for beneficial proposals without proper support or ask for support for existing proposals that are not beneficial.

There are two elements to deliberation strategies that Walton deliberately does not go into: revising knowledge and the combination of deliberation with other dialogue types, especially persuasion. As agents evaluate the dialogue moves of other agents, they might or might not agree with the statements of other agents. If an agent is convinced by the arguments of other agents, it may want to update its own knowledge, a process known as belief revision or knowledge revision. While being a widely researched topic in agent systems, not many studies have as of yet considered belief revision in the light of argument-based dialogues. Paglieri and Castelfranchi (2004) identify several problems with classic non-argumentative belief revision and show how an argumentation-based approach can derive logical conclusions even while the underlying data are inconsistent. An example application is given that uses trust for accepting or refuting beliefs. Snaith and Reed (2012a) have proposed a specific strategy for adopting and dropping beliefs. Interestingly, it shows how only an approach with structured arguments, as opposed to abstract argumentation models, can display certain benefits of argumentation-based belief revision.

The final relevant aspect of the deliberation decision making process is the

combination of practical and epistemic reasoning. In Chapter 2 it was shown how practical arguments can be combined with epistemic ones in the ASPIC− framework. Agents will therefore consider playing epistemic as well as practical arguments in the dialogue. Fortunately, the dialogue model of Chapter 3 models the connection between the two forms of arguments already, by way of allowing epistemic argue moves to be played that defeat practical arguments through rebuttal, undermining or undercutting. This way, agents can combine the practical reasoning about options with epistemic knowledge that was assigned to them.

To sum up, a deliberation strategy can be seen as a decision process of four distinct steps:

1. Evaluate new dialogue moves and revise knowledge
2. Determine desirability of known options by considering personal goals
3. Evaluate the status of dialogue moves and identify points of attack
4. Make a move, such as proposing an option or playing an attacking or defending argument

While this decision process is still abstract, and certainly allows for a great variety of concrete strategies, it can be used as a template for deliberating agents, in the form of a knowledge revision approach and a deliberation strategy model that formalises the decision procedure.

6.2 Knowledge revision

The first part of agent behaviour concerns the revision of personal knowledge, based on new moves that are made in the dialogue. This step is distinct from the agent's deliberation strategy, which models the last three steps in the decision process as a strategy in a game-theoretic sense. This separation, as well as the separation of the strategy's individual steps, has two purposes. First, it isolates and makes concrete the decisions that an agent has to make. Second, the isolation of steps allows for a comparison of strategies on individual parts of the decision process. This is, for example, useful in an experiment with deliberation strategies. Individual elements of the strategy can be isolated for experimentation, such as belief revision or attitude assignment.

Knowledge revision is formalized as a function that relates to a specific ongoing dialogue as described in Chapter 3 and with agents that are assigned knowledge as in Chapter 4. New moves in the ongoing dialogue provide new information and an agent might want to update its internal model accordingly.

Definition 6.1 Given a deliberation dialogue context $\mathcal{DK} = \langle AS, L_t, L_c, \mathcal{P}, \mathcal{A}, g_d \rangle$ and an agent $a = \langle r, R_a, B_a, G_a, O_a \rangle \in \mathcal{A}$, a *knowledge revision function* $\mathcal{B}_a : D \times Pow(AR) \times Pow(L_b) \times Pow(L_g) \times Pow(L_o) \longrightarrow Pow(AR) \times Pow(L_b) \times Pow(L_g) \times Pow(L_o)$ maps dialogue d and the agent a's rules R_a, beliefs B_a, goals G_a and options O_a to a revised set of rules R'_a, beliefs B'_a, goals G'_a and options O'_a.

Revising knowledge can be as simple as not changing any rules, beliefs, goals or options, that is, $\mathcal{B}_a(d, R_a, B_a, G_a, O_a) = R_a, B_a, G_a, O_a$. On the other hand, the knowledge revision function may incorporate more complex concepts, such as trust or opponent modelling.

The above mentioned-work of Snaith and Reed (2012a) is one, fairly complex, example of an implementation of the knowledge revision function. The set of beliefs is updated by adopting or dropping beliefs from it, based on the notion of minimal change. Options and goals do not exist in their model and the set of rules is never updated.

6.3 Strategy model

The strategy model involves the last three steps of the deliberation decision process described above. These steps are modelled as three functions, where the output of one is used as the input of the next step. A concrete strategy may implement a step in a very trivial or very complex way. For example, attitude assignment can simply mark to pursue every known option or it can evaluate options based on individual goal utility and feasibility predication methods. It is eventually up to the designer of a concrete strategy to make these decisions.

Every function defined below will assume a deliberation dialogue context $\mathcal{DK} = \langle AS, L_t, L_c, \mathcal{P}, \mathcal{A}, g_d \rangle$, an associated ongoing dialogue d and apply to an agent $a = \langle r, R_a, B_a, G_a, O_a \rangle \in \mathcal{A}$.

6.3.1 Attitude assignment

After an agent has potentially updated its knowledge base, it may perform an analysis on how beneficial the various options are that it knows about. Before any move can be made, an agent will want to determine which options it should support, or pursue, and which it should attack. The idea is that an agent assigns an attitude to each option it knows about.

Definition 6.2 Let $H = \{\mathsf{build}, \mathsf{destroy}, \mathsf{indifferent}\}$ be the set of all strategy attitudes. An agent a's *attitude assignment function* $\mathcal{H}_a : D \times L_o \longrightarrow H$ maps a dialogue d and one of the agent's options $o \in O_a$ to a *strategy attitude* $h \in H$.

The specific implication of assigning one of the attitudes to an option is not specified, but there is an intuitive meaning to building or destroying an option. Options which the agent deems beneficial are assigned a build attitude, options that seem disagreeable are assigned a destroy attitude and an indifferent attitude is given to those options that the agent considers of no particular interest at the time or at least requires no action in the form of new moves. An agent will probably want to assign an attitude to every option it knows about, that is, the options that were assigned to the agent as well as options that were already proposed in the dialogue. However, it is up to the actual implementation to make this design decision.

Below, in Section 6.4.3, a concrete heuristic is proposed that self-interestedly assigns attitudes using the private goals of an agent. In the literature, the assignment of attitudes to individual proposals has seen some interest in the form of layered strategy designs for persuasion dialogues. Amgoud and Maudet (2002) use build and destroy attitudes (which they refer to as strategies) that are chosen on the basis of a so-called level of prudence, indicating how reluctant an agent is to expose its arguments. A local argumentation system is used to determine the status of propositions. A build attitude is chosen by the agent if it can, on the basis of its current knowledge, construct an acceptable argument for the persuasion topic. Instead, if it can construct an acceptable argument for the contrary, it will choose a destroy attitude. The level of prudence determines if the argument's conclusion, the persuasion topic or its contrary, needs to be sceptically or credulously acceptable. In consequence, the level of prudence is a parameter that makes the agent more or less strict on the required arguments for an option.

Amgoud and Maudet's attitudes can be modelled as an attitude assignment function, adjusted for deliberation by considering the individual options as persuasion topic.

Example 6.1 Consider some dialogue d and agent $a = \langle R_a, B_a, G_a, O_a \rangle$ with associated local ASPIC− argumentation system AS with set of rules R_a and knowledge base $K = B_a \cup G_a \cup O_a$. For an Amgoud and Maudet-style attitude assignment with prudence level l it holds that for every option $o \in O_a$, $\mathcal{H}_a(d, o) = h$ such that $h = \mathsf{build}$ if o is l-acceptable in AS, $h = \mathsf{destroy}$ if $\neg o$ is l-acceptable in AS or otherwise $o = \mathsf{indifferent}$. Prudence level l requires either a credulously or sceptically acceptable argument for $\mathsf{conc}(A)$.

6.3.2 Attack point identification

After establishing which options it should build or destroy, an agent needs to identify the points of attack in a dialogue. As explained in Chapter 3, every proposed option in the dialogue is associated with a proposal tree, with which the dialogical status of moves is determined. Typically, an agent wants to influence the status of moves, so that a beneficial proposal is built and one that is not beneficial is destroyed. Moves that an agent likely wants to attack are $\mathsf{propose}$ moves, since these usually, depending on the protocol, influence directly which proposed option is selected as dialogue outcome. In any case, an agent needs to identify the points in the dialogue tree where it can attack.

Definition 6.3 An agent a's *attack point identification function* $\mathcal{I}_a = D \times L_o \longrightarrow Pow(M)$ maps a dialogue d and an option $o \in O_a$ to a set of *attack points* N such that for every $m \in N$ it holds that $m \in d$.

Example 6.2 Consider the example dialogue of Figure 6.1, originally proposed in Section 3.2.3. Two agents are deliberating on where to go for dinner. Two proposals were made and various practical and epistemic arguments were played, which gave rise to two proposals trees, with both the propose moves being *in*.

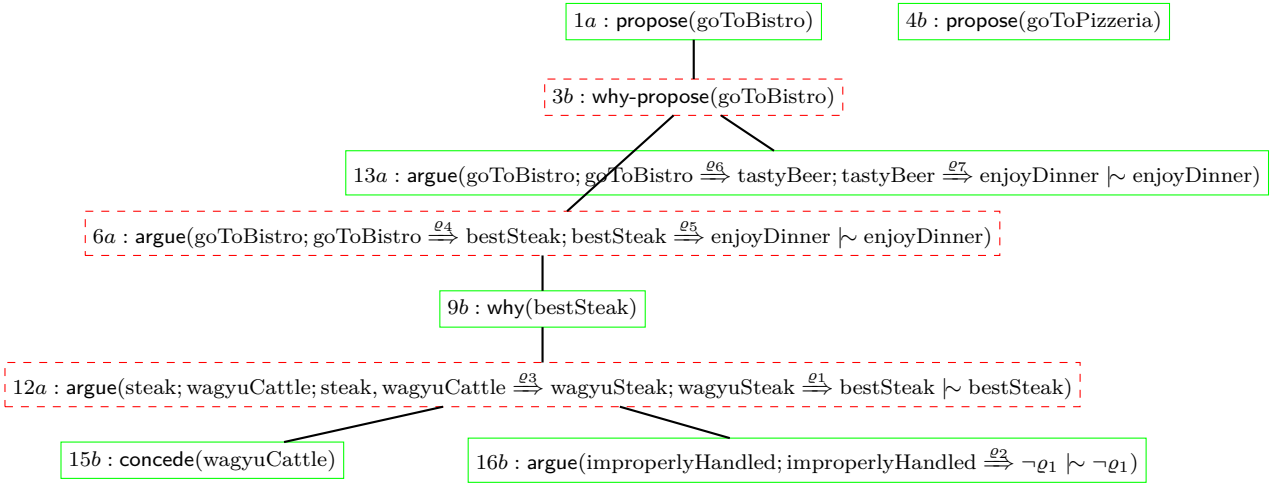


Figure 6.1: Copy of the proposal trees example dialogue of Figure 3.1

Say that an agent wants to prevent the option to goToBistro to be selected as dialogue outcome, that is, $\mathcal{H}_a(d, \text{goToBistro}) = \mathsf{destroy}$. This can be realized by flipping the status of the associated propose move to *out*. The obvious move to identify as attack point is propose move $\mathsf{propose}$(goToBistro). In addition, there are more moves that would make the $\mathsf{propose}$(goToBistro) move *out*, such as attacking move 13. While an attacking move targeting 16 would not flip the status of the original propose move, some agent strategies might still want to include 16 as an attack point. For example, if a dialogue protocol is used that only requires agents to play at least weakly relevant moves, an attack on 16 is a legal move, creating a new winning part, which an agent might consider useful.

Existing models for argumentation-based dialogue strategies have generally not considered attack point identification as a separate step in the strategic decision process. One reason is that most proposed protocols only allow replies to the last move, so no attack point has to be chosen. Nevertheless, sometimes the separation with move generation has been modelled more or less implicitly. Black and Atkinson (2011) have introduced a strategy for deliberation dialogues where agents need to decide between different plausible moves. They propose a solution through opponent modelling. Interestingly, the separation between gathering possible plausible moves and the actual move selection is similar to the separation of attack point identification and actual move generation. One difference is that the deliberation model of Black and Atkinson only allows for practical arguments, while the model in this thesis combines practical and epistemic arguments and treats them similarly in the identification of attack points.

6.3.3 Move generation

The final step in the strategic decision making process is to select the actual move to play. The assigned attitudes and the identified attack points are used in this move generation process. The selected move is required to be legal according to the dialogue protocol that is in place.

Definition 6.4 An agent a's *move generation function* $\mathcal{G}_a = D \times Pow(AR) \times Pow(L_b) \times Pow(L_g) \times Pow(L_o) \longrightarrow M$ maps current dialogue d, rules R_a, beliefs B_a, goals G_a and options O_a to a single move m such that $m \in \mathcal{P}(d)$.

Generating a move to play is not an easy task, even if the attitude towards individual options is already determined using $\mathcal{H}_a$ and attack points have been identified with $\mathcal{I}_a$. To begin with, if an agent wants to attack an identified attack point there might be multiple attacking locutions available. Next, aside from attacking, an agent may want to play surrendering moves, such as conceding to a claim by another agent to divert the focus of the discussion. Then, with the type of locution chosen, the actual content still needs to be determined. An argue move needs to contain an argument that defeats the target, but such an argument may use rebuttals, underminers and undercutters on various premises and rules. On top of that, there might be many attack points and a move for only one of them can be made. Finally, there are other types of moves to consider, such as propose, inform, prefer and prefer-equal.

Most studies into argumentation dialogue strategies have focused on the move generation step of the four-step decision process. Often a local argumentation system is used to evaluate a claim or argument in an attack point. For instance, the strategy presented in Amgoud and Maudet (2002) considers the last move and, if it is an attack point, chooses an acceptable argument with the contrary conclusion, if one exists.

Example 6.3 Consider an Amgoud and Maudet-style move generation strategy on the example dialogue d of Figure 6.1 and an agent a with $\mathcal{H}_a(d, \text{goToBistro}) = \mathsf{destroy}$ and $\mathcal{I}_a(d, \text{goToBistro}) = N$ such that last move $m_{16} \in N$. Note that $\mathsf{content}(m_{16}) = \mathsf{argue}(A)$ where $\mathsf{conc}(A) = \neg\varrho_1$ and $\mathsf{prem}(A) = \{\text{improperlyHandled}\}$. Given a local ASPIC− argumentation system AS with set of rules R_a and knowledge base $K = B_a \cup G_a \cup O_a$ and prudence level l, $\mathcal{G}_a(d, R_a, B_a, G_a, O_a) = m$ where

- $\mathsf{target}(m) = m_{16}$ and $\mathsf{content}(m) = \mathsf{argue}(B)$ if $\mathsf{cf}(\mathsf{conc}(A))$ is l-acceptable on the grounds of argument B in AS and $\mathsf{argue}(B) \notin d$, or else,
- $\mathsf{target}(m) = m_{16}$ and $\mathsf{content}(m) = \mathsf{why}(\mathsf{conc}(A))$ if $\mathsf{why}(\mathsf{conc}(A)) \notin d$, or else,
- $\mathsf{content}(m) = \mathsf{skip}$.

Note that the original strategy from Amgoud and Maudet (2002) only considers the last move in the dialogue as attack point, but it is straightforward to extend this. Section 6.4.6 will show an algorithm that supports the consideration of all attack points in move selection, among other properties.

6.4 Basic heuristics

The proposed knowledge revision function and strategy model for deliberating agents are still merely templates. The four functions that define agent behaviour still have to be implemented. Some example implementations from the literature have already been discussed. Several useful alternatives are now introduced, which will also be used in the experimental evaluation of Chapter 8. This includes several heuristics that are not yet full implementations of the functions, but rather help narrow down the choices from the very general agent behaviour functions.

6.4.1 Simple belief revision

Knowledge revision has been widely studied in the field of agent technology. Therefore, it might be expected that there is largely an agreement on the approach of updating beliefs, goals and options. However, this is not the case. By any means, knowledge revision is a difficult issue and the introduction of an argumentation-based approach, albeit with lots of potential, only makes it more delicate. For example, while agent systems commonly use logics that disallow conflicting beliefs, an argumentation-based approach can comfortably belief both p and $\neg p$ as an argumentation system proves acceptability through underlying arguments.

Instead of offering a complex knowledge revision alternative here, this thesis will primarily adopt a simple revision function. The idea is that an agent never updates its personal beliefs, with the exception of adopting knowledge of any proposed option in the dialogue.

Recall definition 3.17 The set of *proposed options* for a dialogue d is the set of options $Q_d = \{o | \mathsf{propose}(o) \in d\}$.

Definition 6.5 A knowledge revision function for agent a satisfies the *minimal knowledge revision heuristic* if $\mathcal{B}_a(d, R_a, B_a, G_a, O_a) = (R'_a, B'_a, G'_a, O'_a)$ such that $Q_d \subseteq O'_a$.

Having all proposals in the knowledge base is a reasonable heuristic, because an agent will generally want to consider all options, not just those that it happened to know about at the start of the dialogue.

6.4.2 Naive and considerate revision

A more elaborate implementation of knowledge revision $\mathcal{B}_a$ function is to adopt beliefs and rules used in the dialogue into the personal knowledge. As the argue and inform moves reveal knowledge, the agent can, for example, directly adopt them. Alternatively, an agent can adopt only those beliefs for which it has no conflicting knowledge.

Definition 6.6 Let a be an agent with an ASPIC− argumentation framework AS with $R_d = R_a$ and $K = B_a \cup O_a$. Let $d = \langle m_0, \ldots, m_n \rangle$ be a dialogue and $B_{m_n} \subseteq L_b$ be a set of revealed beliefs by move m_n such that:

- $B_{m_n} = \mathsf{prem}(A)$ if $\mathsf{content}(m_n) = \mathsf{argue}(A)$,
- $B_{m_n} = \{p\}$ if $\mathsf{content}(m_n) = \mathsf{inform}(p)$,
- $B_{m_n} = \emptyset$ otherwise.

A knowledge revision function for agent a satisfies the *naive revision heuristic* if $\mathcal{B}_a(d, R_a, B_a, G_a, O_a) = (R_a, B_a \cup B_{m_n}, G_a, O_a \cup Q_d)$.

Let B'_{m_n} be a set of beliefs such that $B'_{m_n} = \langle p | p \in B_{m_n}$ and no argument A such that $\mathsf{conc}(A) = \neg p$ can be constructed on the basis of $AS \rangle$. A knowledge revision function for agent a satisfies the *considerate revision heuristic* if $\mathcal{B}_a(d, R_a, B_a, G_a, O_a) = (R_a, B_a \cup B'_{m_n}, G_a, O_a \cup Q_d)$.

The naive variation revises knowledge by adopting any premise used in an argue move and any belief used in an inform move. The considerate variation only adopts a belief if it can not construct a counterargument to it. This alternative is therefore more careful. In any case, from the moment the agent incorporates a belief, it can use the belief in its reasoning process and play it in a dialogue move. It is expected that this style of knowledge revision influences the course of a dialogue, as agents can learn that certain (possibly negated) beliefs hold, making it possible to play arguments that it could otherwise not have constructed. An experiment will have to show whether the various metrics for deliberation will indeed show differences.

6.4.3 Goal-based attitudes

Agents in this thesis are partially self-interested, something which is implemented in BDI agents by means of their personal goals. For this reason it

makes sense to use personal goals in the assignment of attitudes. More precisely, attitudes can be assigned based on the potential that an option has to satisfy personal goals. To start with, each goal that an agent has is supposed to have a certain utility, expressed as a numeric value.

Definition 6.7 An agent $a = \langle r, R_a, B_a, G_a, O_a \rangle$ has for every goal $g \in G_a$ a *goal utility* $U_a^g \in \mathbb{N}$ and has a *combined goal utility function* $\mathcal{U}_a : Pow(L_g) \longrightarrow \mathbb{N}$ that maps a set of goals $G \subseteq G_a$ to a *combined goal utility* $u \in \mathbb{N}$.

The potential of an option can be determined by connecting to a personal goal through a practical argument, following the notion of defensible options as introduced in Chapter 5.

Recall definition 5.4 Given an agent $a = \langle r, R_a, B_a, G_a, O_a \rangle$, some option $o \in O_a$, some goal $g \in G_a$ and an ASPIC− argumentation framework AS with $R_d = R_a$ and $K = B_a \cup \{o\}$, o is an *a-g-defensible option* if an argument A can be constructed such that $\mathsf{conc}(A) = g$ and $o \in \mathsf{prem}(A)$.

An option that is *a-g*-defensible has a demonstrable benefit for the agent. Moreover, there can be multiple goals that promise to be satisfied from a single option. In that case an agent might actually prefer the option over another *a-g*-defensible option that only has promise to satisfy fewer goals, or rather, promise less utility. In any case, the self-interestness of a BDI agent is modelled through the utility that it attributes to an option by means of satisfiable goals, backed up by practical arguments.

Definition 6.8 For every option $o \in O_a$ the *option utility* is defined by $U_a^o = \mathcal{U}_a(G)$ where $G \subseteq G_a$ such that for every $g \in G$, o is a *a-g*-defensible option and such that for every $G' \subseteq G_a$ where $G \neq G'$, for every $g' \in G'$, o is a *a-g'*-defensible option and $\mathcal{U}_a(G') \leq \mathcal{U}_a(G)$.

Hence, the utility for an option is defined by the set of goals that the option promises to satisfy that give the highest combined utility. How this utility of a specific set of goals is determined is left undefined, but an implementation of the attitude assignment function that satisfies this heuristic is found below.

Example 6.4 Consider again agent a_1 from running Example 5.1, which was assigned rules, beliefs, options and goals from a scenario.

	a_1	
O_{a_1}		o_1, o_2
G_{a_1}		g_d, g_1, g_2, g_4
R_{a_1}	$R^{r_1}_{a_1}$	$o_1 \overset{\varrho 1}{\Longrightarrow} p_5,\ p_5 \overset{\varrho 2}{\Longrightarrow} p_2,\ p_2 \overset{\varrho 3}{\Longrightarrow} g_2,$
		$o_1 \overset{\varrho 4}{\Longrightarrow} p_6,\ p_6 \overset{\varrho 5}{\Longrightarrow} p_4,\ p_4 \overset{\varrho 6}{\Longrightarrow} g_d,$
		$o_2 \overset{\varrho 7}{\Longrightarrow} p_5,\ p_5 \overset{\varrho 2}{\Longrightarrow} p_2,$
		$o_2 \overset{\varrho 9}{\Longrightarrow} p_9,\ p_9 \overset{\varrho 10}{\Longrightarrow} p_1,$
	$R^{\bar{r}_1}_{a_1}$	$o_1 \overset{\varrho 23}{\Longrightarrow} p_{10},\ p_8 \overset{\varrho 25}{\Longrightarrow} g_3$
		$o_2 \overset{\varrho 26}{\Longrightarrow} p_3,\ p_5 \overset{\varrho 27}{\Longrightarrow} g_2,$
B_{a_1}	$B^{r_1}_{a_1}$	$\neg\varrho_{17}, \neg p_3,$
	$B^{\bar{r}_1}_{a_1}$	$\neg\varrho_{27}$

As was shown, the agent can connect known options to its personal goals by constructing practical arguments. Concretely, the agent can construct arguments that connect options o_1 and o_2 to its personal goal g_2. With the ASPIC− argumentation system with $R_d = R_{a_1}$ and $K = B_{a_1} \cup \{o_1, o_2\}$ of Example 5.1, the agent can construct:

$$B'' = \frac{\dfrac{o_1}{p_5}\,\varrho_1}{g_2}\,\varrho_3 \qquad C'' = \frac{\dfrac{o_2}{p_5}\,\varrho_7}{g_2}\,\varrho_3$$
$$B' = \frac{\dfrac{p_5}{p_2}\,\varrho_2}{g_2}\,\varrho_3 \qquad C' = \frac{\dfrac{p_5}{p_2}\,\varrho_2}{g_2}\,\varrho_3$$
$$B = \frac{\dfrac{p_2}{g_2}\,\varrho_3}{g_2}\,\varrho_3 \qquad C = \frac{\dfrac{p_2}{g_2}\,\varrho_3}{g_2}\,\varrho_3$$

$$D'' = \frac{\dfrac{o_1}{p_5}\,\varrho_1}{g_2}\,\varrho_{27} \qquad E'' = \frac{\dfrac{o_2}{p_5}\,\varrho_7}{g_2}\,\varrho_{27}$$
$$D = \frac{\dfrac{p_5}{g_2}\,\varrho_{27}}{g_2}\,\varrho_{27} \qquad E = \frac{\dfrac{p_5}{g_2}\,\varrho_{27}}{g_2}\,\varrho_{27}$$

To determine the utility of option o_1, the utility of every set of goals G should be considered where for every $g \in G$, o_1 is an a_1-g-defensible option and such that this set G has the highest possible utility. Out of all of agent a_1's options, o_1 is only a_1-g_1-defensible, so we only have to consider the sets $G' = \{g_1\}$ and the empty set $G'' = \emptyset$. Finally, $U^{o_1}_a = \mathcal{U}_a(G')$ if $\mathcal{U}_a(G'') \leq \mathcal{U}_a(G')$ or, alternatively, $U^{o_1}_a = \mathcal{U}_a(G'')$ if $\mathcal{U}_a(G') \leq \mathcal{U}_a(G'')$.

In a real world agent implementation the combined goal utility for the empty set G'' will probably be lower than that for G', since that includes g_1 which the agent beliefs will be satisfied. However, this is eventually determined by the concrete implementation of the attitude assignment function.

Finally, the utility of an option can be used to assign an attitude to it. Again, various implementations are possible. A reasonable assumption of rationality is that options that are assigned a build attitude never have a lower utility than another option that is assigned a destroy utility.

Definition 6.9 An attitude assignment function for agent a satisfies the *self-interested attitude heuristic* if for any two options $o, o' \in O_a$ in dialogue d it holds that if $U_a^o < U_a^{o'}$ then $\mathcal{H}_a(d, o) \neq$ build and if $U_a^{o'} < U_a^o$ then $\mathcal{H}_a(d, o) \neq$ destroy.

This is a fair assumption of basic rationality because otherwise the agent would pursue to build an option while it will destroy an option that it actually has a higher utility for, and vice versa.

6.4.4 Focused attitudes

The self-interested attitude assignment can be made concrete in various ways. Two alternatives are introduced here. Both are based on a simple way of combining the utility of a set of goals by summing the individual utilities for each goal. Although this does not cover the full complexities of real world assessment of utilities for a set of goals, it models in a straightforward way that multiple goals together have a single combined utility value.

The two attitude assignment variants will use a slight variation on which options to build. Although both will destroy any option that it does not want to build, a realistic albeit quite self-interested approach, the first is willing to build any option with positive utility while the second is even more focussed.

Definition 6.10 Given a set of goals G, the combined goal utility $\mathcal{U}_a(G) = \sum_{g \in G} U_a^g$. An attitude assignment function for agent a satisfies the *spread attitude heuristic* if for every option $o \in O_a$ in dialogue d:

- $\mathcal{H}_a(d, o) =$ build iff $U_a^o > 0$,
- $\mathcal{H}_a(d, o) =$ destroy otherwise,

An attitude assignment function for agent a satisfies the *focused attitude heuristic* if for every option $o \in O_a$ in dialogue d:

- $\mathcal{H}_a(d, o) =$ build iff for each $o' \in O_a$ such that $o \neq o'$ it holds that $U_a^o > U_a^{o'}$,

- $\mathcal{H}_a(d, o) = \mathsf{destroy}$ otherwise.

The spread attitude heuristic builds all options for which it has a positive utility, while the focused variant only build the option with the highest utility. Although the spread variant is already self-interested, it is arguably slightly less focused on maximal personal gain. How this difference in attitude assignment is reflected in a dialogue is a topic for experimentation.

6.4.5 Relevant attack points

Identifying attack points connects the attitudes on options to the moves made in the dialogue in order to find the attack points. A short example was given above, but an algorithm will now be introduced that better suits the specifics of the deliberation model of this paper. Since not only the last move can be replied to but any earlier move as well, an agent will want to find any point that allows a legal attacking move. Moreover, if a dialogue protocol restricts moves to be strongly or weakly relevant, as the confined deliberation protocol of Definition 3.15, the attack point identification function can already discard moves to which an attack would not be relevant.

The following relevant attack point algorithm identifies moves as attack points if they can be attacked in the current dialogue, while being compatible with the confined deliberation protocol of Definition 3.15 and only considering strongly relevant moves.

Definition 6.11 Given a dialogue d, an attack point identification function satisfies the *relevant attack point heuristic* for an option $o \in d$ with associated proposal move $m = \mathsf{propose}(o)$ if $\mathcal{I}_a(d, o) = \mathsf{getRelevantAttackPoints}(d, \emptyset, m, \top)$ following Algorithm 1.

The algorithm to find relevant attack points in a proposal tree works by recursively traversing, depth first, through the moves in a proposal tree. It relies on the fact that moves that are *out* are never relevant attack points and moves that are *in* are only relevant attack points if they flip the status of the propose move. Accordingly, the algorithm collects propose and attacking moves that are *in* as attack points, while stopping the traversal when a move is *out* and also replies to a move that is *out*, implemented through the parent-is-attacker parameter *pia*. Traversal has to stop at this point because from there on there

Algorithm 1 getRelevantAttackPoints

Input: dialogue d, attack point set N, move m, parent-is-attacker pia

```
1: if m is a propose move or an attacking move then
2:   if m is in then
3:     {Only include moves that are in}
4:     N = N ∪ {m}
5:     for all m' ∈ d where target(m') = m do
6:       getRelevantAttackPoints(d, N, m', ⊤)
7:     end for
8:   end if
9: else if pia then
10:  for all m' ∈ d where target(m') = m do
11:    getRelevantAttackPoints(d, N, m', ⊥)
12:  end for
13: end if
14: return N
```

will be no strongly relevant moves in the tree, since even when making the move at this point in will never flip the status of any move upwards in the tree.

Proposition 6.1 Given a dialogue d, $m \in \mathsf{getRelevantAttackPoints}(d, \emptyset, m, \top)$ iff any attacking move m', such that $\mathsf{target}(m') = m$, is strongly relevant with respect to Definition 3.11.

Proof $\rightarrow$ Recall that a more is strongly relevant if it flips the status of the propose move. First consider if m is in in a branch under two subsequent *out* moves. An attacking reply on m would never be relevant as the tree root, the proposal move, status will never flip. Second, if m is not under two subsequent *out* moves, an attack on m will always flip the status of the attacked move $\mathsf{target}(m) = m'$, which in turn would flip the status of $\mathsf{target}(m')$ until the tree root, the propose move, inevitably flips status. $\leftarrow$ If m is not an attacking or propose move, it is trivially never strongly relevant. If m is an attacking or propose move, first consider if m is in a branch under two subsequent moves that are *out*. It would is not relevant as an attack on m will never flip the propose move status. Second, if m is not under two subsequent *out* moves, an attack on m will not flip the status of the targeted move $\mathsf{target}(m)$ and will therefore never flip the status of the propose move, so it is never strongly relevant.

Example 6.5 Consider again the example deliberation dialogue in Figure 6.1. The getRelevantAttackPoints algorithm identifies the attack points for the goToBistro tree by starting at its root. Since the propose move m_1 is *in*, this needs to be included. It is always a point that can be attacked with strong relevance. Note that there may be no more valid moves under the dialogue protocol, but this is not part of the attack point identification process. The propose move's reply why-propose(goToBistro) is not *in* and therefore not a relevant attack point. After all, an attack on this would not influence the move's status, as it already has an attacker that is *in*, and consequently not influence the status of the propose move. We remember that this move was *out* by setting the parent-is-attacker parameter to $\bot$. This means that its replies need to be *in* in order to keep collecting relevant attack points. Since move m_6 is not, that move and none of the moves further in this branch are collected. This makes sense, since an attacking move here will not flip the proposal move status. Instead, move m_{13} is included, being the reason that why-propose(goToBistro) is *out*. Hence, getRelevantAttackPoints$(\emptyset, m_1, \top, d) = \{m_1, m_{13}\}$.

Note that attitudes towards options are not yet used in an attack point identification function. The result of the attack point identification function is a set of attack points N from which the move generation method can select a move to actually attack, if so desired, which is guaranteed to be strongly relevant.

6.4.6 Thorough attacker generation

Even when just restricting move generation to providing attackers to the already identified attack points, different locution types are available and the actual move content still needs to be generated. A move selection function needs to decide which attack point to reply to. In case of an argue move it may reply with a counterargument or play a why move. One solution is to iterate over the attack points try to find any possible attacking move. If one is found, this is directly returned as new move. Only if no new attacking move can be made will the agent skip its turn, which is why this heuristic is called thorough. Moreover, options that are beneficial for the agent, with an assigned build strategy, are proposed if they were not proposed yet.

Definition 6.12 Let d be some dialogue and a be an agent with sets of rules R_a, beliefs B_a, goals G_a and options O_a. A move generation function satisfies the *ba-*

sic thorough attacker heuristic if $\mathcal{G}_a(d, R_a, B_a, G_a, O_a) = \mathsf{generateMove}(d, R_a, B_a, G_a, O_a)$ following Algorithm 2

Algorithm 2 generateMove

Input: dialogue d, rules R_a, beliefs B_a, goals G_a, options O_a

for all $o \in O_a$ **do**
  $h = \mathcal{H}_a(d, o)$
  {Find a propose move for this option}
  m_o a move such that $\mathsf{content}(m_o) = \mathsf{propose}(o)$
  if $m_o \notin d$ **and** $h = build$ **then**
    return $\mathsf{propose}(o)$
  else if $m_o \in d$ **and** ($h = build$ and m_o is *out*) **or** ($h = \mathsf{destroy}$ and m_o is *in*) **then**
    {Loop through all attack points}
    AS is an ASPIC− argumentation framework with $R_d = R_a$ and $K = B'_a \cup \{o\}$
    for all $m \in \mathcal{I}_a(d, o)$ **do**
      if $m = \mathsf{propose}(o)$, $\mathsf{content}(m') = \mathsf{why\text{-}propose}(o)$ **and** $m' \notin d$ **then**
        return $\mathsf{why\text{-}propose}(o)$
      else if $m = \mathsf{argue}(A)$, $\mathsf{content}(m') = \mathsf{argue}(B)$, B on the basis of AS defeats A **and** $m' \notin d$ **then**
        return $\mathsf{argue}(B)$
      else if $m = \mathsf{why\text{-}propose}(o)$, $\mathsf{content}(m') = \mathsf{argue}(A)$ where A is on the basis of AS with $o \in \mathsf{prem}(A)$ **and** $m' \notin d$ **then**
        return $\mathsf{argue}(A)$
      else if $m = \mathsf{why}(p)$, $\mathsf{content}(m') = \mathsf{argue}(A)$ where A is on the basis of AS **and** $m' \notin d$ **then**
        return $\mathsf{argue}(A)$
      end if
    end for
  end if
end for
return skip

Note that at most one move per proposal tree is made, but this is a characteristic of the algorithm and not enforced by the dialogue protocol. The algorithm is called basic because there are many improvements that can be made. First,

selecting just the first possible attacking move might not be the best solution. Second, the algorithm does not consider to play any surrenders, option preference or inform moves. Finally, although the heuristic is thorough, it might be better not to play certain attackers, even when it would flip the proposal move status in a beneficial way, because the information contained in the attacking move might be used against the agent. In conclusion, the algorithm results in agent behaviour that is self-interested, proactive and adheres to the protocol, but many improvements can still be thought of.

6.4.7 Non-arguing agents

Whether an agent uses an argumentation logic for internal reasoning, revising knowledge and assigning attitudes, is eventually not the determining factor in calling it an arguing agent. Rather, the ultimate factor in being an arguing agent is whether it can motivate its proposals and claims by playing argue moves. Hence, for the purpose of comparing arguing versus non-arguing performance, agents need to be introduced that can internally reason over proposals but that do not attack and support proposals and claims with argue moves.

From this it follows that a non-arguing agent differs to an arguing agent solely on its move generation. The non-arguing variant can still use argumentation to revise knowledge or assign option attitudes. However, it will no play any argue moves. Two move generation functions will now be introduced that implement this idea of non-arguing agents.

Definition 6.13 A move generation function for agent a in dialogue d with sets of rules R_a, beliefs B_a, goals G_a and options O_a satisfies the *propose and reject heuristic* if $\mathcal{G}_a(d, R_a, B_a, G_a, O_a) = m$ such that

- $\mathsf{content}(m) = \mathsf{propose}(o)$ if there exists an $o \in O_a$ such that $\mathcal{H}_a(d, o) = \mathsf{build}$ and $m \notin d$,
- $\mathsf{content}(m) = \mathsf{reject}(o)$ if there exists an $o \in O_a$ such that $\mathcal{H}_a(d, o) = \mathsf{build}$ and some move $m' \notin d$ with $\mathsf{content}(m') = \mathsf{propose}(o)$,
- $\mathsf{content}(m) = \mathsf{skip}$ otherwise.

Put simply, an agent employing a propose and reject heuristic will only play propose and reject moves. If it has a build attitude towards an option, it will propose this option. If the agent has a destroy attitude towards an option, it

will reject the option. Recall that rejecting a proposal is similar to questioning it with a why-propose move, except that a reject move can not be attacked, in contrast with the why-propose move to which an argument can be played showing the connection between an option and the mutual goal.

This propose and reject heuristic might seem simplistic, but it is similar to a simple negotiation strategy where agents make and reject proposals until they find one that they can agree on. Moreover, the agent can still use internal argumentation-based reasoning like the simple rational agent does, so it makes very clear the difference between playing argue moves or not.

Arguably, a non-arguing agent should still be able to expose internal beliefs, for the purpose of informing other agents. As non-arguing agents will not specifically ask for clarification, it can't rely on (relevant) arguments. Instead, if an agent deems it necessary to tell other agents about some belief, it may use the inform locution. A straightforward implementation for an agent is to simply play an inform move for each of its internal beliefs.

Definition 6.14 A move generation function for agent a in dialogue d with sets of rules R_a, beliefs B_a, goals G_a and options O_a satisfies the *propose and inform heuristic* if $\mathcal{G}_a(d, R_a, B_a, G_a, O_a) = m$ such that

- $\mathsf{content}(m) = \mathsf{propose}(o)$ if there exists an $o \in O_a$ such that $\mathcal{H}_a(d, 0) = \mathsf{build}$ and $m \notin d$,
- $\mathsf{content}(m) = \mathsf{why\text{-}propose}(o)$ if there exists an $o \in O_a$ such that $\mathcal{H}_a(d, 0) = \mathsf{build}$ and there exists no $m' \in d$ with $\mathsf{content}(m') = \mathsf{propose}(o)$,
- $\mathsf{content}(m) = \mathsf{inform}(p)$ if there exists an $p \in B_a$ and there exists no $m' \in d$ with $\mathsf{content}(m') = \mathsf{inform}(p)$,
- $\mathsf{content}(m) = \mathsf{skip}$ otherwise.

Agents that employ the propose and inform heuristic will not only make and reject proposals, but also play inform moves for the local beliefs it has. This way it can inform other agents of its personal beliefs, which may influence the others, for example if they adopt those beliefs into their own knowledge.

6.5 Agent behaviour in experiments

Now that every step in the agent's decision procedure has been formalized, they can be combined to form a single agent behaviour. This fully specifies the

strategic behaviour of an agent in a deliberation dialogue situation, by using the output of one step as input into the next.

Definition 6.15 An agent a's *agent behaviour* $\mathcal{S}_a = \langle \mathcal{B}_a, \mathcal{H}_a, \mathcal{I}_a, \mathcal{G}_a \rangle$ is a tuple such that:

- $\mathcal{B}_a(d, R_a, B_a, G_a, O_a) = (R'_a, B'_a, G'_a, O'_a)$,
- for each $o \in O'_a$, $\mathcal{H}_a(d, o) = h$ on the basis of $\mathcal{B}_a$,
- for each $o \in O'_a$, $\mathcal{I}_a(d, o) = N$ on the basis of $\mathcal{B}_a$,
- $\mathcal{G}_a(d, R'_a, B'_a, G'_a, O'_a) = m$ on the basis of $\mathcal{B}_a$, $\mathcal{I}_a$ and $\mathcal{G}_a$,

Hence, the agent behaviour function regulates that the output one step in the decision process is used as input to another. Concretely, the output of the internal state revision step $\mathcal{B}_a$ is used by all other three functions in the agent behaviour, while the attitude assignment $\mathcal{H}_a$ and attack point identification $\mathcal{I}_a$ functions are used in the last move generation step $\mathcal{G}_a$ as well.

It is interesting to see how agent behaviour influences the course of a dialogue. For this purpose, a simple rational agent is now introduced that combines the above defined heuristics into a fully specified agent.

Definition 6.16 A *simple rational agent* is an agent a with agent behaviour $\mathcal{S}_a = \langle \mathcal{B}_a, \mathcal{H}_a, \mathcal{I}_a, \mathcal{G}_a \rangle$ such that:

- $\mathcal{B}_a(d, R_a, B_a, G_a, O_a) = (R_a, B_a, G_a, O_a \cup Q_d)$,
- $\mathcal{H}_a$ implements the spread attitude heuristic,
- $\mathcal{I}_a$ implements the relevant attack point heuristic,
- $\mathcal{G}_a$ implements the basic thorough attacker heuristic.

The knowledge revision function $\mathcal{B}_a$ makes concrete the minimal knowledge revision heuristic by only adding any proposed option to its personal beliefs. Attitude assignment function $\mathcal{H}_a$, attack point identification function $\mathcal{I}_a$ and move generation function $\mathcal{G}_a$ are taken directly from the above proposed heuristics, which are already concrete. Together they form a simple yet rational agent that uses argumentation for internal reasoning and is called an arguing agent as it will play argue moves in a dialogue.

Example 6.6 Consider again the generated example scenario from Chapter 4. All three agents are simple rational agents. On top of the assigned knowledge from the scenario, which can be found in Example 4.5 on page 72, the agents have a specific utility for each of their personal goals.

a_1		a_2		a_3	
$U_{a_1}^{g_d}$	1	$U_{a_2}^{g_d}$	2	$U_{a_3}^{g_d}$	5
$U_{a_1}^{g_1}$	2	$U_{a_2}^{g_1}$	3	$U_{a_3}^{g_3}$	5
$U_{a_1}^{g_2}$	4	$U_{a_2}^{g_2}$	4	$U_{a_3}^{g_4}$	1
$U_{a_1}^{g_4}$	5	$U_{a_2}^{g_3}$	3	$U_{a_3}^{g_2}$	1

The agents will deliberate in a dialogue d and are all committed to find an option that will realise mutual goal g_d. Agent a_1 is the player of the first turn. It will first consider revising its knowledge, but since dialogue d is still an empty sequence it will not add or drop any knowledge yet.

The next step is assigning attitudes to the options it knows about, for which it first has to establish which options are defensible by which goals. For each option it constructs an argument that connects the option to each personal goal, if they exist. As seen above, and originally in Chapter 5, the agent can for both its options o_1 and o_2 construct an argument to its personal goal g_2. In fact, it can construct two of such arguments for each option, but one is enough for the option to be a-g_2-defensible. For both options the agent will now assess the utility by computing the combined goal utility $\mathcal{U}(\emptyset)$ and $\mathcal{U}(\{g_2\})$. Since the specific function will simply sum the utilities of individual goals in the set $\mathcal{U}(\emptyset) = 0$ and $\mathcal{U}(\{g_2\}) = 4$ which, by picking the set with the highest combined utility, will result in option utilities $U_a^{o_1} = 4$ and $U_a^{o_2} = 4$. Finally, the agent assigns a build attitude to options with a positive utility and a destroy utility otherwise. Since both options have a positive utility the resulting attitude assignment $\mathcal{H}_a(d, O_a) = \{(o_1, \mathsf{build}), (o_2, \mathsf{build})\}$.

The agent has determined that it wants to build both options it knows about. Since there are no moves yet in the dialogue it does not need to identify attack points yet. Move generation is also easy. The *generateMove* algorithm will first look if option o_1 is already proposed in d and if not it will propose the option if it has a build attitude. Hence, the move generation function will return $\mathcal{G}(d, S, N) = \mathsf{propose}(o_1)$.

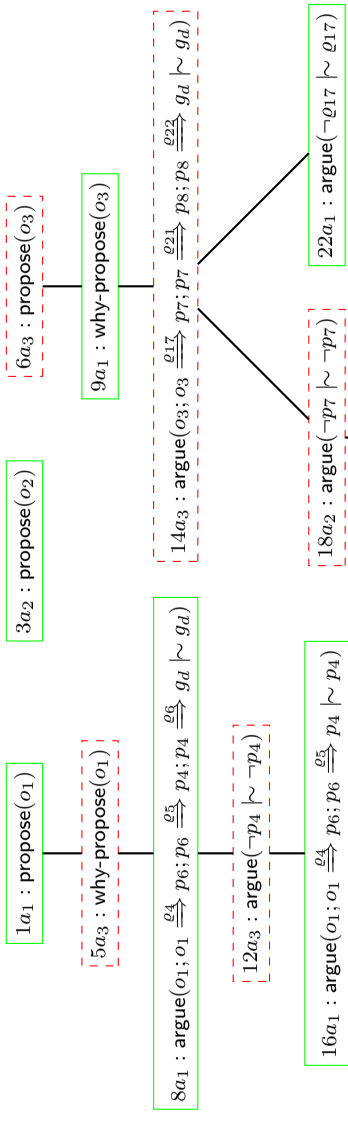


Figure 6.2: The proposal trees a dialogue with three simple relevant agents

Form here on it is still the turn of agent a_1. That will also want to propose the other option o_2 but the dialogue protocol restricts this. Instead it can only skip for now. The turn is given to the next agent in line, agent a_2, who now applies its knowledge revision and strategy functions. Eventually this will lead to the dialogue of Figure 6.2. Note that skip moves are numbered but not shown.

This chapter has formalized agent behaviour in deliberation dialogues. The knowledge revision and strategy functions provide a clear separation of individual decision steps that agents have to make in a deliberation situation. The functions allow for a great variety of possible implementations and several heuristics for concrete implementations have been introduced. Chapter 8 will use the simple rational agent to compare it against an agent that does not argue in the dialogue at all. The dialogues played by these two different agents can then be analysed to compare the benefits of using argumentation in a multi-agent deliberation dialogue by means of various desirable properties, which Chapter 7 will introduce.

Chapter 7

Dialogue metrics

The goal of this thesis is to study if it is beneficial for agents to use argumentation in deliberation-style dialogues. However, as of yet it has not been discussed what these benefits actually are. To understand what benefits argumentation can bring, it first has to be studied what actually is desirable.

This chapter will first discuss which properties are desirable in multi-agent deliberation dialogues. This will lead to the introduction and formalization of four metrics, which can be applied to deliberation dialogues to test how *good* a dialogue was. In this way, metrics can be used to compare dialogues, notably dialogues produced by different agents to compare if it is beneficial to adopt certain behaviour.

7.1 Desirable properties

To be able to measure if a dialogue exposes desirable behaviour, it must first be established what kind of behaviour is desirable. This is not straightforward, as the desirable properties of a multi-agent dialogue system are dependent on the goals of the system. For instance, the designer of a system in which different agents deliberate on a place to have dinner, likely values highly how the final outcome of a dialogue reflects the personal utilities of individual agents. On the other hand, in some systems the information that is exchanged should (also) be minimized, such as in a medical system where privacy issues call for a minimal exchange of data. A final example is a public or governmental system, where decisions need to be understandable by humans and irrelevant aspects should be ignored by the agents. In any case, every system has different desirable

properties and metrics should be defined accordingly.

A number of metrics for argumentation-based dialogue systems have been introduced in the literature, each to be able to measure a different desirable property of the argumentation systems used. Three papers are of special interest. First, Pasquier et al. (2010) have investigated how an exchange of personal goals in a negotiation dialogue may increase the likelihood and quality of agreements. Whether agents can agree with each other is measured by having agents explicitly accept or refuse a certain proposal. The experiments showed that agents that ask for and expose personal goals are more *likely to agree* on a specific proposal. Moreover the quality is measured in terms of the *execution cost* of the proposed option. Options with costly plans have a low utility for the multi-agent system, while cheaper plans are more desirable as the combined utility of the agents is higher.

Second, Karunatillake et al. (2009) have performed experiments with negotiation-style dialogues, but have used slightly different metrics. The quality of outcomes is also measured in terms of *combined utility* for the agents in the system. However, this effectiveness measure is determined by explicitly summing the utilities of agents as expressed in their personal goals, rather than computing the objective cost of a proposal. The other metric used in the experiments relates to dialogue efficiency. By *counting the number of moves* it was possible to measure how different agent behaviours influence the communication costs in terms of number of required utterances.

Third and finally, Amgoud and Dupin De Saint-Cyr (2008) propose several metrics for agent performance in persuasion dialogues. First, moves, specifically argue moves, are assigned a numeric *weight* value, which is used to calculate an agent's contribution in the dialogue and the overall dialogue weight. Second, the number of formulas (beliefs and goals) that are exposed in the dialogue are counted for every agent to assign each agent a *degree of loan*. This indicates to what degree an agent was able to reuse the knowledge of others as opposed to having to expose personal knowledge. Both proposed metrics apply to the performance of an agent, rather than to the performance of the multi-agent system. Hence, these measurement cannot directly be used to derive an objective measurement of the quality of a dialogue. However, see below for a variation of the degree of loan to apply to a dialogue as a whole.

7.2 Metrics for deliberation

Inspired by the literature, four metrics will now be introduced. The approach is that metrics for deliberation dialogues operate on two distinct dimensions: efficiency versus effectiveness and communication versus topic layers. Efficiency pertains to using limited resources, whether it is time, the required number of messages or the required amount of information that was exchanged. On the other hand, effectiveness measures how appropriate the dialogue was, such as moves being relevant or how well the dialogue outcome suits the multi-agent system. Efficiency and effectiveness can be measured for both the communication layer and the topic layer. Communication is the carrier of information, while the topic layer contains the actual arguments and knowledge. This leads to four ways in which dialogues are measured.

	efficiency	effectiveness
communication layer	move count	relevance
topic layer	information concealment	combined utility

Four metrics will now be introduced to measure if dialogues are efficient or effective on the communication or topic layers. The specifics of a multi-agent system will eventually determine which metrics should be applied. For instance, an experiment in a system with sensitive information can apply the concealment metric that tests topic efficiency.

7.2.1 Communication efficiency

Efficiency on the communication layer measures how much resources are used by the agents to play their moves in the dialogue. In a typical deliberation system a dialogue with few moves may already be considered better than one that uses many, but efficiency is of special importance if communication in a system is expensive, for example, if exchange of dialogue moves uses a paid platform. Efficiency can be measured by counting the number of moves made in a dialogue.

Definition 7.1 The *move count* of a dialogue d is measured by

$$\mathsf{movecount}(d) = |\{m | m \in d \text{ and } \mathsf{content}(m) \neq \mathsf{skip}\}|$$

Move efficiency counts all moves in a dialogue that are not skip moves, as those moves are merely used to control the flow of turn taking.

Example 7.1 Consider again the dialogue d between three simple relevant agents, which was visualized in Figure 6.2. $\mathsf{movecount}(d) = 12$ as there are 26 moves in d, but only 12 when not counting the skip moves.

7.2.2 Communication effectiveness

Another requirement on the communication is for it to be effective. Effective communication does not pertain to the amount of used resources, but to how appropriately the resources were used. Intuitively, a move is appropriate when it helps bring understanding to the topic at hand. The dialogue model already introduced a way of determining this in the form of strong and weak relevance of moves. If a move is relevant, it is an effective use of the communication layer. Moves that do not directly relate to a proposed option are considered irrelevant, for instance any inform move. Since they are irrelevant, such moves will bring down communication effectiveness of a dialogue.

Definition 7.2 Given a dialogue d, let d' be a set of moves such that for every move $m \in d$, either $m \in d'$ or $\mathsf{content}(m) \notin \{\mathsf{skip}, \mathsf{prefer}, \mathsf{prefer\text{-}equal}\}$. The *degree of relevance* of dialogue d is measured by

$$\mathsf{relevance}(d) = \frac{|\{m | m \in d' \text{ where } m \text{ was strongly relevant}\}|}{|d'|}$$

or if $|d'| = 0$ then $\mathsf{relevance}(d) = 0$.

This thesis will only measure strong relevance. The definition of strong relevance applies only to moves that are made in a proposal tree. Other moves, such as inform, skip, prefer and prefer-equal moves are never relevant. However, skip move is merely a construct to control turn taking and prefer and prefer-equal moves are useful when establishing a dialogue outcome. Intuitively they don't diminish communication effectiveness. For that reason the moves of these types will not be considered in the degree of relevance metric.

It should be noted that some arguing agent behaviours are restricted to only play relevant moves. Importantly, the arguing agent defined in Chapter 6 uses the relevant attack points heuristic and therefore only plays relevant moves. On the other hand, the non-arguing propose and inform heuristic of Definition 6.14 is an example of an agent that does not play strictly relevant moves. Chapter 9 will provide some ideas for arguing agents that might not restrict themselves to the playing of strongly relevant moves.

Example 7.2 In the example of Figure 6.2 every move that the agents made, not considering skip moves, was strongly relevant, that is, $\mathsf{relevance}(d) = \frac{12}{12} = 1$. Note that this is expected because the attack point identification function of the simple relevant agents only aggregates moves to which an attacker would be strongly relevant. Chapter 8, however, will introduce agents that communicate but are not always strongly relevant.

7.2.3 Topic efficiency

Efficiency can be measured through the number of moves that are made, but this metric does not regard how much information is exposed by a move. Various systems can be thought of where it is desirable to have as little information in the played moves as possible, such as a system dealing with privacy data or in dialogues between business competitors.

Information is exposed in a dialogue by means of three types of moves: propose, argue and inform. Agents that want to conceal their knowledge can refrain from playing these moves, or use rules, beliefs, options and goals that other agents already played in the dialogue. To measure information concealment on a dialogue as a whole, the exposed rules and formulas from the topic language are counted and compared to the total number of rules, beliefs, options and goals.

Definition 7.3 Given a dialogue d, the set of agents $\mathcal{A}$ where each $a \in \mathcal{A}$ is an agent $a = \{d, R_a, B_a, G_a, O_a\}$, the number of *exposed rules, beliefs and options*

$$\begin{aligned} x_d = | \bigcup_{m \in d} \{o | \mathsf{content}(m) = propose(o)\} \cup \\ \bigcup_{m \in d} \{p | \mathsf{content}(m) = inform(p)\} \cup \\ \bigcup_{m \in d} \{\mathsf{prem}(A) | \mathsf{content}(m) = argue(A)\} | + \\ | \bigcup_{m \in d} \{\mathsf{rules}(A) | \mathsf{content}(m) = argue(A)\} | \end{aligned}$$

and the number of *available rules, beliefs and options*

$$n_{\mathcal{A}} = | \bigcup_{a \in \mathcal{A}} R_a | + | \bigcup_{a \in \mathcal{A}} B_a | + | \bigcup_{a \in \mathcal{A}} O_a |$$

the *degree of information concealment* of a dialogue d is measured by

$$\mathsf{conceal}(d) = \frac{n_{\mathcal{A}} - x_d}{n_{\mathcal{A}}}$$

The number of rules, beliefs and options are counted that are used in the moves in the dialogue and this is compared against the total number of available rules, beliefs and options, as were assigned to the agents. Note that goals are not considered here, as only the mutual goal will be played by the agents in this thesis and this is already given in the dialogue context. The degree of information concealment represents the percentage of rules, beliefs and options that agents did not (have to) use in the dialogue, that is, which were concealed.

Example 7.3 The moves in the example dialogue represented by Figure 6.2 expose rules, beliefs and options through propose and argue moves. Exposed are $x = |\{o_1, o_2, o_3, \neg p_4, \neg p_7, \neg \varrho_{17}\}| + |\{o_1 \overset{\varrho_4}{\Longrightarrow} p_6, p_6 \overset{\varrho_5}{\Longrightarrow} p_4, p_4 \overset{\varrho_6}{\Longrightarrow} g_d, o_3 \overset{\varrho_{17}}{\Longrightarrow} p_7, p_7 \overset{\varrho_{21}}{\Longrightarrow} p_8, p_8 \overset{\varrho_{22}}{\Longrightarrow} g_d\}| = 6 + 6 = 12$ and available are $n = 30 + 7 + 3 = 40$ rules, beliefs are goals, hence $\mathsf{conceal}(d) = \frac{40-12}{40} = 0.7$.

7.2.4 Topic effectiveness

The final metric applies to the topic effectiveness, that is, how effective the result of the dialogue was for the multi-agent system as a whole. An effective dialogue result means that the result was desirable with regards to all agents in the system. As discussed above, this can be expressed in terms of the combined utility that the selected dialogue outcome has for each individual agent. The utility that agents in this thesis assign to every option is used here. The utility every agents has for the dialogue outcome is simply summed, as in Karunatillake et al. (2009).

Definition 7.4 The *combined utility* of a dialogue d with mutual goal g_d is measured by

$$\mathsf{utility}(d) = \sum_{a \in A} U_{d,a}^{\mathcal{O}(d,g_d)}$$

Recall that $\mathcal{O}(d, g_d)$ returns the option selected as dialogue outcome and subsequently $U_{d,a}^{\mathcal{O}(d,g_d)}$ returns the utility that an agent assigned to the dialogue outcome. The dialogue outcome is thus determined by the sum of utilities for the dialogue outcome by all agents. Note that in this thesis the dialogue

outcome is determined by selecting an arbitrary proposal whose propose move was *in*.

Example 7.4 Three proposals were made in the running example dialogue of Figure 6.2, but only the propose moves for o_1 and o_2 were *in*. Say that the dialogue outcome selected is $\mathcal{O}(d, g_d) = o_1$. As seen above in Example 6.6 agent a_1 has an option utility $U_{a_1}^{o_1} = 4$ for this proposal. Agent a_2 has practical arguments connecting o_1 to g_1 as well as g_3 and therefore $U_{a_2}^{o_1} = U_{a_2}^{g_1} + U_{a_2}^{g_3} = 3 + 3 = 6$. Agent a_3 has no practical argument connecting o_1 to any of its personal goals, so $U_{a_3}^{o_1} = 0$. Finally, the combined utility of the dialogue is $\mathsf{utility}(d) = 4 + 6 + 0 = 10$. Interestingly, if not o_1 but rather o_2 would have been selected as dialogue outcome then $\mathsf{utility}(d) = 4 + 7 + 1 = 12$. Option o_3 was not *in* and is therefore not considered as dialogue outcome, but the agents have a summed utility for o_3 of $0 + 0 + 1 = 1$.

Measuring the quality of the dialogue outcome by combined utility is only one way to assess topic effectiveness. In game theory a selected outcome is often evaluated on the basis of Pareto optimality. An option in a deliberation dialogue is Pareto optimal if for any other option there is at least one agent that ends up with a lower utility.

Definition 7.5 Given a dialogue d with mutual goal g_d and set of dialogue proposals Q_d, the dialogue outcome $o = \mathcal{O}(d, g_d)$ is *Pareto optimal* iff there exists no $o' \in Q_d$ where $o' \neq o$ such that for each agent $a \in \mathcal{A}$, $U_{d,a}^{o'} \geq U_{d,a}^{o}$ and for some agent $a \in \mathit{agents}$, $U_{d,a}^{o'} > U_{d,a}^{o}$.

In other words, an option is Pareto optimal of there is no other option for which every agent has at least the same utility and for which at least one agent has a greater utility. Note that a dialogue may have multiple Pareto optimal options or none at all and that the option with the highest combined utility is always Pareto optimal. Whether the combined utility or the Pareto optimality of a dialogue outcome should be used by an experiment designer as a measure for topic effectiveness depends on the specific dialogue system. Combined utility aims at a maximization of system utility, while the Pareto optimality balances individual agent preferences.

Example 7.5 The option utilities for the agents in the running example of Figure 6.2 are, as discussed above, as follows:

	o_1	o_2	o_3
a_1	4	4	0
a_2	6	7	0
a_3	0	0	1

Say again that the dialogue outcome $\mathcal{O}(d, g_d) = o_1$. This outcome is not Pareto optimal since alternative o_2 would have at least one agent, that is, agent a_2, that would be better off, without having an agent that would get a lower utility. Outcome $\mathcal{O}(d, g_d) = o_2$, however, is Pareto optimal, as option o_1 would not make any agent better off and a switch to option o_3 would make a_3 better off but not without making agents a_1 and a_2 worse off. Finally consider o_3. As the propose move for o_3 is *out*, the dialogue outcome function would never select o_3 as dialogue outcome. Recall that options whose propose moves are *out* are not considered as in the dialogue the proposal was successfully attacked. In other words, the agents provided arguments, to which no counter evidence was given, that showed why the proposal should not be considered as dialogue outcome. However, if it would be considered, o_3 would actually be a Pareto optimal outcome. Although it has a very low combined utility of 1, there is actually no alternative option that would make another agent better off without agent a_3 loosing its utility.

7.3 Application of metrics

This chapter has introduced four metrics to test deliberation dialogues on four distinct desirable properties. It focusses on testing the multi-agent system as a whole, rather than on properties that individual agents may desire. Depending on the concrete dialogue system some metrics may be more or less applicable. This is also the reason why the metrics can not be accumulated into a single generally applicable metric for dialogue performance. Individual goals of the multi-agent system have to be tested separately.

Through the four introduced metrics, objective measurements on the performance of agents in an argumentation-supported dialogue can be made. This means that it is now possible to have agents engage in a deliberation dialogue given some scenario and compare performance of various agents. Through an experimental setup the performance of arguing and non-arguing agents can be compared on all four metrics for desirable properties for deliberation dialogues.

Chapter 8

Testing the benefits of argumentation

At the start of this thesis a goal was set to investigate the benefits of argumentation in multi-agent deliberation dialogues. Put simply, benefits of argumentation can be investigated by comparing the performance of arguing versus non-arguing agents. This chapter will consider several variations of arguing and non-arguing agents and experimentally show how they perform in terms of the four metrics for deliberation dialogues defined in the last chapter.

8.1 Experimental design

An experimental platform was designed and implemented for the comparison of the performance of agent behaviours. The primary purpose was to compare arguing and non-arguing strategies, while the various alternative implementations introduced in Chapter 6 were tested as well.

8.1.1 Connecting the experimental models

The experimental platform consists of implementations for the full stack of models required to perform automatic experiments with deliberation, that is, implementations of the dialogue model, the scenario generation model, the agent behaviour model and the four deliberation dialogue metrics. The general idea is to generate many scenarios, which will then be played by specific agents within the deliberation dialogue model. Every playing of a scenario represents a single experiment run, which can be analysed using the dialogue metrics. Within a

single run every agent uses the same implementation of the four functions that define an agent behaviour. This way, alternative behaviours can be compared, such as arguing and non-arguing agents, by testing the performance of many experiment runs for each behaviour.

Definition 8.1 A *deliberation experiment run* is a tuple $\mathcal{X} = \langle \mathcal{DK}, \mathcal{SK}, \mathcal{A}, \mathcal{S}_a, d_t, \mathcal{O} \rangle$ such that

- $\mathcal{DK} = \langle AS, L_t, L_c, \mathcal{P}, \mathcal{A}, g_d \rangle$ is a deliberation dialogue context following Definition 3.1 such that
  - $\mathcal{P}$ is the confined deliberation protocol of Definition 3.15,
  - Q_j is the set of justifiable options of Definition 3.20,
- $\mathcal{SK}$ is a scenario generation context following Definition 5.1,
- $\mathcal{S} = \langle \mathcal{B}, \mathcal{H}, \mathcal{I}, \mathcal{G} \rangle$ is an agent behaviour following Definition 6.15,
- for every agent $a \in \mathcal{A}$ it holds that
  - $a = \langle r, R_a, B_a, G_a, O_a \rangle$ generated under $\mathcal{SK}$,
  - agent behaviour $\mathcal{S}_a = \mathcal{S}$,
- d_t is the terminated dialogue between agents $\mathcal{A}$ with mutual goal g_d in context $\mathcal{DK}$,
- $\mathcal{O}$ is the dialogue outcome function given context $\mathcal{DK}$ where $\mathcal{O}(d_t, g_d) = o$ such that o is an arbitrary option $o \in Q_j$.

For the purpose of the experiments in this chapter, the confined deliberation protocol was used. As the minimal protocol, it regulates turn taking, forbids repeating moves and forces agents to explain propose moves through arguments concluding the mutual goal. However, it also restricts agents to play relevant moves, allows only one propose move in a turn and disallows replies to ones own moves.

The scenarios generated for each experiment run use the model introduced in Chapter 4. Using a specific scenario generation context $\mathcal{SK}$, the agents in the dialogue run are initialised with rules, beliefs, goals and options. Every agent plays the dialogue using the same behaviour $\mathcal{S}$, following Chapter 6, consisting of four concrete implementations of the deliberation decision steps.

It is by varying the concrete functions $\mathcal{B}, \mathcal{H}, \mathcal{I}$ and $\mathcal{G}$ in the strategy that a comparison between agent behaviours is possible. Finally, the four metrics of Chapter 7 are applied at the end of an experiment run to test the performance, specifically to terminated dialogue d_t with dialogue outcome $\mathcal{O}(d_t, g_d)$. Which metric is interesting is dependent on the used agent behaviours and the goal of the specific experiment.

8.1.2 Scenario generation configuration

The scenarios generated under context $\mathcal{SK}$, which result in a set of agents $\mathcal{A}$ with assigned rules, beliefs, goals and options, are generated using a concrete configuration of input parameters. In Chapter 5 an experiment was performed to find the ideal configuration of input parameters on the basis of four distinct metrics for scenario interestingness. As the four metrics cannot be combined directly, a choice has to be made which ideal configuration is used for the experimental platform. Although all four of the introduced metrics provide desirable properties on the scenario generation process, the option defensibility metrics are of most importance. Recall that this metric tests to what extent agents can couple the assigned options to their personal goals. This is important in deliberation experiments since we want to compare self-interested agents. For example, the simple relevant agent bases the attitude assignment to an option on the goals to which it can connect the option.

Table 8.1 shows the ideal setting for each input parameter of the scenario generation model as was originally listed in Table 5.3. The ideal setting promises to give the highest number of defensible options for every agent, based on the prediction from the underlying multiple linear regression model.

Unfortunately, the parameter configuration is in practice not able to generate viable scenarios. The ideal setting would theoretically produce the highest possible option defensibility, but when scenarios were actually generated with the software implementation, a problem arises. Because of the assignment of (relatively) many rules, through high values of $n_{B_a^r}$ and $n_{B_a^{\bar{r}}}$, the resulting scenarios almost always contain rule loops. As explained in Section 5.3.3, rule loops are not supported in the ASPIC Java Components library as the software would then run infinitely. The experimentation platform can, in such a case, generate a new scenario, but with the ideal parameter configuration the problem will persist. For this reason the values for $n_{B_a^r}$ and $n_{B_a^{\bar{r}}}$ are lowered until a workable situation occurs. As it happens, the size of the seed set of options

Table 8.1: Statistically ideal (for $j_{\mathcal{A}}^{g_d}$) and experimentally used input parameter configurations

	Statistically ideal	Used in experiments
Conflicts	Chained	Chained
$n_{\mathcal{A}}$	7	7
$n_{\mathcal{R}}$	8	8
$n_{B_{S\mathcal{K}}}$	40	40
$n_{O_{S\mathcal{K}}}$	34	17
$n_{G_{S\mathcal{K}}}$	12	12
n_{O_r}	8	8
n_{G_r}	5	5
l	2	2
$n_{G_a^{\bar{r}}}$	2	2
$n_{B_a^r}$	70	60
$n_{B_a^{\bar{r}}}$	45	35

$n_{O_{S\mathcal{K}}}$ has to be adjusted accordingly to prevent the now more scarce rules assigned to agents to be unrelated to each other, dropping the expected option defensibility again. The final input parameter configuration that was used for all experiments in this chapter is shown in Table 8.1.

8.1.3 Implementation details

Every model in the experimental platform was implemented as part of a Java application, which uses the ASPIC Java Components of South and Vreeswijk (2009) to construct and evaluate arguments. The dialogue model introduced in Chapter 3 forms the base of the platform that starts and terminates dialogues in some context $\mathcal{DK}$ and allows agents to play moves. Several noteworthy decisions were made during the implementation of the experimentation platform.

First of all, the agent behaviours that were experimented with in this chapter all rely on goals to have a single utility value. As presented in Definition 6.7, the agent has a numeric value for each goal indicating the utility. This is used to establish option utility and subsequently in the assignment of option attitudes. However, the scenario generation model, which assigns goals to agents, does not yet assign a utility to goals.

For each experiment run the agents will be assigned a utility for each of their goals. One important aspect is that agents are all allocated the same summed amount of utility. This is vital since the utilities between different agents need to be comparable. For instance, the combined utility metric for deliberation dialogues sums the utilities that different agents have for an option. If one agent had much higher utilities, then this agent would have a stronger influence on the metric than others, which is not desirable within the experiments of this thesis. Instead, in the experiments for this chapter a simple assignment of utilities is used, which ensures a constant amount of utility per agents and every goal for an agent to still have a different valuation.

Definition 8.2 Let $\mathcal{X} = \langle \mathcal{DK}, \mathcal{SK}, \mathcal{A}, \mathcal{S}_a, d_t, \mathcal{O} \rangle$ be an experiment run. Each agent $a \in \mathcal{A}$ with set of goals $G_a = \{g_0, \ldots, g_i, \ldots, g_n\}$ is assigned a sequence of goals $\vec{G_a} = \langle g_0, \ldots, g_i, \ldots, g_n \rangle$, where for each goal $g \in G_a$, $g \in \vec{G_a}$ and $i \in \mathbb{N}$ is the sequence index. Goal $g_i \in \vec{G_a}$ has a goal utility $U_a^{g_i} = i$.

Example 8.1 Simply put, every goal has a utility that is the value of its index in the set (turned into a sequence) of goals. For instance, if an agent a_1 was assigned a set of $G_{a_1} = \{g_2, g_4, g_5, g_d\}$ and thus the sequence $\vec{G_{a_1}} = \{g_2, g_4, g_5, g_d\}$, then it has goal utilities $U_{a_1}^{g_2} = 1$, $U_{a_1}^{g_4} = 2$, $U_{a_1}^{g_5} = 3$ and finally $U_{a_1}^{g_d} = 4$. The total utility assigned to the agent is $\sum_{g \in G_a} U_a^g = 10$. The scenario generation model ensures a fixed size $n_{G_r} + n_{G_a^r}$ for the set of goals for every agent and therefore the total assigned utility to each agent is equal.

One can think of situations where a different utility allocation is desirable. Perhaps certain agents have a more authoritative role and should be assigned more utility accordingly. Another alternative is when the utility for a combination of goals G, defined by $\mathcal{U}_a(G)$, cannot simply be determined by summing their individual utilities. In that case the option utility function needs to be adjusted from the simple summing specified in Definition 6.16. Such situations fall outside the scope of this thesis.

Finally, there are some points to be made concerning the application of the dialogue metrics. The combined utility and Pareto optimality metrics operate on a dialogue outcome, for which there needs to be at least one proposal that was *in*. This is because the dialogue outcome function only selects an outcome from the proposed options whose propose move was *in*. To make sure that the metrics return a value even without dialogue outcome, any dialogue d without

a dialogue outcome is considered to have a zero dialogue utility $\mathsf{utility}(d) = 0$ and is considered to not have a Pareto optimal outcome.

8.1.4 Method

Experiments were performed using the implemented experimentation platform to compare the performance of individual agent behaviours. Each experiment consisted of two distinct behaviours, for which 1000 experiment runs were performed per behaviour, to make a total of $N = 2000$ experiment runs. Running time of an experiment depended heavily on the two agent behaviours used but roughly varied between 10 to 90 minutes.

The two distinct behaviours in an experiment can be compared directly as all other elements of the experimental platform are kept constant. Two specific behaviours are chosen each time to test a specific hypothesis. The scope and configurability of the experimentation platform offer a wide variety of experiments to perform. The experiments in this chapter focus on the differences between arguing and non-arguing agents and on the effect of the individual decision steps in an arguing agent, in order to answer the general research question posed at the start of this thesis whether it is beneficial for the multi-agent system to allow argumentation in deliberation situations.

Table 8.2 lists the six behaviours that were experimented with. For each experiment the simple rational agent was compared to a single other behaviour, with only one of the four decision step functions being different. This makes for a total of five experiments for which the differences in metrics directly reflect the different decision step implementation.

Table 8.2: Decision functions used in each agent behaviour $\mathcal{S} = \langle \mathcal{B}, \mathcal{H}, \mathcal{I}, \mathcal{G} \rangle$ (with original definition numbers in parenthesis)

$\mathcal{S}$	$\mathcal{B}$	$\mathcal{H}$	$\mathcal{I}$	$\mathcal{G}$
simple rational agent (6.16)	simple belief revision (6.5)	spread attitude (6.10)	relevant (6.11)	basic thorough attacker (6.12)
naive revision agent	naive revision (6.6)	spread attitude	relevant	basic thorough attacker
considerate revision agent	considerate revision (6.6)	spread attitude	relevant	basic thorough attacker
focused attitude agent	simple belief revision	focused attitude (6.10)	relevant	basic thorough attacker
simple non-arguing agent	simple belief revision	spread attitude	relevant	propose and reject (6.13)
chatty non-arguing agent	simple belief revision	spread attitude	relevant	propose and inform (6.14)

The data for each experiment will be visualized through box plots of the two agent behaviours, which show, the median, lower and upper quartiles and the smallest and largest included samples, plus any outliers. The box is drawn from the lower to the higher quantile, showing the locations of the 25th and 75th percentile of the data distribution. The medium, drawn inside the box as bold line, represents the middle value in the distribution of all samples. The total spread of included data samples is indicated using the whiskers. Any outliers are plotted separately. The mean is also plotted in the graphs, using the ◆ sign, showing the average value over all samples.

8.2 Results

Four hypotheses on the effects of argumentation in deliberating agents will now be discussed, based on the experiments that were performed with the experimental platform.

8.2.1 Arguing versus non-arguing agents

The most prominent experiment compared the simple rational and simple non-arguing agents. As explained above, the non-arguing agents are effectively not asking for and providing arguments in the deliberation dialogue. Instead, they have to rely on proposing options and rejecting those. Attitude assignment is still argumentation-based, just as with the simple arguing agent, and knowledge revision only adopts all proposed options in both agents.

As explained in the beginning of this thesis, argumentation-based dialogue models often claim to promote efficiency benefits. This leads to the following hypothesis on argumentation efficiency in deliberation dialogues.

Hypothesis 1 A deliberation process is more efficient if agents are allowed to argue, requiring less communication to come to a decision.

Figure 8.1 shows the communication efficiency, defined as the move count, and the topic efficiency, defined as the degree of information concealment, for the two simple deliberating agents. Clearly, the arguing agents typically played more moves, indicated by the movecount, than the agents that do not argue in the dialogues. This result is explained by the fact that arguing agents can, and indeed do, ask for and provide reasons for questing proposals. Disputes are solved through the playing of arguments, while the non-arguing agents only

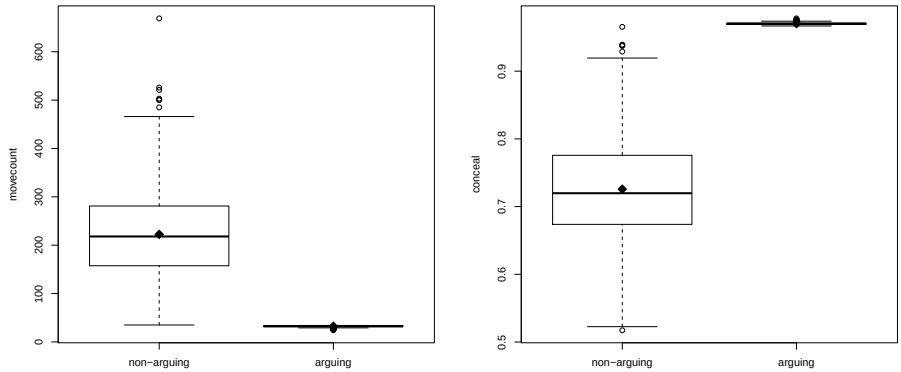


Figure 8.1: movecount and conceal for simple rational arguing and simple non-arguing agents

propose options and reject them. Similarly, the arguing agents also revealed much more of their knowledge, shown by the lower values for relevance. All that the non-arguing agents reveal are their options.

In Chapter 6 it was argued that non-arguing agents might still want to share knowledge with the other agents, but by using inform statements rather than arguments relevant to some proposal. This was modelled through the propose and inform heuristic, where agents will inform others of the facts in their knowledge. A separate experiment was performed that compares the simple non-arguing agent to an agent that also plays inform moves.

Figure 8.2 shows that the communication effectiveness is obviously affected, as the non-arguing agent will now also play inform moves. Degree of information concealment is similarly affected, as agents will now not only play relevant propose and why-propose moves, but also inform statements, which reveal personal beliefs to the other agents. Still, the number of moves as well as the concealment of information is still better with the chatty non-arguing agent than with arguing agents, as pictured in Figure 8.3. Concluding the discussion on efficiency, the simple arguing agents use over 200 moves in a dialogue

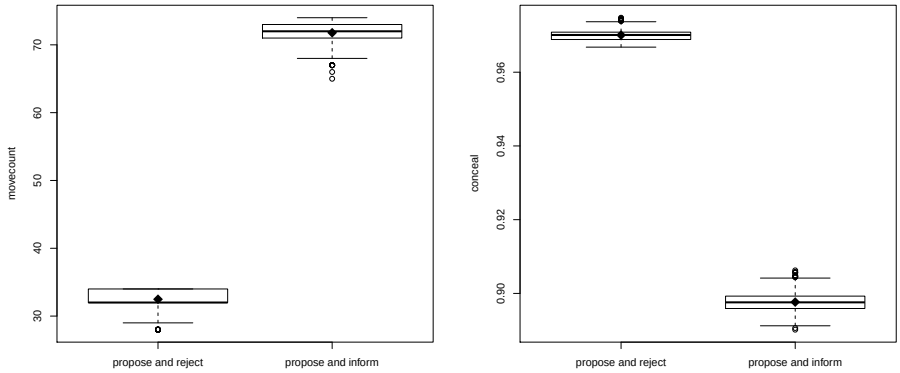


Figure 8.2: movecount and conceal for simple non-arguing and chatty non-arguing agents

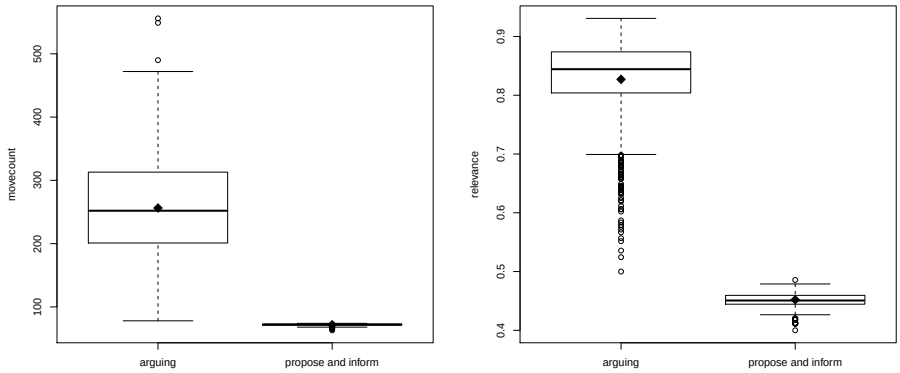


Figure 8.3: movecount and relevance for arguing and chatty non-arguing agents

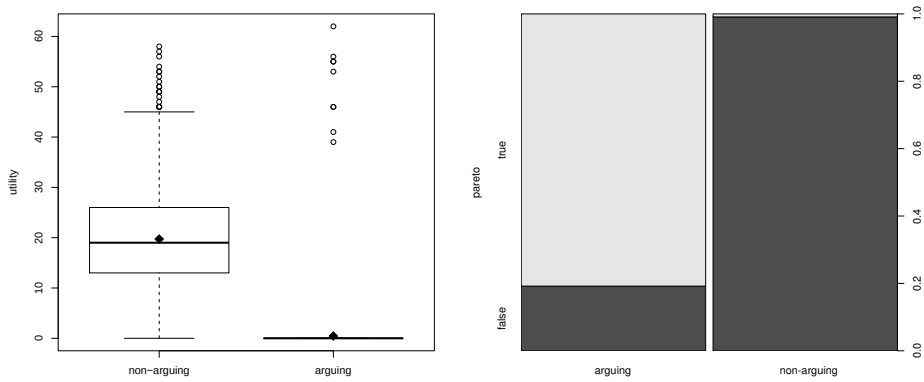


Figure 8.4: utility and Pareto optimality for simple rational arguing and simple non-arguing agents

on average, the chatty non-arging agents, who play informs, use over 70, while the non-arging agents that only propose and reject need just over 30 moves on average. Hypothesis 1 should therefore be rejected.

The other important aspect to compare arguing and non-arguing agents on is effectiveness. Again, models for argumentation-based dialogues as proposed in the literature are justified through the potential effectiveness benefits. The experiment between arguing and non-arguing agents can show how effectiveness is influenced by the introduction of argumentation.

Hypothesis 2 A deliberation process is more effective if agents are allowed to argue, leading to an outcome with higher utility for the multi-agent system.

Figure 8.4 shows the topic effectiveness, measured as combined utility for the agents utility, for arguing versus non-arguing agents. The arguing agents, with an average utility of around 20, outperformed the non-arguing agents, with an average utility around 1.

The explanation for this is found in the lack of agreement in non-arguing agents. In a typical dialogue between non-arguing agents there is much disagreement, as agents have different utilities for options and it is therefore common

that no single option is interesting for all agents. In such a situation the proposal for the option will be directly attacked. Since attacked proposals (without arguments to back up) will always be *out*, they are never selected as dialogue outcome. In short, it is very common that a dialogue between merely non-arguing agents will have no outcome, as there is often at least one agent that attacks the option. A dialogue without an outcome has a combined utility of 0, explaining the many zero-utility dialogues and the accordingly low topic effectiveness average.

It is good to note that the arguing agents also typically have options that are immediately refuted by another agent. However, in contrast to the non-arguing agents, the simple rational agents can explain why it is still a viable proposal, by showing how it supports the mutual goal. Indeed in a typical dialogue between arguing agents proposals are both attacked and defended by various arguments.

A dialogue that has no outcome also has no Pareto optimal outcome. It is therefore expected that the simple non-arguing agents, who commonly can not settle on a dialogue outcome, also have a very low percentage of cases with Pareto optimal outcomes. The simple rational agents, on the other hand, typically select an outcome that is Pareto optimal in about 80% of the cases.

Finally, effectiveness was measured in terms of how relevant the agents were in their moves. As seen, the simple arguing, but also the simple non-arguing agents, play only relevant moves. Hence, this direct comparison does not show a clear benefit. However, the case is different for the chatty non-arguing agents, who expose beliefs in the dialogue through inform statements. inform moves can not directly be related to the ongoing discussion of some claim and the relevance metric therefore considers such moves to degrade the communication effectiveness of a dialogue. This is shown in Figure 8.3, where the arguing agents on average conceal about 85% of their information, while the chatty non-arguing agents conceal only about 45% of their beliefs and options. Concluding, Hypothesis 2 can be accepted, as arguing agents perform better on the effectiveness metrics than the non-arguing agents.

8.2.2 Baseline performance

As seen, the simple rational agents can argue which proposal should be selected, which results in a combined utility of around 20. This raises the question whether this is high or low. Or rather, how does this level compare to the situation where a random proposal is picked?

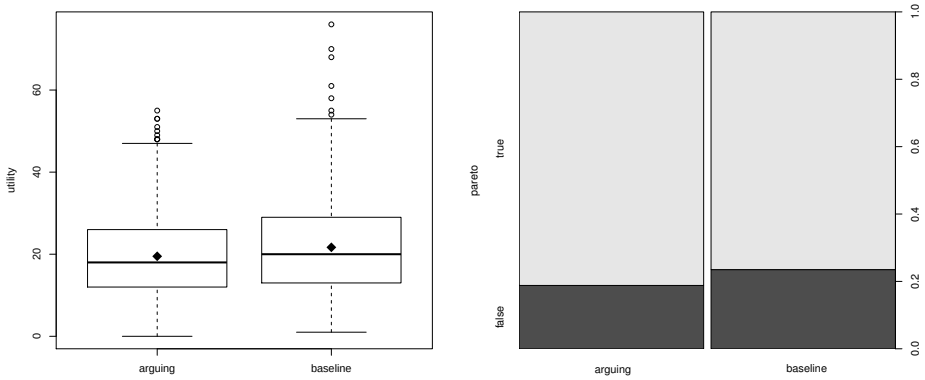


Figure 8.5: utility and Pareto optimality for simple rational agents versus baseline performance

To test this, the simple rational agent was compared to a baseline performance. The baseline is set by selecting a random proposed option, rather than one that is *in*, but still not considering options that were not proposed in the dialogue. This allows to compare the effect that making argue moves has on the average combined utility of the agents.

Figure 8.5 shows the topic effectiveness, measured by the combined utility utility, and the percentage of Pareto optimal outcomes for the simple rational agents and the baseline. As it happens, there is no statistical difference between agents that argue about the propose options and agents that do not argue at all in the dialogue. Both the average combined utility and the Pareto optimality metrics do not show that arguing in a deliberation dialogue results in a better performance.

This somewhat discouraging result can be explained by the fact that the combined utilities that agents have for the various options are usually not very dispersed. Although individual agents typically have a strong preference, the utility for all agents combined for some option is usually very comparable to that of other options. As a result, even picking a proposed option at random

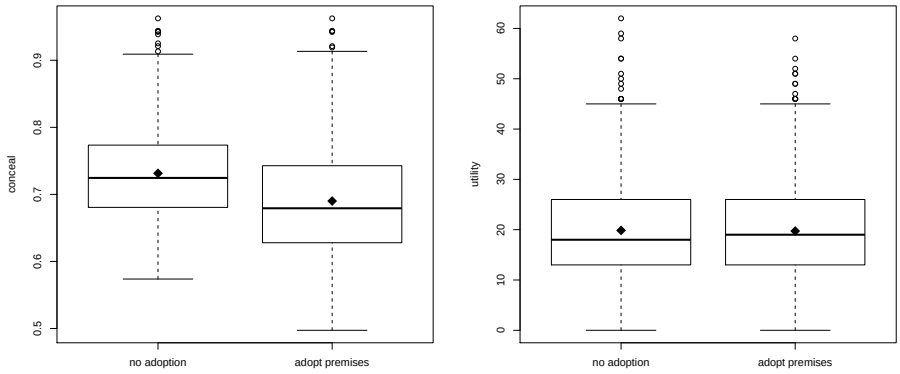


Figure 8.6: relevance and utility for simple rational and naive revision agents

will typically yield a good utility. It is expected that a different, more skewed utility distribution will cause the arguing agents to start performing better than the baseline.

8.2.3 Knowledge revision

So far the arguing agent behaviour that was used did not use any sort of knowledge revision beyond adding proposed options to its knowledge. However, it was suggested that the adoption of exposed knowledge in the dialogue can directly influence the topic efficiency and effectiveness. The underlying intuition is that agents can use the earlier exposed knowledge to prevent having to expose their still concealed beliefs.

Hypothesis 3 Adopting knowledge of others allows an agent to conceal information while combined utility is not negatively influenced.

An experiment was performed that compares the simple rational agent with one that adopts any premise that other agents use. The earlier presented naive revision heuristic of Definition 6.6 is used by this behaviour to have the agents adopt any premise that was used in an argument in the dialogue.

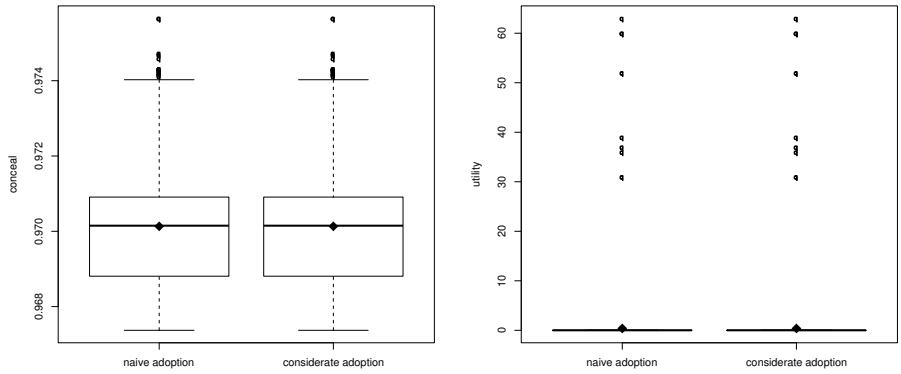


Figure 8.7: relevance and utility for naive and considerate revision agents

Figure 8.6 shows the effect of adopting beliefs in a naive matter on the effectiveness of communication and topic layers. Contrary to the simple arguing agent, the naive revision agent was not able to conceal more of its information. Actually, it exposed more knowledge in the dialogue. Hypthesis 3 should therefore be rejected, even though, as hypothesized, the average combined utility was not affected by the naive way of adopting beliefs.

The reason for the naive revision agent actually exposing move knowledge is found in how the adopted knowledge is used. What happens is that the naive agents adopt premises, which allows them to construct new arguments. As agents will play any relevant argue move they know about, the agent will now play the new arguments, based on the learned premises, which often expose defeasible rules that were not exposed earlier in the dialogue. Consequently, the adoption of premises by other agents results in a higher number of defeasible rules that are exposed, by which the concealment metric goes down.

An alternative was proposed to the naive way of adopting beliefs. The original idea was that agents would not accept beliefs, and would therefore not reason with these beliefs, which the agent could construct counterarguments for. A new agent, called the considerate revision agent, was used to compare to

the naive approach.

Figure 8.7 shows the results for the comparison between the naive and considerate way of adopting beliefs. As it happens, the degree of concealment nor the combined utility were different. Being considerate, only adopting beliefs to which there is no counterargument, does not influence how much information agents expose or the utility of a dialogue. This is a result on how many beliefs are actually adopted. As it turns out, almost all beliefs that are exposed will be adopted by the considerate agent, as it will rarely have a counterargument.

8.2.4 Self-interestedness

The final experiment performed in the interest of this chapter was a comparison between two alternative ways of assigning attitudes to options. The simple relevant agent builds any option that it has a positive utility for, based on the goals that it expects to achieve. An alternative was proposed that is more focused, only assigning a build attitude to the option with the highest utility and destroy to any other option. This is more self-interested than the spread attitude assignment of the simple arguing agent, as it pursues solely its most promising alternative. The intuition is that this approach will cause the discussion to be more focused, with fewer proposals being discussed and therefore a lower number of dialogue moves. As a consequence it is also expected that agents can withhold more of their information.

Hypothesis 4 Dialogue efficiency, for topic as well as communication, will increase if agents are more self-interested.

Figure 8.8 shows the effect of a more self-interested attitude assignment on the dialogue efficiency, that is, the number of moves the agents used and the information concealment. The agents that only assigned a build attitude to the option with the most utility needed less than half the number of moves on average to complete a dialogue. Moreover, significantly less beliefs are exposed by the agents.

Looking at the experimental data, the focussed attitude agents will propose much less options, as they only have a build attitude for the option with maximum utility. The dialogues therefore focus on less options, resulting in shorter dialogues with less exposed beliefs and rules. Concluding, Hypothesis 4 can be accepted, as efficiency is increased when agents are more self-interested.

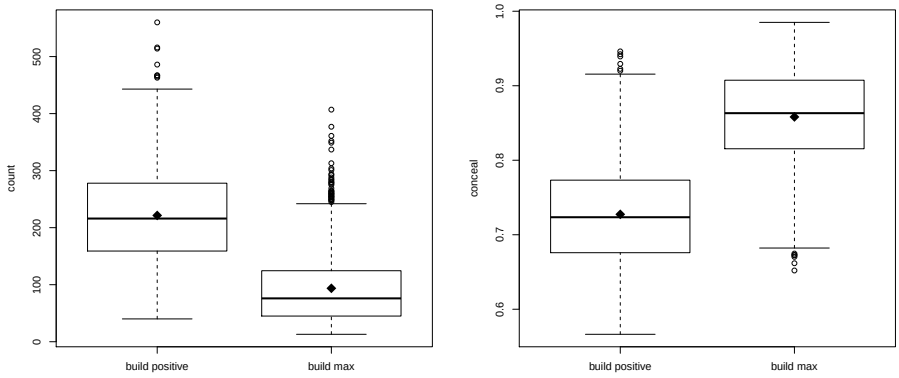


Figure 8.8: movecount and conceal for simple arguing and focussed attitude agents

Karunatillake et al. (2009) have also compared agents that are more or less self-interested in argumentation-style dialogues. One of their findings was that the withholding of information, like the focussed attitude agents of this chapter, resulted in a lower effectiveness as measured by the combined utility. To see if this result can be reproduced, the topic effectiveness of the focussed attitude agents should be lower than that of the simple arguing agents.

Hypothesis 5 Topic effectiveness, measured as combined utility, will decrease if agents are more self-interested.

Figure 8.9 shows the topic effectiveness and the Pareto optimality. Contrary to the findings of Karunatillake et al. (2009), restricting the sharing of options did not significantly lower the combined utility. However, the outcome is less likely to be Pareto optimal, which means that, as expected, the effectiveness is negatively influenced by focussed, more restricted proposing of options. Hence, as Pareto optimality is concerned, Hypothesis 5 can be accepted.

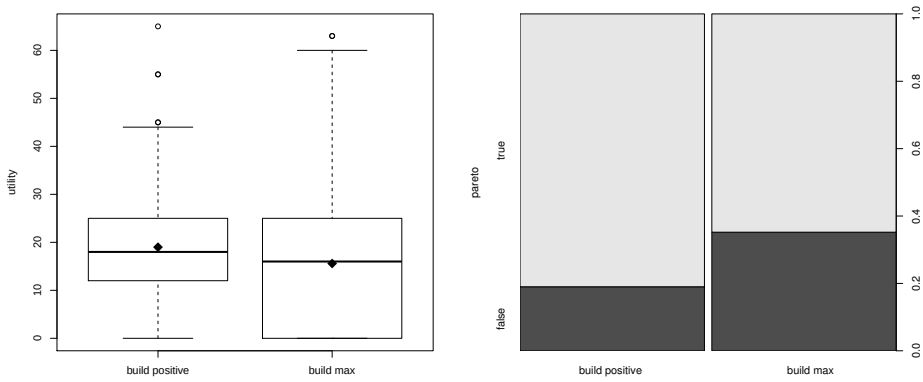


Figure 8.9: utility and Pareto optimality for simple arguing and focussed attitude agents

8.3 Discussion

At the start of this thesis a research question was posed to discover the benefits for a multi-agent system to use argumentation in deliberation dialogues. The experiments of this chapter answer this question by the comparison of arguing and non-arguing agents. Each of the desirable properties for deliberation dialogues have been tested to validate if these benefits arose in the experimental setup.

8.3.1 Argumentation benefits on desirable deliberation properties

Employing arguments in a deliberation dialogue has shown to not increase communication or topic efficiency, as illustrated by the rejection of Hypothesis 1. More specifically, it has the reverse effect. Arguing agents need a lot of moves to settle their debate, while the non-arguing agents can suffice with merely proposing and rejecting options. Correspondingly, the topic efficiency when using argumentation is also lower, as more information is shared in the dialogue.

Non-arguing agents that share personal beliefs through inform statements are less efficient than their simple non-arguing counterparts, but still more efficient than the arguing agents.

While the use of argumentation does not improve dialogue efficiency, communication and topic effectiveness are positively influenced, as Hypothesis 2 was accepted. The deliberation scenarios often cause strong disagreement about the options and the non-arguing agents can do no more than proposing and rejecting those. The arguing agents, on the other hand, have the opportunity to contest option rejection, by showing how their mutual goal will be achieved by an option. For this reason the arguing agents are much more likely to decide on an outcome and therefore achieve a higher shared utility. Moreover, if non-arguing agents expose personal information, it is not directly tight to an option, bringing down communication relevance. Hence, arguing allows agents to be more effective in a deliberation dialogue.

The simple arguing agent as specified in Chapter 6 is really just that: a simple agent capable of playing arguments in a deliberation setting. It is representative of an arguing agent, but many variations of arguing agents can be realized in the agent behaviour framework. Some alternatives have been considered in this chapter and it is shown how they influence the benefits of using arguments in multi-agent deliberation. Admittedly, more sophisticated non-arguing agents might realize a better effectiveness using alternative approaches to settle disputes. However, the same can be said of the arguing agents in this chapter. For example, if in two distinct proposal trees a dispute arises on some claim p, the simple arguing agents will extensively discuss this claim in both proposal trees. In fact, this situation can even occur (repeatedly) within one proposal tree. Smarter arguing agents could recognize this and decide not to continue the discussion any further. Of course, many other improvements can be though of.

8.3.2 Existing experimental work with argumentation

Although scarce, there have been studies in the multi-agent research field that try to experimentally proof the benefits of using argumentation in multi-agent dialogue. Above, the study by Karunatillake et al. (2009) was already cited. They study the performance of agents in a multi-agent society that negotiate on responsibilities and the execution of tasks. Interestingly, their experiments showed that the introduction of an argumentation phase in the dialogues de-

creased communication efficiency but allowed for a higher topic effectiveness, resulting in a higher shared utility. This is exactly the result acquired in this chapter. However, the argumentation phase in the experiments of Karunatillake et al. was implemented as a simple function stating whether the discussion on the option was successful or not. In other words, the argumentation phase in their dialogues did not involve the actual construction and playing of structured arguments. In fact, their agents were abstract, having no knowledge base, reasoning mechanism or move generation step. In conclusion, it is motivating to find very similar results while having a fully developed framework with agents that can use structured argumentation.

Pasquier et al. (2010) have proposed an experimental framework for negotiating agents that use reframing, a process of asking for and providing background information to bring about a better understanding of the reasons for rejecting a proposal. Information is supplied in the form of goals and options that are implicitly connected, without providing (defeasible) inference rules that connect them. For this reason, it is not a structured argument and there is no underlying model in which statements can rebut, undermine or undercut each other. Regardless, agents that ask for and supply underlying goals are roughly similar to the deliberating agents in this chapter. Pasquier et al. show that the agents that use reframing statistically are more effective than the bargaining-only agents, as measured by the cost of the outcome as inverted utility. Hence, their negotiating agents, although without the explicit adoption of an argumentation logic, show comparable results as the arguing agents of this chapter.

Finally, Pajares Ferrando and Onaindia (2012) have designed and implemented a framework for the evaluation of argumentation-based multi-agent planning. In a dialogue-like interaction the agents restructure a shared plan until every agent agrees on it. The plan is revised by contesting a step in a plan through the moving of an attacking argument. In a classical setting, the plan would then be changed and in turn the other agents can rework it. On the other hand, with an argumentation-based approach the supplied argument can in turn be attacked with arguments as so to use the non-monotonic aspect of argumentation to mutually find agreement. An experimental validation was performed on a complex yet typical multi-agent planning problem. The results show that argumentation requires additional interactions and therefore are less efficient. However, the resulting plans achieve a higher effectiveness, as the agents can prevent failed plans and the resulting plans have a higher team satisfaction, measured by the individual agents' preferences. These re-

sults are directly in line with the experiments in this chapter, albeit in the more restrictive multi-agent context of a planning problem.

8.3.3 Trade-offs of using argumentation

Combining the just discussed previous work on experimental validation of the uses of argumentation with the deliberation experiments in this chapter, there seems to be a strong trade-off between efficiency and effectiveness whether argumentation is beneficial or not. In other words, it is strongly dependent on the specific system in which agents are implemented, whether it is a good idea to allow for argumentation between agents. When efficiency is most important in a system, argumentation may be too costly. For example, when communicative acts are expensive in terms of money or time or when the information is very sensitive and should be shared as little as possible. When effectiveness is most important in a system, argumentation can help improve the performance of the multi-agent system as a whole.

The trade-off between effectiveness and efficiency does not mean that only the one or the other can be designed for. Instead, the proposed framework in this thesis allows for future creation of agents that can smartly determine when to use argumentation and when not to argue. Both the arguing and arguing agents of this thesis are still very simple in many respects. However, how agents can reliably make the decision to use argumentation and if there are perhaps benefits for individual agents to (not) argue needs to be studied further. Non-arguing agent can perhaps draw upon work on negotiation, while the arguing agent may likely benefit from a more sophisticated way of selecting a proposal as outcome based on their explicitly expressed preferences. Fortunately, the method of experimentation and the deliberation platform of this thesis provide the tools to design such agents as well as experimentally validate their performance.

Chapter 9

Argumentation testbed

As often with research, the answering of one question results in the asking of many others. The formal framework developed in this thesis, and the experimental methodology to go with it, supplies a solid basis for the answering of many new questions on argumentation in multi-agent dialogues. This chapter will show how the experimental platform of this thesis is suitable as general testbed for studying argumentation in deliberation dialogues.

9.1 From platform to testbed

Already during the construction of the formal framework of this thesis, it became clear that there is an enormous variety of deliberation protocols, scenarios and arguing and non-arguing agents. With this in mind, it became clear that the framework of this thesis, implemented as the experimental platform, could serve as a starting point for further experimental work on argumentation in dialogues. After all, many strong assumptions and restrictions have been made in the design of the platform and agent behaviours, each of which are opportunities for additional experimentation.

Yuan et al. (2008) have proposed to construct a testbed for arguing agents to spur the interest in argumentation for multi-agent systems. A testbed, which can work as an Arguing Agents Competition, can increase interest in argumentation-based dialogue frameworks, as well as arguing agent strategies. Yuan et al. propose six requirements for an argumentation testbed. A testbed should:

1. provide an interesting and fair scenario to play,

2. enforce legal moves according to some protocol,
3. allow agents to join into a dialogue,
4. control turn taking,
5. allow for reviewing of played dialogues,
6. allow running of repeated experiments.

It will now be explained how these requirements are satisfied by the experimentation platform as implemented for this thesis.

9.1.1 Asynchronous platform

During the implementation of the experimentation platform, special attention has been given to the way experiments are run, dialogue communication is organized and how agents are executed. Importantly, agents are considered to be autonomous entities, which can decide by themselves which dialogue to join and which have full control over their own behaviour. Hence, the platform and each agent all need to run separately, or more specifically, within their own computational thread. A dedicated dialogue control thread is used to handle the joining of agents as well as the passing of messages, including moves that agents make. It is also responsible for turn taking and determining the dialogue outcome.

To support the repeated execution of experiment runs, allowing agents to participate in many subsequent dialogues, a dedicated experimentation thread is used that initializes and starts individual experiment runs. It gathers dialogue statistics to measure the performance of the multi-agent system in terms of each of the four deliberation metrics of Chapter 7. In short, the experiment, the dialogue control and each of the agents are separate entities that run asynchronously and are aligned through message broadcasts. Figure 9.1 shows the sequence diagram of how experiments are executed in the testbed implementation.

9.1.2 Experiment run output and analysis

To run repeated argumentation experiments, the Java software implementation of the experimentation platform offers a command line interface called baidd-exp, where *baidd* stands for BDI Agents Interacting in Deliberation Dialogues.

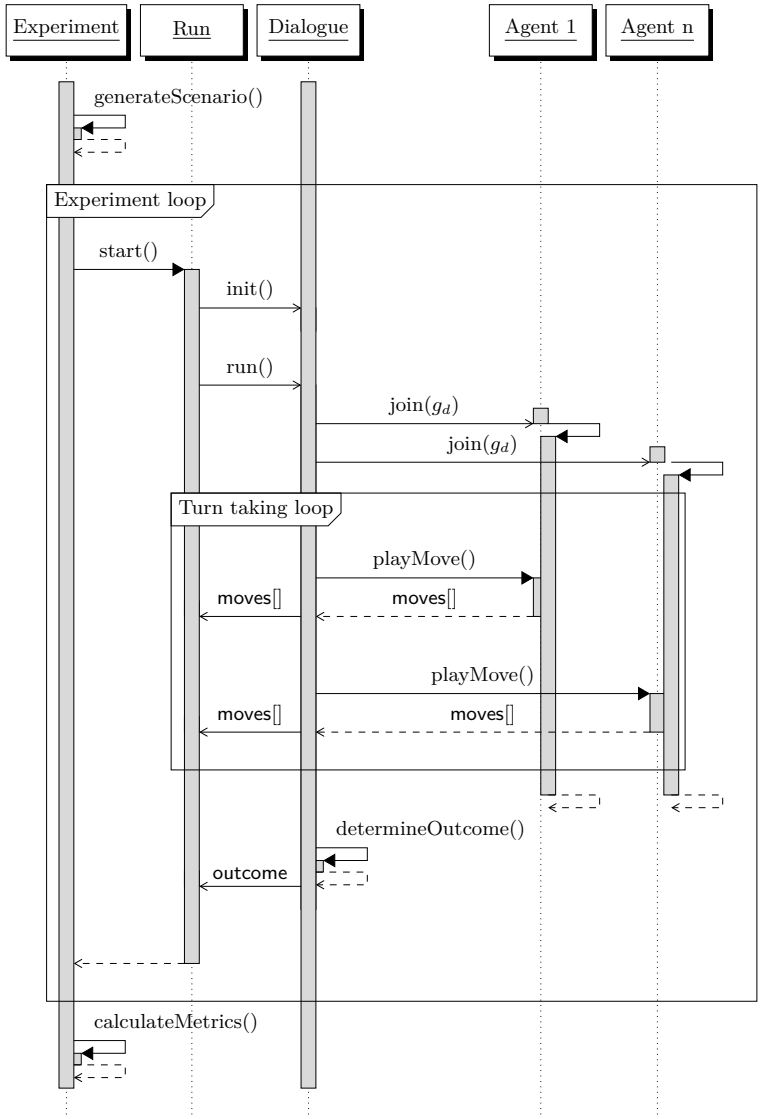


Figure 9.1: Experiment execution in the asynchronous platform and agents

Table 9.1: Options to run experiments with the baidd-exp program

Option	Description
-?, -h, --help	Print usage message
-f, --file <File>	File to write output to
-l, --level <Level>	Set message console print level
-o, --output <Writer>	Set experiment results output (Console, Csv)
-p, --property <Property>	Experiment with a BDI strategy property
-r, --runs <Integer>	Number of dialogues to run
-s, --strategy <LocalAgent>	Set agent class (BDIAgent, NonArguingAgent)
-v, --version	Version info
-y, --history <File>	Directory to store generated scenarios
-D, <DeliberationRule>	Use a specific deliberation rule
-O, <OutcomeSelectionRule>	Use a specific outcome selection rule
-T, <TerminationRule>	Use a specific termination rule

Configuring the dialogue protocol and the agents, as well as the experimental parameters, is performed by simply providing the right set of options to the program. The available options to the baidd-exp program are listed in Table 9.1.

For example, to start 100 experiment runs with agents of the BDIAgent class, while testing the AdoptBeliefs property of that agent and writing the results to a local file, it suffices to execute:

./baidd-exp -s BDIAgent -r 100 -p AdoptBeliefs -o Csv -f ./results.csv

This will result in a local result.csv data file that contains, for all experiment runs, the experiment run configuration and the performance in terms of the four deliberation metrics. The data is formatted for easy loading into data analysis software such as SPSS or R. Optionally, the full history of generated scenarios can be written to local XML files. These XML files contain the specification for each experiment run, including the agent knowledge and dialogue protocol configuration, and can be used to replay the specific experiment run. Specifically, they can be replayed with the exp-viewer tool.

Example 9.1 The baidd-exp application generates one XML file for every participating agent, containing the knowledge that was assigned and optionally any experimentation properties that the agent is requested to use. Listing 9.1 shows an example XML file for an agent that is assigned 3 options, 4 goals and 15

Listing 9.1: Example agent knowledge XML file

```
<?xml version="1.0" encoding="UTF-8"?>
<baidd-agent>
   <beliefbase><![CDATA[
      [r65] g_d <- p_3.
      [r11] g_d <- p_1.
      [r29] g_d <- p_18.
      [r24] ~r5 <- p_6.
      [r91] ~p_17 <- p_18.
      [r17] p_14.
      [r32] p_16 <- do(o_2).
      [r28] p_18 <- do(o_2).
      [r92] p_7.
      [r39] p_7 <- do(o_4).
      [r62] p_8 <- do(o_5).
      [r41] p_1 <- do(o_4).
      [r95] g_4 <- p_4.
      [r67] g_4 <- p_9.
      [r12] p_6 <- p_15.]]>
   </beliefbase>
   <options>
      <option>do(o_4)</option>
      <option>do(o_5)</option>
      <option>do(o_2)</option>
   </options>
   <goals>
      <goal>g_2. 1</goal>
      <goal>g_d. 2</goal>
      <goal>g_d. 3</goal>
      <goal>g_4. 4</goal>
   </goals>
   <properties>
      <property name="AdoptBeliefs" type="boolean">true</property>
   </properties>
</baidd-agent>
```

beliefs. The multi-agent system configuration is written to another XML file, shown in Listing 9.2, and specifies the mutual goal for the agents (g_d), the dialogue protocol rules and the agents to load, including the agent behaviour Java class file to use (BDIAgent).

The baidd-viewer program provides a GUI to load and (re)play stored deliberation scenarios from XML files. The dialogue settings, such as protocol rules, loaded agents, played moves and proposal trees are visualized. Dialogues can be played outright or step by step, pausing the dialogue after every dialogue turn. Figure 9.2 shows a screen shot of the application with an ongoing dialogue between two agents. After joining the dialogue, the agent proposed two options

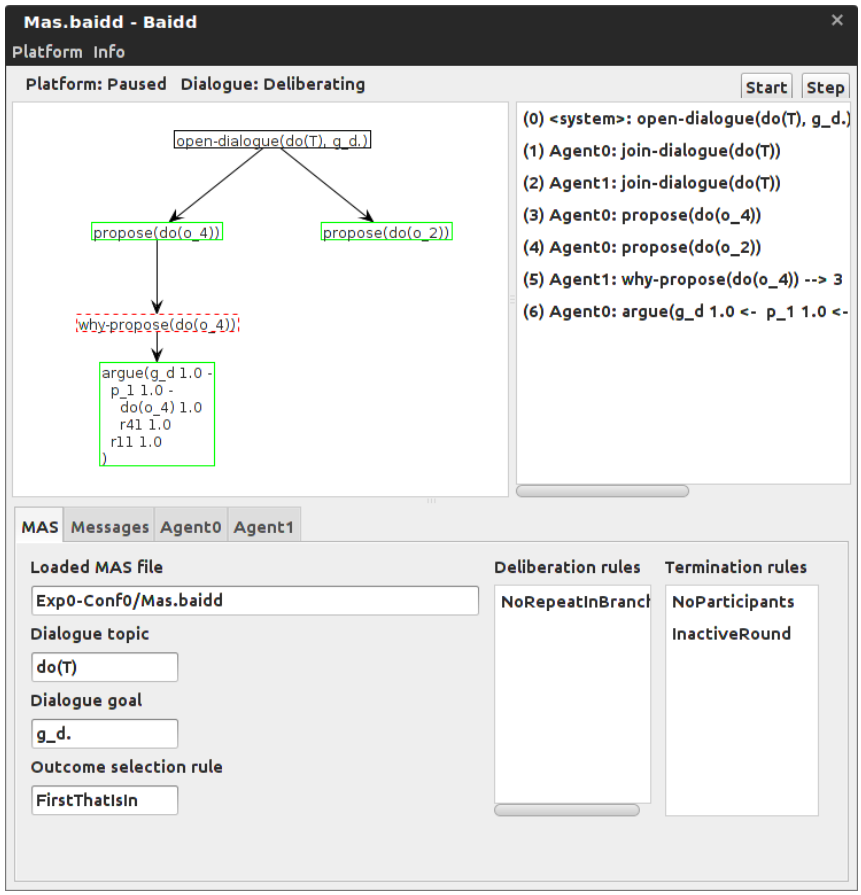


Figure 9.2: Screen shot of the baidd-viewer program

Listing 9.2: Example multi-agent system XML file

```
<?xml version="1.0" encoding="UTF-8"?>
<baidd-mas>
   <topic goal="g_d.">do(T)</topic>
   <protocol>
      <deliberation-rules>
         <rule>NoRepeatInBranch</rule>
      </deliberation-rules>
      <termination-rules>
         <rule>NoParticipants</rule>
         <rule>InactiveRound</rule>
      </termination-rules>
   </protocol>
   <agents>
      <agent file="Agent0.agent" name="Agent0" type="BDIAgent" />
      <agent file="Agent1.agent" name="Agent1" type="BDIAgent" />
   </agents>
</baidd-mas>
```

already and submitted one argument supporting a proposal. The proposal trees are pictured to easily see how the status of moves in the dialogue is affected. Moves that are *in* are drawn with a green, solid outline, while moves that are *out* are drawn with a red, dashed outline.

9.1.3 Scalable scenarios

Each argumentation experiment requires a different type of scenario. Perhaps the designer of the experiment is interested in dialogues between many agents or with a high amount of incomplete knowledge. On top of this, the designer might define scenario interestingness in different ways, as explained and formalized in Chapter 5. The testbed supports supports these designer requirements through the input parameter configuration of the scenario generation method, such as the number of beliefs, goals and options, the length of argument chains or the number of roles in a dialogue.

To ensure a certain type of scenario for some testbed experiment, it is possible to fix input parameters. The ideal setting for parameters that are not fixed can be determined by applying the scenario generation method experiment of Chapter 5. Although the software does not fully automate this process of finding the ideal setting yet, rather requiring the manual running of a Java program and R script, it could easily be integrated and included as baidd-exp command line parameter. This way, the type of scenario can be controlled by the designer

Listing 9.3: DeliberationRule interface and an example implementation

```
/**
 * Evaluates whether a submitted move in the dialogue is allowed
 * @param dialogue The current state of the dialogue
 * @param move The new move that was submitted
 * @return The reason why this move was fallacious; or null if it wasn't
 */
public abstract ProtocolException evaluateMove(Dialogue dialogue, Move<?
    extends Locution> newMove);

@Override
public ProtocolException evaluateMove(Dialogue dialogue, Move<? extends
    Locution> newMove) {
   if (newMove != null && newMove.getTarget() != null &&
       newMove.getTarget().getPlayer() == newMove.getPlayer()) {
      return new ProtocolException(newMove, "The new move attacks one's own:
          An agent may only attack moves of other players");
   }
   return null;
}
```

of the experiment, while maintaining fair scenarios with maximum potential for deliberation.

9.1.4 Protocol and outcome rules

As shown above, the baidd-exp program allows for the configuration of dialogue protocol and outcome rules. The testbed software's dialogue control thread checks every move submitted by the agents for validity. For example, agents can be disallowed to reply to their own moves or play non-relevant moves, by supplying the appropriate deliberation rules as options to the baidd-exp program. Moreover, a single outcome rule can be supplied to the software, which specifies the dialogue outcome selection function to use.

The designer of a deliberation experiment can supply new protocol and dialogue outcome rules. These are implemented as single Java functions. There are two types of protocol rules: rules that can prohibit moves and rules that cause the dialogue to terminate. A DeliberationRule should check move legality and return a ProtocolException if a new move violates the specific protocol rule constraint. The testbed takes care of broadcasting this error and will discard the erroneous move. The protocol rule interface and an example implementation are given in Listing 9.3.

A TerminationRule is a protocol rule that checks if the dialogue should be

Listing 9.4: TerminationRule interface and an example implementation

```
/**
 * Evaluates the dialogue on whether it should terminate
 * @param dialogue The current state of the dialogue
 * @param participants The number of participants in the dialogue
 * @param skips The number of subsequent skips (non-moving) by the
     participating agents
 * @return The reason, in text, why the dialogue should terminate; or null
     if it should not
 */
public abstract TerminationMessage shouldTerminate(Dialogue dialogue, int
     participants, int skips);

@Override
public TerminationMessage shouldTerminate(Dialogue dialogue, int
     participants, int skips) {
   if (skips >= participants + 1) {
      return new TerminationMessage("The dialogue was inactive for a full
           round: All of the agents skipped their turn and the first skipped
           twice.");
   }
   return null;
}
```

terminated. In this thesis, the dialogue terminates when every agent skips their turn and the first agent skips twice. A termination rule implementation should return a TerminationMessage if a dialogue should be terminated, after which the testbed will broadcast this message, stop accepting new moves and finalize the dialogue by determining the outcome. Listing 9.4 shows the function signature and the implementation used with the experiments of this thesis.

When a dialogue terminates, the dialogue control thread will determine the dialogue outcome and notify the agents as well as any platform listeners, such as the experimentation run or the dialogue viewer tool. Only a single outcome rule is used for a dialogue, as it implementes the outcome selection function of Definition 3.19. An OutcomeSelectionRule should return one of the proposals that the agents made during the dialogue. The method signature and an example implementation are shown in Listing 9.5.

While the outcome selection implementation in the thesis simply selects an arbitrary proposal that is *in*, an alternative implementation might consider other factors, such as the explicit prefer and prefer-equal moves made by the agents or the role of an agent. Similarly, the designer of a deliberation experiment might want to introduce new protocol or termination rules. In fact, the

Listing 9.5: OutcomeSelectionRule interface and an example implementation

```
/**
 * Selects a proposal from those made in the dialogue as dialogue outcome
 * @param dialogue The final, terminated dialogue
 * @param allMoves All legal moves made by the agents, including prefer and
     other moves not affecting a proposal tree
 * @return The winning proposal, or null if there is no dialogue outcome
 */
public abstract Proposal determineOutcome(Dialogue dialogue, List<Move<?
    extends Locution>> allMoves);

@Override
public Proposal determineOutcome(Dialogue dialogue, List<Move<? extends
    Locution>> allMoves) {
   // Create a local set of made proposals and shuffle it
   ArrayList<Proposal> proposals = new
       ArrayList<Proposal>(dialogue.getProposals());
   Collections.shuffle(proposals);
   // Now select the first proposal that is in
   for (Proposal proposal : proposals) {
      if (proposal.isIn()) {
         return proposal;
      }
   }
   return null;
}
```

modular enabling of protocol and outcome selection rules opens up the possibility for a wide variety of argumentation-enabled dialogue experiments. The designer of such an experiment can reuse the experimental tools and BDI agents of the testbed and focus on the effects of the particular protocol and outcome selection rules that are of interest.

9.2 Implementing agent behaviour

To experiment with new types of agents, the designer needs to implement a simple interface that allows the testbed software to start the agent, give it the turn to make moves and inform it of events in the dialogue, such as other agents making moves. The testbed will ensure that any agent is run on a separate thread and is instantiated with the right knowledge, as defined by the scenario. Is it up to the agent implementation to store knowledge, a model of the ongoing dialogue and other state information that the agent requires. The full Agent interface is shown in Listing 9.6.

Listing 9.6: Agent interface

```
public interface Agent {

    /**
     * This should return how the agent wants to be called in the dialogue
     */
    public String getName();

    /**
     * An agent that needs to initialize itself can do so here. It is only
         called once per session.
     * @param participant The Participant object used to represent this agent
         in a Dialogue structure
     */
    public void initialize(Participant participant);

    /**
     * The dialogue is starting and the agent should return either a
         deny-dialogue or join-dialogue move.
     * @param openDialogue The open-dialogue location containing the
         deliberation topic and topic goal
     */
    public Move<? extends Locution> decideToJoin(OpenDialogueLocution
        openDialogue);

    /**
     * An agent has the chance here to make moves. It is a full single turn,
         so a list of all the desired moves has to be returned at once.
     */
    public List<Move<? extends Locution>> makeMoves();

    /**
     * When some agent makes new moves in the dialogue (including the
         sender). An agent can use this to update its model of the dialogue.
     * @param moves The new moves that were submitted to the dialogue
     */
    public void onNewMovesReceived(List<Move<? extends Locution>> moves);

    /**
     * When some message was received from the platform (e.g. when the
         dialogue terminates).
     * @param messages The dialogue messages, with textual information
     */
    public void onDialogueMessagesReceived(List<DialogueMessage> messages);

    /**
     * When some exception occurs in the ongoing dialogue, an agent may be
         notified of this. For example, when it returns invalid moves.
     * @param e The exception that was thrown, with information on the cause
     */
    public void onDialogueException(DialogueException e);

}
```

Example 9.2 A minimal agent that at least wants to present its knowledge of options to other agents would merely have to join the dialogue and play propose moves for each of the options it knows about. An extension would be to at least maintain a model of the ongoing dialogue, by monitoring the moves of other players and updating a model of proposal trees accordingly. Listing 9.7 shows the full implementation of a simple agent that maintains a model of the ongoing dialogue and plays propose moves for each option it was assigned from the scenario.

Note that the example agent is strongly autonomous. Although the software controls the start of the agent, it is not tied to a running dialogue platform. Instead, it can constantly process data, make external connections, start additional threads, and so on.

9.2.1 Behaviour model agent template

Although the testbed puts few constraints on the way that agents function, it is useful to have an agent template with a more limited scope, to be used for local experiments with arguing agents. It matches the flexible yet concise model for agent behaviour presented in Chapter 6, by maintaining the state of the dialogue and dividing the move-making process into the four distinct decision steps of deliberation. In other words, the behaviour model agent simplifies the implementation of a behaviour model-compatible agent for the designer of an experiment, as only the four behaviour decision steps have to be implemented. Moreover, it offers several helper methods that a typical BehaviourModelAgent would want to use, such as querying its knowledge base to find proof for a claim. Listing 9.8 shows the available class members storing the agent state and the function signatures of the abstract functions in the BehaviourModelAgent class that should be implemented by the agent designer.

As the behaviour model maps directly to the four Java functions of the DeliberatingAgent class, the heuristics introduced in Chapter 6 are direct implementations. For example, the considerate revision heuristic of Definition 6.6 is implemented in Listing 9.9 by adding any proposed option, any informed belief and any premise used in an argue move to the knowledge of the agent.

Listing 9.7: An example SimpleAgent that plays proposals and maintains a model of the ongoing dialogue

```
public class SimpleAgent implements Agent {

   private Participant participant;
   private List<Rule> options;
   private Dialogue dialogue;

   public InactiveAgent(AgentXmlData xmlDataFile) {
      // The XML data contains the knowledge we were assigned
      this.options = xmlDataFile.getOptions();
   }

   public String getName() {
      return "Simple agent";
   }

   public void initialize(Participant participant) {
      this.participant = participant;
   }

   public Move<? extends Locution> decideToJoin(OpenDialogueLocution
        openDialogue) {
      // Start our model of the ongoing dialogue
      this.dialogue = new Dialogue(openDialogue.getTopic(),
           openDialogue.getTopicGoal());
      // Always join the dialogue
      Move<JoinDialogueLocution> join = Move.buildMove(participant, null,
           new JoinDialogueLocution(openDialogue.getTopic()));
      return join;
   }

   public List<Move<? extends Locution>> makeMoves() {
      // Propose every option in our knowledge
      List<Move<? extends Locution>> moves = new ArrayList<Move<? extends
           Locution>>();
      for (Rule option : this.options) {
         moves.add(Move.buildMove(this.participant, null, new
              ProposeLocution(option.getClaim())));
      }
      return moves;
   }

   public void onNewMovesReceived(List<Move<? extends Locution>> moves) {
      // Update the model of the ongoing dialogue
      if (this.dialogue != null)
         this.dialogue.update(moves);
   }

   public void onDialogueException(DialogueException e) {
   }

   public void onDialogueMessagesReceived(List<DialogueMessage> messages) {
   }

}
```

Listing 9.8: BehaviourModelAgent's internal state and four decision steps

```
protected Dialogue dialogue;
protected KnowledgeBase beliefs;
protected List<Rule> options;
protected List<Goal> goals;

protected abstract void reviseKnowledge(List<Move<? extends Locution>>
    newMoves);
protected abstract List<ValuedOption> assignAttitudes(List<Constant>
    options);
protected abstract List<Move<? extends Locution>>
    identifyAttackPoints(Dialogue dialogue);
protected abstract List<Move<? extends Locution>>
    generateMoves(List<ValuedOption> valuedOptions);
```

9.2.2 Parameterized BDI agents

The BDIAgent built for use in the experiments of this thesis already implements a large number of simple yet intuitive heuristics. The designer of a deliberation experiment, especially when not focussing on different types of agents, might simply configure the existing BDI agent to its liking, rather than making a full new implementation. The agent behaviour is controlled by setting one of the various configuration properties that the BDIAgent adheres to. Without having to change any code of the testbed software, the agent can be configured using the --property command line option to the baidd-exp program. For every property that is supplied, the testbed software will run some scenario with the specific Property set to true and again with the Property set to to false.

Listing 9.10 lists the properties that are available. Note that some are dependant on each other. For example, AdoptOnlyBeliefsWithoutCounterargument will only work if the agent is configured to AdoptBeliefs in the first place. Similarly, only one of the attitude assignment methods is used at the time, such that with BuildMaxDestroyMin set to true any of the other attitude assignment properties are ignored. By default the agent is configured to play as the simple relevant arguing agent from Definition 6.16. To disable argumentation altogether, the PlayOnlyPropose property can be used, while a chatty agent can be configured by using the PlayInformMoves property.

Listing 9.9: Considerate revision implemented as reviseKnowledge function

```
@Override
protected void reviseKnowledge(List<Move<? extends Locution>> moves) {

   for (Move<? extends Locution> move : moves) {
      if (move.getLocution() instanceof ProposeLocution) {

         // We will adopt any new proposed option
         Constant proposal =
              ((ProposeLocution)move.getLocution()).getConcreteProposal();
         // Do we know about it already?
         if (!isBeliefInOptions(proposal)) {
            // Is not known yet: add it as option belief
            this.optionBeliefs.add(new Rule(proposal));
         }

      } else if (move.getLocution() instanceof DeliberationLocution) {

         // Gather the newly exposed beliefs
         Set<Constant> exposed = new HashSet<Constant>();
         ((DeliberationLocution)move.getLocution()).gatherPublicBeliefs(exposed);
         for (Constant b : exposed) {

            // We add any newly exposed constant if it is not an option or
                 the mutual goal
            if (b instanceof Constant &&
                 !dialogue.getTopicGoal().getGoalContent().equals(b) &&
                 !dialogue.getTopic().isUnifiable(b) &&
                 !beliefs.ruleExists(new Rule(b))) {

               // Check if no argument for the contrary can be constructed
               List<RuleArgument> proofs = helper.findProof(new
                    ConstantList(b.negation()), 0.0, this.beliefs,
                    this.optionBeliefs, null);
               if (proofs.size() == 0)
                  beliefs.addRule(new Rule(b));

            }

         }

      }
   }

}
```

Listing 9.10: BDIAgent's Property enumeration to configure behaviour

```
public enum Property {

   // Adopting beliefs
   AdoptBeliefs,
   AdoptOnlyBeliefsWithoutCounterargument,

   // Attitude assignment
   BuildMaxDestroyMin,
   BuildOnlyMaxUtility,
   BuildAllAboveAverage,
   BuildAllPositiveUtility,

   // Move generation
   PlayRejects,
   OnlyRejectIfCounterArgument,
   OnlyWhyProposeIfCounterArgument,
   OnlyWhyRejectIfArgument,
   OnlyWhyIfCounterArgument,
   SkipIfPossibe,

   // Disable arguing
   PlayOnlyRejects,
   PlayOnlyPropose,
   PlayInformMoves

}
```

9.3 Possibilities and restrictions

The testbed as presented in this chapter opens up a wide array of possibilities for experimentation with arguing agents. Since it is fully developed, with protocol, scenario generation method and various arguing and non-arguing agents, an experiment designer can reuse those parts that it does not want to focus on. For example, if a study is performed on the effect of knowledge dispersion, only the scenario generation method has to be customized, while the dialogue framework, agents and experimentation tool are readily available.

Notable about the testbed presented here is that the platform, including the agents, supports the full expressiveness of structured argumentation. This includes concepts that were not yet used in this thesis, such as strict versus defeasible rules, axioms versus ordinary premises and different acceptability semantics. Because of this, the testbed can not only be of interest for research into argumentation-based dialogue frameworks and strategies for such dialogues, but also for experimental research with argumentation logics. However, it must be

noted that the Java ASPIC implementation currently used has some omissions with respect to the ASPIC framework specification. This issue can be solved by using a more recent implementation, as will be discussed below.

Besides the ideas already presented in this and the previous chapter, possibilities for experimental research with argumentation in multi-agent dialogues include:

- Different agent types competing within a dialogue
- Allow agents to, or even force to, explicitly accept proposals
- Benefits of argumentation in more/less liberal protocols
- Unequal roles and knowledge distributions between agents
- Outcome selection through preference aggregation or a voting phase

Of course, countless other experiments are possible. In any case, each of these studies can be performed with little additional modelling and programming. However, there are also some restrictions that require a more structured reworking of or additions to the platform. One important restriction of the current testbed software is that all agents run on a single machine. No architecture is designed or implemented for experiments with distributed agents. Wells et al. (2008) have proposed the idea to set up an Arguing Agents Competition, much similar to the deliberating agents of this thesis, but set up with a distributed software architecture to allow for competition-like experimentation. Researchers in different physical locations could implement arguing agents and have them compete against each other using the distributed testbed system.

Wells et al. provide a description of how an argumentation testbed can support distributed experiments with competing arguing agents. Some issues are already addressed in our experimentation platform. For instance, evaluation of arguments is handled through the assignment of a status to moves and truthfulness of agents is not a requirement for the used dialogue framework. However, some technical considerations are still valid. A client-server architecture is required for agents to connect to a testbed service, which generates a scenario to play, distributes knowledge to agents and gathers dialogue result data for analysis. Luckily, the existing testbed software already has dialogue control and agent execution decoupled, with message exchange as way of communicating. A distributed version of the testbed software only needs to replace

the local instantiation of agents with a service that allows remote agents to connect. Message exchange, such as requesting and supplying dialogue moves, can, for example, be implemented using an XML-RPC protocol, with arguments encoded in the XML-compatible AIF+ language (Reed et al., 2008). Detailed modelling and implementing this distributed testbed architecture remains as future work.

Some restrictions are in place with respect to the dialogue framework. First, agents are assumed to know the rules of the dialogue game and be capable of communicating in the framework language, including the understanding of the used ASPIC− argumentation framework. Note, however, that they are not enforced to be honest or cooperative and are not required to agree of the evaluation of arguments. This is taken care of by the protocol and the move status evaluation.

Second, although an interesting topic on its own, an experiment designer probably wants to adopt the rules of the minimal deliberation protocol of Definition 3.14, or at least prohibit duplicate proposals and repeated moves. Without these two critical rules, agents can repeatedly make the same proposals and repeat arguments continuously. On the other hand, dropping the turntaking rule, effectively allowing agents to move at any time, is perhaps interesting to experiment with.

Third, the arguing agents presented in Chapter 6 use an attack point identification function that ensures playing of relevant moves. However, this requires all agents in the dialogue to only play relevant moves. In the case of our experiment this is enforced by a protocol rule. If an experiment designer loosens this assumption, dropping the relevance protocol rule, then the presented arguing agents are no longer assured to only play relevant moves.

9.4 Case study

As part of the Master thesis research project, de Leng (2012) has employed the testbed software to study two new types of agents: trust-based agents and secretive agents. The idea behind trust-based agents is that an agent might accept an argument played by another agent, adopting the knowledge contained in the argument, only if it has a trust relation with the other agent. If some agent is not trusted, its arguments can be ignored or even actively attacked. Trust is encoded as a single numeric value for each opponent, indicating the trust level, and a threshold value that defines whether some agent's utterance

should be trusted or not.

Secretive agents were designed by de Leng as agents that prefer not to expose knowledge. The idea is that agents sometimes expect to gain a higher utility if they refrain from playing arguments, because the exposed knowledge can be used against them. Secrecy is handled at two levels: active privacy-sensitivity, causing the agent to actively attack arguments with why moves, and passive privacy-sensitivity, causing agents to skip their turn if that seems beneficial. Moreover, secretive agents will try to play arguments that use premises exposed already in the dialogue, rather than arguments based on private beliefs that were not yet exposed.

Implementation of the two new types of agents was performed by extending the BDIAgent. Holding back on playing, instead skip the turn, and questioning other agents' arguments with why moves, was already supported through the OnlyWhyProposeIfCounterArgument and SkipIfPossibe properties. To prefer playing arguments with already exposed knowledge, the reviseKnowledge Java function was used to maintain a local list of all exposed beliefs and the generateMoves function was adjusted to sort arguments on the percentage of already exposed premises, where the standard BDIAgent simply picks the first available argument. Trust-based agents also extend the existing BDIAgent and add a single check before the normal generateMoves behaviour whether the move, which is some relevant attack point, should be attacked or, instead, if the player of the move is trusted and the move should not be attacked.

De Leng argues that benefits of a trust-based or secretive approach might be more apparent if not every agent in a dialogue uses the same strategy. To experiment with this, the agent instantiation in the experimentation software was reworked to instantiate two different types of agents. For example, if the experimental scenario consists of six agents, the testbed software can instantiate two BDIAgents and four TrustBasedAgents. An extra command line option for baidd-exp was introduced to set the ratio of agent types.

An experiment was performed with both new agent types. This showed that trust-based agents require fewer moves on average, improving communication efficiency, while the combined utility for the multi-agent system, the topic effectiveness, was not notably influenced. However, if only one BDIAgent played against multiple TrustBasedAgents, the trust-based agents, which are quite naive, would perform significantly worse than the lone BDIAgent, which realized a much higher individual utility. The experiment with secretive agents showed little effect of active or passive privacy-sensitivity on the communica-

tion efficiency or topic effectiveness. An exception is that the liberal use of why moves reduced the required number of dialogue moves. This is a counterintuitive result, as why moves typically open up new reasons for discussion and it is therefore expected that playing why moves would increase average dialogue length. Unfortunately, de Leng does not provide a reason for this discrepancy.

With respect to this thesis, de Leng's experimental work on trust-based and secretive agents showed that the experimental platform is very capable as testbed for new experimental work with argumentation. Through the isolation of decision steps and the availability of proven and standardized components, including a fully specified arguing agent, even complex ideas with arguing agents can be implemented and tested in a short time span.

This chapter has shown how the software and experimental method developed for this thesis can be used as testbed system for future argumentation research. The main benefit is that the designer of some argumentation-related experiment does not have to model and implement a full experimental platform stack, including arguing agents and a configurable experimentation tool. As shown, experimentation with protocols is supported through easy configuration and extension of deliberation, termination and outcome selection rules. Research focussing on strategies for argumentation-based dialogues can either take full control over the agent process, use the behaviour model-based agent template or even just configure the existing parametrized BDI agent, depending on the scope of the intended agent design. In any case, the scenario generation-powered platform supports execution of many repeated experiment runs and extracting performance-analytical data based on the four distinct deliberation metrics. A case study has shown how experimental work with argumentation is encouraged by the presented testbed system.

Chapter 10

Discussion

The research project for this thesis was set up to investigate claims that employing argumentation in dialogues would benefit a multi-agent system in terms of efficiency and effectiveness. Deliberation dialogues were chosen as prime topic, as it is arguably the most versatile and interesting of the argumentation-based dialogue types. An experimental approach was taken to find evidence concerning the claimed argumentation benefits. It has the potential to show in which practical cases benefits do or do not arise and allows for a performance-wise comparison of arguing and non-arguing agents.

10.1 Conclusions

As broken down in the introductory chapter, testing the performance of agents in deliberation dialogues first of all requires the establishing of desirable properties of deliberation. Performance of the multi-agent system is measured through the introduction of metrics for deliberation dialogues. The resulting performance numbers allow for a quantitative comparison of dialogues and, as an extension, of the agents participating in those dialogues. This raised the question:

Research sub-question 1 Which properties of deliberation dialogues are desirable and how can we measure these properties?

Chapter 7 explored the desirable properties of deliberation dialogues. Although the general terms efficiency and effectiveness are commonly used in the literature, these concepts still needed to be made concrete. In fact, based on the

literature, a distinction was made into communication and topic layer metrics. This led to four metrics, each providing a numeric performance figure on the dialogue itself and the agents that participated in the dialogue. Additionally, it was shown how the notion of Pareto optimality can be used to value a dialogue outcome quantitatively, as alternative to the summed utility values of agents.

The designer of a deliberation experiment can select the appropriate metrics depending on the use case. For example, if information is highly sensitive, the designer can focus on the topic efficiency, measuring the degree of information concealment. Combined, the four introduced metrics cover the spectrum of desirable properties for deliberation style dialogues. They can therefore be used as general performance measure to compare (arguing and non-arguing) agents.

Although dialogue frameworks existed for various argumentation-based dialogue types and concrete strategies for such dialogues have been proposed in the literature, no suitable unified framework yet existed for deliberation dialogues and arguing agents. This justified the following research sub-question:

Research sub-question 2 What is a suitable framework for specifying deliberation dialogues and agent behaviours and how can the performance of deliberating agents be compared?

A framework for deliberation style dialogues has been proposed in Chapter 3. It allows agents to take turn to play moves such as proposing new options, providing arguments and stating preferences. Through the explicit reply structure, every move can be assigned a status that can be used to enforce move relevance or select a proposal as dialogue outcome. This dialogue framework is very suitable for experimental work for four reasons. First, it supports argumentation with structured arguments to allow for rich disputes, in line with modern argumentation logics. This includes epistemic as well as practical arguments, combining beliefs with goals and options. Second, dialogues can be evaluated by simply looking at the explicit reply structure of moves. For this reason, agents do not have to agree on the evaluation on submitted arguments in the dialogue. Third, the topic language is extensible with other types of attacking or surrendering moves, without having to specify semantics on how the new move type influences other moves. For example, allowing an explicit accept move as surrendering reply on propose moves would implement agents accepting a proposal explicitly in a natural fashion. Fourth, the dialogue protocol can be made more or less liberal to make it suitable for a specific experiment.

Agents were designed, in Chapter 6, as fully autonomous entities in the experimental framework. Based on the BDI architecture, agents maintain their own world knowledge and model of the dialogue, they can internally reason with arguments and are free to (not) make moves as they see fit, as long as the dialogue protocol is adhered to. This design allows for a wide variety of agent strategies. To aid in the design of agents, and to make heuristics for individual deliberation decision steps directly comparable, a behaviour model was specified. It makes concrete the four steps relevant to deliberation dialogue decision making: knowledge revision, attitude assignment, attack point identification and move generation. Several arguing and non-arguing heuristics have been proposed that implement decision steps and together form simple yet fully functional deliberating agents.

The final part of a fully specified framework for deliberation experiments is to provide agents with scenarios to play. Chapter 4 introduced a method to generate scenarios that reflect the characteristics of typical deliberation situations. The core idea is that goals and options known to agents are not disconnected entities, but are connected by defeasible rules to form a rule chain. The chains allow means-end reasoning that combines epistemic with practical reasoning to show how an option promises to satisfy a certain goal. Through chaining, the resulting scenarios are not incoherent bodies of knowledge, but provide a structure that allows agents to propose and support options in a dialogue. Chapter 5 shows how the scenario generation method can best be configured for an experiment, to maximize the potential for interesting deliberation dialogues.

To find evidence that using argumentation in deliberation dialogues brings benefits, agents that use argumentation need to be compared to agents that do not argue. Of course, agents use argumentation by playing argue moves in a dialogue, but argumentation can be used internally by the agents as well, as internal reasoning method. To study how internal reasoning and playing of argue moves are related, the following question was asked:

Research sub-question 3 How can agents use arguments to reason about options and generate appropriate moves?

Playing a deliberation dialogue is not an easy task. Both the revision of knowledge and the strategic reasoning towards move generation require a combination of knowledge, personal goals and utilities, the current dialogue status and option potential. As already shown in Chapter 5, agents can use argumentation internally to support the reasoning process over knowledge. Through

construction of practical arguments, option potential can be established for both the mutual goal, required to propose an option, as well as personal goals.

Chapter 6 has provided several heuristics that take advantage of option potential, using the power of the underlying argumentation logic to derive defensible arguments for and against proposals. Knowledge can be revised selectively by only adopting beliefs for which there is no proof to the contrary. Attitudes can be assigned to options based on their potential to realize personal goals. Finally, move generation, choosing one move to make from the many legal moves, was supported by an explicit attack point identification step. The notion of relevance from the dialogue framework was reused to realize agents that only play arguments relevant to the topic at hand, that is, the status of some existing dialogue proposal.

With the formal framework, metrics and fully specified agents at hand, it is finally possible to experimentally explore if (and when) argumentation brings about benefits for a multi-agent system. Through experiments with arguing and non-arguing agents, the main research question of this thesis can now be answered:

Research question What are the benefits of using argumentation in multi-agent deliberation dialogues?

The experiments of Chapter 8 provide several answers to this question. Most importantly, an experiment between a simple rational arguing agent and simple and chatty non-arguing agents showed that argumentation decreases efficiency but increases effectiveness. This is true for the communication layer as well as the topic layer. In fact, it was concluded that there is a trade-off between efficiency and effectiveness. The designer of an agent will have to consider the specifics of the deliberation system where an agent will be deployed, in order to determine when to have agents argue and when argumentation is best not used. However, more sophisticated agents than those used in this thesis will likely have a less strict reverse correlation between expected number of moves and combined utility. For example, not engaging in a dispute on claims that are already discussed elsewhere in the dialogue may strongly improve the communication efficiency performance of the arguing agent.

Some discouraging results were found as well. Prominently, the arguing agent was not able to outperform the baseline performance of randomly selecting a proposed option and doing away with argumentation altogether. As

hypothesized in Chapter 8, this is likely the result of a strong dispersion of utilities for options between the agents in the dialogue. Further experimentation is necessary to validate this intuition. Another influencing factor is the simplistic outcome selection function of picking a random proposal that is *in*. A selection function that adheres to explicit prefer and prefer-equal moves or even introduces a separate voting phase might already show how arguing agents can outperform the baseline performance, as not just any proposal would be picked, but rather one that is *in* and agents explicitly prefer best. In any case, the use of argumentation should not only be found in the utility of the selected dialogue outcome, but also in the providing of proof, by means of practical arguments, on why the selected proposal is indeed a good outcome.

Other evidence of there being a trade-off between efficiency and effectiveness was provided with the experiment on self-interested agents. It was shown that employing a more self-interested attitude assignment increases dialogue topic and communication efficiency, while it decreases the topic effectiveness, albeit only measured in terms of Pareto optimality. While this further validates our general conclusion on the benefits of argumentation, it also shows that there is room for arguing agents that perform better on efficiency while not, or only to a limited degree, having to give in on effectiveness performance. One of the challenges for future experimental research with arguing agents is therefore to devise agents that perform well on all four specified metrics, or at least on those that are important for the specific system.

10.2 Contributions

With the research performed for and described in this thesis, several concrete advancements have been made to the existing work in argumentation-based dialogue research. Although all have been discussed implicitly throughout the previous chapters, it is useful to enumerate them here.

The first contribution of this thesis is a fully defined framework for deliberation dialogues and deliberating agents. The framework for deliberation dialogues developed for this thesis was first proposed in Kok et al. (2010). Deliberation dialogues are particularly interesting with respect to experimental evaluation as multiple agents need to compete and cooperate at the same time, while combining epistemic and practical reasoning. The presented framework is the first of sorts that allows the use of the full expressive power of modern structured argumentation logics in deliberation dialogues. Although several

strategies for argumentation-based dialogues have been proposed, no generalized framework for agent behaviour in argumentation-based dialogues, as proposed in thesis, was readily available. The proposed framework makes concrete the individual decision steps that were left implicit in previous work, by which easier comparison of heuristics for these steps is now possible.

A second contribution is a method for performing experiments with argumentation-enabled agents in multi-agent dialogues that is extensible and general enough to cater for many future research directions. Importantly, instead of having to rely on an existing set of cases for agents to deliberate on, scenarios can be generated that provide interesting situations for the evaluation of arguing agents. Scenario generation was first proposed in Kok et al. (2012c). Moreover, contrary to existing experimental work with argumentation dialogues, the scenarios provide agents with beliefs, goals and options that allow them to use structured argumentation, to reason internally and to argue in the dialogue. The metrics for deliberation dialogues complete the experimental platform and, accompanying the platform, a method to experimentally compare agents that use argumentation with agents that do not argue.

Preliminary results have been obtained, as third research contribution, that show a trade-off between efficiency and effectiveness whether to employ argumentation or not. This result was first presented in Kok et al. (2012b) and, in more detail, in Kok et al. (2012a). The results are said to be preliminary as there are hints that sophisticated agent behaviour can overcome to a certain extent the inverse correlation between efficiency and effectiveness. Several simple alternative heuristics have been proposed that support this finding.

This thesis concluded with an explanation on how the experimental platform of this thesis can be used as testbed system for future research, which is the fourth contribution. A full software stack is now available that covers dialogue framework, agents and performance-analytical capabilities, and that can be reused (in part or completely) by new studies with argumentation in multi-agent dialogues. This allows for focus on the specific topic at hand, without having to implement a full platform, but still having access to the power of a full experimentation platform. Although not yet built as distributed platform, designers of future experiments with deliberating agents can reuse the components that are of less interest to their experiments and focus on, for instance, a specific knowledge revision method or study the influence of unequally distributed knowledge between agents.

10.3 Future work

The work of this thesis is only one further step towards a true understanding of the role and benefits of argumentation in agent communication. In fact, several new lines of research can be started, stemming from the results presented in this thesis. Several recent directions in the field of computational argumentation can prove useful. To start with, the formal dialogue framework can be generalized to enable other types of argumentation-based dialogue types, such as Black and Hunter (2007)'s inquiry system, to be embedded in a deliberation dialogue. Alternatively, the topic language can be enriched with novel argumentation logic enhancements. Value-based practical reasoning (van der Weide et al., 2009) seems especially useful as it naturally combines internal motivations in agents to option preferences. The deliberation dialogue-compatible decomposition of options into plans, proposed by Toniolo et al. (2012), is another attractive extension, as it can help agents understand the relations between options and therefore potentially improve deliberation performance.

Generating interesting and realistic scenarios is at the heart of useful and successful experimentation with arguing agents. The method presented in this thesis could be augmented with techniques that translate natural language arguments into a formal model argumentative model. Wyner et al. (2012) are using argumentation-based models to extract logical arguments from natural language. The extracted arguments based on domain-specific argumentation schemes, can then be evaluated using readily available argumentation logics. A challenging yet highly useful addition would be to apply this approach to generate even richer and more realistic scenarios.

Regarding the agent behaviour model, it would be good to validate the expressiveness of the proposed model by implementing state-of-the-art argumentation dialogue heuristics, such as Snaith and Reed (2012a)'s knowledge revision. More generally, the testbed system challenges researchers to devise strategies that work well on various, or even all, of the deliberation metrics at once. For instance, advanced strategies could recognise which claims have already been discussed elsewhere in the dialogue. If agents have multiple encounters, it is interesting to investigate how opponent models can improve argumentation-based reasoning in deliberating agents.

To continue experimental research with the presented testbed platform, it could be useful to update the technical implementation in two ways. First, the platform should adopt a distributed architecture. This would allow agents from

different researchers to compete and thereby make state-of-the-art agent comparisons easier. The improved architecture should allow different agent types to join in a single dialogue. Second, the currently used implementation of the ASPIC+ argumentation framework should be replaced by the recent implementation by Snaith and Reed (2012b). This would not only solve the implementation inconsistency issues described in Chapter 2, but also offer the full feature set of the most recent ASPIC+ framework definition to the agents. This further enriches the experimentation platform, allowing for even more interesting and sophisticated agents and, by extension, deliberation experiments.

A final very interesting research question is what the benefits of argumentation are for individual agents, not only for the multi-agent system as a whole. Desirable properties of deliberation and accompanying metrics need to be introduced that reflect individual agent performance, rather than that of the system. If experiments with benefits for individual agents showed a similar trade-off between efficiency and effectiveness, a big step forwards would be make into understanding how argumentation benefits multi-agent communication.

Bibliography

Amgoud, L., Bodenstaff, L., Caminada, M., McBurney, P., Parsons, S., Prakken, H., van Veenen, J., and Vreeswijk, G. A. W. (2006). ASPIC Deliverable D2.6: Final review and report on formal argumentation system.

Amgoud, L. and Dupin De Saint-Cyr, F. (2008). Measures for Persuasion Dialogs: A Preliminary Investigation. In Besnard, P., Doutre, S., and Hunter, A., editors, *Proceedings of the 2th International Conference on Computational Models of Argument (COMMA2008)*, volume 172 of *Frontiers in Artificial Intelligence and Applications*, pages 13–24, Toulouse, France. IOS Press.

Amgoud, L. and Maudet, N. (2002). Strategical Considerations for Argumentative Agents (preliminary report). In Benferhat, S. and Giunchiglia, E., editors, *Proceedings of the 9th International Workshop on Non-Monotonic Reasoning (NMR2002)*, pages 399–407, Toulouse, France. n.p.

Amgoud, L., Maudet, N., and Parsons, S. (2000). Modelling Dialogues using Argumentation. In Durfee, E. H., editor, *Proceedings of the 4th International Conference on Multi-Agent Systems (ICMAS2000)*, pages 31–38, Boston, MA, USA. IEEE Computer Society.

Amgoud, L. and Prade, H. (2009). Using Arguments for Making and Explaining Decisions. *Artificial Intelligence*, 173(3-4):413–436.

Atkinson, K., Bench-Capon, T. J. M., and McBurney, P. (2005a). A Dialogue Game Protocol for Multi-Agent Argument over Proposals for Action. *Autonomous Agents and Multi-Agent Systems*, 11(2):153–171.

Atkinson, K., Bench-Capon, T. J. M., and McBurney, P. (2005b). Arguing about Cases as Practical Reasoning. In Gardner, A., editor, *Proceed-*

ings of the 10th International Conference on Artificial Intelligence and Law (ICAIL2005), pages 35–44, Bologna, Italy. ACM Press.

Atkinson, K., Bench-Capon, T. J. M., and Walton, D. (2012). Distinctive features of persuasion and deliberation dialogues. *Journal of Logic and Computation*, 3:1–23.

Baroni, P., Caminada, M., and Giacomin, M. (2011). An Introduction to Argumentation Semantics. *The Knowledge Engineering Review*, 26(4):365–410.

Bench-Capon, T. J. M. (2003). Persuasion in Practical Argument using Value-based Argumentation Frameworks. *Journal of Logic and Computation*, 13(3).

Bench-Capon, T. J. M. and Prakken, H. (2006). Justifying Actions by Accruing Arguments. In Dunne, P. E. and Bench-Capon, T. J. M., editors, *Proceedings of the 1th International Conference on Computational Models of Argument (COMMA2006)*, volume 144 of *Frontiers in Artificial Intelligence and Applications*, pages 247–258, Liverpool, UK. IOS Press.

Black, E. and Atkinson, K. (2010). Agreeing What to Do. *Proceedings of the 7th International Workshop on Argumentation in Multi-Agent Systems (ArgMAS2010)*, 6614:1–18.

Black, E. and Atkinson, K. (2011). Choosing Persuasive Arguments for Action. In Sonenberg, L., Stone, P., Tumer, K., and Yolum, P., editors, *Proceedings of the 10th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS2011)*, volume 1-3, pages 905–912, Taipei, Taiwan. IFAAMAS.

Black, E. and Hunter, A. (2007). A Generative Inquiry Dialogue System. In Durfee, E. H., Yokoo, M., Huhns, M. N., and Shehory, O., editors, *Proceedings of the 6th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007)*, volume 5, page 241, Honolulu, Hawaii, USA. IFAAMAS.

Carlson, L. (1983). *Dialogue Games: An Approach to Discourse Analysis*. Reidel Publishing Company, Dordrecht, The Netherlands.

de Leng, D. (2012). *Experimenting with Strategies in Inter-Agent Argumentation Dialogues*. Master's thesis, University of Utrecht, The Netherlands.

Dijkstra, P., Bex, F., Prakken, H., and de Vey Mestdagh, K. (2005). Towards a Multi-Agent System for Regulated Information Exchange in Crime Investigations. *Artificial Intelligence and Law*, 13(1):133–151.

Dung, P. M. (1995). On the Acceptability of Arguments and its Fundamental Role in Non-Monotonic Reasoning, Logic Programming and N-Person Games. *Artificial Intelligence*, 77(2):321–357.

FIPA (2000). FIPA ACL Message Structure Specication. http://www.fipa.org/specs/fipa00061/.

Karunatillake, N. C., Jennings, N. R., Rahwan, I., and McBurney, P. (2009). Dialogue Games that Agents Play within a Society. *Artificial Intelligence*, 173(9-10):935–981.

Kok, E. M., Meyer, J.-J. C., Prakken, H., and Vreeswijk, G. A. W. (2010). A Formal Argumentation Framework for Deliberation Dialogues. In McBurney, P., Parsons, S., and Rahwan, I., editors, *Proceedings of the 7th International Workshop on Argumentation in Multi-Agent Systems (ArgMAS2010)*, volume 6614 of *Lecture Notes in Computer Science*, pages 31–48, Toronto, Canada. Springer.

Kok, E. M., Meyer, J.-J. C., Prakken, H., and Vreeswijk, G. A. W. (2012a). Testing the Benefits of Structured Argumentation in Multi-Agent Deliberation Dialogues. In McBurney, P., Parsons, S., and Rahwan, I., editors, *Proceedings of the 9th International Workshop on Argumentation in Multi-Agent Systems (ArgMAS2012)*, Lecture Notes in Computer Science, pages 87–106, Valencia, Spain. Springer.

Kok, E. M., Meyer, J.-J. C., Prakken, H., and Vreeswijk, G. A. W. (2012b). Testing the Benefits of Structured Argumentation in Multi-Agent Deliberation Dialogues (extended abstract). In van der Hoek, W., Padgham, L., Conitzer, V., and Winikoff, M., editors, *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, pages 1411–1412, Valencia, Spain. IFAAMAS.

Kok, E. M., Meyer, J.-J. C., van Oostendorp, H., Prakken, H., and Vreeswijk, G. A. W. (2012c). A Methodology for the Generation of Multi-Agent Argumentation Dialogue Scenarios. In Cossentino, M., Kaisers, M., Tuyls, K., and Weiss, G., editors, *EUMAS'11 Selected and Revised Papers*, number 7541 in

Lecture Notes in Computer Science, pages 160–174, Maastricht, The Netherlands. Springer.

McBurney, P., Hitchcock, D., and Parsons, S. (2007). The Eightfold Way of Deliberation Dialogue. *International Journal of Intelligent Systems*, 22(1):95–132.

McBurney, P. and Parsons, S. (2002). Games that Agents Play: A Formal Framework for Dialogues between Autonomous Agents. *Journal of Logic, Language and Information*, 11(3):315–334.

McBurney, P., Parsons, S., and Wooldridge, M. (2002). Desiderata for Agent Argumentation Protocols. In *Proceedings of the 1st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2002)*, pages 402–409, Bologna, Italy. ACM.

Paglieri, F. and Castelfranchi, C. (2004). Revising Beliefs Through Arguments: Bridging the Gap Between Argumentation and Belief Revision in MAS. In Rahwan, I., Moraitis, P., and Reed, C., editors, *Proceedings of the 1st International Workshop on Argumentation in Multi-Agent Systems (ArgMAS2004)*, volume 3366 of *Lecture Notes in Computer Science*, pages 78–94, New York, NY, USA. Springer.

Pajares Ferrando, S. and Onaindia, E. (2012). Defeasible Argumentation for Multi-Agent Planning in Ambient Intelligence Applications. In van der Hoek, W., Padgham, L., Conitzer, V., and Winikoff, M., editors, *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, pages 509–516, Valencia, Spain. IFAAMAS.

Parsons, S. and Jennings, N. R. (1996). Negotiation through Argumentation – A Preliminary Report. In Tokoro, M., editor, *Proceedings of the 2nd International Conference on Multiagent Systems (ICMAS1996)*, pages 267–274, Kyoto, Japan. The AAAI Press.

Parsons, S., Sierra, C., and Jennings, N. R. (1998). Agents that Reason and Negotiate by Arguing. *Journal of Logic and Computation*, 8(3):261–292.

Pasquier, P., Hollands, R., Rahwan, I., Dignum, F., and Sonenberg, L. (2010). An Empirical Study of Interest-based Negotiation. *Autonomous Agents and Multi-Agent Systems*, 22(2):249–288.

Pini, M. S., Rossi, F., Venable, K. B., and Walsh, T. (2008). Aggregating Partially Ordered Preferences. *Journal of Logic and Computation*, 19(3):475–502.

Pollock, J. L. (1987). Defeasible Reasoning. *Cognitive Science*, 11(4):481–518.

Prakken, H. (2005). Coherence and Flexibility in Dialogue Games for Argumentation. *Journal of Logic and Computation*, 15(6):1009–1040.

Prakken, H. (2010). An Abstract Framework for Argumentation with Structured Arguments. *Argument and Computation*, 1(2):93–124.

Prakken, H. and Vreeswijk, G. A. W. (2002). *Logics for Defeasible Argumentation*, volume 4, pages 218–319. Kluwer Academic Publishers.

Rahwan, I. (2005). Guest Editorial: Argumentation in Multi-Agent Systems. *Autonomous Agents and Multi-Agent Systems*, 11(2):115–125.

Rahwan, I. and Amgoud, L. (2006). An Argumentation-based Approach for Practical Reasoning. *Proceedings of the 5th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS2006)*, pages 347–354.

Rahwan, I., Pasquier, P., Sonenberg, L., and Dignum, F. P. M. (2007). On the Benefits of Exploiting Underlying Goals in Argument-based Negotiation. In Cohn, A., editor, *Proceedings of 22nd Conference on Artificial Intelligence (AAAI2007)*, number 1, pages 116–121, Vancouver, Canada. AAAI Press.

Rahwan, I., Ramchurn, S. D., Jennings, N. R., McBurney, P., Parsons, S., and Sonenberg, L. (2004). Argumentation-based Negotiation. *The Knowledge Engineering Review*, 18(04):343–375.

Rao, A. and Georgeff, M. (1991). Modeling Rational Agents within a BDI-Architecture. In Allen, J. F., Fikes, R., and Sandewall, E., editors, *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR1991)*, pages 473–484, Cambridge, MA, USA. Morgan Kaufmann.

Reed, C., Wells, S., Devereux, J., and Rowe, G. (2008). Aif+: Dialogue in the Argument Interchange Format. In Besnard, P., Doutre, S., and Hunter, A., editors, *Proceedings of the 2th International Conference on Computational Models of Argument (COMMA2008)*, volume 172 of *Frontiers in Artificial Intelligence and Applications*, pages 311–323, Toulouse, France. IOS Press.

Searle, J. (1976). A Classification of Illocutionary Acts. *Language in Society*, 5(1):1–23.

Sierra, C., Jennings, N. R., Noriega, P., and Parsons, S. (1997). A Framework for Argumentation-based Negotiation. In Singh, M. P., Rao, A. S., and Wooldridge, M., editors, *Proceedings of the 4th International Workshop on Agent Theories, Architectures, and Languages (ATAL1997)*, volume 1365 of *Lecture Notes in Computer Science*, pages 177–192, Rhode Island, USA. Springer-Verlag.

Snaith, M. and Reed, C. (2012a). Justified Argument Revision in Agent Dialogue. In McBurney, P., Parsons, S., and Rahwan, I., editors, *Proceedings of the 9th International Workshop on Argumentation in Multi-Agent Systems (ArgMAS2012)*, Lecture Notes in Computer Science, pages 87–106, Valencia, Spain. Springer.

Snaith, M. and Reed, C. (2012b). TOAST: Online ASPIC+ Implementation. In Verheij, B., Szeider, S., and Woltran, S., editors, *Proceedings of the 4th International Conference on Computational Models of Argument (COMMA2012)*, volume 245 of *Frontiers in Artificial Intelligence and Applications*, pages 509–510, Vienna, Austria. IOS Press.

South, M. and Vreeswijk, G. A. W. (2009). ASPIC Java Components. http://aspic.cossac.org/.

Toniolo, A., Norman, T. J., and Sycara, K. P. (2012). An Empirical Study of Argumentation Schemes for Deliberative Dialogue. In de Raedt, L., Bessière, C., Dubois, D., Doherty, P., Frasconi, P., Heintz, F., and Lucas, P. J. F., editors, *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI2012)*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, pages 756–761, Montpellier, France. IOS Press.

van der Weide, T. L., Dignum, F. P. M., Meyer, J.-J. C., Prakken, H., and Vreeswijk, G. A. W. (2009). Practical Reasoning using Values: Giving meaning to Values. In McBurney, P., Rahwan, I., Parsons, S., and Maudet, N., editors, *Proceedings of the 6th International Workshop on Argumentation in Multi-Agent Systems (ArgMAS2009)*, volume 6057 of *Lecture Notes in Computer Science*, pages 225–240, Budapest, Hungary. Springer.

Visser, W. (2008). Implementation of Argument-based Practical Reasoning.

Vreeswijk, G. A. W. (1997). Abstract Argumentation Systems. *Artificial Intelligence*, 90(1-2):225–279.

Walton, D. (2007). How to Make and Defend a Proposal in a Deliberation Dialogue. *Artificial Intelligence and Law*, 14(3):177–239.

Walton, D. N. and Krabbe, E. C. W. (1995). *Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning*. State University of New York Press, New York, NY, USA.

Wells, S., Łoziński, P., and Nhat Pham, M. (2008). Towards An Arguing Agents Competition: Architectural Considerations. In *Proceedings of the 8th Workshop on Computational Models of Natural Argument (CNMA2008)*, Patras, Greece. n.p.

Wooldridge, M., Jennings, N. R., and Kinny, D. (2000). The Gaia Methodology for Agent-Oriented Analysis and Design. *Autonomous Agents and Multi-Agent Systems*, 3(3):285–312.

Wyner, A., Schneider, J., Atkinson, K., and Bench-Capon, T. J. M. (2012). Semi-Automated Argumentative Analysis of Online Product Reviews. In Verheij, B., Szeider, S., and Woltran, S., editors, *Proceedings of the 4th International Conference on Computational Models of Argument (COMMA2012)*, volume 245 of *Frontiers in Artificial Intelligence and Applications*, pages 43–50, Vienna, Austria. IOS Press.

Yuan, T., Schulze, J., Devereux, J., and Reed, C. (2008). Towards an Arguing Agents Competition: Building on Argumento. In *Proceedings of the 8th Workshop on Computational Models of Natural Argument (CNMA2008)*, pages 61–874, Patras, Greece. n.p.

Samenvatting

Argumentatie is voor mensen zo vanzelfsprekend als het voeren van een dialoog. Het is dan ook een uitstekend middel om keuzes te maken en te bespreken. Immers kan de vinger snel op een pijnpunt worden gelegd en meningsverschillen kunnen slagvaardig worden opgelost. Het is dan ook niet gek dat er in allerlei vormen van wetenschap interesse is voor argumentatie.

Binnen de kunstmatige intelligentie belooft het gebruik van argumenten heel wat voordelen te bieden. Agenten, de spelers in een intelligent systeem, zouden met argumentatie beter kunnen redeneren en onderlinge dialogen verlopen effectiever en efficiënter. Hierom zijn er argumentatielogica's ontwikkeld en raamwerken en protocollen gemaakt voor diverse argumentatie-gebaseerde dialoogtypen. Echter is er maar weinig onderzoek uitgevoerd waarbij de voordelen die argumentatie zou bieden op praktische wijze worden onderzocht.

In dit proefschrift wordt een compleet raamwerk voorgesteld, waarmee op experimentele wijze de praktische voordelen van argumentatie in dialogen kunnen worden onderzocht. Een model voor beraadslaging biedt meerdere agenten de mogelijkheid om een dialoog aan te gaan over welke actie er moet worden ondernomen, om een gezamenlijk doel te realiseren. Door het spelen van argumenten proberen de agenten elkaar te overtuigen van de beste actie.

Om te onderzoeken of argumentatie daadwerkelijk de dialoogeffectiviteit en -efficiëntie positief beïnvloed, dienen argumenterende en niet-argumenterende agenten te worden vergeleken. Hiervoor worden een model en een methode voorgesteld om scenario's te genereren en vervolgens te laten spelen door de agenten. Diverse vormen van eenvoudige doch rationele strategieën worden voorgesteld en gebruikt om vele dialogen te spelen. Effectiviteit en -efficiëntie van iedere dialoog wordt vervolgens vanuit gezichtspunt van zowel het gespreksonderwerp als de communicatie getoetst.

Vanuit experimenten met argumenterende en niet-argumenterende agenten blijkt dat efficiëntie op zowel de onderwerp- als de communicatielaag niet positief maar juist negatief worden beïnvloed. Het aanvoeren en verdedigen van argumenten zorgt namelijk voor langere dialogen, waarbij meer informatie moet worden geopenbaard dan wanneer slechts voorstellen worden geopperd en afgekeurd. De effectiviteit blijkt echter positief te worden beïnvloed als agenten argumenten gebruiken. Argumentatie helpt agenten, zo blijkt, om afgewezen voorstellen te verdedigen, waardoor het probleem verholpen kan worden dat geen enkele actie gekozen uiteindelijk gekozen wordt. Bovendien helpt argumentatie de agenten om relevante toevoegingen aan de dialoog te blijven doen, terwijl bij het simpelweg delen van informatie uitingen niet coherent zijn. Concluderend lijkt er een omgekeerde balans (trade-off) te zijn tussen efficiëntie en effectiviteit. Of argumentatie beter achterwege kan worden gelaten is afhankelijk van de eisen van het specifieke systeem.

Dit proefschrift levert niet alleen verder bewijs dat argumentatie in agent-dialoogsystemen voordelen kan hebben. Het experimentele raamwerk dat is beschreven biedt tevens vele mogelijkheden tot experimenteel vervolgonderzoek naar argumentatielogica's, -dialogen en -strategieën. Het geïntroduceerde softwaresysteem leent zich uitstekend tot hergebruik, waarbij reeds een complete implementatie beschikbaar is voor generatie van scenario's, een dialoogmodel dat gestructureerde argumentatie ondersteund en een toetsingsmethode voor simpele maar ook zeer complexe beraadslagingsstrategieën.

Dankwoord

Hoewel een proefschrift op het eerste gezicht een zuiver wetenschappelijke uiteenzetting lijkt, in werkelijkheid belichaamt het zo veel meer. Na vier jaar proberen, testen, falen, schrijven, bouwen, nachtwerk, uitstel, ontspanning, deadlines, presenteren, afleiding en delibereren is iedere bladzijde op zich een klein stukje geschiedenis. En natuurlijk zijn er heel wat mensen zonder wie die bladzijdes nooit realiteit waren geworden.

Allereerst bedank ik uiteraard graag mijn promotoren, Henry, John-Jules en Gerard. Van mijn naïef enthousiasme en via harde realiteit tot een gedegen proefschrift komen is alleen mogelijk geweest door jullie kennis, geduld en aanmoedigingen. Fijn dat ik het project zo naar eigen hand mocht zetten, met alle uitdagingen en kansen die dat heeft gegeven.

Wat ik zeker ook ga missen zijn mijn collega's. Ook al veranderde door de jaren steeds de bezetting, de lunches, avonden, weekenden weg en conferentiebezoeken behoren absoluut tot mijn leukste herinneringen tijdens mijn promotietraject. Bas (2x), Hado, Joost (2x), Loïs, Maaike, Marieke, Max, Michal, Nick, Nieske, Paolo, Pouyan, Sjoerd en Tom, super bedankt. Gennaro, I really enjoyed our talks and beer trips, thanks. Liz, it was awesome having you as a roommate, thank you.

De onvoorwaardelijke steun van familie wordt wel eens onderschat, maar ik weet zeker dat ik zonder ze het spoor bijster zou zijn geraakt. Mam, ik hoop dat ik je trots heb gemaakt. Peter, ik vind het bijzonder hoe we theoriën op elkaar afvuren en zonder het echt te begrijpen, volledig eens kunnen zijn. Paul, Door en Mees, ik kan geen toekomst voorspellen, maar dat het super wordt lijkt me duidelijk. Ego, thanks man, ik heb al zin in een nieuwe filmmarathon.

Anna, Cécile, Erica, Johan, Leonie, Martijn, Miranda, Mirte, Nicolas, Sjoerd, jullie waren onontbeerlijk de afgelopen jaren. Ik vind het bijzonder dat wij na

zo'n lange tijd nog zo veel lol kunnen hebben, zelfs nu de afstand plots zo groot is. Dat dit nog maar vele jaren mag duren.

Lieve Bertanne, niemand is zo belangrijk voor mij als jij dat bent. Soms kon alleen jij me motiveren en aan het werk zetten, maar bovenal breng ik met niemand zo lief tijd door als met jou. Ons nieuwe avontuur in Frankrijk is nu al zo leuk, dat ik niet kan wachten om te weten wat we de volgende 10 jaar samen gaan doen, en daarna.

Curriculum vitae

Publications

Kok, E. M., Meyer, J.-J. C., van Oostendorp, H., Prakken, H., and Vreeswijk, G. A. W. (2012c). **A Methodology for the Generation of Multi-Agent Argumentation Dialogue Scenarios**. In Cossentino, M., Kaisers, M., Tuyls, K., and Weiss, G., editors, *EUMAS'11 Selected and Revised Papers*, number 7541 in Lecture Notes in Computer Science, pages 160–174, Maastricht, The Netherlands. Springer.

Kok, E. M., Meyer, J.-J. C., Prakken, H., and Vreeswijk, G. A. W. (2012b). **Testing the Benefits of Structured Argumentation in Multi-Agent Deliberation Dialogues (extended abstract)**. In van der Hoek, W., Padgham, L., Conitzer, V., and Winikoff, M., editors, *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, pages 1411–1412, Valencia, Spain. IFAAMAS.

Kok, E. M., Meyer, J.-J. C., Prakken, H., and Vreeswijk, G. A. W. (2012a). **Testing the Benefits of Structured Argumentation in Multi-Agent Deliberation Dialogues**. In McBurney, P., Parsons, S., and Rahwan, I., editors, *Proceedings of the 9th International Workshop on Argumentation in Multi-Agent Systems (ArgMAS2012)*, Lecture Notes in Computer Science, pages 87–106, Valencia, Spain. Springer.

Kok, E. M., Meyer, J.-J. C., Prakken, H., and Vreeswijk, G. A. W. (2010). **A Formal Argumentation Framework for Deliberation Dialogues**. In McBurney, P., Parsons, S., and Rahwan, I., editors, *Proceedings of*

the 7th International Workshop on Argumentation in Multi-Agent Systems (ArgMAS2010), volume 6614 of *Lecture Notes in Computer Science*, pages 31–48, Toronto, Canada. Springer.

Professional experience

Dec 2008 – Dec 2012	**PhD candidate in Computer Science** Utrecht University, Intelligent Systems Group
Sep 2008 – Nov 2008	**Technical Software Engineer** Deltares (Delft) as contractor for Alten Nederland (Rotterdam)
Sep 2007 – Jun 2008	**Student Assistent** Utrecht University, Intelligent Systems Group
Nov 2003 – Jun 2007	**Software Engineer** MostWare Automatisering (Leiden)

Education

Sep 2006 – Jul 2008	**Master in Agent Technology** Utrecht University
Sep 2000 – Jul 2004	**Bachelor in Information and Communication Technology** The Hague University of Applied Sciences

SIKS Dissertation Series

1998

1998-1 **Johan van den Akker** (CWI), DEGAS - An Active, Temporal Database of Autonomous Objects.

1998-2 **Floris Wiesman** (UM), Information Retrieval by Graphically Browsing Meta-Information.

1998-3 **Ans Steuten** (TUD), A Contribution to the Linguistic Analysis of Business Conversations within the Language/Action Perspective.

1998-4 **Dennis Breuker** (UM), Memory versus Search in Games.

1998-5 **E.W. Oskamp** (RUL), Computerondersteuning bij Straftoemeting.

1999

1999-1 **Mark Sloof** (VU), Physiology of Quality Change Modelling; Automated modelling of Quality Change of Agricultural Products.

1999-2 **Rob Potharst** (EUR), Classification using decision trees and neural nets.

1999-3 **Don Beal** (UM), The Nature of Minimax Search.

1999-4 **Jacques Penders** (UM), The practical Art of Moving Physical Objects.

1999-5 **Aldo de Moor** (KUB), Empowering Communities: A Method for the Legitimate User-Driven Specification of Network Information Systems.

1999-6 **Niek J.E. Wijngaards** (VU), Redesign of compositional systems.

1999-7 **David Spelt** (UT), Verification support for object database design.

1999-8 **Jacques H.J. Lenting** (UM), Informed Gambling: Conception and Analysis of a Multi-Agent Mechanism for Discrete Reallocation.

2000

2000-1 **Frank Niessink** (VU), Perspectives on Improving Software Maintenance.

2000-2 **Koen Holtman** (TUE), Prototyping of CMS Storage Management.

2000-3 **Carolien M.T. Metselaar** (UVA), Sociaal-organisatorische gevolgen van kennistechnologie; een procesbenadering en actorperspectief.

2000-4 **Geert de Haan** (VU), ETAG, A Formal Model of Competence Knowledge for User Interface Design.

2000-5 **Ruud van der Pol** (UM), Knowledge-based Query Formulation in Information Retrieval.

2000-6 **Rogier van Eijk** (UU), Programming Languages for Agent Communication.

2000-7 **Niels Peek** (UU), Decision-theoretic Planning of Clinical Patient Management.

2000-8 **Veerle Coup** (EUR), Sensitivity Analyis of Decision-Theoretic Networks.

2000-9 **Florian Waas** (CWI), Principles of Probabilistic Query Optimization.

2000-10 **Niels Nes** (CWI), Image Database Management System Design Considerations, Algorithms and Architecture.

2000-11 **Jonas Karlsson** (CWI), Scalable Distributed Data Structures for Database Management.

2001

2001-1 **Silja Renooij** (UU), Qualitative Approaches to Quantifying Probabilistic Networks.

2001-2 **Koen Hindriks** (UU), Agent Programming Languages: Programming with Mental Models.

2001-3 **Maarten van Someren** (UvA), Learning as problem solving.

2001-4 **Evgueni Smirnov** (UM), Conjunctive and Disjunctive Version Spaces with Instance-Based Boundary Sets.

2001-5 **Jacco van Ossenbruggen** (VU), Processing Structured Hypermedia: A Matter of Style.

2001-6 **Martijn van Welie** (VU), Task-based User Interface Design.

2001-7 **Bastiaan Schonhage** (VU), Diva: Architectural Perspectives on Information Visualization.

2001-8 **Pascal van Eck** (VU), A Compositional Semantic Structure for Multi-Agent Systems Dynamics.

2001-9 **Pieter Jan 't Hoen** (RUL), Towards Distributed Development of Large Object-Oriented Models, Views of Packages as Classes.

2001-10 **Maarten Sierhuis** (UvA), Modeling and Simulating Work Practice BRAHMS: a multiagent modeling and simulation language for work practice analysis and design.

2001-11 **Tom M. van Engers** (VUA), Knowledge Management: The Role of Mental Models in Business Systems Design.

2002

2002-01 **Nico Lassing** (VU), Architecture-Level Modifiability Analysis.

2002-02 **Roelof van Zwol** (UT), Modelling and searching web-based document collections.

2002-03 **Henk Ernst Blok** (UT), Database Optimization Aspects for Information Retrieval.

2002-04 **Juan Roberto Castelo Valdueza** (UU), The Discrete Acyclic Digraph Markov Model in Data Mining.

2002-05 **Radu Serban** (VU), The Private Cyberspace Modeling Electronic Environments inhabited by Privacy-concerned Agents.

2002-06 **Laurens Mommers** (UL), Applied legal epistemology; Building a knowledge-based ontology of the legal domain.

2002-07 **Peter Boncz** (CWI), Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications.

2002-08 **Jaap Gordijn** (VU), Value Based Requirements Engineering: Exploring Innovative E-Commerce Ideas.

2002-09 **Willem-Jan van den Heuvel** (KUB), Integrating Modern Business Applications with Objectified Legacy Systems.

2002-10 **Brian Sheppard** (UM), Towards Perfect Play of Scrabble.

2002-11 **Wouter C.A. Wijngaards** (VU),

Agent Based Modelling of Dynamics: Biological and Organisational Applications.

2002-12 **Albrecht Schmidt** (UVA), Processing XML in Database Systems.

2002-13 **Hongjing Wu** (TUE), A Reference Architecture for Adaptive Hypermedia Applications.

2002-14 **Wieke de Vries** (UU), Agent Interaction: Abstract Approaches to Modelling, Programming and Verifying Multi-Agent Systems.

2002-15 **Rik Eshuis** (UT), Semantics and Verification of UML Activity Diagrams for Workflow Modelling.

2002-16 **Pieter van Langen** (VU), The Anatomy of Design: Foundations, Models and Applications.

2002-17 **Stefan Manegold** (UVA), Understanding, Modeling, and Improving Main-Memory Database Performance.

2003

2003-01 **Heiner Stuckenschmidt** (VU), Onotology-Based Information Sharing In Weakly Structured Environments.

2003-02 **Jan Broersen** (VU), Modal Action Logics for Reasoning About Reactive Systems.

2003-03 **Martijn Schuemie** (TUD), Human-Computer Interaction and Presence in Virtual Reality Exposure Therapy.

2003-04 **Milan Petkovic** (UT), Content-Based Video Retrieval Supported by Database Technology.

2003-05 **Jos Lehmann** (UVA), Causation in Artificial Intelligence and Law - A modelling approach.

2003-06 **Boris van Schooten** (UT), Development and specification of virtual environments.

2003-07 **Machiel Jansen** (UvA), Formal Explorations of Knowledge Intensive Tasks.

2003-08 **Yongping Ran** (UM), Repair Based Scheduling.

2003-09 **Rens Kortmann** (UM), The resolution of visually guided behaviour.

2003-10 **Andreas Lincke** (UvT), Electronic Business Negotiation: Some experimental studies on the interaction between medium, innovation context and culture.

2003-11 **Simon Keizer** (UT), Reasoning under Uncertainty in Natural Language Dialogue using Bayesian Networks.

2003-12 **Roeland Ordelman** (UT), Dutch speech recognition in multimedia information retrieval.

2003-13 **Jeroen Donkers** (UM), Nosce Hostem - Searching with Opponent Models.

2003-14 **Stijn Hoppenbrouwers** (KUN), Freezing Language: Conceptualisation Processes across ICT-Supported Organisations.

2003-15 **Mathijs de Weerdt** (TUD), Plan Merging in Multi-Agent Systems.

2003-16 **Menzo Windhouwer** (CWI), Feature Grammar Systems - Incremental Maintenance of Indexes to Digital Media Warehouses.

2003-17 **David Jansen** (UT), Extensions of Statecharts with Probability, Time, and Stochastic Timing.

2003-18 **Levente Kocsis** (UM), Learning Search Decisions.

2004

2004-01 **Virginia Dignum** (UU), A Model for Organizational Interaction: Based on Agents, Founded in Logic.

2004-02 **Lai Xu** (UvT), Monitoring Multi-party Contracts for E-business.

2004-03 **Perry Groot** (VU), A Theoretical and Empirical Analysis of Approximation in Symbolic Problem Solving.

2004-04 **Chris van Aart** (UVA), Organizational Principles for Multi-Agent Architectures.

2004-05 **Viara Popova** (EUR), Knowledge discovery and monotonicity.

2004-06 **Bart-Jan Hommes** (TUD), The Evaluation of Business Process Modeling Techniques.

2004-07 **Elise Boltjes** (UM), Voorbeeldig onderwijs; voorbeeldgestuurd onderwijs, een opstap naar abstract denken, vooral voor meisjes.

2004-08 **Joop Verbeek** (UM), Politie en de Nieuwe Internationale Informatiemarkt, Grensregionale politiële gegevensuitwisseling en digitale expertise.

2004-09 **Martin Caminada** (VU), For the Sake of the Argument; explorations into argument-based reasoning.

2004-10 **Suzanne Kabel** (UVA), Knowledge-rich indexing of learning-objects.

2004-11 **Michel Klein** (VU), Change Management for Distributed Ontologies.

2004-12 **The Duy Bui** (UT), Creating emotions and facial expressions for embodied agents.

2004-13 **Wojciech Jamroga** (UT), Using Multiple Models of Reality: On Agents who Know how to Play.

2004-14 **Paul Harrenstein** (UU), Logic in Conflict. Logical Explorations in Strategic Equilibrium.

2004-15 **Arno Knobbe** (UU), Multi-Relational Data Mining.

2004-16 **Federico Divina** (VU), Hybrid Genetic Relational Search for Inductive Learning.

2004-17 **Mark Winands** (UM), Informed Search in Complex Games.

2004-18 **Vania Bessa Machado** (UvA), Supporting the Construction of Qualitative Knowledge Models.

2004-19 **Thijs Westerveld** (UT), Using generative probabilistic models for multimedia retrieval.

2004-20 **Madelon Evers** (Nyenrode), Learning from Design: facilitating multidisciplinary design teams.

2005

2005-01 **Floor Verdenius** (UVA), Methodological Aspects of Designing Induction-Based Applications.

2005-02 **Erik van der Werf** (UM), AI techniques for the game of Go.

2005-03 **Franc Grootjen** (RUN), A Pragmatic Approach to the Conceptualisation of Language.

2005-04 **Nirvana Meratnia** (UT), Towards Database Support for Moving Object data.

2005-05 **Gabriel Infante-Lopez** (UVA), Two-Level Probabilistic Grammars for Natural Language Parsing.

2005-06 **Pieter Spronck** (UM), Adaptive Game AI.

2005-07 **Flavius Frasincar** (TUE), Hypermedia Presentation Generation for Semantic Web Information Systems.

2005-08 **Richard Vdovjak** (TUE), A Model-driven Approach for Building Distributed Ontology-based Web Applications.

2005-09 **Jeen Broekstra** (VU), Storage, Querying and Inferencing for Semantic Web Languages.

2005-10 **Anders Bouwer** (UVA), Explaining Behaviour: Using Qualitative Simulation in Interactive Learning Environments.

2005-11 **Elth Ogston** (VU), Agent Based Matchmaking and Clustering - A Decentralized Approach to Search.

2005-12 **Csaba Boer** (EUR), Distributed Simulation in Industry.

2005-13 **Fred Hamburg** (UL), Een Computermodel voor het Ondersteunen van Euthanasiebeslissingen.

2005-14 **Borys Omelayenko** (VU), Web-Service configuration on the Semantic Web; Exploring how semantics meets pragmatics.

2005-15 **Tibor Bosse** (VU), Analysis of the Dynamics of Cognitive Processes.

2005-16 **Joris Graaumans** (UU), Usability of XML Query Languages.

2005-17 **Boris Shishkov** (TUD), Software Specification Based on Re-usable Business Components.

2005-18 **Danielle Sent** (UU), Test-selection strategies for probabilistic networks.

2005-19 **Michel van Dartel** (UM), Situated Representation.

2005-20 **Cristina Coteanu** (UL), Cyber Consumer Law, State of the Art and Perspectives.

2005-21 **Wijnand Derks** (UT), Improving Concurrency and Recovery in Database Systems by Exploiting Application Semantics.

2006

2006-01 **Samuil Angelov** (TUE), Foundations of B2B Electronic Contracting.

2006-02 **Cristina Chisalita** (VU), Contextual issues in the design and use of information technology in organizations.

2006-03 **Noor Christoph** (UVA), The role of metacognitive skills in learning to solve problems.

2006-04 **Marta Sabou** (VU), Building Web Service Ontologies.

2006-05 **Cees Pierik** (UU), Validation Techniques for Object-Oriented Proof Outlines.

2006-06 **Ziv Baida** (VU), Software-aided Service Bundling – Intelligent Methods & Tools for Graphical Service Modeling.

2006-07 **Marko Smiljanic** (UT), XML schema matching – balancing efficiency and effectiveness by means of clustering.

2006-08 **Eelco Herder** (UT), Forward, Back and Home Again – Analyzing User Behavior on the Web.

2006-09 **Mohamed Wahdan** (UM), Automatic Formulation of the Auditor's Opinion.

2006-10 **Ronny Siebes** (VU), Semantic Routing in Peer-to-Peer Systems.

2006-11 **Joeri van Ruth** (UT), Flattening Queries over Nested Data Types.

2006-12 **Bert Bongers** (VU), Interactivation – Towards an e-cology of people, our technological environment, and the arts.

2006-13 **Henk-Jan Lebbink** (UU), Dialogue and Decision Games for Information Exchanging Agents.

2006-14 **Johan Hoorn** (VU), Software Requirements: Update, Upgrade, Redesign - towards a Theory of Requirements Change.

2006-15 **Rainer Malik** (UU), CONAN: Text Mining in the Biomedical Domain.

2006-16 **Carsten Riggelsen** (UU), Approximation Methods for Efficient Learning of Bayesian Networks.

2006-17 **Stacey Nagata** (UU), User Assistance for Multitasking with Interruptions on a Mobile Device.

2006-18 **Valentin Zhizhkun** (UVA), Graph transformation for Natural Language Processing.

2006-19 **Birna van Riemsdijk** (UU), Cognitive Agent Programming: A Semantic Approach.

2006-20 **Marina Velikova** (UvT), Monotone models for prediction in data mining.

2006-21 **Bas van Gils** (RUN), Aptness on the Web.

2006-22 **Paul de Vrieze** (RUN), Fundaments of Adaptive Personalisation.

2006-23 **Ion Juvina** (UU), Development of Cognitive Model for Navigating on the Web.

2006-24 **Laura Hollink** (VU), Semantic Annotation for Retrieval of Visual Resources.

2006-25 **Madalina Drugan** (UU), Conditional log-likelihood MDL and Evolutionary MCMC.

2006-26 **Vojkan Mihajlovic** (UT), Score Region Algebra: A Flexible Framework for Structured Information Retrieval.

2006-27 **Stefano Bocconi** (CWI), Vox Populi: generating video documentaries from semantically annotated media repositories.

2006-28 **Borkur Sigurbjornsson** (UVA), Focused Information Access using XML Element Retrieval.

2007

2007-01 **Kees Leune** (UvT), Access Control and Service-Oriented Architectures.

2007-02 **Wouter Teepe** (RUG), Reconciling Information Exchange and Confidentiality: A Formal Approach.

2007-03 **Peter Mika** (VU), Social Networks and the Semantic Web.

2007-04 **Jurriaan van Diggelen** (UU), Achieving Semantic Interoperability in Multi-agent Systems: A Dialogue-based Approach.

2007-05 **Bart Schermer** (UL), Software Agents, Surveillance, and the Right to Privacy: a Legislative Framework for Agent-enabled Surveillance.

2007-06 **Gilad Mishne** (UVA), Applied Text Analytics for Blogs.

2007-07 **Natasa Jovanovic** (UT), To Who It May Concern - Addressee Identification in Face-to-Face Meetings.

2007-08 **Mark Hoogendoorn** (VU), Modeling of Change in Multi-Agent Organizations.

2007-09 **David Mobach** (VU), Agent-Based Mediated Service Negotiation.

2007-10 **Huib Aldewereld** (UU), Autonomy vs. Conformity: an Institutional Perspective on Norms and Protocols.

2007-11 **Natalia Stash** (TUE), Incorporating Cognitive/Learning Styles in a General-Purpose Adaptive Hypermedia System.

2007-12 **Marcel van Gerven** (RUN), Bayesian Networks for Clinical Decision Support: A Rational Approach to Dynamic Decision-Making under Uncertainty.

2007-13 **Rutger Rienks** (UT), Meetings in Smart Environments; Implications of Progressing Technology.

2007-14 **Niek Bergboer** (UM), Context-Based Image Analysis.

2007-15 **Joyca Lacroix** (UM), NIM: a Situated Computational Memory Model.

2007-16 **Davide Grossi** (UU), Designing Invisible Handcuffs. Formal investigations in Institutions and Organizations for Multi-agent Systems.

2007-17 **Theodore Charitos** (UU), Reasoning with Dynamic Networks in Practice.

2007-18 **Bart Orriens** (UvT), On the development an management of adaptive business collaborations.

2007-19 **David Levy** (UM), Intimate relationships with artificial partners.

2007-20 **Slinger Jansen** (UU), Customer Configuration Updating in a Software Supply Network.

2007-21 **Karianne Vermaas** (UU), Fast diffusion and broadening use: A research on residential adoption and usage of broadband internet in the Netherlands between 2001 and 2005.

2007-22 **Zlatko Zlatev** (UT), Goal-oriented design of value and process models from patterns.

2007-23 **Peter Barna** (TUE), Specification of Application Logic in Web Information Systems.

2007-24 **Georgina Ramírez Camps** (CWI), Structural Features in XML Retrieval.

2007-25 **Joost Schalken** (VU), Empirical Investigations in Software Process Improvement.

2008

2008-01 **Katalin Boer-Sorbán** (EUR), Agent-Based Simulation of Financial Markets: A modular,continuous-time approach.

2008-02 **Alexei Sharpanskykh** (VU), On Computer-Aided Methods for Modeling and Analysis of Organizations.

2008-03 **Vera Hollink** (UVA), Optimizing hierarchical menus: a usage-based approach.

2008-04 **Ander de Keijzer** (UT), Management of Uncertain Data - towards unattended integration.

2008-05 **Bela Mutschler** (UT), Modeling and simulating causal dependencies on process-aware information systems from a cost perspective.

2008-06 **Arjen Hommersom** (RUN), On the Application of Formal Methods to Clinical Guidelines, an Artificial Intelligence Perspective.

2008-07 **Peter van Rosmalen** (OU), Supporting the tutor in the design and support of adaptive e-learning.

2008-08 **Janneke Bolt** (UU), Bayesian Networks: Aspects of Approximate Inference.

2008-09 **Christof van Nimwegen** (UU), The paradox of the guided user: assistance can be counter-effective.

2008-10 **Wauter Bosma** (UT), Discourse oriented summarization.

2008-11 **Vera Kartseva** (VU), Designing Controls for Network Organizations: A Value-Based Approach.

2008-12 **Jozsef Farkas** (RUN), A Semiotically Oriented Cognitive Model of Knowledge Representation.

2008-13 **Caterina Carraciolo** (UVA), Topic Driven Access to Scientific Handbooks.

2008-14 **Arthur van Bunningen** (UT), Context-Aware Querying; Better Answers with Less Effort.

2008-15 **Martijn van Otterlo** (UT), The Logic of Adaptive Behavior: Knowledge Representation and Algorithms for the Markov Decision Process Framework in First-Order Domains.

2008-16 **Henriette van Vugt** (VU), Embodied agents from a user's perspective.

2008-17 **Martin Op 't Land** (TUD), Applying Architecture and Ontology to the Splitting and Allying of Enterprises.

2008-18 **Guido de Croon** (UM), Adaptive Active Vision.

2008-19 **Henning Rode** (UT), From Document to Entity Retrieval: Improving Precision and Performance of Focused Text Search.

2008-20 **Rex Arendsen** (UVA), Geen bericht, goed bericht. Een onderzoek naar de effecten van de introductie van elektronisch berichtenverkeer met de overheid op de administratieve lasten van bedrijven.

2008-21 **Krisztian Balog** (UVA), People Search in the Enterprise.

2008-22 **Henk Koning** (UU), Communication of IT-Architecture.

2008-23 **Stefan Visscher** (UU), Bayesian network models for the management of ventilator-associated pneumonia.

2008-24 **Zharko Aleksovski** (VU), Using background knowledge in ontology matching.

2008-25 **Geert Jonker** (UU), Efficient and Equitable Exchange in Air Traffic Management Plan Repair using Spender-signed Currency.

2008-26 **Marijn Huijbregts** (UT), Segmentation, Diarization and Speech Transcription: Surprise Data Unraveled.

2008-27 **Hubert Vogten** (OU), Design and Implementation Strategies for IMS Learning Design.

2008-28 **Ildiko Flesch** (RUN), On the Use of Independence Relations in Bayesian Networks.

2008-29 **Dennis Reidsma** (UT), Annotations and Subjective Machines - Of Annotators, Embodied Agents, Users, and Other Humans.

2008-30 **Wouter van Atteveldt** (VU), Semantic Network Analysis: Techniques for Extracting, Representing and Querying Media Content.

2008-31 **Loes Braun** (UM), Pro-Active Medical Information Retrieval.

2008-32 **Trung H. Bui** (UT), Toward Affective Dialogue Management using Partially Observable Markov Decision Processes.

2008-33 **Frank Terpstra** (UVA), Scientific Workflow Design; theoretical and practical issues.

2008-34 **Jeroen de Knijf** (UU), Studies in Frequent Tree Mining.

2008-35 **Ben Torben Nielsen** (UvT), Dendritic morphologies: function shapes structure.

2009

2009-01 **Rasa Jurgelenaite** (RUN), Symmetric Causal Independence Models.

2009-02 **Willem Robert van Hage** (VU), Evaluating Ontology-Alignment Techniques.

2009-03 **Hans Stol** (UvT), A Framework for Evidence-based Policy Making Using IT.

2009-04 **Josephine Nabukenya** (RUN), Improving the Quality of Organisational Policy Making using Collaboration Engineering.

2009-05 **Sietse Overbeek** (RUN), Bridging Supply and Demand for Knowledge Intensive Tasks - Based on Knowledge, Cognition, and Quality.

2009-06 **Muhammad Subianto** (UU), Understanding Classification.

2009-07 **Ronald Poppe** (UT), Discriminative Vision-Based Recovery and Recognition of Human Motion.

2009-08 **Volker Nannen** (VU), Evolutionary Agent-Based Policy Analysis in Dynamic Environments.

2009-09 **Benjamin Kanagwa** (RUN), Design, Discovery and Construction of Service-oriented Systems.

2009-10 **Jan Wielemaker** (UVA), Logic programming for knowledge-intensive interactive applications.

2009-11 **Alexander Boer** (UVA), Legal Theory, Sources of Law & the Semantic Web.

2009-12 **Peter Massuthe** (TUE, Humboldt-Universitaet zu Berlin), Perating Guidelines for Services.

2009-13 **Steven de Jong** (UM), Fairness in Multi-Agent Systems.

2009-14 **Maksym Korotkiy** (VU), From ontology-enabled services to service-enabled ontologies. making ontologies work in e-science with ONTO-SOA

2009-15 **Rinke Hoekstra** (UVA), Ontology Representation - Design Patterns and Ontologies that Make Sense.

2009-16 **Fritz Reul** (UvT), New Architectures in Computer Chess.

2009-17 **Laurens van der Maaten** (UvT), Feature Extraction from Visual Data.

2009-18 **Fabian Groffen** (CWI), Armada, An Evolving Database System.

2009-19 **Valentin Robu** (CWI), Modeling Preferences, Strategic Reasoning and Collaboration in Agent-Mediated Electronic Mar-

kets.

2009-20 **Bob van der Vecht** (UU), Adjustable Autonomy: Controling Influences on Decision Making.

2009-21 **Stijn Vanderlooy** (UM), Ranking and Reliable Classification.

2009-22 **Pavel Serdyukov** (UT), Search For Expertise: Going beyond direct evidence.

2009-23 **Peter Hofgesang** (VU), Modelling Web Usage in a Changing Environment.

2009-24 **Annerieke Heuvelink** (VUA), Cognitive Models for Training Simulations.

2009-25 **Alex van Ballegooij** (CWI), "RAM: Array Database Management through Relational Mapping".

2009-26 **Fernando Koch** (UU), An Agent-Based Model for the Development of Intelligent Mobile Services.

2009-27 **Christian Glahn** (OU), Contextual Support of Social Engagement and Reflection on the Web.

2009-28 **Sander Evers** (UT), Sensor Data Management with Probabilistic Models.

2009-29 **Stanislav Pokraev** (UT), Model-Driven Semantic Integration of Service-Oriented Applications.

2009-30 **Marcin Zukowski** (CWI), Balancing vectorized query execution with bandwidth-optimized storage.

2009-31 **Sofiya Katrenko** (UVA), A Closer Look at Learning Relations from Text.

2009-32 **Rik Farenhorst and Remco de Boer** (VU), Architectural Knowledge Management: Supporting Architects and Auditors.

2009-33 **Khiet Truong** (UT), How Does Real Affect Affect Affect Recognition In Speech?.

2009-34 **Inge van de Weerd** (UU), Advancing in Software Product Management: An Incremental Method Engineering Approach.

2009-35 **Wouter Koelewijn** (UL), Privacy en Politiegegevens; Over geautomatiseerde normatieve informatie-uitwisseling.

2009-36 **Marco Kalz** (OUN), Placement Support for Learners in Learning Networks.

2009-37 **Hendrik Drachsler** (OUN), Navigation Support for Learners in Informal Learning Networks.

2009-38 **Riina Vuorikari** (OU), Tags and self-organisation: a metadata ecology for learning resources in a multilingual context.

2009-39 **Christian Stahl** (TUE, Humboldt-Universitaet zu Berlin), Service Substitution – A Behavioral Approach Based on Petri Nets.

2009-40 **Stephan Raaijmakers** (UvT), Multinomial Language Learning: Investigations into the Geometry of Language.

2009-41 **Igor Berezhnyy** (UvT), Digital Analysis of Paintings.

2009-42 **Toine Bogers** (UvT), Recommender Systems for Social Bookmarking.

2009-43 **Virginia Nunes Leal Franqueira** (UT), Finding Multi-step Attacks in Computer Networks using Heuristic Search and Mobile Ambients.

2009-44 **Roberto Santana Tapia** (UT), Assessing Business-IT Alignment in Networked Organizations.

2009-45 **Jilles Vreeken** (UU), Making Pattern Mining Useful.

2009-46 **Loredana Afanasiev** (UvA), Querying XML: Benchmarks and Recursion.

2010

2010-01 **Matthijs van Leeuwen** (UU), Patterns that Matter.

2010-02 **Ingo Wassink** (UT), Work flows in Life Science.

2010-03 **Joost Geurts** (CWI), A Document Engineering Model and Processing Frame-

work for Multimedia documents.

2010-04 **Olga Kulyk** (UT), Do You Know What I Know? Situational Awareness of Co-located Teams in Multidisplay Environments.

2010-05 **Claudia Hauff** (UT), Predicting the Effectiveness of Queries and Retrieval Systems.

2010-06 **Sander Bakkes** (UvT), Rapid Adaptation of Video Game AI.

2010-07 **Wim Fikkert** (UT), A Gesture interaction at a Distance.

2010-08 **Krzysztof Siewicz** (UL), Towards an Improved Regulatory Framework of Free Software. Protecting user freedoms in a world of software communities and eGovernments.

2010-09 **Hugo Kielman** (UL), Politiële gegevensverwerking en Privacy, Naar een effectieve waarborging.

2010-10 **Rebecca Ong** (UL), Mobile Communication and Protection of Children.

2010-11 **Adriaan Ter Mors** (TUD), The world according to MARP: Multi-Agent Route Planning.

2010-12 **Susan van den Braak** (UU), Sensemaking software for crime analysis.

2010-13 **Gianluigi Folino** (RUN), High Performance Data Mining using Bio-inspired techniques.

2010-14 **Sander van Splunter** (VU), Automated Web Service Reconfiguration.

2010-15 **Lianne Bodenstaff** (UT), Managing Dependency Relations in Inter-Organizational Models.

2010-16 **Sicco Verwer** (TUD), Efficient Identification of Timed Automata, theory and practice.

2010-17 **Spyros Kotoulas** (VU), Scalable Discovery of Networked Resources: Algorithms, Infrastructure, Applications.

2010-18 **Charlotte Gerritsen** (VU), Caught in the Act: Investigating Crime by Agent-Based Simulation.

2010-19 **Henriette Cramer** (UvA), People's Responses to Autonomous and Adaptive Systems.

2010-20 **Ivo Swartjes** (UT), Whose Story Is It Anyway? How Improv Informs Agency and Authorship of Emergent Narrative.

2010-21 **Harold van Heerde** (UT), Privacy-aware data management by means of data degradation.

2010-22 **Michiel Hildebrand** (CWI), End-user Support for Access to Heterogeneous Linked Data.

2010-23 **Bas Steunebrink** (UU), The Logical Structure of Emotions.

2010-24 **Dmytro Tykhonov** (), Designing Generic and Efficient Negotiation Strategies.

2010-25 **Zulfiqar Ali Memon** (VU), Modelling Human-Awareness for Ambient Agents: A Human Mindreading Perspective.

2010-26 **Ying Zhang** (CWI), XRPC: Efficient Distributed Query Processing on Heterogeneous XQuery Engines.

2010-27 **Marten Voulon** (UL), Automatisch contracteren.

2010-28 **Arne Koopman** (UU), Characteristic Relational Patterns.

2010-29 **Stratos Idreos** (CWI), Database Cracking: Towards Auto-tuning Database Kernels.

2010-30 **Marieke van Erp** (UvT), Accessing Natural History - Discoveries in data cleaning, structuring, and retrieval.

2010-31 **Victor de Boer** (UVA), Ontology Enrichment from Heterogeneous Sources on the Web.

2010-32 **Marcel Hiel** (UvT), An Adaptive Service Oriented Architecture: Automatically solving Interoperability Problems.

2010-33 **Robin Aly** (UT), Modeling Repre-

sentation Uncertainty in Concept-Based Multimedia Retrieval.

2010-34 **Teduh Dirgahayu** (UT), Interaction Design in Service Compositions.

2010-35 **Dolf Trieschnigg** (UT), Proof of Concept: Concept-based Biomedical Information Retrieval.

2010-36 **Jose Janssen** (OU), Paving the Way for Lifelong Learning; Facilitating competence development through a learning path specification.

2010-37 **Niels Lohmann** (TUE), Correctness of services and their composition.

2010-38 **Dirk Fahland** (TUE), From Scenarios to components.

2010-39 **Ghazanfar Farooq Siddiqui** (VU), Integrative modeling of emotions in virtual agents.

2010-40 **Mark van Assem** (VU), Converting and Integrating Vocabularies for the Semantic Web.

2010-41 **Guillaume Chaslot** (UM), Monte-Carlo Tree Search.

2010-42 **Sybren de Kinderen** (VU), Needs-driven service bundling in a multi-supplier setting - the computational e3-service approach.

2010-43 **Peter van Kranenburg** (UU), A Computational Approach to Content-Based Retrieval of Folk Song Melodies.

2010-44 **Pieter Bellekens** (TUE), An Approach towards Context-sensitive and User-adapted Access to Heterogeneous Data Sources, Illustrated in the Television Domain.

2010-45 **Vasilios Andrikopoulos** (UvT), A theory and model for the evolution of software services.

2010-46 **Vincent Pijpers** (VU), e3alignment: Exploring Inter-Organizational Business-ICT Alignment.

2010-47 **Chen Li** (UT), Mining Process Model Variants: Challenges, Techniques, Examples.

2010-48 **Milan Lovric** (EUR), Behavioral Finance and Agent-Based Artificial Markets.

2010-49 **Jahn-Takeshi Saito** (UM), Solving difficult game positions.

2010-50 **Bouke Huurnink** (UVA), Search in Audiovisual Broadcast Archives.

2010-51 **Alia Khairia Amin** (CWI), Understanding and supporting information seeking tasks in multiple sources.

2010-52 **Peter-Paul van Maanen** (VU), Adaptive Support for Human-Computer Teams: Exploring the Use of Cognitive Models of Trust and Attention.

2010-53 **Edgar Meij** (UVA), Combining Concepts and Language Models for Information Access.

2011

2011-01 **Botond Cseke** (RUN), Variational Algorithms for Bayesian Inference in Latent Gaussian Models.

2011-02 **Nick Tinnemeier** (UU), Organizing Agent Organizations. Syntax and Operational Semantics of an Organization-Oriented Programming Language.

2011-03 **Jan Martijn van der Werf** (TUE), Compositional Design and Verification of Component-Based Information Systems.

2011-04 **Hado van Hasselt** (UU), Insights in Reinforcement Learning - Formal analysis and empirical evaluation of temporal-difference learning algorithms.

2011-05 **Base van der Raadt** (VU), Enterprise Architecture Coming of Age - Increasing the Performance of an Emerging Discipline.

2011-06 **Yiwen Wang** (TUE), Semantically-Enhanced Recommendations in Cultural Heritage.

2011-07 **Yujia Cao** (UT), Multimodal Information Presentation for High Load Human Computer Interaction.

2011-08 **Nieske Vergunst** (UU), BDI-based Generation of Robust Task-Oriented Dialogues.

2011-09 **Tim de Jong** (OU), Contextualised Mobile Media for Learning.

2011-10 **Bart Bogaert** (UvT), Cloud Content Contention.

2011-11 **Dhaval Vyas** (UT), Designing for Awareness: An Experience-focused HCI Perspective.

2011-12 **Carmen Bratosin** (TUE), Grid Architecture for Distributed Process Mining.

2011-13 **Xiaoyu Mao** (UvT), Airport under Control; Multiagent Scheduling for Airport Ground Handling.

2011-14 **Milan Lovric** (EUR), Behavioral Finance and Agent-Based Artificial Markets.

2011-15 **Marijn Koolen** (UVA), The Meaning of Structure: the Value of Link Evidence for Information Retrieval.

2011-16 **Maarten Schadd** (UM), Selective Search in Games of Different Complexity.

2011-17 **Jiyin He** (UVA), Exploring Topic Structure: Coherence, Diversity and Relatedness.

2011-18 **Mark Ponsen** (UM), Strategic Decision-Making in complex games.

2011-19 **Ellen Rusman** (OU), The Mind's Eye on Personal Profiles.

2011-20 **Qing Gu** (VU), Guiding service-oriented software engineering - A view-based approach.

2011-21 **Linda Terlouw** (TUD), Modularization and Specification of Service-Oriented Systems.

2011-22 **Junte Zhang** (UVA), System Evaluation of Archival Description and Access.

2011-23 **Wouter Weerkamp** (UVA), Finding People and their Utterances in Social Media.

2011-24 **Herwin van Welbergen** (UT), Behavior Generation for Interpersonal Coordination with Virtual Humans On Specifying, Scheduling and Realizing Multimodal Virtual Human Behavior.

2011-25 **Syed Waqar ul Qounain Jaffry** (VU), Analysis and Validation of Models for Trust Dynamics.

2011-26 **Matthijs Aart Pontier** (VU), Virtual Agents for Human Communication.

2011-27 **Aniel Bhulai** (VU), Dynamic website optimization through autonomous management of design patterns.

2011-28 **Rianne Kaptein** (UVA), Effective Focused Retrieval by Exploiting Query Context and Document Structure.

2011-29 **Faisal Kamiran** (TUE), Discrimination-aware Classification.

2011-30 **Egon van den Broek** (UT), Affective Signal Processing (ASP): Unraveling the mystery of emotions.

2011-31 **Ludo Waltman** (EUR), Computational and Game-Theoretic Approaches for Modeling Bounded Rationality.

2011-32 **Nees Jan van Eck** (EUR), Methodological Advances in Bibliometric Mapping of Science.

2011-33 **Tom van der Weide** (UU), Arguing to Motivate Decisions.

2011-34 **Paolo Turrini** (UU), Strategic Reasoning in Interdependence: Logical and Game-theoretical Investigations.

2011-35 **Maaike Harbers** (UU), Explaining Agent Behavior in Virtual Training.

2011-36 **Erik van der Spek** (UU), Experiments in serious game design: a cognitive approach.

2011-37 **Adriana Burlutiu** (RUN), Machine Learning for Pairwise Data, Applications for

Preference Learning and Supervised Network Inference.

2011-38 **Nyree Lemmens** (UM), Bee-inspired Distributed Optimization.

2011-39 **Joost Westra** (UU), Organizing Adaptation using Agents in Serious Games.

2011-40 **Viktor Clerc** (VU), Architectural Knowledge Management in Global Software Development.

2011-41 **Luan Ibraimi** (UT), Cryptographically Enforced Distributed Data Access Control.

2011-42 **Michal Sindlar** (UU), Explaining Behavior through Mental State Attribution.

2011-43 **Henk van der Schuur** (UU), Process Improvement through Software Operation Knowledge.

2011-44 **Boris Reuderink** (UT), Robust Brain-Computer Interfaces.

2011-45 **Herman Stehouwer** (UvT), Statistical Language Models for Alternative Sequence Selection.

2011-46 **Beibei Hu** (TUD), Towards Contextualized Information Delivery: A Rule-based Architecture for the Domain of Mobile Police Work.

2011-47 **Azizi Bin Ab Aziz(VU** (Exploring Computational Models for Intelligent Support of Persons with Depression),

. 2011-48 **Mark Ter Maat** (UT), Response Selection and Turn-taking for a Sensitive Artificial Listening Agent.

2011-49 **Andreea Niculescu** (UT), Conversational interfaces for task-oriented spoken dialogues: design aspects influencing interaction quality.

2012

2012-01 **Terry Kakeeto** (UvT), Relationship Marketing for SMEs in Uganda.

2012-02 **Muhammad Umair(VU** (Adaptivity, emotion, and Rationality in Human and Ambient Agent Models),

. 2012-03 **Adam Vanya** (VU), Supporting Architecture Evolution by Mining Software Repositories.

2012-04 **Jurriaan Souer** (UU), Development of Content Management System-based Web Applications.

2012-05 **Marijn Plomp** (UU), Maturing Interorganisational Information Systems.

2012-06 **Wolfgang Reinhardt** (OU), Awareness Support for Knowledge Workers in Research Networks.

2012-07 **Rianne van Lambalgen** (VU), When the Going Gets Tough: Exploring Agent-based Models of Human Performance under Demanding Conditions.

2012-08 **Gerben de Vries** (UVA), Kernel Methods for Vessel Trajectories.

2012-09 **Ricardo Neisse** (UT), Trust and Privacy Management Support for Context-Aware Service Platforms.

2012-10 **David Smits** (TUE), Towards a Generic Distributed Adaptive Hypermedia Environment.

2012-11 **J.C.B. Rantham Prabhakara** (TUE), Process Mining in the Large: Preprocessing, Discovery, and Diagnostics.

2012-12 **Kees van der Sluijs** (TUE), Model Driven Design and Data Integration in Semantic Web Information Systems.

2012-13 **Suleman Shahid** (UvT), Fun and Face: Exploring non-verbal expressions of emotion during playful interactions.

2012-14 **Evgeny Knutov(TUE** (Generic Adaptation Framework for Unifying Adaptive Web-based Systems),

. 2012-15 **Natalie van der Wal** (VU), Social Agents. Agent-Based Modelling of Integrated Internal and Social Dynamics of Cognitive and Affective Processes.

2012-16 **Fiemke Both** (VU), Helping people by understanding them - Ambient Agents supporting task execution and depression treatment.

2012-17 **Amal Elgammal** (UvT), Towards a Comprehensive Framework for Business Process Compliance.

2012-18 **Eltjo Poort** (VU), Improving Solution Architecting Practices.

2012-19 **Helen Schonenberg** (TUE), What's Next? Operational Support for Business Process Execution.

2012-20 **Ali Bahramisharif** (RUN), Covert Visual Spatial Attention, a Robust Paradigm for Brain-Computer Interfacing.

2012-21 **Roberto Cornacchia** (TUD), Querying Sparse Matrices for Information Retrieval.

2012-22 **Thijs Vis** (UvT), Intelligence, politie en veiligheidsdienst: verenigbare grootheden?.

2012-23 **Christian Muehl** (UT), Toward Affective Brain-Computer Interfaces: Exploring the Neurophysiology of Affect during Human Media Interaction.

2012-24 **Laurens van der Werff** (UT), Evaluation of Noisy Transcripts for Spoken Document Retrieval.

2012-25 **Silja Eckartz** (UT), Managing the Business Case Development in Inter-Organizational IT Projects: A Methodology and its Application.

2012-26 **Emile de Maat** (UVA), Making Sense of Legal Text.

2012-27 **Hayrettin Gurkok** (UT), Mind the Sheep! User Experience Evaluation & Brain-Computer Interface Games.

2012-28 **Nancy Pascall** (UvT), Engendering Technology Empowering Women.

2012-29 **Almer Tigelaar** (UT), Peer-to-Peer Information Retrieval.

2012-30 **Alina Pommeranz** (TUD), Designing Human-Centered Systems for Reflective Decision Making.

2012-31 **Emily Bagarukayo** (RUN), A Learning by Construction Approach for Higher Order Cognitive Skills Improvement, Building Capacity and Infrastructure.

2012-32 **Wietske Visser** (TUD), Qualitative multi-criteria preference representation and reasoning.

2012-33 **Rory Sie** (OUN), Coalitions in Cooperation Networks (COCOON).

2012-34 **Pavol Jancura** (RUN), Evolutionary analysis in PPI networks and applications.

2012-35 **Evert Haasdijk** (VU), Never Too Old To Learn – On-line Evolution of Controllers in Swarm- and Modular Robotics .

2012-36 **Denis Ssebugwawo** (RUN), Analysis and Evaluation of Collaborative Modeling Processes.

2012-37 **Agnes Nakakawa** (RUN), A Collaboration Process for Enterprise Architecture Creation.

2012-38 **Selmar Smit** (VU), Parameter Tuning and Scientific Testing in Evolutionary Algorithms.

2012-39 **Hassan Fatemi** (UT), Risk-aware design of value and coordination networks.

2012-40 **Agus Gunawan** (UvT), Information Access for SMEs in Indonesia.

2012-41 **Sebastian Kelle** (OU), Game Design Patterns for Learning.

2012-42 **Dominique Verpoorten** (OU), Reflection Amplifiers in self-regulated Learning.

2012-44 **Anna Tordai** (VU), On Combining Alignment Techniques.

2012-45 **Benedikt Kratz** (UvT), A Model and Language for Business-aware Transactions.

2012-46 **Simon Carter** (UVA), Exploration and Exploitation of Multilingual Data for Statistical Machine Translation.

2012-47 **Manos Tsagkias** (UVA), Mining Social Media: Tracking Content and Predicting Behavior.

2012-48 **Jorn Bakker** (TUE), Handling Abrupt Changes in Evolving Time-series Data.

2012-49 **Michael Kaisers** (UM), Learning against Learning - Evolutionary dynamics of reinforcement learning algorithms in strategic interactions.

2012-50 **Steven van Kervel** (TUD), Ontologogy driven Enterprise Information Systems Engineering.

2012-51 **Jeroen de Jong** (TUD), Heuristics in Dynamic Sceduling; a practical framework with a case study in elevator dispatching.

2013

2013-01 **Viorel Milea** (EUR), News Analytics for Financial Decision Support.

2013-02 **Erietta Liarou** (CWI)), MonetDB/DataCell: Leveraging the Column-store Database Technology for Efficient and Scalable Stream Processing.

2013-03 **Szymon Klarman** (VU), Reasoning with Contexts in Description Logics.

2013-04 **Chetan Yadati** (TUD)), Coordinating autonomous planning and scheduling.

2013-05 **Dulce Pumareja** (UT), Groupware Requirements Evolutions Patterns.

2013-06 **Romulo Gonzalves** (CWI)), The Data Cyclotron: Juggling Data and Queries for a Data Warehouse Audience.

2013-07 **Giel van Lankveld** (UT), Quantifying Individual Player Differences.

2013-08 **Robbert-Jan Merk** (VU)), Making enemies: cognitive modeling for opponent agents in fighter pilot simulators.

2013-09 **Fabio Gori** (RUN), Metagenomic Data Analysis: Computational Methods and Applications.

2013-10 **Jeewanie Jayasinghe Arachchige** (UvT)), A Unified Modeling Framework for Service Design.

2013-11 **Evangelos Pournaras** (TUD), Multi-level Reconfigurable Self-organization in Overlay Services.

2013-12 **Maryam Razavian** (VU), Knowledge-driven Migration to Services.

2013-13 **Mohammad Zafiri** (UT), Service Tailoring: User-centric creation of integrated IT-based homecare services to support independent living of elderly.

2013-14 **Jafar Tanha** (UVA), Ensemble Approaches to Semi-Supervised Learning Learning.

2013-15 **Daniel Hennes** (UM), Multiagent Learning - Dynamic Games and Applications.

2013-16 **Eric Kok** (UU), Exploring the practical benefits of argumentation in multi-agent deliberation.