Project N 002307

ASPIC

Argumentation Service Platform with Integrated Components

Instrument: Specific Targeted Research Project
Thematic Priority: Information Society Technologies, objective 2.3.1.7.

Deliverable D2.6 - Final review and report on formal argumentation system

Due date of deliverable: 31/12/2005
Actual submission date: 15/02/2006

Start date of project: 01/01/2004 Duration: 36 Months

Université Paul Sabatier (UPS)

Version 1.1 - 31/12/2006

Partners:

LogicDIS S.A. (Co-ordinator)	LogicDIS	Greece
Cancer Research UK	CRUK	UK
Zeus Consulting S.A.	ZEUS	Greece
NAVUS	NAVUS	Germany
University of Ljubljana	ULFRI	Slovenia
Technical University of Catalonia	UPC	Spain
Institut de Recherche en Informatique de Toulouse	IRIT	France
University of Surrey	UNiS	UK
University of Liverpool	UNiL	UK
Utrecht University	UU	Netherlands
City University of New York	CUNY University	USA

Distribution List	
ASPIC Consortium	ASPIC Consortium, EC Project Officer

Written by:	Leila Amgoud (IRIT) Lianne Bodenstaff (Utrecht) Martin Caminada (Utrecht) Peter McBurney (Liverpool) Simon Parsons (CUNY) Henry Prakken (Utrecht) Jelle van Veenen (Utrecht) Gerard Vreeswijk (Utrecht)	

Document Change Log

Version #	Issue Date	Sections Affected	Relevant Information
1.1	31/12/2006	UPS, Utrecht, Liverpool	Final document after contribution of partners

Contents

1 Introduction **1**

2 A general argumentation system for inference **3**

2.1 Argumentation process 3
2.2 Logical language 5
2.3 The notion of argument 6
2.4 Comparing arguments 7
2.4.1 Last link principle 8
2.4.2 Weakest link principle 8
2.5 The conflicts between arguments 9
2.6 The acceptability of arguments 10
2.7 The consequence relation 11
2.8 Procedural form 12
2.8.1 A general framework for argument games 13
2.8.2 Credulous reasoning under the preferred semantics 15
2.8.2.1 Linear argument games 16
2.8.2.2 Tree-like argument games 17
2.8.3 Skeptical reasoning under the preferred semantics 18
2.8.3.1 Skeptical reasoning as credulous meta-reasoning 18
2.8.3.2 Argument game 20
2.8.4 Skeptical reasoning under the grounded semantics 22
2.8.5 Algorithms 22
2.8.5.1 Credulous reasoning under the preferred semantics 23
2.8.5.2 Skeptical reasoning under the grounded semantics 25

3 A general argumentation system for decision making **28**

3.1 Introduction 28
3.2 Logical language and the different bases 28
3.3 The arguments 30
3.3.1 Recommending arguments 30
3.3.2 Decision arguments 31
3.4 Comparing arguments 33
3.5 Acceptability of arguments 34
3.6 Decision criteria 35

4 A general system for dialogue 37
4.1 Introduction . . . 37
4.2 A general model for dialogue . . . 39
4.3 A formal framework for persuasion dialogues . . . 40
4.3.1 The framework: general ideas . . . 41
4.3.2 The framework formally defined . . . 41
4.3.3 Liberal dialogue systems . . . 45
4.3.3.1 The communication language . . . 45
4.3.3.2 The commitment rules . . . 46
4.3.3.3 Turntaking . . . 46
4.3.3.4 The protocol . . . 46
4.3.3.5 Termination and outcome of dialogues . . . 47
4.3.4 Protocols for relevant dialogues . . . 49
4.3.4.1 Motivation . . . 49
4.3.4.2 Relevance defined . . . 51
4.3.5 Protocols for weakly relevant dialogues . . . 53
4.3.6 Respecting commitments . . . 54
4.4 Formalising dialogue games with the Event Calculus . . . 55
4.4.1 The Event Calculus . . . 56
4.4.2 Formalisation of a persuasion game . . . 59
4.4.2.1 Additional fluents . . . 59
4.4.2.2 The protocol . . . 60
4.4.2.3 Commitment set . . . 61
4.4.2.4 Turntaking . . . 62
4.5 Another illustration of our approach . . . 63
4.5.1 The PWA system . . . 63
4.5.1.1 Communication Language . . . 63
4.5.1.2 Protocol . . . 63
4.5.1.3 Effect Rules . . . 65
4.5.1.4 Turntaking . . . 65
4.5.1.5 Termination Rules . . . 65
4.5.2 Formalisation of PWA in Event calculus . . . 65
4.5.2.1 Additional fluents . . . 65
4.5.2.2 Begin situation . . . 66
4.5.2.3 Legal moves . . . 68
4.5.2.4 Claiming a set of propositions . . . 70
4.5.2.5 Attitudes . . . 73
4.5.2.6 Turntaking . . . 74
4.5.2.7 Effect of a move on the commitment set . . . 75
4.5.2.8 Ending the dialogue . . . 76
4.6 Argument-based negotiation . . . 76
4.6.1 A language and protocol for multi-attribute negotiation . 77
4.6.2 The combination . . . 79
4.6.3 Properties of the combined protocol . . . 81
4.6.4 An example . . . 82
4.6.5 Adding dialogue policies for the e-Consent Scenario . . . 84

4.6.5.1 Notation 85
4.6.5.2 Negotiation policies 85
4.6.5.3 Persuasion policies 87
4.7 Implementation 88
4.7.1 Persuasion protocol implementation 88
4.7.1.1 The basics 88
4.7.1.2 Implementation 90
4.7.1.3 Running the program 92
4.7.2 Information-seeking protocol implementation 95
4.8 Combining Dialogues 97

5 Conclusion **99**

Chapter 1

Introduction

A rational agent can express claims and judgments, aiming at reaching a decision, a conclusion, or informing, convincing, negotiating with other agents. Pertinent information may be insufficient or contrastedly there may be too much relevant but partially incoherent information. And, in case of multi-agent interaction, conflicts of interest are inevitable. So, agents can be assisted by argumentation, a process based on the exchange and the valuation of interacting arguments which support opinions, claims, proposals, decisions, ...

Argumentation has become an Artificial Intelligence keyword for the last fifteen years, especially in sub-fields such as nonmonotonic reasoning, inconsistency-tolerant reasoning, multiple-source information systems, natural language processing and human-machine interface also in connection with multi-agents systems. There are several introductory survey or collections of papers available on these AI research trends [71, 45, 69].

In [5], the ASPIC consortium has presented an overview of existing argumentation models for inference, decision, dialogue and finally for learning. It is clear from that overview that the issues of inference and decision have, for a long time, been considered as two distinct problems and been studied separately. As a result of this, several models have been proposed for each problem. The basic idea behind inference is to draw conclusions from a set of premisses, in other words, to determine whether a given conclusion can be regarded as justified on the basis of the existing information. The decision problem consists of defining a pre-ordering on a set of possible choices or alternatives, on the basis of available information. Indeed, an alternative cannot be true or false, but it can be preferred to the other alternatives in the current state of the world.

In [4] a first argumentation-based model has been proposed for inference and decision. Two important features of that model can be outlined. The first one is its *generality* compared to existing argumentation systems developed for inference purposes. Indeed, the new ASPIC model is defined on a unspecified logical language. The only requirement is that strict and defeasible rules should be distinguished. The second important feature of the ASPIC model is the fact that both inference and decision are captured and analyzed. In [2] Amgoud has argued that inference is part of a decision problem. The basic idea is to infer from all the available information, the formulas which are "correctly" supported, then to classify the different decisions on the basis of these formulas. Moreover, the proposed framework is general enough to capture different kinds of decision problems such as decision under uncertainty, multiple criteria decision and rule-based decision.

The acceptability semantics used in ASPIC are exactly the ones defined by Dung in [33]. In [4] different proofs of acceptability have been proposed.

When defining the ASPIC-formalism, we discovered some very interesting problems and undesirable results. One may think that this problem is proper to ASPIC model, unfortunately not. Several existing systems such as Prakken and Sartor's system [68], the argument-theoretic version of Nute's Defeasible Logic [42] and Garcia and Simari's system [38] suffer from the same problem. In order to avoid such anomalies, in [21] Caminada and Amgoud have proposed different *principles* (what they call *rationality postulates* or *axioms*) that any argumentation system should fulfill. These postulates will govern the well definition of an argumentation system and will ensure the correctness of its results. Three important postulates have been defined: the *closeness* and the *consistency* of the results that an argumentation system may return, and also the phenomenon of *non-contamination*. These postulates are violated in systems such as [68, 42, 38]. In [21], Caminada and Amgoud have also proposed different solutions in which these postulates can be warranted in the ASPIC model. In [6], all these results have been reported in detail.

The aim of this deliverable D2.6 is to report the solution chosen to be implemented by ASPIC consortium.

Concerning dialogue, in this document we propose a formal model of dialogue instantiating the general formal framework for dialogues presented in Section 4 of ASPIC Deliverable D2.1 [5]. The model clearly shows how the argumentation framework defined in the first part for inference and decision can be used in dialogues. Within the framework of [5] a formal model of persuasion is defined and then combined with a model of negotiation dialogue and a partial design of dialogical agents. Then two possible routes to implementation of the proposed dialogue models are discussed.

The document is organized in three main chapters. Chapter 2 presents a general argumentation framework for inference. Chapter 3 extends the first argumentation framework to deal with decision-making. Chapter 4 is devoted to models of dialogue. The deliverable ends with a short Conclusion in Chapter 5.

Chapter 2

A general argumentation system for inference

2.1 Argumentation process

Argumentation is a reasoning model which follows the five following steps:

1. Constructing *arguments* (in *favor* of / *against* a "statement") from bases.
2. Defining the *strengths* of those arguments.
3. Determining the different *conflicts* between the arguments.
4. Evaluating the *acceptability* of the different arguments.
5. Concluding or defining the *justified conclusions*.

Indeed, argumentation systems are built around an underlying logical language $\mathcal{L}$ and an associated notion of logical consequence, defining the notion of argument. The argument construction is a monotonic process: new knowledge cannot rule out an argument but only gives rise to new arguments which may interact with the first argument. Since the knowledge bases may be inconsistent, the arguments may be conflicting too. Consequently, it is important to determine among all the available arguments, the ones which will be *justified*. In [33], an argumentation system is defined as follows:

Definition 1 (Argumentation system) *An* argumentation system *(AF) is a pair $\langle \mathcal{A},\ Def \rangle$. $\mathcal{A}$ is a set arguments and $Def \subseteq \mathcal{A} \times \mathcal{A}$ is a defeasibility relation. We say that an argument A defeats an argument B iff $(A, B) \in Def$ (or $A\ Def\ B$).*

Among all the conflicting arguments, it is important to know which are the arguments which will be kept for inferring conclusions and for making decisions. In [33], different semantics for the notion of acceptability have been proposed. Let's recall them here.

Definition 2 (Conflict-free, Defence) *Let $\mathcal{B} \subseteq \mathcal{A}$.*

- *A set $\mathcal{B}$ is* conflict-free *iff there exist no A_i, A_j in $\mathcal{B}$ such that A_i Def A_j.*
- *A set $\mathcal{B}$* defends *an argument A_i iff for each argument $A_j \in \mathcal{A}$, if A_j Def A_i, then there exists A_k in $\mathcal{B}$ such that A_k Def A_j.*

Definition 3 (Acceptability semantics) *Let $\mathcal{B}$ be a conflict-free set of arguments, and let $\mathcal{F}$: $2^{\mathcal{A}} \mapsto 2^{\mathcal{A}}$ be a function such that $\mathcal{F}(\mathcal{B}) = \{A \mid \mathcal{B}$ defends $A\}$.*

- *$\mathcal{B}$ is* admissible *iff $\mathcal{B} \subseteq \mathcal{F}(\mathcal{B})$.*
- *$\mathcal{B}$ is a* complete extension *iff $\mathcal{B} = \mathcal{F}(\mathcal{B})$.*
- *$\mathcal{B}$ is a* grounded extension *iff it is the minimal (w.r.t. set-inclusion) complete extension.*
- *$\mathcal{B}$ is a* preferred extension *iff it is a maximal (w.r.t. set-inclusion) complete extension.*
- *$\mathcal{B}$ is a* stable extension *iff it is a preferred extension that defeats all arguments in $\mathcal{A} \backslash \mathcal{B}$.*

Let $\mathcal{E} = \{E_1, \ldots, E_n\}$ be the set of all possible extensions under a given semantics.

Note that there is only one grounded extension. It contains all the arguments which are not defeated and also the arguments which are defended directly or indirectly by non-defeated arguments. Moreover, the following results can be shown:

Lemma 1 *Let $\langle \mathcal{A}, Def \rangle$ be any argumentation framework and $\mathcal{B} \subseteq \mathcal{A}$. If $\mathcal{B}$ is admissible, then $\mathcal{B} \subseteq \mathcal{F}(\mathcal{B})$.*

Lemma 2 *Let $\langle \mathcal{A}, Def \rangle$ be any argumentation framework and $\mathcal{B} \subseteq \mathcal{A}$. If $\mathcal{B}$ is admissible, then $\mathcal{F}(\mathcal{B})$ is also admissible.*

So far, we have defined the acceptability of sets of arguments. In what follows, we propose different status of a single argument.

Definition 4 (Status of an argument) *Let $\langle \mathcal{A}, Def \rangle$ be an argumentation framework, $\mathcal{E} = \{E_1, \ldots, E_n\}$ its possible extensions under a given semantics. Let A be an argument.*

- *A is* skeptically accepted *under a given semantics iff A belongs to all the extensions under the semantics.*
- *A is* credulously accepted *under a given semantics iff A belongs to at least one extension under the semantics.*

- *A is* rejected *under a given semantics iff A does not belong to any extension under the semantics.*

The last step of an argumentation process consists of determining, among all the conclusions of the different arguments, the "good" ones called *justified conclusions.* Let `Output` denote this set of justified conclusions. One way of defining `Output` is to consider the conclusions which are supported by at least one argument in each extension.

Definition 5 (Justified conclusions) *Let $\langle \mathcal{A}, Def \rangle$ be an argumentation system and $\{E_1, \ldots, E_n\}$ be its set of extensions (under a given semantics).*
`Output` $= \{\psi | \forall E_i, \exists A \in E_i$ *such that* $\texttt{Conc}(A) = \psi\}$ *where* $\texttt{Conc}(A)$ *stands for the conclusion of the argument A.*

2.2 Logical language

In what follows, $\mathcal{L}$ will denote a logical language closed under negation. Moreover, we assume the availability of a function "$-$", which works with $\mathcal{L}$, such that $-\psi = \phi$ iff $\psi = \neg\phi$ and $-\psi = \neg\phi$ iff $\psi = \phi$. Let $\mathcal{K}$ be a knowledge base containing formulas of $\mathcal{L}$.

Definition 6 (Consistent set) *Let $\mathcal{P} \subseteq \mathcal{L}$. $\mathcal{P}$ is* consistent *iff $\nexists$ ψ, $\phi \in \mathcal{P}$ such that $\psi = -\phi$, otherwise it is said* inconsistent.

We distinguish between *strict* rules which will enable to define *conclusive* inferences and *defeasible* rules which will enable to define *defeasible* inferences.

Definition 7 (Strict and defeasible rules) *Let ϕ_1, ..., ϕ_n, ψ be elements of $\mathcal{L}$.*

A strict rule *is of the form ϕ_1, ..., $\phi_n \to \phi$ meaning that if $\phi_1, \ldots, \phi_n$ hold, then* without exception *it holds that ψ.*

A defeasible rule *is of the form ϕ_1, ..., $\phi_n \Rightarrow \phi$ meaning that if $\phi_1, \ldots, \phi_n$ hold, then it* usually *holds that ψ.*

$\phi_1, \ldots, \phi_n$ will be called the antecedent *of the rule and ϕ its* consequent. *$\mathcal{S}$ will denote the set of all strict rules and $\mathcal{R}$ will denote the set of all defeasible rules.*

Definition 8 (Transposition) *A strict rule s is a* transposition *of ϕ_1, ..., $\phi_n \to \psi$ iff $s = \phi_1$, ..., ϕ_{i-1}, $\neg\psi$, ϕ_{i+1}, ..., $\phi_n \to \neg\phi_i$ for some $1 \leq i \leq n$.*

Based on the thus defined notion of transposition, we now define a closure operator for the set $\mathcal{S}$.

Definition 9 (Transposition operator) *Let $\mathcal{S}$ be a set of strict rules. $Cl_{tp}(\mathcal{S})$ is a minimal set such that:*

- *$\mathcal{S} \subseteq Cl_{tp}(\mathcal{S})$, and*
- *If $s \in Cl_{tp}(\mathcal{S})$ and t is a transposition of s then $t \in Cl_{tp}(\mathcal{S})$.*

We say that $\mathcal{S}$ is closed under transposition *iff $Cl_{tp}(\mathcal{S}) = \mathcal{S}$.*

Definition 10 (Theory) *A* theory *$\mathcal{T}$ is a tuple $\langle (Cl_{tp}(\mathcal{S}),\ \mathcal{R}\rangle$.*

Definition 11 (Closure of a set of formulas) *Let $\mathcal{P} \subseteq \mathcal{K}$. The* closure *of $\mathcal{P}$ under the set $\mathcal{S}$ of strict rules, denoted $Cl_{\mathcal{S}}(\mathcal{P})$, is the smallest set such that:*

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

If $\mathcal{P} = Cl_{\mathcal{S}}(\mathcal{P})$, then $\mathcal{P}$ is said closed.

2.3 The notion of argument

The basic idea behind an argument, called here *epistemic*, is the fact that a given premise is justified on the basis of the available knowledge in $\mathcal{K}$. Such arguments have a *deductive* form.

In what follows, for a given argument, the function `PREM` returns all the formulas of $\mathcal{K}$ (called *premises*) used to build the argument, `PROP` returns all the propositions used in that argument, `CONC` returns its conclusion, `SUB` returns all its sub-arguments and finally the function `DefRules` returns all the defeasible rules of the argument.

Definition 12 (Epistemic Argument) *An* epistemic *argument A is:*

- *ϕ if $\phi \in \mathcal{K}$ with:*
  $\texttt{PREM}(A) = \{\phi\}$,
  $\texttt{PROP}(A) = \{\phi\}$,
  $\texttt{CONC}(A) = \phi$,
  $\texttt{SUB}(A) = \phi$,
  $\texttt{DefRules}(A) = \emptyset$.

- *$A_1, \ldots A_n \rightarrow \psi$ if $A_1, \ldots, A_n$ are epistemic arguments such that there exists a strict rule $\texttt{CONC}(A_1), \ldots, \texttt{CONC}(A_n) \rightarrow \psi$ in $Cl_{tp}(\mathcal{S})$.*
  $\texttt{PREM}(A) = \texttt{PREM}(A_1) \cup \ldots \cup \texttt{PREM}(A_n)$,
  $\texttt{PROP}(A) = \texttt{PROP}(A_1) \cup \ldots \cup \texttt{PROP}(A_n) \cup \{\psi\}$,
  $\texttt{CONC}(A) = \psi$,
  $\texttt{SUB}(A) = \texttt{SUB}(A_1) \cup \ldots \cup \texttt{SUB}(A_n) \cup \{A\}$.
  $\texttt{DefRules}(A) = \texttt{DefRules}(A_1) \cup \ldots \cup \texttt{DefRules}(A_n)$.

- *$A_1, \ldots A_n \Rightarrow \psi$ if $A_1, \ldots, A_n$ are epistemic arguments such that there exists a defeasible rule $\texttt{CONC}(A_1), \ldots, \texttt{CONC}(A_n) \Rightarrow \psi$.*
  $\texttt{PREM}(A) = \texttt{PREM}(A_1) \cup \ldots \cup \texttt{PREM}(A_n)$,
  $\texttt{PROP}(A) = \texttt{PROP}(A_1) \cup \ldots \cup \texttt{PROP}(A_n) \cup \{\psi\}$,
  $\texttt{CONC}(A) = \psi$,
  $\texttt{SUB}(A) = \texttt{SUB}(A_1) \cup \ldots \cup \texttt{SUB}(A_n) \cup \{A\}$,
  $\texttt{DefRules}(A) = \texttt{DefRules}(A_1) \cup \ldots \cup \texttt{DefRules}(A_n) \cup \{A_1, \ldots A_n \Rightarrow \psi\}$

Let $\mathcal{A}_e$ be the set of all epistemic arguments.

Definition 13 (Top rule) *Let A be an argument. The* top rule *of the argument A, denoted* `TopRule`(A)*, is the last rule used to build it.*

Let us illustrate the above definitions on the following example.

Example 1 *Let $\mathcal{K} = \emptyset$, $\mathcal{S} = \{\rightarrow a, \rightarrow d\}$ and $\mathcal{R} = \{a \Rightarrow b, d \Rightarrow \neg b\}$. The following arguments can be built:*

$A_1 : [\rightarrow a]$

$A_2 : [\rightarrow d]$

$A_3 : [A_1 \Rightarrow b]$

$A_4 : [A_2 \Rightarrow \neg b]$

Let us consider another example.

Example 2 *Let $\mathcal{K} = \{a, d\}$, $\mathcal{S} = \emptyset$ and $\mathcal{R} = \{a \Rightarrow b, d \Rightarrow \neg b\}$. The following arguments can be built:*

$A_1 : [a]$

$A_2 : [d]$

$A_3 : [A_1 \Rightarrow b]$

$A_4 : [A_2 \Rightarrow \neg b]$

Definition 14 (Strict vs. defeasible argument) *Let A be an epistemic argument. A is* strict *iff* `DefRules`$(A) = \emptyset$*, otherwise A is said* defeasible.

2.4 Comparing arguments

In [7, 68], it has been argued that arguments may have forces of various strengths. These forces will play at least two roles:

1. they allow an agent to compare different arguments in order to select the 'best' ones.
2. they are useful for determining the acceptable arguments among the conflicting ones.

In what follows $\succeq$ will denote any preference relation between epistemic arguments. For two arguments A and B, $A \succeq B$ means that A is at least as 'good' as B. $\succ$ denotes the strict ordering associated with $\succeq$. $A \succ B$ means that A is strictly preferred over B.

Throughout the document, we suppose that there exists a basic ordering $\succ$ on the set of arguments. This basic ordering captures the idea that strict arguments are preferred to the defeasible ones.

Definition 15 (Basic ordering) *Let A, B be two arguments. $A \succ B$ iff A is strict and B is defeasible.*

The basic ordering can be refined by other principles. In the literature there are two well-known principles for comparing pairs of epistemic arguments: the *last link* principle defined in [68] for legal applications, and the *weakest link* principle defined in [7] and applied in the case of handling inconsistency in knowledge bases.

2.4.1 Last link principle

The basic idea behind this principle is to prefer an argument A over another argument B if the last defeasible rule(s) used in B is less preferred than the last defeasible rule(s) in A. This principle takes it for granted the fact that there exists a partial pre-ordering $>_r$ (r for rules) on the set $\mathcal{R}$ of defeasible rules. For two defeasible rules R, R', $R >_r R'$ means that R is preferred to R'.

Before defining this principle, let us first start by introducing formally the concept of "last defeasible rule". Note that an argument may have several defeasible rules. Those rules are returned by a function `LastDefRules` defined as follows:

Definition 16 (Last defeasible rules) *Let A be an argument.*

- $\texttt{LastDefRules}(A) = \emptyset$ *iff* $\texttt{DefRules}(A) = \emptyset$.
- *If $A = A_1, \ldots, A_n \Rightarrow \phi$, then* $\texttt{LastDefRules}(A) = \{\texttt{CONC}(A_1), \ldots, \texttt{CONC}(A_n) \Rightarrow \phi\}$*, otherwise* $\texttt{LastDefRules}(A) = \texttt{LastDefRules}(A_1) \cup \ldots \cup \texttt{LastDefRules}(A_n)$.

The above definition is then used to compare pairs of arguments as follows:

Definition 17 (Last link principle) *Let A, $B \in \mathcal{A}_e$. A is preferred to be B, denoted $A \succ B$, iff $\exists\, R \in \texttt{LastDefRules}(B)$ such that $\forall\, R' \in \texttt{LastDefRules}(A)$, $R' > R$.*

Let us illustrate this definition by the following example.

Example 2 – continued: *Let $\mathcal{K} = \{a, d\}$, $\mathcal{S} = \emptyset$ and $\mathcal{R} = \{a \Rightarrow b, d \Rightarrow \neg b\}$. Suppose that $a \Rightarrow b >_r d \Rightarrow \neg b$. In this case, the argument $[[a] \Rightarrow b]$ is preferred to the argument $[[d] \Rightarrow \neg b]$.*

2.4.2 Weakest link principle

The idea behind the weakest link principle is to prefer an argument A over an argument B if the less entrenched belief of A is preferred to the less entrenched belief of B. In this case, both the knowledge base $\mathcal{K}$ and the set $\mathcal{R}$ of defeasible rules are supposed to be equipped with a partial pre-ordering $>_k$ (resp. $>_r$). For two elements K and K' of $\mathcal{K}$ $K >_k K'$ means that K is preferred to K'.

Definition 18 (Weakest link principle) *Let A, $B \in \mathcal{A}_e$. A is preferred to B, denoted by $A \succ B$, iff:*

- $\forall K \in \texttt{PREM}(A), \exists K' \in \texttt{PREM}(B)$ *such that* $K > K'$, *and*
- $\forall R \in \texttt{DefRules}(A), \exists R' \in \texttt{DefRules}(B)$ *such that* $R > R'$.

Consider again the above example.
Example 2 – continued: *Let $\mathcal{K} = \{a, d\}$ and $\mathcal{R} = \{a \Rightarrow b, d \Rightarrow \neg b\}$. Suppose that $a \Rightarrow b >_r d \Rightarrow \neg b$. If $a >_k d$, then the argument $[[a] \Rightarrow b]$ is preferred to the argument $[[d] \Rightarrow \neg b]$. However, if $d >_k a$ then the two arguments are incomparable.*

2.5 The conflicts between arguments

Since the information may be inconsistent, the arguments may be conflicting. Indeed, arguments supporting beliefs may be conflicting.

Two kinds of conflicts between arguments can be distinguished. The first one corresponds to the case where one argument uses a defeasible rule of which the applicability is disputed by the other argument. In the following definition, $\lceil . \rceil$ stands for the objectivation operator [65], which converts a meta-level expression into an object-level expression.

Definition 19 (Undercutting) *Let A and B be arguments in $\mathcal{A}_e$. A* undercuts *B iff $\exists\ B' \in \texttt{SUB}(B)$ of the form $B''_1, \ldots, B''_n \Rightarrow \psi$ and $\exists\ A' \in \texttt{SUB}(A)$ with $\texttt{CONC}(A') = \neg\lceil\texttt{CONC}(B''_1), \ldots, \texttt{CONC}(B''_n) \Rightarrow \psi\rceil$.*

The second kind of conflicts corresponds to the case where two arguments support contradictory conclusions.

Definition 20 (Rebutting) *Let A and B be arguments. A* rebuts *B iff $\exists\ A' \in \texttt{SUB}(A)$ with $\texttt{CONC}(A') = \phi$ and $\exists\ B' \in \texttt{SUB}(B)$ with $\texttt{CONC}(B') = -\phi$.*

Example 3 *Let $\mathcal{K} = \{a, t\}$, $\mathcal{S} = \{a \rightarrow b\}$, $\mathcal{R} = \{b \Rightarrow c, t \Rightarrow \neg b, \neg b \Rightarrow d\}$. The argument $[[[t] \Rightarrow \neg b] \Rightarrow d]$* rebuts *$[[[a] \rightarrow b] \Rightarrow c]$. The reverse is also true.*

Property 1 *The relation* Rebut *is symmetrical.*

The definition of *rebut* captures the idea of "Assumption attack" defined in the deliverable D2.2. In that deliverable, a set of assumptions has been distinguished from $\mathcal{K}$. However, we argue that those assumptions can be modeled as defeasible rules with an empty antecedent. So, in what follows "Assumption attack" will not be considered.

Definition 21 (Restricted rebutting) *Let A and B be arguments. A* restrictively rebuts *B on (A', B') iff $\exists\ A' \in \texttt{SUB}(A)$ with $\texttt{CONC}(A') = \phi$ and $\exists\ B' \in \texttt{SUB}(B)$ of the form $B''_1, \ldots, B''_n \Rightarrow -\phi$.*

Note that the restricted version of *rebut* is a refinement of this last.

Property 2 Restricted rebut $\Rightarrow$ Rebut. *The reverse is not always true.*

Note 1 *In what follows, we will keep the restricted version of rebutting.*

The two relations: *undercut* and restricted *rebutting* are brought together in a unique definition of "defeat" as follows:

Definition 22 (Defeating) *Let A and B be arguments of $\mathcal{A}_e$. We say that A* defeats *B iff*

1. *A* restrictively rebuts *B on (A', B') and not $(B' \succeq A')$, or*
2. *A undercut B.*

2.6 The acceptability of arguments

Once all the basic concepts introduced, we are now ready to define an argumentation framework for inferring from inconsistent knowledge bases.

Definition 23 (Argumentation framework) *Let $\mathcal{T}$ be a theory. An* argumentation framework *(AF) built on $\mathcal{T}$ is a pair $<\mathcal{A}_e$,* defeat*$>$ s.t:*

- *$\mathcal{A}_e$ is the set of arguments given in Definition 12,*
- defeat *is the relation given in Definition 22.*

Among all the conflicting arguments, it is important to know which are the arguments which will be kept for inferring conclusions, in other terms one should define the acceptable arguments. In what follows, we will use the different *semantics* for the notion of acceptability that have been proposed by Dung [33], and recalled in section 2.1.

$\mathcal{E} = \{E_1, \ldots, E_n\}$ will denote the set of all possible extensions under a given semantics of the argumentation system $<\mathcal{A}_e$, *defeat*$>$.

Property 3 *An argumentation framework $<\mathcal{A}_e$,* defeat*$>$ has a unique grounded extension which may be empty. It may have different stable, preferred and complete extensions.*

In [22], it has been shown that each argument which is in an extension, has all its sub-arguments in that extension.

Proposition 1 *Let $<\mathcal{A}_e$,* defeat*$>$ be an argumentation system and $E_1, \ldots, E_n$ its different extensions under a given semantics. $\forall\ A \in E_i$, then $\texttt{Sub}(A) \subseteq E_i$, $\forall E_i$.*

Another interesting property concerns the consistency of the results returned by this system. This is of great importance since it shows clearly that the above system returns *safe* conclusions.

Proposition 2 *Let* $<\mathcal{A}_e,$ defeat$>$ *be an argumentation system built from a theory* $\mathcal{T}$ *with* $\mathcal{S}$ *consistent, and* $E_1, \ldots, E_n$ *its different extensions under a given semantics.* $\forall\ E_i$*, the set* $\{\texttt{Conc}(A) \mid A \in E_i\}$ *is consistent.*

The above result shows also that the rationality postulate about direct consistency defined in [21] is satsfied by this argumentation system.

Proposition 3 *Let* $<\mathcal{A}_e,$ defeat$>$ *be an argumentation system built from a theory* $\mathcal{T}$ *with* $\mathcal{S}$ *consistent, and* $E_1, \ldots, E_n$ *its different extensions under a given semantics.* $\forall E_i$, $\{\texttt{Conc}(A)|A \in E_i\} = Cl_{\mathcal{S}}(\{\texttt{Conc}(A)|A \in E_i\})$.

The above result shows that the rationality postulate about closedness is satisfied by the argumentation framework defined in this chapter. The idea of closedness is that the answer of an argumentation-engine should be closed under strict rules. That is, if we provide the engine with a strict rule $a \rightarrow b$ ("if a then it is also *unexceptionally* the case that b"), together with various other rules, and our inference engine outputs a as justified conclusion, then it should also output b as justified conclusion. Consequently, b should also be supported by an acceptable argument.

Proposition 4 *Let* $<\mathcal{A}_e,$ defeat$>$ *be an argumentation system built from a theory* $\mathcal{T}$ *with* $\mathcal{S}$ *consistent, and* $E_1, \ldots, E_n$ *its different extensions under a given semantics.* $\forall E_i$*, the set* $Cl_{\mathcal{S}}(\{\texttt{Conc(A)}|A \in E_i\})$ *is consistent.*

The argumentation framework proposed here satisfies the third rationality postulate which is about indirect consistency.

2.7 The consequence relation

Once the acceptable arguments defined, conclusions may be inferred from a knowledge base.

Definition 24 (Inferring) *Let* $\mathcal{T}$ *be a theory,* $AF = <\mathcal{A}_e,$ defeat$>$ *be an argumentation framework and* $\mathcal{E}$ *its set of extensions under a given semantics.*
ψ *is* inferred *from* $\mathcal{T}$*, denoted by* $\mathcal{T}\ \mid\!\sim\ \psi$*, iff* $\forall\ E_i \in \mathcal{E}$, $\exists\ A \in E_i$ *such that* $\texttt{CONC}(A) = \psi$.
$\texttt{Output}(AF) = \{\psi \mid \mathcal{T}\ \mid\!\sim\ \psi\}$.

An important result is that the set of all conclusions inferred from $\mathcal{T}$ is consistent.

Proposition 5 *Let* $<\mathcal{A}_e,$ defeat$>$ *be an argumentation system built from a theory* $\mathcal{T}$ *with* $\mathcal{S}$ *consistent, and* $E_1, \ldots, E_n$ *its different extensions under a given semantics.* $\texttt{Output}(AF)$ *is consistent.*

The set of inferences made from a base using the proposed argumentation framework is closed under strict rules. Formally:

Proposition 6 *Let* $<\mathcal{A}_e$, defeat$>$ *be an argumentation system built from a theory* $\mathcal{T}$ *with* $\mathcal{S}$ *consistent, and* $E_1, \ldots, E_n$ *its different extensions under a given semantics.* $\texttt{Output}(AF) = Cl_{\mathcal{S}}(\texttt{Output}(AF))$.

Finally, it is to show that the closure under strict rules of the set of inferences is also consistent.

Proposition 7 *Let* $<\mathcal{A}_e$, defeat$>$ *be an argumentation system built from a theory* $\mathcal{T}$ *with* $\mathcal{S}$ *consistent, and* $E_1, \ldots, E_n$ *its different extensions under a given semantics. The set* $Cl_{\mathcal{S}}(\texttt{Output}(AF))$ *is consistent.*

2.8 Procedural form

The credulous and the skeptical acceptability of an argument under a given semantics have been defined in section 2.1. The problem which consists in deciding if an argument is credulously (resp. skeptically) accepted under a given semantics is called the credulous (resp. the skeptical) acceptance problem. In this chapter, we are interested in giving "proofs" of the acceptability.

As indicated in [24], in everyday life, a "proof" often takes the form of a set of pieces of information, such that these pieces of information, taken together, can prove the fact, in some sense. The proof can include the way in which the pieces of information have to be articulated in order to actually prove the fact, but not always: the idea is then that, given the pieces of information, it is not too difficult to reconstruct a proof. One interesting aspect of such proofs is that they provide a kind of "explanation" as to why a given fact is believed to be true. In this respect, "good" proofs are usually concise and avoid irrelevant information.

Considering our acceptance problems, we would like to be able to explain why a given argument is accepted. If we consider the credulous acceptance problem for the preferred semantics, a proof that an argument a is accepted could simply be an admissible set that contains a. However, this will probably not be informative enough as an explanation; it is usually interesting to know why the set is admissible: against which other arguments it defends itself, and how. In other words, a more satisfactory proof should exhibit the admissible set as well as all its attackers, and how the admissible set defends itself against these attackers.

This implies that a proof of acceptance for a given argument a should at least separate its arguments in two classes: the ones that are in favor of a in this proof (the elements of the admissible set in the case of the credulous acceptance problem under the preferred semantics); and the ones that are "against" a in this proof (the attackers of the admissible set). A proof can be given a more refined structure if we take a conciseness requirement into account: in this case, the only arguments in favor of a that it contains should be there for a good reason, that is, because they directly, or indirectly, defend a or another defender of a. Similarly, the only arguments against a in the proof should be the attackers of a or their defenders.

Argument games between a proponent (PRO) and an opponent (OPP) can be interpreted as constructing proofs of acceptance that have the dialectical structure underlined above: the proponent starts with the argument to be "proved", and attempts to defend that argument against any attack coming from the opponent. The precise rules of the argument game depend on the semantics to be captured.

[79] outlines argument games (called Two Party Immediate response Disputes, or TPI-disputes) which have been formalized by [34]. Argument games have also been formalized in [44], where a general framework is proposed which enables to define argument games for winning positions in argumentation frameworks.

Following the formal approach of [44], but with slightly different definitions, a general framework for argument games has been proposed in [24]. In section 2.8.1, we present this framework, which is a particular dialogue system (see Deliverable D2.1, section 4.2). The two argument games for the credulous acceptance problem under the preferred semantics which were introduced in [23, 24] are presented in section 2.8.2, and the argument game for the skeptical acceptance problem introduced in [32] is presented in section 2.8.3. An argument game for the acceptance problem under the grounded semantics is presented in section 2.8.4. Finally, some algorithms computing the previous argument games are presented in section 2.8.5.

2.8.1 A general framework for argument games

The common elements of dialogue systems have been presented in Deliverable D2.1, section 4.2. An argument game (or dialectical proof theory, as defined in [24]), is a particular dialogue system, whose elements are instantiated on the following way:

Dialogue goal: To prove the credulous or the skeptical acceptability of an argument under a given semantics.

Participants are two, the proponent (PRO) and the opponent (OPP). The participants are also called the *players*. The role of PRO is to defend the argument under consideration against any attack; the role of OPP is to outline these attacks. The set of commitments associated to each participant is empty[1].

Context: An argumentation system $<\mathcal{A}, Def>$, fixed and finite.

Topic language: The participants play arguments of $\mathcal{A}$.

Communication language: A *move* in $\mathcal{A}$ is a pair $[P, X]$ where $P \in \{\text{PRO}, \text{OPP}\}$ and $X \in \mathcal{A}$. A *dialogue* is a countable sequence of moves.

[1]Each participant is implicitly committed to the arguments he or she plays, but in order to make the framework as simple as possible, we choose to let the set of commitments always empty.

Protocol: A *legal-move function* defines, at every step in the dialogue, what moves can be used to continue the dialogue. The players play in turn and PRO plays first. The dialogue is terminated when the set of arguments returned by the legal-move function is empty.

Effect rules: None since the set of commitments of a participant is always empty.

Outcome rules: *Winning criteria* are defined in order to determine if the argument under consideration is successfully defended with an argument game.

Each participant has a **strategy** depending on the acceptance problem to solve. This strategy is represented in the legal-move function.

We define now argument games more formally, inspired by [44, 24, 32]. To this end, we will first define the notion of a dialogue type (involving the formal definition of a move and a legal-move function), then of a dialogue about an argument and finally of two winning criteria. All these definitions are set for a *basic* argumentation framework (see section 2.1).

Definition 25 (Dialogue type) *Let* $<\mathcal{A}, Def>$ *be an argumentation framework.*

A move *in* $\mathcal{A}$ *is a pair* $[P, X]$ *where* $P \in \{\mathrm{PRO}, \mathrm{OPP}\}$ *and* $X \in \mathcal{A}$. *For a move* $\mu = [P, X]$, *we use* $\mathrm{pl}(\mu)$ *to denote* P *and* $\arg(\mu)$ *to denote* X. *The set of moves in* $\mathcal{A}$ *is denoted by* $\mathcal{M}$. $\mathcal{M}^*$ *denotes the set of finite sequences of moves.*

A dialogue type *is a tuple* $<\mathcal{A}, Def, \phi>$ *where* $\phi : \mathcal{M}^* \rightarrow 2^{\mathcal{A}}$ *is a function called* legal-move function.

Definition 26 (Dialogue about an argument) *Let* $<\mathcal{A}, Def, \phi>$ *be a dialogue type. A* dialogue d *in* $<\mathcal{A}, Def, \phi>$ *(or* ϕ*-dialogue for short) is a countable sequence* $\mu_0\mu_1\ldots$ *of moves in* $\mathcal{A}$ *such that:*

1. $\mathrm{pl}(\mu_0) = \mathrm{PRO}$
2. $\mathrm{pl}(\mu_i) \neq \mathrm{pl}(\mu_{i+1})$
3. $\arg(\mu_{i+1}) \in \phi(\mu_0 \ldots \mu_i)$

We say that d is about *argument* $\arg(\mu_0)$.

Remind that when the set of arguments returned by the legal-move function is empty, the dialogue cannot be continued.

Although [44] allow any conflict-free set of arguments to appear in a move, we restrict a move to contain one argument only: actually, it is possible to check easily whether a particular argument can be advanced whereas allowing "arbitrary" conflict-free sets may lead to difficulties or inefficiencies in deciding if a particular subset could be used. We have dropped [44]'s requirement that $\arg(\mu_{i+1})$ must attack $\arg(\mu_i)$ because, in order to have sequential argument games of credulous acceptance with respect to the preferred semantics, we need to let OPP try all possible attacks against any of PRO's arguments, but only one at a time.

Let us now introduce some notations. Let $d = \mu_0\mu_1 \dots \mu_i$ be a finite ϕ-dialogue:

- μ_i is denoted by $\text{last}(d)$;
- $\phi(\mu_0 \dots \mu_i)$ is denoted by $\phi(d)$;
- $\text{PRO}(d)$ will denote the set of arguments advanced by PRO during d.
- The extension of d with a move μ in $\mathcal{A}$ such that $\mu_0\mu_1 \dots \mu_i\mu$ is a ϕ-dialogue is denoted by the juxtaposition $d.\mu$.

We consider two winning criteria: a given dialogue about an argument x can be won, or there can be a winning strategy for x, that is a way for PRO to defend x against all attacks of OPP. To give this second criterion, the following definition is needed: the sequence y is a *prefix* of the sequence x or x is an *extension* of y if and only if there exists a sequence z such that x is obtained by the concatenation of y and z, $x = y.z$.

Definition 27 (Winning criteria) *Given a dialogue type* $<\mathcal{A}, Def, \phi>$*:*

Dialogue won [44]: *A ϕ-dialogue d is* won by PRO *if and only if d is finite, cannot be continued (that is $\phi(d) = \emptyset$), and* $\text{pl}(\text{last}(d)) = \text{PRO}$.

Winning strategy [24]: *A* ϕ-winning strategy *for an argument x is a non-empty finite set S of finite ϕ-dialogues about x won by* PRO *such that:* $\forall d \in S, \forall d'$ *prefix of d such that* $\text{last}(d')$ *is played by* PRO, $\forall y \in \phi(d'), \exists d'' \in S$ *such that d'' is an extension of the juxtaposition* $d'.[\text{OPP}, y]$.

In other words, a ϕ-winning strategy must show that any ϕ-dialogue about x where PRO plays the last move can be extended in a ϕ-dialogue won by PRO whatever the response of OPP to this last move.

The definition of a dialogue which is won is used to define sequential argument games for the credulous acceptance of an argument under the preferred semantics (section 2.8.2.1), and for the skeptical acceptance of an argument under the preferred semantics (section 2.8.3). Winning strategies are used in section 2.8.2.2 to define tree-like argument games for the credulous acceptance of an argument under the preferred semantics: the dialogues that compose the strategy correspond to paths in the tree from the root to leaves of the tree.

2.8.2 Credulous reasoning under the preferred semantics

We first introduce some additional notations.

Notation 1 *Let* $<\mathcal{A}, Def>$ *be an argumentation framework. Given an argument $a \in \mathcal{A}$, we denote:*

- *by* $R^+(a) = \{b \in \mathcal{A} \mid aDefb\}$ *the set of the* successors *of* a*,*
- *by* $R^-(a) = \{b \in A \mid bDefa\}$ *the set of the* predecessors *of* a*, and*
- *by* $R^{\pm}(a)$ *the set* $R^+(a) \cup R^-(a)$*.*

Moreover, given a set $S \subseteq \mathcal{A}$ *of arguments and* $\varepsilon \in \{+, -, \pm\}$*,* $R^{\varepsilon}(S) = \bigcup_{a \in S} R^{\varepsilon}(a)$*.*

Finally, $\text{Refl}(<\mathcal{A}, Def>) = \{x \in \mathcal{A} \mid xDefx\}$ *is the set of arguments that attack themselves (or* self-attacking arguments*). This set will be denoted by* Refl *for short.*

The credulous acceptance problem is to decide if a given argument x belongs to at least one preferred extension. As already mentioned, a proof that x is credulously accepted exhibits an admissible set of arguments that contains x; the proof must also exhibit attackers of this set, and the structure of the proof must show how the set defends itself against these attackers.

According to [23, 24], the dialogues introduced in the previous section enable us to distinguish arguments that defend x from those that attack it: the former, as well as x itself, are labelled "PRO" in the dialogue, whereas the latter are labelled "OPP". The two proof theories that we present below are based on dialogues such that for every attacker, that is, for every move $[\text{OPP}, z]$, there is always a preceding move $[\text{PRO}, y]$ in the sequence, such that z attacks y. This PRO-argument y justifies/explains the presence of z in the proof. Similarly, every defender in a proof, that is, every move of the form $[\text{PRO}, y]$, must be *immediately* preceded in the dialogue by an attacker against which it defends, that is a move $[\text{OPP}, x]$ such that y attacks x.

Another restriction can be put on the moves that can appear in a proof of credulous acceptance: let d be a finite ϕ-dialogue. $R^{\pm}(\text{PRO}(d))$ contains the arguments which attack or which are attacked by an argument advanced by PRO during d. If d is to be a proof of acceptance, then PRO attempts to build an admissible set of arguments, so PRO cannot choose any argument in $R^{\pm}(\text{PRO}(d))$ for pursuing the dialogue d, nor any self-attacking argument. In the sequel, $\text{POSS}(d)$ denotes the set of arguments which may be chosen by PRO for extending an already conflict-free set $\text{PRO}(d)$:

$$\text{POSS}(d) = A \setminus (\text{PRO}(d) \cup R^{\pm}(\text{PRO}(d)) \cup \text{Refl}).$$

Note that it is useless for OPP to advance an argument which is attacked by $\text{PRO}(d)$.

2.8.2.1 Linear argument games

The following legal-move function leads to a dialectical proof theory in which OPP is not obliged to respond to the last argument advanced by PRO. Hence, the argument games obtained are linear.

Definition 28 (Legal-move function ϕ_1) *[23] Given an argumentation framework* $<\mathcal{A}, Def>$*, let* $\phi_1 : \mathcal{M}^* \to 2^{\mathcal{A}}$ *be defined by:*

- *if d is a dialogue of odd length (next move is by* OPP*),*

$$\phi_1(d) = R^-(\text{PRO}(d)) \setminus R^+(\text{PRO}(d));$$

- *if d is a dialogue of even length (next move is by* PRO*),*

$$\phi_1(d) = R^-(\text{arg}(\text{last}(d))) \cap \text{POSS}(d).$$

Combining a dialogue type $<\mathcal{A}, Def, \phi_1>$ and the first winning criterion (dialogue won, cf. Def. 27), we obtain ϕ_1-proofs:

Definition 29 (ϕ_1-proof for an argument) *[23] A ϕ_1-proof for the argument x is a ϕ_1-dialogue about x won by* PRO*.*

The following result establishes that the ϕ_1-proof theory is sound, and complete for finite argumentation framework.

Property 4 *[23] If d is a ϕ_1-proof for the argument x, then* PRO(d) *is an admissible set containing x.*

If the argument x is in a preferred extension of the argumentation framework $<\mathcal{A}, Def>$, and if $\mathcal{A}$ is finite, then there exists a ϕ_1-proof for x.

2.8.2.2 Tree-like argument games

It is also convenient to present proofs in a more traditional way, on a tree-like form, where at each stage of the proof, the advanced argument attacks the previous one. Such proofs are obtained using the following legal-move function.

Definition 30 (Legal-move function ϕ_2) *[23] Given an argumentation framework $<\mathcal{A}, Def>$, let $\phi_2 : \mathcal{M}^* \rightarrow 2^{\mathcal{A}}$ be defined by:*

- *if d is a dialogue of odd length (next move is by* OPP*),*

$$\phi_2(d) = R^-(\text{arg}(\text{last}(d))) \setminus R^+(\text{PRO}(d));$$

- *if d is a dialogue of even length (next move is by* PRO*),*

$$\phi_2(d) = \phi_1(d) = R^-(\text{arg}(\text{last}(d))) \cap \text{POSS}(d).$$

ϕ_2 is a restriction of ϕ_1 since, according to ϕ_2, OPP must advance an argument which attacks the argument advanced by PRO in the previous move.

Combining a dialogue type $<\mathcal{A}, Def, \phi_2>$ and the second winning criterion (winning strategy, Def. 27), we obtain ϕ_2-proofs:

Definition 31 (ϕ_2-proof for an argument) *[23] A ϕ_2-proof for the argument x is a ϕ_2-winning strategy S for x such that $\bigcup_{d \in S}(\text{PRO}(d))$ is conflict-free.*

The ϕ_2-proof theory is proved sound and complete for the credulous acceptance problem, according to the following result.

Property 5 *[23] There exists a ϕ_1-proof for the argument x if and only if there exists a ϕ_2-proof for x.*

An advantage of a ϕ_2-proof over a ϕ_1-proof is that one can immediately see, with a ϕ_2-proof, which argument is attacked by the argument advanced by OPP in a move (the argument of the immediately preceding move).

2.8.3 Skeptical reasoning under the preferred semantics

The skeptical acceptance problem under the preferred semantics, which consists in determining if a given argument is in every extension of a given argumentation system, is studied in [32]. We present here the results of [32].

2.8.3.1 Skeptical reasoning as credulous meta-reasoning

Determining if an argument is in every extension of an argumentation framework can be easily (but not efficiently) answered if we can enumerate all the extensions of the system: we consider a first extension E_1 and test if $x \in E_1$. If it is, this suggests that x may indeed be in every extension (as opposed to the case where $x \notin E_1$). We then consider a second extension E_2: it may happen that $x \notin E_2$, so the existence of E_2 *a priori* casts a doubt over the fact that x is in every extension. However, if it turns out that $x \in E_2$, this reinforces the possibility that x may be in every extension. Continuing the process, each extension E starts, with its sole existence, by being an argument suggesting that x may not be in every extension, to become, if it turns out that $x \in E$, an argument reinforcing the possibility that x is in every extension. Of course, enumerating all the extensions will generally not be efficient. We study in the remainder of this section how we can refine this approach, by enumerating smaller sets that can be interpreted as "meta"-arguments for or against the possibility that x is in every extension.

Since every argumentation system has at least one (preferred) extension, an argument x must be in at least one extension in order to be in all of them, so x must be in at least one admissible set. Now, suppose we have found one admissible set P that contains x; so we know that x is in at least one extension $E \supseteq P$. What could prevent x from being in every extension? If there is an extension E' such that $x \notin E'$, then $P \nsubseteq E'$, so there must be a conflict between P and E' (otherwise, since P and E' defend themselves, $P \cup E'$ would be admissible, which is not possible since E' is maximally admissible and $P \nsubseteq E'$). Thus if x is not in every extension, there must be some admissible set P' that attacks P and such that P' is not in any extension that contains x (take for instance $P' = E'$). In a sense, P can be seen as a "meta"-argument suggesting that x may well be in every extension of the system; whereas P' can be seen as a counter-argument: it suggests that, since there is an admissible set that contradicts P, there may be some maximal admissible set of arguments that does not contain x. This "meta" counter-argument is in turn contradicted if there is some admissible set of arguments P'' that contains both P' and x.

In order to formalise this approach, let us introduce the following definitions and notations:

Definition 32 (Def-attacks) *[32] Let $<\mathcal{A}, Def>$ be an argumentation system. Let $x \in \mathcal{A}$ and $S, S' \subseteq \mathcal{A}$. x Def-*attacks *S if there exists $y \in S$ such that $xDefy$. S Def-*attacks *x if there exists $y \in S$ such that $yDefx$. Finally, S Def-*attacks *S' if there exists $x \in S$ such that x Def-attacks S.*

Notation 2 *Given an argumentation system $<\mathcal{A}, Def>$,* $\mathrm{Adm}(<\mathcal{A}, Def>)$ *denotes the collection of admissible sets of $<\mathcal{A}, Def>$.*

Let us define a relation Def_x on the admissible subsets of some argumentation system $<\mathcal{A}, Def>$:

Definition 33 (Def_x-attacks) *[32] Let $<\mathcal{A}, Def>$ be an argumentation system. Let $X, Y \in$* $\mathrm{Adm}(<\mathcal{A}, Def>)$*. Then $XDef_xY$ (or X Def_x-attacks Y) if:*

1. *$x \in Y \setminus X$ and X Def-attacks Y; or*
2. *$x \in X \setminus Y$ and $X \supseteq Y$.*

In case 1., Y suggests that x may be in every extension: it is at least in all the extensions that contain Y; but X suggests that there may in fact be some extensions that do not contain x: those that contain X cannot contain Y because X Def-attacks Y.

In case 2., Y suggests that x may not be in every extension, since it is admissible and does not contain x; but X shows that Y can be extended to an admissible set that does contain x.

Note that X Def_x-attacks Y is not equivalent to X Def-attacks Y. The latter means that there is $(x, y) \in X \times Y$ such that $xDefy$.

It may be sufficient to restrict Def_x to some admissible sets of $<\mathcal{A}, Def>$ only. To this end, we define the set $\mathcal{A}_x$ as follows:

Definition 34 *[32] Let $<\mathcal{A}, Def>$ be an argumentation system. Let $x \in \mathcal{A}$. Then $\mathcal{A}_x = \mathcal{A}_x^{\mathrm{PRO}} \cup \mathcal{A}_x^{\mathrm{OPP}}$, where:*

$\mathcal{A}_x^{\mathrm{PRO}}$ *is the set of the admissible sets of $<\mathcal{A}, Def>$ that contain x;*

$\mathcal{A}_x^{\mathrm{OPP}}$ *is the set of the admissible sets X of $<\mathcal{A}, Def>$ that do not contain x and are of the form $X = \bigcup_{Y \in \mathcal{Y}} Y$, where the $Y \in \mathcal{Y}$ are parts of $\mathcal{A}$ minimal such that Y is an admissible set of $<\mathcal{A}, Def>$ and Y Def-attacks some element of $\mathcal{A}_x^{\mathrm{PRO}}$.*

We are now able to express the problem of skeptical acceptance of an argument x in terms of admissibility, or credulous acceptance, in a "meta"-argumentation framework. Such a framework has for "meta"-arguments the admissible sets of $\mathcal{A}_x$, and Def_x is its "meta"-defeat relation. Our first result is that if there is a "meta"-admissible set in favor of a given argument x, then x is in every extension of the theory:

Proposition 8 *[32] An argument x of an argumentation framework $<\mathcal{A}, Def>$ is in every preferred extension of $<\mathcal{A}, Def>$ if there exist $P \in \mathrm{Adm}(<\mathcal{A}, Def>)$ and $\mathcal{P} \in \mathrm{Adm}(< \mathcal{A}_x, Def_x >)$ such that $x \in P$ and $P \in \mathcal{P}$.*

Our next result shows that the approach is complete. It guarantees that if we can find an admissible set P of $<\mathcal{A}, Def>$ that contains x, and an admissible set of $< \mathcal{A}_x, Def_x >$ that contains P, then x is in every extension. However, this result alone would not guarantee the completeness of the approach: if we find P, but then cannot find $\mathcal{P}$, is it that x is not in every preferred extension of $<\mathcal{A}, Def>$, or could it be that we just picked the wrong P? The proposition below shows that any P can be part of a meta-proof for x, if x is in every extension:

Proposition 9 *[32] If an argument x of an argumentation framework $<\mathcal{A}, Def>$ is in every preferred extension of $<\mathcal{A}, Def>$, then:*

1. *for every $P \in \mathrm{Adm}(<\mathcal{A}, Def>)$ such that $x \in P$ there exists $\mathcal{P} \in \mathrm{Adm}(< \mathcal{A}_x, Def_x >)$ such that $P \in \mathcal{P}$;*
2. *there exist $P \in \mathrm{Adm}(<\mathcal{A}, Def>)$ and $\mathcal{P} \in \mathrm{Adm}(< \mathcal{A}_x, Def_x >)$ such that $x \in P$ and $P \in \mathcal{P}$.*

2.8.3.2 Argument game

Since we have characterized the skeptical acceptance problem as a credulous acceptance problem in a meta-argumentation framework, any dialectical proof theory designed for the credulous acceptance problem can be used to solve the skeptical acceptance problem. We will illustrate this below with a linear argument game.

Note that we will have to consider the credulous acceptability of a set of arguments under the preferred semantics, which consists in deciding if a set of arguments is included in at least one preferred extension. So we extend the definition of a dialogue in order to define a proof theory for the credulous acceptance of a set of arguments.

Definition 35 (Dialogue about a set of arguments) *Let $<\mathcal{A}, Def, \phi>$ be a dialogue type. A* dialogue *d in $<\mathcal{A}, Def, \phi>$ (or ϕ-dialogue for short)* for a finite set $S = \{a_1, a_2, \ldots, a_n\} \subseteq \mathcal{A}$ *is a countable sequence $\mu_{0_1}\mu_{0_2}\ldots\mu_{0_n}\mu_1\mu_2\ldots$ of moves in $\mathcal{A}$ such that:*

1. $\mathrm{pl}(\mu_{0_k}) = \mathrm{PRO}$ *and* $\mathrm{arg}(\mu_{0_k}) = a_k$*, for* $1 \leq k \leq n$
2. $\mathrm{pl}(\mu_1) = \mathrm{OPP}$ *and* $\mathrm{pl}(\mu_i) \neq \mathrm{pl}(\mu_{i+1})$*, for* $i \geq 1$
3. $\mathrm{arg}(\mu_{i+1}) \in \phi(\mu_{0_1}\ldots\mu_{0_n}\mu_1\ldots\mu_i)$

We say that d is about *S.*

In a dialogue about a set of arguments, the first n moves are played by PRO to put forward the elements of the set, and the subsequent moves are played alternatively by OPP and PRO. Note that a dialogue about an argument x is equivalent to a dialogue about the set of arguments $\{x\}$.

Definition 36 (ϕ_1-proof for a set of arguments) *Given a dialogue type $<\mathcal{A}, Def, \phi_1>$, a ϕ_1-proof for a set of arguments $S \subseteq \mathcal{A}$ is a ϕ_1-dialogue about S won by* PRO*.*

The following proposition ensures the soundness and completeness of the above proof theory for set-credulous acceptance.

Proposition 10 *[32] Let $<\mathcal{A}, Def, \phi_1>$ be a dialogue type where $<\mathcal{A}, Def>$ is an argument system such that $\mathcal{A}$ is finite. Let $S \subseteq \mathcal{A}$ be a conflict-free set. If d is a ϕ_1-proof for S, then* PRO(d) *is an admissible set containing S. If S is included in a preferred extension of $<\mathcal{A}, Def>$ then there exists a ϕ_1-proof for S.*

Let us now describe informally how this type of dialogue can be used as a proof theory for the skeptical acceptance problem. Suppose that we want to prove that some argument x of an argumentation system $<\mathcal{A}, Def>$ is in every extension of $<\mathcal{A}, Def>$. All we need to do is to find an admissible set P that contains x, and then find a dialogue for $\{P\}$ won by PRO with respect to the argumentation system $< \mathcal{A}_x, R_x >$.

In order to find the initial admissible set P that contains x, Proposition 10 says that we can look for a ϕ_1-dialogue d for $\{x\}$ won by PRO w.r.t. $<\mathcal{A}, Def>$: we can then take $P = \mathrm{PRO}(d)$. In order to establish that x is in every extension of the theory, we then start a dialogue with the move $[\mathcal{PRO}, P]$, where $\mathcal{PRO}$ denotes the player that tries to establish the acceptability of P in the meta-graph. In fact, a more detailed dialogue can start with the move $[\mathcal{PRO}, d]$, showing not only P but the entire dialogue that established the admissibility of P.

In order to continue the meta-dialogue, we need a move of the form $[\mathcal{OPP}, d_1]$, where $\mathcal{OPP}$ denotes the player who tries to establish that P is not credulously accepted in the meta-argumentation framework, and where d_1 must be a dialogue in $<\mathcal{A}, Def>$ for an argument that Def-attacks P.

$\mathcal{PRO}$ must then put forward a dialogue for $\mathrm{PRO}(d_1) \cup \{x\}$ won by PRO, thereby showing that the admissible set found by $\mathcal{OPP}$ in the preceding move can be "returned" in favor of $\mathcal{PRO}$.

This type of meta-dialogue is best illustrated on an example.

Example 4 *Consider the following system:*

$$\begin{tikzcd}
b \arrow[rd] \arrow[dd, leftrightarrow] & & & & \\
& c \arrow[r] & d \arrow[r] & e \arrow[r] & f \\
a \arrow[ru] & & g \arrow[ru] & &
\end{tikzcd}$$

A meta-dialogue proving that d is in every extension of the theory is depicted below (note that the moves of the dialogues in $<\mathcal{A}, Def>$ are in columns, whereas the moves of the meta-dialogue in $< \mathcal{A}_x, R_x >$ are in line):

$$\begin{bmatrix} \mathcal{PRO}, & [\text{PRO}, d] \\ & [\text{OPP}, c] \\ & [\text{PRO}, a] \end{bmatrix} \qquad [\mathcal{OPP}, [\text{PRO}, b]] \qquad \begin{bmatrix} \mathcal{PRO}, & [\text{PRO}, d] \\ & [\text{PRO}, b] \end{bmatrix}$$

$\mathcal{PRO}$'s first move shows that $\{a, d\}$ is admissible	*$\mathcal{OPP}$ then plays an admissible set $\{b\}$ that attacks $\{a, d\}$*	*$\mathcal{PRO}$ concludes by proving that $\{b, d\}$ is admissible in $<\mathcal{A}, Def>$*

2.8.4 Skeptical reasoning under the grounded semantics

An argument game for the grounded semantics was described in Section 2.5 of [13]. We here briefly outline a reformulation of this argument game in the present framework.

Definition 37 (Legal-move function ϕ_g) *Given an argumentation framework $<\mathcal{A}, Def>$, let $\phi_g : \mathcal{M}^* \rightarrow 2^A$ be defined by:*

- *if d is a dialogue of odd length (next move is by* OPP*),*
$$\phi_g(d) = R^-(\text{arg}(\text{last}(d)));$$
- *if d is a dialogue of even length (next move is by* PRO*),*
$$\phi_g(d) = R^-(\text{arg}(\text{last}(d))) \setminus (R^+(\text{arg}(\text{last}(d))) \cup \text{PRO}(d)).$$

Definition 38 (ϕ_g-proof for an argument) *A ϕ_g-proof for an argument x is a ϕ_g-winning strategy S for x.*

Soundness and completeness (Proposition 2.1 in [13]) can be proved as a straightforward generalisation of proofs for specific systems in [68] and [8]. Such a generalised proof can be found in [20].

2.8.5 Algorithms

In this section, we present an algorithm which searches for a proof of the credulous acceptability under the preferred semantics of an argument. This algorithm, first introduced in [24], relies upon the ϕ_1-proof theory presented in Section 2.8.2. We extend this algorithm to search for a proof of the credulous acceptability under the preferred semantics of a set of arguments. An algorithm searching for a proof of the skeptical acceptability under the preferred semantics of an argument would rely upon the two previous algorithms. We finally present an algorithm which searches for a proof of the acceptability of a set of arguments under the grounded semantics.

2.8.5.1 Credulous reasoning under the preferred semantics

Because the ϕ_1-proof theory is sound and complete with respect to credulous acceptance of an argument under the preferred semantics (see Section 2.8.2.1), we can restate the credulous acceptance problem of an argument as follows:

> Given an argumentation framework $<\mathcal{A}, Def>$, and an argument $a \in \mathcal{A}$, is there a ϕ_1-proof for a?

The algorithm that we present below, first introduced in [24], searches for a ϕ_1-proof for a. It builds a proof from left to right, thereby progressing backwards in the graph of the defeat relation. The algorithm recursively enumerates all attackers, that is, predecessors, of arguments put forward by PRO: they are OPP's arguments in the proof being built. For each of OPP's argument that is not already attacked by PRO, the algorithm chooses one of its predecessors as defender of the argument that has just been attacked: it is put forward by PRO.

The choices made by the algorithm when looking for defenders may lead to an unsuccessful line of defence. In this case, the algorithm must look for another line of defence: the algorithm comes back to an earlier stage and tries another choice of defence if any is left; when no other choice is left, it shows that no ϕ_1-proof can be found for the original argument a. Not everything is forgotten after such a backtrack: PRO can remember which lines of defence have been lost before the backtrack, in order not to loose time again with lines of defence bound to be lost. The fact that such lines of defence can never be won by PRO is shown by [24], Theorem 4.1.

We are now ready to present our credulous query answering algorithm. We will do so from the point of view of PRO: its primary goal is to construct an admissible set "around" a given argument a. Suppose that, at a given stage, a ϕ_1-dialogue d about a has been built, and that it is OPP's turn to make a move; we know that PRO(d) is conflict-free. At that point, OPP must choose an attacker of the arguments contained in PRO(d), which is not already attacked by PRO(d), that is, an element x of $\phi_1(d) = R^-(\mathrm{PRO}(d)) \setminus R^+(\mathrm{PRO}(d))$. If none exists, then PRO has won the dialogue. However, if such an x is put forward by OPP, then PRO must find an attacker y of x. There are some restrictions: y must be in $\mathrm{POSS}(d) = \mathcal{A} \setminus (\mathrm{PRO}(d) \cup R^{\pm}(\mathrm{PRO}(d)) \cup \mathrm{Refl})$, and y must not lead to a line of defence tried without success earlier. Let O denote, at any stage, the arguments that have not yet been put forward by PRO, and cannot/must not be put forward by PRO (they will be "Out" of the set of arguments built by PRO): O must contain $R^{\pm}(\mathrm{PRO}(d)) \cup \mathrm{Refl}$, and the arguments that have already lead to some unsuccessful line of defence.

Our algorithm can be expressed as a function $\texttt{CredQA}_{\texttt{rec}}$ which, given a ϕ_1-dialogue d and a set O disjoint from PRO(d), returns a ϕ_1-dialogue d' won by PRO and extending d, such that $\mathrm{PRO}(d') \cap O = \emptyset$, if such a d' exists; and returns $\perp$ if no such d' exists. The function $\texttt{CredQA}_{\texttt{rec}}$ is called by the main function `CredQA`, which first checks that $a \notin \mathrm{Refl}$, where a is the argument on query: if this is the case, $\texttt{CredQA}_{\texttt{rec}}$ is called with $O = \mathrm{Refl} \cup R^{\pm}(a)$, and $d = [\mathrm{PRO}, a]$.

Function 1 $\texttt{CredQA}_{\texttt{rec}}(Def, d, O)$

```
Parameters: a binary relation $Def$, a $\phi_1$-dialogue $d$, and a set $O$ disjoint from
    PRO($d$)
Result: a $\phi_1$-dialogue $d'$ won by PRO extending $d$, such that PRO$(d') \cap O = \emptyset$,
    if such a $d'$ exists; $\perp$ if no such $d'$ exists
if there is $x \in R^-(\mathrm{PRO}(d)) \setminus R^+(\mathrm{PRO}(d))$ such that $R^-(x) \subseteq O$ then
    return $(\perp)$;
if there is $x \in R^-(\mathrm{PRO}(d)) \setminus R^+(\mathrm{PRO}(d))$ such that $R^-(x) \not\subseteq O$ then
    $y \leftarrow$ some element of $R^-(x) \setminus O$;
    $res \leftarrow \texttt{CredQA}_{\texttt{rec}}(Def, d.[\mathrm{OPP}, x].[\mathrm{PRO}, y], O \cup R^{\pm}(y))$;
    if $res \neq \perp$ then return $(res)$;
    else return $(\texttt{CredQA}_{\texttt{rec}}(Def, d, O \cup \{y\})$;
else return $(d)$;
```

Function 2 $\texttt{CredQA}(Def, a)$

```
Parameters: a binary relation $Def$, an argument $a$
Result: a $\phi_1$-proof for $a$ if $a$ is credulously accepted; $\perp$ otherwise
if $a \in \mathrm{Refl}$ then return $(\perp)$;
else return $(\texttt{CredQA}_{\texttt{rec}}(Def, [\mathrm{PRO}, a], \mathrm{Refl} \cup R^{\pm}(a)))$;
```

The algorithm for the credulous reasoning for an argument can be easily extended to search for a proof of the credulous acceptability under the preferred semantics of a set of arguments. This algorithm is expressed as the function `CredQAset`. This function first checks if the set to be tested is conflict-free; if this is the case, a dialogue d putting forward the arguments of the set is built, and then $\texttt{CredQA}_{\texttt{rec}}$ is called with $O = \mathrm{Refl} \cup R^{\pm}(S)$ and d.

Function 3 $\texttt{CredQAset}(Def, S)$

```
Parameters: a binary relation $Def$, a set of arguments $S$
Result: a $\phi_1$-proof for $S$ if $S$ is credulously accepted; $\perp$ otherwise
if there is $x, y \in S$ such that $x Def y$ then return $(\perp)$;
else
    let $d$ be an empty dialogue;
    $O \leftarrow \emptyset$;
    while $S \neq \emptyset$ do
        $x \leftarrow$ some argument of $S$;
        $d \leftarrow d.[\mathrm{PRO}, x]$;
        $O \leftarrow O \cup R^{\pm}(x)$;
        $S \leftarrow S \setminus \{x\}$;
    return $(\texttt{CredQA}_{\texttt{rec}}(Def, d, \mathrm{Refl} \cup O))$;
```

2.8.5.2 Skeptical reasoning under the grounded semantics

We can restate the skeptical acceptance problem under the grounded semantics as follows:

> Given an argumentation framework $<\mathcal{A}, Def>$, and an argument $a \in \mathcal{A}$, is there a ϕ_g-proof for a?

The algorithm that we present below searches for a ϕ_g-proof for a. In other words, it tries to build a ϕ_g-winning strategy S for a, that is, a set of ϕ_g-dialogues about a won by PRO such that: $\forall d \in S$, $\forall d'$ prefix of d such that $\text{pl}(\text{last}(d')) = \text{PRO}$, $\forall x \in \phi_g(d')$, that is, $\forall x$ which attacks $\text{arg}(\text{last}(d'))$, $\exists d'' \in S$ such that d'' is an extension of $d'.[\text{OPP}, x]$.

We define a *ϕ_g-winning strategy for a dialogue* d as a set S of ϕ_g-dialogues won by PRO that extend d and such that: $\forall d' \in S$, $\forall d''$ prefix of d' and extension of d, such that $\text{pl}(\text{last}(d'')) = \text{PRO}$, $\forall x \in \phi_g(d'')$, $\exists d''' \in S$ such that d''' is an extension of $d''.[\text{OPP}, x]$.

Given an argument a, a ϕ_g-winning strategy for the dialogue $[\text{PRO}, a]$ is obviously a ϕ_g-winning strategy for a. Moreover, a ϕ_g-winning strategy for a dialogue d is $\{d\}$ if d is won by PRO; otherwise, if the last move of d is by PRO, one has to consider the dialogues $d_1, \ldots, d_n$ such that, given $\phi_g(d) = \{x_1, \ldots, x_n\}$, $d_i = d.[\text{OPP}, x_i]$. If one of these dialogues has no ϕ_g-winning strategy, then there is no ϕ_g-winning strategy for d; this happens if no $y \in \phi_g(d_i) = R^-(x) \setminus (R^+(x) \cup \text{PRO}(d_i))$ is such that $d.[\text{OPP}, x].[\text{PRO}, y]$ has a ϕ_g-winning strategy. Otherwise, a ϕ_g-winning strategy for d is the union of the ϕ_g-winning strategies for the dialogues $d_1, \ldots, d_n$. A ϕ_g-winning strategy for a dialogue d_i is in fact a ϕ_g-winning strategy for a dialogue $d'_i = d_i.[\text{PRO}, y] = d.[\text{OPP}, x].[\text{PRO}, y]$, with $y \in \phi_g(d_i)$.

Our algorithm relies upon these properties. It is expressed as a function called $\texttt{GroundedQA}_{\texttt{rec}}$ which, given a ϕ_g-dialogue d, the last move of which is by PRO, returns a ϕ_g-winning strategy S for d, if one exists; and returns $\bot$ if no such S exists. The function $\texttt{GroundedQA}_{\texttt{rec}}$ is called by the main function `GroundedQA`, with $d = [\text{PRO}, a]$.

Testing if a set of arguments is included in the grounded extension can be simply done by testing if each argument of the set is included in the grounded extension. This is what function `GroundedQAset` does.

Function 4 `GroundedQA`$_{\texttt{rec}}(Def, d)$

Parameters: a binary relation Def, a ϕ_g-dialogue d, the last move of which is by PRO

Result: a ϕ_g-winning strategy for d, if such a winning strategy exists; $\bot$ otherwise.

```
1:  $X \leftarrow R^-(\text{arg}(\text{last}(d)))$;
2:  **if** $X = \emptyset$ **then return** $\{d\}$;
3:  **else**
4:      $S \leftarrow \emptyset$;
5:      failure $\leftarrow \bot$;
6:      **while** $X \neq \emptyset$ and $\neg$ failure **do**
7:          $x \leftarrow$ some element of $X$;
8:          $Y \leftarrow R^-(x) \setminus (R^+(x) \cup \text{PRO}(d))$;
9:          success $\leftarrow \bot$;
10:         **while** $Y \neq \emptyset$ and $\neg$ success **do**
11:             $y \leftarrow$ some element of $Y$;
12:             $res \leftarrow \texttt{GroundedQA}_{\texttt{rec}}(Def, d.[\text{OPP}, x].[\text{PRO}, y])$;
13:             **if** $res = \bot$ **then** $Y \leftarrow Y \setminus \{y\}$;
14:             **else** success $\leftarrow \top$;
15:         **if** success **then**
16:             $S \leftarrow S \cup res$;
17:             $X \leftarrow X \setminus \{x\}$;
18:         **else** failure $\leftarrow \top$;
19:     **if** failure **then return** $\bot$;
20:     **else return** $S$;
```

Function 5 `GroundedQA`(Def, a)

Parameters: a binary relation Def, an argument a

Result: a ϕ_g-proof for a if a belongs to the grounded extension; $\emptyset$ otherwise

```
1:  **return** $(\texttt{GroundedQA}_{\texttt{rec}}(Def, [\text{PRO}, a]))$;
```

Function 6 `GroundedQAset`(Def, X)

Parameters: a binary relation Def, a set of arguments X

Result: a set $\mathcal{S}$ that contains a ϕ_g-proof for each argument of X, if such a set exists; $\bot$ otherwise

```
1: $\mathcal{S} \leftarrow \emptyset$;
2: failure $\leftarrow \bot$;
3: **while** $X \neq \emptyset$ and $\neg$ failure **do**
4:    $a \leftarrow$ some element of $X$;
5:    $res \leftarrow$ GroundedQA$(Def, a)$;
6:    **if** $res = \bot$ **then** failure $\leftarrow \top$;
7:    **else**
8:       $\mathcal{S} \leftarrow \mathcal{S} \cup \{res\}$;
9:       $X \leftarrow X \setminus \{a\}$;
10: **if** failure **then return** $\bot$;
11: **else return** $\mathcal{S}$;
```

Chapter 3

A general argumentation system for decision making

3.1 Introduction

In the previous chapter, we have presented a general argumentation framework for inferring from inconsistent knowledge bases. That framework satisfies the rationality postulates introduced in [21], this means that it delivers safe conclusions. Moreover, the framework handles one type of arguments, called epistemic arguments. In this section, we will extend that framwork in order to make decisions. For that purpose, in addition to the epistemic arguments, we will show that two other types of arguments are considered: recommended arguments and decision arguments. Indeed, epistemic arguments support beliefs, whereas recommended and decision arguments support decisions. The three types of arguments may conflict with each other exactly in the two ways described in the previous chapter. However, in order to avoid in kind of wishful thinking, epistemic arguments take precedence over recommended and decision ones. An argumentation framework for decision making takes then as input three kinds of arguments, as well as the different conflict which may exist between them, and returns the acceptable ones using the acceptability semantics of Dung. Once the acceptable arguments are known, decisions will be compared using a principle on the basis of the quality of their supporting arguments.

3.2 Logical language and the different bases

In what follows we will consider the same language as in Chapter 1 but with some refinements. $\mathcal{L}$ denotes a logical language closed under negation. We assume that formulas of $\mathcal{L}$ are built on two kinds of variables: *decision variables* $\mathcal{DV}$ and *non-decision variables* $\mathcal{NDV}$ such that $\mathcal{DV} \cap \mathcal{NDV} = \emptyset$. In what follows, the function `VAR` will return for an element ϕ of $\mathcal{L}$ the set of all its variables.

Thus, from $\mathcal{L}$, the four following sets will be distinguished:

1. $\mathcal{D}$ will contain all the possible *decisions* such that $\forall\ d \in \mathcal{D}$, $\texttt{VAR}(d) \subseteq \mathcal{DV}$.
2. $\mathcal{K}$ will represent the *knowledge* about the environment of an agent such that $\forall\ \phi \in \mathcal{K}$, $\texttt{VAR}(\phi) \subseteq \mathcal{NDV}$.
3. $\mathcal{G}^+$ will gather the *goals* of an agent, i.e what an agent wants to achieve, such that $\forall\ g \in \mathcal{G}^+$, $\texttt{VAR}(g) \subseteq \mathcal{DV}$.
4. $\mathcal{G}^-$ will gather the *rejections* of an agent, i.e what an agent wants to avoid, such that $\forall\ g \in \mathcal{G}^-$, $\texttt{VAR}(g) \subseteq \mathcal{DV}$.

The set $\mathcal{S}$ of strict rules (resp. $\mathcal{R}$ of defeasible rules) contains three forms of strict (resp. defeasible rules) rules:

- $\phi_1, \ldots, \phi_n \rightarrow \phi$ (resp. $\phi_1, \ldots, \phi_n \Rightarrow \phi$) where $\forall \phi_{i=1,n}$, ϕ, $\texttt{VAR}(\phi_i)$, $\texttt{VAR}(\phi)$ $\subseteq \mathcal{NDV}$. These rules are used to infer a belief from a set of premises. $\mathcal{S}_1$ denotes the set of such rules.
- $\phi_1, \ldots, \phi_n \rightarrow d$ (resp. $\phi_1, \ldots, \phi_n \Rightarrow d$) where $\forall \phi_{i=1,n}$, $\texttt{VAR}(\phi_i) \subseteq \mathcal{NDV}$ and $d \in \mathcal{D}$. Such rules are mainly used in rule-based decision making and they mean "if it is the case that $\phi_1, \ldots, \phi_n$, then the decision d should (resp. may) be taken". $\mathcal{S}_2$ denotes the set of such rules.
- $\phi_1, \ldots, \phi_n, d \rightarrow g$ (resp. $\phi_1, \ldots, \phi_n, d \Rightarrow g$) where $\forall \phi_{i=1,n}$, $\texttt{VAR}(\phi_i) \subseteq \mathcal{NDV}$, $d \in \mathcal{D}$, $g \in \mathcal{G}^+$ (resp. $\mathcal{G}^-$). These rules are encountered in decision under uncertainty as well as in multiple criteria decision making. They mean that "if $\phi_1, \ldots, \phi_n$ hold, then the decision d will (resp. may) lead to the satisfaction of the goal g". $\mathcal{S}_3$ denotes the set of such rules.

It is clear that $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2 \cup \mathcal{S}_3$.

Example 5 *Let us consider the well-known example about taking an umbrella or not, knowing that the sky is cloudy. The different bases are:* $\mathcal{K} = \{c\}$, $\mathcal{D} = \{u, \neg u\}$, $\mathcal{G}^+ = \{\neg w,\ \neg l\}$, $\mathcal{G}^- = \{w,\ l\}$, *with: l: to be overloaded, r: it rains, w: being wet, u: taking an umbrella, c: the sky is cloudy. Let us suppose the following strict rules:* $\mathcal{S} = \{c \rightarrow r,\ u \rightarrow l,\ \neg u \rightarrow \neg l,\ u \rightarrow \neg w,\ r;$[1] $\neg u \rightarrow w,\ \neg r \rightarrow \neg w\}$. *We suppose that* $\mathcal{R} = \emptyset$.

Note 2 *In the above example, the positive goals are the negation of the negative ones. However, in practice this is not always the case. The same remark holds for decisions. In a decision making problem, one does not always try to decide between d and ¬d. Consider, for instance, the case of an agent who wants to go somewhere for holidays and hesitates between the three following destinations: Toulouse (d_1), Utrecht (d_2) and Liverpool (d_2).*

Definition 39 (Theory) *A* theory $\mathcal{T}$ *is a tuple* $\langle(\mathcal{D},\ \mathcal{K},\ \mathcal{G}^+,\ \mathcal{G}^-),\ Cl_{tp}(\mathcal{S}_1) \cup \mathcal{S}_2 \cup \mathcal{S}_3,\ \mathcal{R}\rangle$.

[1] Here the semicolon plays the role of a conjunction.

3.3 The arguments

Two categories of arguments will be defined: *epistemic* arguments for supporting *beliefs* (see Definition 12) and *non-epistemic* arguments for supporting *decisions*. Among non-epistemic arguments, one may distinguish between *recommending* arguments and *decision* arguments. The idea is that a given decision may be justified in two ways:

1. it is recommended in a given situation (in the case of *rule-based decision*), or

2. it satisfies / violates some goals of the decision maker (in the case of decision under uncertainty and in multiple criteria decision [3, 11]).

3.3.1 Recommending arguments

In rule-based decision making, one generally specifies when a decision can be applied. Such rules are captured in this framework in terms of strict or defeasible rules of the form $\phi_1, \ldots \phi_n \rightarrow d$ (resp. $\phi_1, \ldots \phi_n \Rightarrow d$) with $\phi_1, \ldots \phi_n \in \mathcal{K}$ and $d \in \mathcal{D}$. These rules mean that if the situation $\phi_1, \ldots \phi_n$ holds then one "should" take the decision d (resp. "can" take the decision d). Recommending arguments are thus based on epistemic ones and have a *deductive* form. Formally:

Definition 40 (Recommending Argument) *A* recommending *argument is*

- $A_1, \ldots A_n \rightarrow d$ *if* $A_1, \ldots, A_n$ *are epistemic arguments,* $d \in \mathcal{D}$ *and there exists a strict rule* $\texttt{CONC}(A_1), \ldots, \texttt{CONC}(A_n) \rightarrow d$*,*
  $\texttt{PREM}(A) = \texttt{PREM}(A_1) \cup \ldots \cup \texttt{PREM}(A_n)$*,*
  $\texttt{PROP}(A) = \texttt{PROP}(A_1) \cup \ldots \cup \texttt{PROP}(A_n) \cup \{d\}$*,*
  $\texttt{CONC}(A) = d$*,*
  $\texttt{SUB}(A) = \texttt{SUB}(A_1) \cup \ldots \cup \texttt{SUB}(A_n) \cup \{A\}$*,*
  $\texttt{DefRules}(A) = \texttt{DefRules}(A_1) \cup \ldots \cup \texttt{DefRules}(A_n)$*.*

- $A_1, \ldots A_n \Rightarrow d$ *if* $A_1, \ldots, A_n$ *are epistemic arguments,* $d \in \mathcal{D}$ *and there exists a defeasible rule* $\texttt{CONC}(A_1), \ldots, \texttt{CONC}(A_n) \Rightarrow d$*,*
  $\texttt{PREM}(A) = \texttt{PREM}(A_1) \cup \ldots \cup \texttt{PREM}(A_n)$*,*
  $\texttt{PROP}(A) = \texttt{PROP}(A_1) \cup \ldots \cup \texttt{PROP}(A_n) \cup \{d\}$*,*
  $\texttt{CONC}(A) = d$*,*
  $\texttt{SUB}(A) = \texttt{SUB}(A_1) \cup \ldots \cup \texttt{SUB}(A_n) \cup \{A\}$*,*
  $\texttt{DefRules}(A) = \texttt{DefRules}(A_1) \cup \ldots \cup \texttt{DefRules}(A_n) \cup \{\texttt{CONC}(A_1), \ldots, \texttt{CONC}(A_n) \Rightarrow d\}$*.*

$\mathcal{A}_r$ *stands for the set of all recommended arguments. If* A *is a recommended argument with* $\texttt{CONC}(A) = d$ *, then* d *is called a* recommended *decision.*

Note 3 *Note that the recommended decision appears in the top rule of the argument. Moreover, the argument is based only on beliefs.*

Property 6 *Let $A \in \mathcal{A}_r$. $\forall A' \in$ SUB(A) such that $A \neq A'$, A' is an epistemic argument.*

Let us illustrate this category of arguments through the following example.

Example 6 *Let $\mathcal{K} = \{age > 18\}$, $\mathcal{D} = \{v, \neg v\}$ and $\mathcal{G}^+ = \mathcal{G}^- = \emptyset$. Let also $\mathcal{S}$ $= \{age >= 18 \rightarrow v, age < 18 \rightarrow v\}$, $\mathcal{R} = \emptyset$, with v stands for "to vote".*

In this case, one can construct the following argument for v: $[[age >= 18] \rightarrow v]$.

3.3.2 Decision arguments

In decision under uncertainty (resp. in multiple criteria decision), the preferred decisions are generally the ones which "highly" satisfy the goals/preferences (resp. the criteria). As shown in [3, 11], a decision is related to the goals by means of rules of the form $\phi_1, \ldots \phi_n, d \rightarrow g$ (resp. $\phi_1, \ldots \phi_n, d \Rightarrow g$) meaning that in the case where $\phi_1, \ldots \phi_n$ are true, if the decision d is taken then this leads to the satisfaction of the goal g.

In what follows, for a given decision argument, the functions GOALS$^+$ and GOALS$^-$ will return respectively the goals and the rejections satisfied by the decision supported by that argument.

Definition 41 (Decision Argument) *A* decision *argument is*

- *$A_1, \ldots A_n, d \rightarrow g$ if $A_1, \ldots, A_n$ are epistemic arguments and there exits a strict rule CONC(A_1), ..., CONC(A_n), $d \rightarrow g$ such that $d \in \mathcal{D}$ and $g \in \mathcal{G}^+$ (resp. $g \in \mathcal{G}^-$).*
  PREM(A) = PREM$(A_1) \cup \ldots \cup$ PREM(A_n),
  PROP(A) = PROP$(A_1) \cup \ldots \cup$ PROP$(A_n) \cup \{g\}$,
  GOALS$^+(A) = \{g\}$ *(resp.* GOALS$^-(A) = \{g\}$*)*,
  CONC$(A) = d$,
  SUB(A) = SUB$(A_1) \cup \ldots \cup$ SUB$(A_n) \cup \{A\}$.
  DefRules(A) = DefRules$(A_1) \cup \ldots \cup$ DefRules(A_n).

- *$A_1, \ldots A_n, d \Rightarrow g$ if $A_1, \ldots, A_n$ are epistemic arguments and there exits a defeasible rule CONC(A_1), ..., CONC(A_n), $d \Rightarrow g$ such that $d \in \mathcal{D}$ and $g \in \mathcal{G}^+$ (resp. $g \in \mathcal{G}^-$).*
  PREM(A) = PREM$(A_1) \cup \ldots \cup$ PREM(A_n),
  PROP(A) = PROP$(A_1) \cup \ldots \cup$ PROP$(A_n) \cup \{g\}$,
  GOALS$^+(A) = \{g\}$ *(resp.* GOALS$^-(A) = \{g\}$*)*,
  CONC$(A) = d$,
  SUB(A) = SUB$(A_1) \cup \ldots \cup$ SUB$(A_n) \cup \{A\}$.
  DefRules(A) = DefRules$(A_1) \cup \ldots \cup$ DefRules$(A_n) \cup \{$CONC$(A_1), \ldots,$ CONC$(A_n) \Rightarrow d\}$.

Let $\mathcal{A}_d$ be the set of all decision arguments.

Note 4 *Note that the decision and the satisfied goal appear in the top rule of the argument. Moreover, the argument is based only on beliefs. A decision will have as many decision arguments as it satisfies goals.*

Example 5 – continued: *From the theory $\mathcal{T}$ given in example 5, the two epistemic arguments A_1 and A_2 are built:*

$A_1 : [c]$

$A_2 : [A_1 \rightarrow r]$

In addition to the above arguments, the following decision arguments can also be built:

$A_3 : [u \rightarrow l]$

$A_4 : [u \rightarrow \neg w]$

$A_5 : [\neg u \rightarrow \neg l]$

$A_6 : [A_2, u \rightarrow \neg w]$

As for recommending arguments, the subarguments of a decision argument are epistemic. Formally:

Property 7 *Let $A \in \mathcal{A}_d$. $\forall A' \in \texttt{SUB}(A)$ such that $A \neq A'$, A' is an epistemic argument.*

In [9], it has been argued that arguments are presented in a bipolar way since arguments in favor of (or PRO) a conclusion can be considered as *positive* and arguments against (CON) the conclusion as *negative* ones. In some sense this is true since the two kinds of arguments have different and opposite roles. Indeed, the positive arguments will strengthen their conclusions, whereas the negative ones will weaken them.

In an inference problem, one tries to decide what to believe: ϕ or $\neg\phi$? In this particular case, it is easy to guess that an argument against ϕ is an argument in favor of $\neg\phi$. Moreover, these arguments are conflicting. Thus, for each of the two conclusions only one set of arguments is considered: the arguments in favor of it. The decision is then made on the basis of the quality of those arguments.

In a decision making problem, things seem different since in this case the decision maker wants to decide between different alternatives (what we call in this document decisions): d_1, …, d_n. For instance, consider the case of Peter and Mary who discuss about the place of their next holidays. They have three alternatives: Toulouse, Utrecht and Liverpool. Here each alternative cannot be true or false, but it can be more/less preferred to the other alternatives. In order to make this preference ordering, Mary and Peter compute the arguments in favor of and arguments against each alternative, then the alternatives are compared on the basis of those arguments. For instance, the alternative which has less arguments against it may be preferred. Note that in this case, for each alternative the pair <arguments in favor of, arguments against> is considered in the reasoning. It is also clear in this case, that an argument against Toulouse is not necessarily an argument in favor of Utrecht or Liverpool. For example, an argument against Toulouse is the fact that it is too warm. This argument is unfortunately against even Utrecht. Moreover, the arguments in favor of and against a given decision are not necessarily conflicting to.

Let us define two functions which return respectively for a given decision the arguments which are in favor of it and the arguments against it. Intuitively, an argument is in favor of a decision if that decision leads to the satisfaction of a positive goal. The arguments which recommend decisions are also in favor of that decision.

Definition 42 (Arguments PRO) *Let $d \in \mathcal{D}$ and $\mathcal{B} \subseteq \mathcal{A}_r \cup \mathcal{A}_d$.*

$$\texttt{ArgPRO}(d, \mathcal{B}) = \{A \in \mathcal{B} | \texttt{CONC}(A) \ = \ d \textit{ and } (\texttt{GOALS}^+(A) \neq \emptyset, \textit{ or } d \in \texttt{PROP}(A))\}.$$

An argument is against a decision if the decision leads to the satisfaction of a negative goal. Hence, arguments PRO a decision stress the *positive consequences* of the decision, while arguments CON are only focusing on the negative ones.

Definition 43 (Arguments CON) *Let $d \in \mathcal{D}$ and $\mathcal{B} \subseteq \mathcal{A}_d$.*

$$\texttt{ArgCON}(d, \mathcal{B}) = \{A \in \mathcal{B} | \texttt{CONC}(A) \ = \ d \textit{ and } \texttt{GOALS}^-(A) \neq \emptyset\}.$$

Note 5 *The notions of strict/defeasible/inconsistent (recommending/decision) argument are defined exactly in the same way as for epistemic arguments.*

3.4 Comparing arguments

In what follows $\succeq$ will denote any preference relation between arguments. For two arguments A and B, $A \succeq B$ means that A is at least as 'good' as B. $\succ$ denotes the strict ordering associated with $\succeq$. $A \succ B$ means that A is strictly preferred over B.

Throughout the document, we suppose that there exists a basic ordering $\succ$ on the set of arguments. This basic ordering captures the idea that strict arguments are preferred to the defeasible ones. Moreover, epistemic arguments always take precedence over arguments for decisions. The reason is that a decision cannot be well supported if the beliefs on which it is based are not justified. Finally, since recommending arguments are built from laws and obligations, it is natural to prefer a recommending argument to a decision one.

Definition 44 (Basic ordering) *Let A, B be two arguments. $A \succ B$ iff:*

- *A is strict and B is defeasible, or*
- *A is an epistemic argument and B is a non-epistemic argument, or*

- *A be a recommending argument and B a decision one.*

Recommending arguments are compared exctaly in the same way as epistemic arguments, i.e two recommending arguments are compared using either the last link principle or the weakest link principle defined respectively in [68] and [7].

Unlike epistemic (and recommending) arguments, arguments in favor of and arguments against decisions involve both *goals* and *beliefs*. Thus, the force of such arguments depends not only on the quality of beliefs used in these arguments, but also on the *importance* of the *satisfied* (resp. *violated*) goals.

This means that the knowledge base $\mathcal{K}$ is supposed to be equipped with a partial pre-ordering $>_k$, and the two bases of goals $\mathcal{G}^+$ and rejections $\mathcal{G}^-$ are equipped with a partial ordering $>_g$. Indeed, for two goals g_1 and g_2, $g_1 >_g g_2$ means that the goal g_1 is more important for the agent than the goal g_2.

Intuitively, a decision is 'good' if, according to the most certain beliefs, it satisfies an important goal. A decision is weaker if it involves beliefs with a low certainty, or if it only satisfies a goal with low importance. In other terms, the force of an argument represents to what extent the decision maker is certain that the decision will satisfy its most important goals.

This suggests the use of a *conjunctive* combination of the certainty and the priority of the most important satisfied (resp. violated) goal.

Definition 45 (Decision arguments) *Let A, B be two decision arguments. $A \succeq B$ iff:*

1. $\texttt{GOAL}^2(A) >_g \texttt{GOAL}(B)$, *and*
2. $\forall K \in \texttt{PREM}(A), \exists K' \in \texttt{PREM}(B)$ *such that* $K >_k K'$, *and*
3. $\forall R \in \texttt{DefRules}(A), \exists R' \in \texttt{DefRules}(B)$ *such that* $R >_r R'$.

In [4], different other principles for comparing decision arguments have been suggested.

3.5 Acceptability of arguments

In what follows, let $\mathcal{A} = \mathcal{A}_e \cup \mathcal{A}_r \cup \mathcal{A}_d$. The argumentation framework which will be used for making decisions is as follows:

Definition 46 (Argumentation framework) *Let $\mathcal{T}$ be a theory. An* argumentation framework *(AF) built on $\mathcal{T}$ is a pair <$\mathcal{A}$,* defeat> *s.t:*

- $\mathcal{A} = \mathcal{A}_e \cup \mathcal{A}_r \cup \mathcal{A}_d$,
- defeat *is the relation given in Definition 22 by replacing $\mathcal{A}_e$ by $\mathcal{A}$ and the preference relation between arguments by the new one.*

To compute the acceptable arguments, any semantics (grounded, preferred, complete, stable) can be used. Let $\mathcal{E} = \{E_1, \ldots, E_n\}$ be the set of extensions under a given semantics.

[2]The function $\texttt{GOAL}$ here stands for either $\texttt{GOAL}^+$ or $\texttt{GOAL}^-$.

3.6 Decision criteria

Elements of $\texttt{Output}(AF)$ are considered as true. Note that decisions are not inferred. The reason is that one cannot say that a given decision is true or false. A decision is intrinsic to the decision maker and depends on its preferences. The idea in a decision problem, is to construct the arguments in favor of and against each decision. Then among all those arguments, only the strong (acceptable) ones are kept and the different decisions are compared on the basis of them. Comparing decisions is an important step in a decision process. Below we present an example of *intuitive* principle which is reminiscent of classical principles in decision.

Definition 47 (Comparing decisions) *Let $\mathcal{T}$ be a theory, $AF = <\mathcal{A}, \text{defeat}>$ be an argumentation framework, and E its grounded extension. Let d_1, $d_2 \in \mathcal{D}$. Let $\texttt{ArgPRO}(d_1, E)$ = (P_1, ..., P_r) and $\texttt{ArgPRO}(d_2, E)$ = (P'_1, ..., P'_s). Each of these vectors is assumed to be decreasingly ordered w.r.t $\succeq$ (e.g. $P_1 \succeq \ldots \succeq P_r$). Let $v = min(r, s)$.*
A pre-ordering $\rhd$ on $\mathcal{D}$ is defined as follows: $d_1 \rhd d_2$ iff:

- *$P_1 \succ P'_1$, or*
- *$\exists\ k \leq v$ such that $P_k \succ P'_k$ and $\forall\ j < k$, $P_j \approx P'_j$, or*
- *$r > v$ and $\forall\ j \leq v$, $P_j \approx P'_j$.*

The above principle takes into account only the arguments pro, and prefers a decision which has at least one acceptable argument pro which is preferred (or stronger) to any acceptable argument pro the other decision. When the strongest arguments in favor of d_1 and d_2 have equivalent strengths (in the sense of $\approx$), these arguments are ignored.
In Deliverable D2.2 [4] three categories of principles for comparing decisions have been proposed: *unipolar* principles in which only arguments PRO or CON decisions are considered, *bipolar* principles in which both arguments PRO and CON are considered, and finally *non-polar* principles. These last consist of aggregating all the arguments PRO and against a decision into a unique argument, then to compare pairs of decisions on the basis of their aggregated arguments.

In [2], it has been shown that that modeling decision making and inference in the same framework does not affect the result of inference. Before that, let us define when two argumentation frameworks are equivalent.

Definition 48 (Equivalent frameworks) *Let $\mathcal{T}$ be a theory and $AF = <\mathcal{A}, \text{defeat}>$, $AF' = <\mathcal{A}', \text{defeat}'>$ be two argumentation frameworks built on the theory $\mathcal{T}$.*
The argumentation framework AF is equivalent *to the argumentation framework AF' iff $\texttt{Output}(AF) = \texttt{Output}(AF')$.*

In [2] it has been shown that an argumentation framework in which only epistemic arguments are taken into account will return exactly the same inferences as an argumentation framework is which all the different kinds of arguments are considered at the same time.

Proposition 11 *The two argumentation frameworks* $<\mathcal{A}_e$, defeat$_e>$ *and* $<\mathcal{A}$, defeat$>$ *are equivalent, with* defeat$_e \subseteq$ defeat *and* defeat$_e \subseteq \mathcal{A}_e \times \mathcal{A}_e$.

Chapter 4

A general system for dialogue

4.1 Introduction

With the rise of distributed computation, and especially the Internet, interaction and communication between distinct computational entities has become of increasing importance. Almost every request for a web-page sent across the World-Wide-Web, for example, will involve a dialogue between a requesting client machine and a requested server machine using the automated dialogue protocol known as Hyper-Text Transfer Protocol, or HTTP. Although this protocol was designed for transfers of static information (in fact, for sharing of physics research papers) between two parties — what are now called information-seeking dialogues — the protocol is now used for interactions between multiple parties, interactions which may involve co-ordination over actions and/or transfers of dynamic information or even computer programs, in addition to static information. The dialogues involved may include negotiation (as in e-commerce), deliberation (as in e-democracy and computer-supported group working), or persuasion (as in weblogs and chatboards), with such applications implemented conceptually at layers above that of the HTTP used for communication.

In parallel to these practical developments computer scientists have also begun to consider dialogue between intelligent and autonomous entities. These entities, which may be humans and/or software programs, are known as *agents*, and provide an effective means to conceptualize the distinct entities which comprise a distributed computational system. Agents will execute actions proactively to achieve some goal or goals, which they, to some greater or lesser extent, will have selected, using means which they have also selected [82]. Real-world applications of agent systems have become significant — for example, see the applications described in [63] — and many observers believe the agent paradigm will be as important in future software engineering as is the objects paradigm currently [49].

To the extent that agents are autonomous they cannot normally be commanded to act (or not act) in given situations by other agents; they can only be requested to act (or not act). To the extent that they are intelligent, agents will require reasons and justifications for such requests, since an intelligent agent would not necessarily accede to every request it receives. Systems of intelligent autonomous agents will therefore require argumentation to enable effective interaction and communications between the constituent agents.

This has been recognized in computer science, starting with the work of Parsons, Jennings and Sierra [59, 60], Kraus, Sycara and Evenchik [47], Reed [70] and Dignum and colleagues [27, 28]. Since Reed [70], work on argumentation-based dialogue in computer science has been strongly influenced by a model of human dialogues due to argumentation theorists Doug Walton and Erik Krabbe [81]. As part of an effort exploring commitment in dialogues, particularly in persuasion dialogues, Walton and Krabbe created a classification of dialogue types between human participants. This classification was based upon: firstly, the various beliefs (pertinent to the topic of the dialogue) which the participants have at the commencement of the dialogue; secondly, the goals of the individual participants at commencement; and, thirdly, the goals at commencement that are shared by the participants, goals we may view as those of the dialogue itself. With this structure, Walton and Krabbe identified six primary types of human dialogues (re-ordered from [81]):

Information-Seeking Dialogues: One participant seeks the answer to some question(s) from another participant, who is believed by the first participant to know the answer(s).

Inquiry Dialogues: The participants collaborate to answer some question or questions whose answers are not known to any one participant.

Persuasion Dialogues: One participant seeks to persuade another to endorse a belief or statement he or she does not currently endorse. These dialogues typically begin with one participant supporting a particular statement which another participant does not, and the first seeks to convince the second to adopt the statement. The second party may not share this goal of the dialogue.

Negotiation Dialogues: The participants bargain over the division of some scarce resource in a manner acceptable to all participants, with each individual party aiming to maximize his or her share. In these dialogues, the individual goals of the participants may well be in conflict.[1]

Deliberation Dialogues: Participants collaborate to decide what action or course of action to take in some situation. Participants commence the dialogue with a common belief that they share a responsibility to decide

[1]Note that this definition of Negotiation is that of Walton and Krabbe. Within Artificial Intelligence, the word *negotiation* is often used to refer to interactions involving other matters besides the division of scarce resources.

the course of action, which may be executed by others not present in the dialogue.

Eristic Dialogues: Participants quarrel verbally as a substitute for physical fighting, with participants aiming to give vent to feelings or emotions.

Significant recent effort in AI has now been devoted to formal models and studies of these different types of dialogue, including: Information-Seeking dialogues [29, 30]; Inquiry dialogues [53]; Persuasion dialogues [19, 41, 50, 55, 66, 72, 84]; Negotiation dialogues [47, 60, 73, 51, 77, 83]; and Deliberation dialogues [14, 52, 76]. Even Eristic dialogues have been studied, since they find potential application in the design of customer service scripts in call-centers, e.g., [37, 36]. Walton and Krabbe do not claim their classification is comprehensive, and indeed, other dialogue types have been described, for example, command dialogues [40] and various types of query dialogues [25].

The main purpose of this chapter is to present a framework in which dialogues and combinations of dialogues may be represented formally, and readily implemented. The work presented starts from the general formal framework for dialogues presented in Section 4 of ASPIC Deliverable D2.1 [5], and so this framework is first summarized again (informally) in Section 4.2. The following section, Section 4.3, then presents an instantiation of this framework with a Persuasion dialogue of the same form as was presented in Section 3.3 of ASPIC Deliverable D2.5 [6]. A formalization of this protocol using a variant of the well-known *Event Calculus* is then presented in Section 4.4. To demonstrate that this approach is not specific to the protocol selected, it is applied to another protocol developed prior to ASPIC, a Persuasion dialogue protocol, here called *PWA*, of Parsons, Wooldridge and Amgoud [62]; this is presented in Section 4.5. We next present an example of a protocol for negotiation dialogues containing embedded persuasion dialogues, along with some of the properties of the combined protocol, in Section 4.6. The design of agent dialogue policies (or strategies) under such a combined protocol is considered here, in the context of the *e-Consent Scenario* articulated in Section 6 of ASPIC Deliverable D2.3 [10]. The penultimate section, Section 4.7, then describes two implementations, one of the protocol studied in Sections 4.3 and 4.4, and the other of the Information-seeking dialogue protocol presented for the *e-Consent Scenario* [10]. Finally, the chapter concludes with a discussion of Combinations of Dialogues and the issues this raises, in Section 4.8.

4.2 A general model for dialogue

This section presents briefly and informally the framework for dialogues first presented in Section 4 of ASPIC Deliverable D2.1 [5]. It is presented here simply as a reminder of the framework and its elements, prior to the new material presented in the later sections of the Chapter. Dialogue systems comprise the following elements:

- A logical language for representation of topics, and a possibly non-monotonic logic for this language.
- A communication language for utterances of speech acts regarding the topics.
- A dialogue purpose.
- Finite sets of participants and of roles, with participants assigned to roles. Participants have a set of commitments and may have a databases of beliefs.
- A context, being that knowledge which all participants pre-suppose throughout the dialogue.
- A set of effect rules specifying the effects of utterances on the commitments of participants.
- A protocol for specifying the legal moves at each stage of the dialogue.
- Rules defining the outcomes of dialogues under the protocol.

Full details of the formal framework can be found in Section 4.2 of ASPIC Deliverable D2.1 [5].

4.3 A formal framework for persuasion dialogues

This section presents a formal framework for dialogue games for persuasion and instantiates it with some specific games. The formal definitions of the framework are repeated with some minor revisions from Section 3.3 of ASPIC Deliverable D2.5 [6]. For an introductory discussion, examples and proofs of formal results the reader is referred to that Deliverable. First, the essentials of logical systems for defeasible argumentation are sketched, which provide the logical basis for dialogue systems for argumentation. Then, the framework is briefly outlined with its fixed and variable elements indicated, after which it is formally defined.

The discussion in this section will, whenever possible, abstract from the logical structure of the individual reasoning of the dialogue participants. Nevertheless, some choices have to be made. Since argumentation typically involves defeasible reasoning, we need a nonmonotonic logic. Since we are dealing with dialogues for argumentation, one particular form of nonmonotonic logic is very appropriate, viz. argumentation systems. In particular, the framework presented will assume an argumentation system conforming to the *Dung-plus* format of Chapter 2 above, with inference defined according to grounded semantics.

4.3.1 The framework: general ideas

The framework allows for variations on a number of issues: for different underlying argument-based logics (but all with grounded semantics), for various sets of locutions, for different turntaking rules and different rules on whether multiple replies, postponing of replies and coming back to earlier choices is allowed. On the other hand, the framework imposes some basic common structure on all dialogues, most importantly, an explicit reply structure on moves, where each move either attacks or surrenders to one earlier (but not necessarily the last) move of the other player. This structure is exploited to allow for various degrees of coherence and flexibility when it comes to maintaining focus of a dialogue (for instance, whether they may make more than one move in a turn, or whether they may move alternative replies to a move). A reply structure is implicit in the protocols of many existing systems but usually not made explicit. It seems especially suited for "verbal struggles" (a term coined by [15] in their classification of speech act verbs). We do not claim, however, that all dialogues should or do conform to this structure. It may, for instance, be less suited for dialogues where the focus is more on inquiry or deliberation than on settling a conflict of opinion. Another assumption of the framework is that during a dialogue the players implicitly build a structure of arguments and counterarguments related to the dialogue topic.

Thus, according to the present approach a dialogue can be regarded in three ways. One can look at the order in which the moves are made, in which case a dialogue is regarded as a linear structure. One can also look at the reply relations between the moves, in which case the dialogue is conceived of as a tree. Finally, one can look at the arguments that are exchanged in reply to each other, in which case the dialogue is regarded as a dialectical structure of arguments and counterarguments.

4.3.2 The framework formally defined

Now the framework will be formally defined. All dialogues are assumed to be for two parties arguing about a single dialogue topic $t \in L_t$, the *proponent* (P) who defends t and the *opponent* (O) who challenges t. As for notation, for any player p, we define $\overline{p} = O$ iff $p = P$ and $\overline{p} = P$ iff $p = O$.

The top level definition of the framework is as follows.

Definition 49 *[Dialogue games for argumentation]. A dialogue system for argumentation (dialogue system for short) is a pair* $(\mathcal{L}, \mathcal{D})$*, where* $\mathcal{L}$ *is a logic for defeasible argumentation and* $\mathcal{D}$ *is a dialogue system proper.*

The elements of the top level definition are in turn defined as follows. Logics for defeasible argumentation are defined as an instance of Dung's abstract framework [33] with a specific, tree-based form of arguments and conforming to grounded semantics. Essentially, this format instantiates the definitions of Chapter 2 with any system for grounded semantics. Since for present purposes not all detail of Chapter 2 is needed, we will use a slightly modified notation.

Definition 50 *A logic for defeasible argumentation $\mathcal{L}$ is a tuple $(L_t, R, Args, \rightarrow)$, where L_t (the topic language) is a logical language, R is a set of inference rules over L_t, Args (the arguments) is a set of AND-trees of which the nodes are in L_t and the AND-links are inferences instantiating rules in R, and $\rightarrow$ a binary relation of defeat defined on Args. For any argument A, prem(A) is the set of leaves of A (its premises) and conc(A) is the root of A (its conclusion).*

An argumentation theory T_F within $\mathcal{L}$ (where $F \subseteq L_t$) is a pair $(A, \rightarrow_{/A})$ where A consists of all arguments in Args with only nodes from F and $\rightarrow_{/A}$ is $\rightarrow$ restricted to $A \times A$. T_{FS} is called finitary if none of its arguments has an infinite number of defeaters.

For any set $A \subseteq Args$ the information base $I(A)$ is the set of all formulas that are a premise of an argument in A. The closure $Cl(A)$ of a set of arguments $A \in Args$ is the argumentation theory $T_{I(A)}$.

An argument B extends an argument A if conc$(B) = \varphi$ and $\varphi \in$ prem(A). The concatenation of A and B (where B extends A) is denoted by $B \otimes A$. Defeasible inference in $\mathcal{L}$ is assumed to be defined according to grounded semantics. The defeat relation of $\mathcal{L}$ is assumed to satisfy the following property: if A defeats B, then for all C extending A and D extending B it holds that $C \otimes A$ defeats $D \otimes B$.

(Here, our L_t corresponds to Chapter 2's $\mathcal{L}$, R corresponds to $\mathcal{S} \cup \mathcal{R}$ and $\rightarrow$ corresponds to 'defeat'. For a precise definition of the notion of arguments as AND-trees see Definition 12.)

The idea of an argumentation theory is that it contains all arguments that are constructible on the basis of a certain theory or knowledge base. Note that each link of an argument corresponds to a (deductive or defeasible) inference rule in R. The present framework fully abstracts from the nature of these rules. Note also that the assumption on the defeat relation is not completely innocent: it is not satisfied in systems where arguments are compared on their 'weakest links', as, for instance, in the work of Pollock [64, 65].

Definition 51 *A dialogue system proper is a triple $\mathcal{D} = (L_c, P, C)$ where L_c (the communication language) is a set of locutions, Pr is a protocol for L_c, and C is a set of effect rules of locutions in L_c, specifying the effects of the locutions on the participants' commitments.*

A communication language is a set of locutions and two relations of attacking and surrendering reply defined on this set.

Definition 52 *A communication language is a tuple $L_c = (S, R_a, R_s)$, where S is a set of locutions and R_a and R_s are two binary relations of attacking and surrendering reply on S. Each $s \in S$ is of the form $p(c)$ where p is an element of a given set P of performatives and c either is a member or subset of L_t, or is a member of Args (of some given logic $\mathcal{L}$). Both R_a and R_s are irreflexive and in addition satisfy the following conditions:*

1. $R_a \cap R_s = \emptyset$

2. $\forall a, b, c : (a, b) \in R_a \Rightarrow (a, c) \notin R_s$

3. $\forall a, b, c : (a, b) \in R_s \Rightarrow (c, a) \notin R_a$

The function $att : R_s \longrightarrow \mathcal{P}(R_a)$ *assigns to each pair* $(a, b) \in R_s$ *one or more attacking counterparts* $(c, b) \in R_a$*.*

Condition (1) says that a locution cannot be an attack and a surrender at the same time, condition (2) says that a locution cannot be an attack on one locution and a surrender to another locution, and condition (3) says that surrenders cannot be attacked (this is since they effectively end a line of dispute).

The protocol for L_c is defined in terms of the notion of a dialogue, which in turn is defined with the notion of a move:

Definition 53 (Moves and dialogues.)

- *The set* M *of moves is defined as* $\mathbb{N} \times \{P, O\} \times L_c^p \times \mathbb{N}$*, where the four elements of a move* m *are denoted by, respectively:*
  - $id(m)$*, the identifier of the move,*
  - $pl(m)$*, the player of the move,*
  - $s(m)$*, the speech act performed in the move,*
  - $t(m)$*, the target of the move.*
- *The set of dialogues, denoted by* $M^{\leq\infty}$*, is the set of all sequences* $m_1, \ldots, m_i, \ldots$ *from* M *such that*
  - *each* i^{th} *element in the sequence has identifier* i*,*
  - $t(m_1) = 0$*;*
  - *for all* $i > 1$ *it holds that* $t(m_i) = j$ *for some* m_j *preceding* m_i *in the sequence.*

  The set of finite dialogues, denoted by $M^{<\infty}$*, is the set of all finite sequences that satisfy these conditions. For any dialogue* $d = m_1, \ldots, m_n, \ldots$*, the sequence* $m_1, \ldots, m_i$ *is denoted by* d_i*, where* d_0 *denotes the empty dialogue.*

Note that the definition of dialogues implies that several speakers cannot speak at the same time.

When $t(m) = id(m')$ we say that m *replies to* m' in d and also that m' is the *target of* m in d. We will sometimes slightly abuse notation and let $t(m)$ denote a move instead of just its identifier. When $s(m)$ is an attacking (surrendering) reply to $s(m')$ we will also say that m is an attacking (surrendering) reply to m'.

A protocol also assumes a turntaking rule.

Definition 54 *A turntaking function* T *is a function*

- $T : M^{<\infty} \longrightarrow \mathcal{P}(\{P, O\})$

such that $T(\emptyset) = \{P\}$. A turn of a dialogue is a maximal sequence of stages in the dialogue where the same player moves.

When $T(d)$ is a singleton, the brackets will be omitted. Note that this definition allows that more than one speaker has the right to speak next.

We are now in the position to define the central element of a dialogue game, the 'rules of the game', in other words, the protocol.[2]

Definition 55 *A protocol on M is a set $P \subseteq M^{<\infty}$ satisfying the condition that whenever d is in P, so are all initial sequences that d starts with.*

A partial function $Pr : M^{<\infty} \longrightarrow \mathcal{P}(M)$ is derived from P as follows:

- *$Pr(d)$ = undefined whenever $d \notin P$;*
- *$Pr(d) = \{m \mid d, m \in P\}$ otherwise.*

The elements of dom(Pr) (the domain of Pr) are called the legal finite dialogues. The elements of $Pr(d)$ are called the moves allowed after d. If d is a legal dialogue and $Pr(d) = \emptyset$, then d is said to be a terminated dialogue.

All protocols are further assumed to satisfy the following basic conditions for all moves m_i and all legal finite dialogues d.

If $m \in Pr(d)$, then:

- *R_1: $pl(m) \in T(d)$;*
- *R_2: If $d \neq d_0$ and $m \neq m_1$, then $s(m)$ is a reply to $s(t(m))$ according to L_c;*
- *R_3: If m replies to m', then $pl(m) \neq pl(m')$;*
- *R_4: If there is an m' in d such that $t(m) = t(m')$ then $s(m) \neq s(m')$.*
- *R_5: For any $m' \in d$ that surrenders to $t(m)$, m is not an attacking counterpart of m'.*

Together these conditions capture a lower bound on coherence of dialogues. Note that they state only necessary conditions for move legality. Rule R_1 says that a move is legal only if moved by the player-to-move. R_2 says that a replying move must be a reply to its target according to L_c, and R_3 says that one cannot reply to one's own moves. Rule R_4 states that if a player backtracks, the new move must be different from the first one. ('backtracking' in this chapter is taken to mean any alternative reply to the same target in a later turn). Finally, R_5 says that surrenders may not be 'revoked'. At first sight, it would seem that R_5 could be formulated as "$t(m)$ does not have a surrendering reply in d". However, later we will see that it makes sense to attack one premise of an argument even if another of its premises has been surrendered.

Finally, a commitment function is a function that assigns to each player at each stage of a dialogue a set of propositions to which the player is committed at that stage.

[2]The first part of the following definition is taken from [16, p. 160].

Table 4.1: Speech acts for liberal dialogues

Acts	Attacks	Surrenders
claim φ	*why* φ	*concede* φ
why φ	*argue* A $(conc(A) = \varphi)$	*retract* φ
argue A	*why* φ $(\varphi \in prem(A))$ *argue* B (B defeats A)	*concede* φ $(\varphi \in prem(A)$ or $\varphi = conc(A))$
concede φ		
retract φ		

Definition 56 *A commitment function is a function*

- C: $M^{\leq \infty} \times \{P, O\} \longrightarrow \mathcal{P}(L_t)$.

such that $C_\emptyset(p) = \emptyset$. $C_d(p)$ *denotes player p's commitments in dialogue d.*

4.3.3 Liberal dialogue systems

So far any 'verbal struggle' could fit the framework. It will now be specialised for argumentation with a particular communication language and some basic protocol rules motivated by this language. In fact, a class of liberal dialogue systems will be defined (parametrised by a logic $\mathcal{L}$), in which the participants have much freedom, and which is intended to be the core of all other dialogue systems of this study.

The communication language allows for making a claim, for challenging, conceding and retracting a claim, for supporting a claim with an argument, and for attacking arguments with counterarguments or by challenging their premises.

Commitment rules are defined, but the commitments are only used in defining termination and outcome of dialogues; they do not constrain move legality (see for that Section 4.3.6 below). There are only weak relevance requirements of moves, viz. those given by the reply structure of L_c, and there are no restrictions at all on length of turns. Basically, a speaker may continue speaking as long as he is not interrupted by the listener, and he may make any move as long as according to L_c it is a well-formed reply to some earlier move of the listener. So liberal dialogues greatly rely for their coherence on the cooperativeness of the dialogue participants.

4.3.3.1 The communication language

The communication language is listed in Table 4.1. In examples below, when an argument contains a single inference, it will usually be listed as *conclusion since premises*. Note that counterarguments must defeat their target according to $\mathcal{L}$. Attacking counterparts of a surrender are at the same line of the surrender except for the second line of the *argue* A row: *argue* B is an attacking counterpart of *concede* φ only if the conclusion of B negates or is negated by φ. (So the attacking counterpart of conceding a premise is an premise-attack and the attacking counterpart of conceding a conclusion is a rebuttal.)

4.3.3.2 The commitment rules

The following commitment rules seem to be uncontroversial and can be found throughout the literature. (Below s denotes the speaker of the move; effects on the other parties' commitments are only specified when a change is effected.)

- If $s(m) = \mathit{claim}(\varphi)$ then $C_s(d,m) = C_s(d) \cup \{\varphi\}$
- If $s(m) = \mathit{why}(\varphi)$ then $C_s(d,m) = C_s(d)$
- If $s(m) = \mathit{concede}(\varphi)$ then $C_s(d,m) = C_s(d) \cup \{\varphi\}$
- If $s(m) = \mathit{retract}(\varphi)$ then $C_s(d,m) = C_s(d)\ - \{\varphi\}$
- If $s(m) = \mathit{argue}(A)$ then $C_s(d,m) = C_s(d) \cup \mathit{prem}(A) \cup \{\mathit{conc}(A)\}$

4.3.3.3 Turntaking

As for the turntaking function, proponent starts with a unique move (which introduces the topic of the dialogue), opponent then replies and after that it is simply assumed that it is always the speaker's turn; in other words, the turn shifts as soon a new speaker succeeds in saying something.

- T_L: $T(d_0) = P$, $T(d_1) = O$, else $T(d) = \{P, O\}$.

Thus protocol rule R_1 is always satisfied for any dialogue with at least two moves.

4.3.3.4 The protocol

The protocol for liberal dialogues adds two protocol rules to those of the general framework.

If $m \in Pr(d)$, then:

- R_6: If $d = \emptyset$, then $s(m)$ is of the form *claim*(φ) or *argue* A.
- R_7: If m concedes the conclusion of an argument moved in m', then m' does not reply to a *why* move.

R_6 says that each dialogue begins with either a claim or an argument. The initial claim or, if a dialogue starts with an argument, its conclusion is the *topic* of the dialogue. R_7 restricts concessions of an argument's conclusion to conclusions of counterarguments. This ensures that propositions are conceded at the place in which they were introduced. Consider the following dialogue:

P_1:	*p since q*
O_2:	*why q*
P_3:	*q since r*
$O_4[P_3]$:	*concede q*

R_7 invalidates O_4 as a reply to P_3; it should instead be targeted at P_1, which is when the proponent introduced q.

Definition 57 *A dialogue system for liberal dialogues is now defined as any dialogue system with L_c as specified in Table 4.1, with turntaking rule T_L and such that a move is legal if and only if it satisfies protocol rules R_1-R_7.*

Note that systems for subsets of L_c can be defined as slight variations of systems for liberal dialogues by simply declaring the use of certain moves illegal at all times.

4.3.3.5 Termination and outcome of dialogues

Next termination and outcome of dialogues must be defined. In practice, termination of dialogues is often conventional so that an 'any time' definition of a dialogue outcome is called for.

In the philosophical literature on two-party-persuasion, the most usual termination criterion is that a dialogue terminates if and only if the opponent concedes proponent's main claim or the proponent retracts his main claim. Above in Definition 55 instead the usual 'mathematical' approach was followed, in which a dialogue is defined as terminated just in case no legal continuation is possible. So to capture the 'philosophical' definition, a dialogue system should ideally be defined such that the players run out of legal moves just in case the main claim is conceded or retracted.

However, more can be said about termination of dialogues. In general the individual knowledge bases of the players will evolve during a dialogue: the players may learn from each other, they may ask advice of third parties, or they may perform other knowledge-gathering actions, such as consulting databases or making observations. For this reason, a player will rarely run out of attacking moves, since it is (theoretically) always possible to find an argument for a claim or a counterargument to an argument. So it will rarely be possible to *force* the other player to concede or retract the main claim. In addition, a 'filibustering' player can always challenge the premises of any new argument. For these reasons realistic dialogues will often not terminate by retraction or concession of the main claim, but by external agreement or decision to terminate it, so formal termination results are of limited practical value.

When the traditional philosophical termination rule is adopted, the obvious outcome rule is to declare proponent the winner if opponent has conceded his main claim and to declare opponent the winner if proponent has retracted his main claim. The winner can then be defined such that the proponent wins if the opponent has conceded his main claim and the opponent wins if the proponent has retracted his main claim.

However, for dialogues that can terminate by convention this may in certain contexts be too restrictive; a player may avoid losing simply by never giving in and continue debating till the other player becomes tired and agrees to terminate. To deal with contexts where this is undesirable, 'any time' outcome definitions need to be studied, which allocate 'burdens to attack' to the players, so that if at a certain dialogue stage a participant has not yet fulfilled his burden to attack, he may be the 'current' loser even if he has not conceded (opponent) or retracted (proponent) the main claim. Besides for identifying the current winner, such a notion can also be used to regulate turntaking and to define relevance of moves, as will be explained in detail in Section 4.3.4.

Consider the following simple liberal dialogue:

P_1: *claim p*
O_2: *why p*

At this stage it seems reasonable that P's main claim is not successfully defended, since there is an unanswered challenge. So P has the burden to attack this challenge on the penalty of being the current loser. Suppose P fulfills this burden with

P_3: *p since q*

Then it seems reasonable to say that P's claim is successfully defended, since its only challenge has been met, so the burden to attack has shifted back to the opponent.

One way to define an 'any time' outcome notion is simply to apply a 'black-box' logical proof theory for $\mathcal{L}$ to the premises of all arguments moved at a certain dialogue stage that are not challenged or retracted. If the main claim is justified in $\mathcal{L}$ on the basis of these premises, proponent is the current winner, otherwise opponent is the current winner. This is the approach adopted in, for example, [19, 41, 48]. However, we argue that a more natural approach is to incorporate the proof theory of $\mathcal{L}$ into the dialogue protocol as much as possible, and then to prove that the dialogue outcome corresponds to what logically follows. Thus the protocol is arguably more realistic as a model of human dialogues, which may be beneficial in several contexts.

To this end we now define an any-time outcome notion that does not appeal to a black-box logical consequence notion. The definition is in terms of the *dialogical status* of an attacking move, which formalises the informal ideas explained in the previous subsection. The definition assumes a notion of a surrendered move, which needs to be defined separately for each instantiation of the framework of Section 4.3.

Definition 58 *[Dialogical status of moves] All attacking moves in a finite dialogue d are either in or* out *in d. Such a move m is in iff*

1. *m is surrendered in d; or else*
2. *all attacking replies to m are out*

Otherwise m is out.

Definition 59 (The current winner of a dialogue) *The status of the initial move m_1 of a dialogue d is in favour of $P(O)$ and against $O(P)$ iff m_1 is in (out) in d. We also say that m_1 favours, or is against p. Player p currently wins dialogue d if m_1 of d favours p.*

For liberal dialogue systems and all further systems to be discussed in this chapter the notion of a surrendered move is defined as follows.

Definition 60 *A move m in a dialogue d is surrendered in d iff*

- *it is an argue A move and it has a reply in d that concedes A's conclusion; or else*
- *m has a surrendering reply in d.*

Proposition 12 *For each finite dialogue d there is a unique dialogical status assignment.*

A counterexample for infinite dialogues is an infinite sequence of attacking moves $m_1, m_2, \ldots, m_i, \ldots$ each replying to the immediately proceding move. This dialogue has two dialogical status assignments: one in which all even moves are *in* and all odd moves are *out*, and one with the converse assignments.

Now the 'current' winner of a dialogue can be defined as follows:

Definition 61 *For any dialogue d the proponent wins d if m_1 is in, otherwise the opponent wins d.*

4.3.4 Protocols for relevant dialogues

In this section it will be shown that liberal protocols only weakly enforce structural coherence of dialogues and then two notions of strong and weak (structural) relevance of moves will be defined that remedy this.

4.3.4.1 Motivation

Liberal protocols promote relevance through the protocol rules R_2, R_4 and R_5. According to these rules, Figure 4.1 displays two legal liberal dialogues, sharing the first three moves. In dialogue 1 move O_3 is in a certain sense superfluous since with O_2 the opponent already launched another attack on P_1. Also, P in his second turn first attacks O's argument for $\neg q$ with P_4 and then retracts q with P_5. Arguably, this behaviour of the proponent is not very coherent: first he counterattacks an attack on his initial argument and then he surrenders to another attack on that argument. Dialogue 2 displays a variant of such rather incoherent behaviour: proponent first concedes the conclusion of O_3's argument with P_4 and then attacks its premise with P_5.

Figure 4.2 displays another legal liberal dialogue that is not entirely coherent. Here the opponent in his third turn attacks with O_7 in a line of the dialogue which P has meanwhile implicitly retreated with P_5: his current reason for q is v. Arguably O's argument for $\neg u$ is at this point irrelevant for the dialogue topic.

These dialogues illustrate that there is a need for stricter protocols, where each move is relevant to the dialogue topic. A rigorous way to enforce relevance

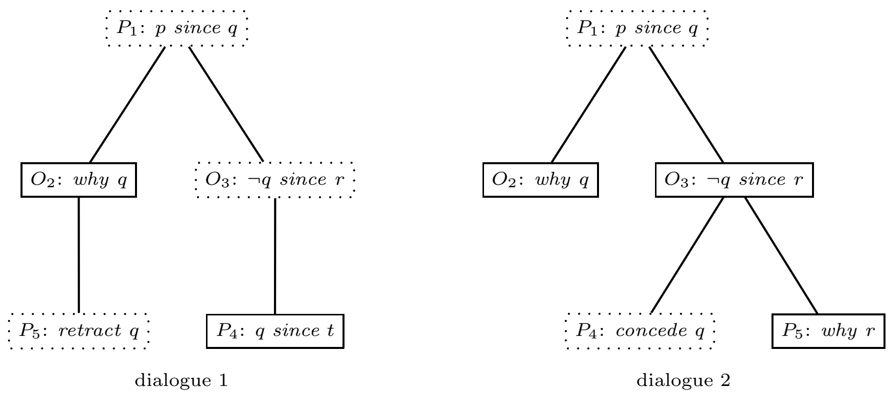

Figure 4.1: Two somewhat incoherent dialogues.

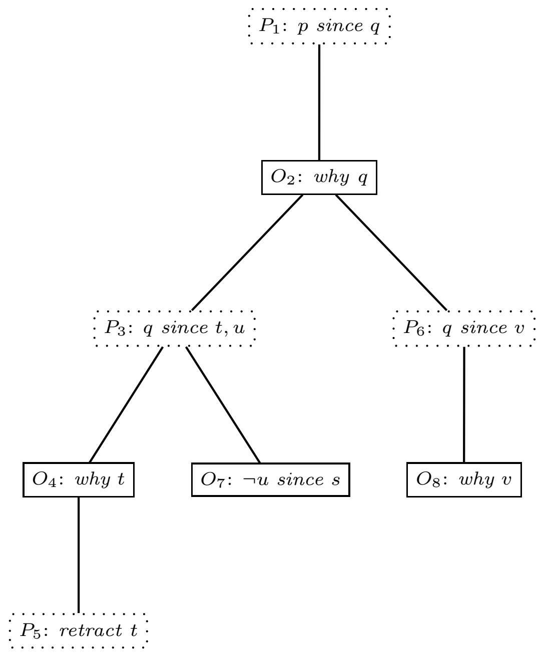

Figure 4.2: Another somewhat incoherent dialogue.

of moves is to have a unique-move and unique-reply protocol. Then in the dialogues of Figure 4.1 O's first turn ends after his challenge of q, after which P has to choose between retracting q or defending it. And in Figure 4.2 the dialogue ends after P_5's retraction and P is penalised for making the 'wrong' choice of argument at P_3. However, this comes with a price. Firstly, as argued in the introduction, it may not in all contexts be fair to disallow the players to repair mistakes or to move alternative arguments in the same turn. Secondly, there are more subtle reasons to allow backtracking. Consider the implied arguments of a dialectical graph. Fairness demands that when such arguments are relevant for the outcome of a dialogue, their moving should be legal at least at one stage during the dialogue. However, as shown in [67], the more a protocol restricts the possibility to move alternatives to earlier moves, the more it runs the risk of making such arguments illegal at any stage. So unique-reply protocols can be unfair.

In general, any 'any time' definition of the dialogue outcome can be used to constrain turntaking and promote relevance: for instance, protocols could be made immediate-reply and and all moves can be required to have an effect on the outcome of the dialogue. These ideas will now be made more precise in terms of the dialogical status of moves.

Intuitively, a replying move is structurally relevant if it is capable of changing the dialogical status of the initial move, given the various ways the players have backtracked and surrendered. Two typical grounds for irrelevance of a move are that it is made in a dialogue branch from which the other adversary has retreated (cf. move O_7 in Figure 4.2), or in a dialogue branch containing a surrendered move, of which the status therefore cannot be changed (cf. move P_5 in dialogue 2 of Figure 4.1).

4.3.4.2 Relevance defined

The requirement that each move be relevant allows the players maximal freedom on issues such as backtracking and postponing replies while yet ensuring a strong focus of a dialogue. The present notion of relevance extends the one of [67], which only applied to argument games.

As for the formal definition of relevance, as just explained, what is crucial is a move's effect on the status of the initial move. In order to determine relevance of surrendering moves, their effect is checked as if they were their attacking counterpart. Thus a move is relevant iff any attacking counterpart with the same target would change the status of the initial move of the dialogue. This is formally defined as follows.

Definition 62 *[Relevance] An attacking move in a dialogue d is* relevant *iff it changes the dialogical status of d's initial move. A surrendering move is relevant iff its attacking counterparts are relevant.*

This definition is not meant to fully capture the notion of relevance in dialogue, since it only looks at structure of a dialogue, not at its content. Clearly, an argument like 'You owe me 50,000 dollars since the earth is round' violates a legal sense of relevance, but it might well be relevant in the present sense.

Together the above definitions imply that a reply to a surrendered move is never relevant. Note also that, if not surrendered, an irrelevant target can become relevant again later in a dialogue, viz. if a player returns to a dialogue branch from which s/he has earlier retreated.

To illustrate these definitions, consider Figure 4.3 (where + means *in* and - means *out*). The dialogue tree on the left is the situation after P_7. The tree in the middle shows the dialogical status of the moves when O has continued after P_7 with O_8, replying to P_5: this move does not affect the status of P_1, so O_8 is irrelevant. Finally, the tree on the right shows the situation where O has instead continued after P_7 with O'_8, replying to P_7: then the status of P_1 has changed, so O'_8 is relevant.

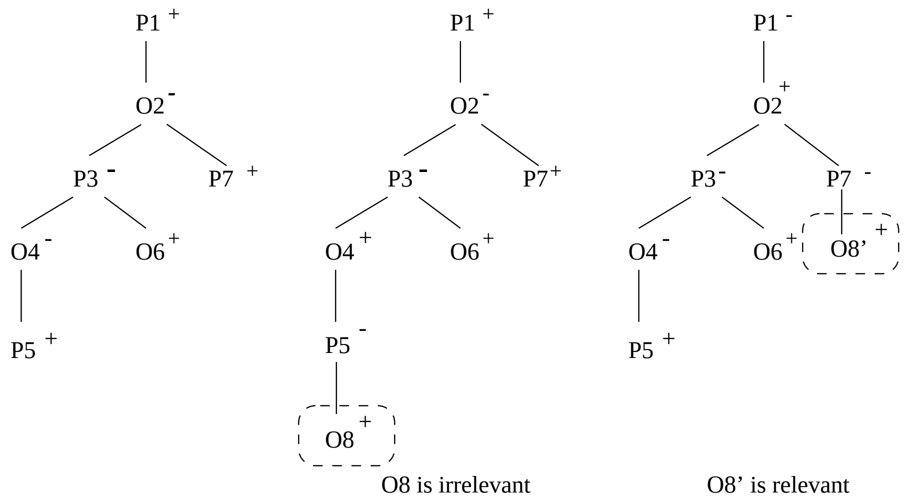

Figure 4.3: Dialogical status of moves.

To be a protocol for relevant dialogue, a protocol must also satisfy some additional conditions on the notions of move legality, turntaking and winning. The following protocol rule is added to those of liberal dialogue systems.

If $m \in Pr(d)$, then:

- R_8: if m is a replying move, then m is relevant in d.

To prevent premature termination of a dialogue this rule must be combined with an immediate-reply turntaking rule (cf. [48]):

- T_i: $T(d_0) = P$ and if $d \neq d_0$ then $T(d) = p$ iff $\overline{p}$ currently wins d.

Together, R_8 and T_i enforce that when a player is to move, s/he keeps moving until s/he has changed the status of the initial move his or her way (since after such a change no further move of the same player can be relevant). In other words, each player first moves zero or more relevant surrenders, and then moves zero or one relevant attacker: if no attackers are moved, this is because the player has no legal moves.

Definition 63 *A dialogue system for relevant dialogues is any dialogue system with L_c as specified in Table 4.1, with turntaking rule T_i and such that a move is legal if and only if it satisfies protocol rules R_1-R_8.*

4.3.5 Protocols for weakly relevant dialogues

Comparing systems for liberal and for relevant dialogues, the main advantage of the relevance requirement is that it keeps a dialogue focussed by ensuring that no resources are wasted on 'superfluous' moves, i.e., moves that have no bearing on the status of the initial move. However, there are reasons to study a weaker sense of relevance. Perhaps the main drawback of the relevance condition is that it must be combined with an immediate-reply turntaking rule, which prevents the moving in one turn of alternative ways to change the status of the initial move. This may be a drawback, for instance, in discussions where the parties cannot immediately reply to each other, and therefore reply to all moves of the preceding turn (as in parliamentary debate).

This disadvantage of the relevance rule can be met with a weakening of the notion of relevance, to require only that each attacking move creates a new 'winning part' of the speaker or removes a 'winning part' of the hearer. First a winning part of a dialogue must be defined. Informally, it is the part of a dialogue that 'makes' the initial move have its dialogical status.

Definition 64 *Let d be a dialogue currently won by player p. A winning part d^p of d is recursively defined as follows.*

1. *First include m_1;*
2. *for each move m of p that is included, if m is surrendered, include all its surrendering replies, otherwise include all its attacking replies;*
3. *for each attacking move m of $\overline{p}$ that is included, include one attacking reply m' that is in in d.*

The idea of this definition is that, by omitting all moves of p that are surrenders or from which p has backtracked, d^p contains that part of d that makes p win. In general, d^p is not unique, since p might have moved alternative attacking replies to a move, neither of which were successfully challenged by $\overline{p}$.

We can now define the notion of weak relevance.

Definition 65 *[Weak relevance.] An attacking move in a dialogue d is* weakly relevant *iff it creates a new winning part of d for the speaker or removes a winning part of the hearer. A surrendering move is weakly relevant iff its attacking counterparts are weakly relevant.*

Relevance according to Definition 62 will now be called *strong relevance*. Clearly, each strongly relevant move is also weakly relevant. The relevance rule is now weakened as follows:

- R_8': if m is a replying move, then m is weakly relevant in d.

Finally, the turntaking rule is relaxed as follows.

- T_w: $T(d_0) = P$. If $d_i \neq d_0$ then $T(d_i) = pl(m_i)$ if $\overline{pl(m_i)}$ currently wins d and $T(d_i) = \{P, O\}$ if $pl(m_i)$ currently wins d.

This says that interrupting the speaker is allowed but not obligatory as soon as the speaker has made himself the current winner.

Definition 66 *A dialogue system for weakly relevant dialogues is any dialogue system with L_c as specified in Table 4.1, with turntaking rule T_w and such that a move is legal if and only if it satisfies protocol rules R_1-R_7, R_8'.*

The structure of weakly relevant dialogues differs in two main respects from that of strongly relevant dialogues. Firstly, a player has some freedom to make additional moves after he has made himself the current winner, possibly creating additional winning parts. Secondly, each player must counterattack all attacks of the other player in order to make himself the current winner.

It is straightforward to prove that the soundness and fairness results for liberal and relevant dialogues still hold for weakly relevant dialogues.

To illustrate the weak notion of relevance, consider again the dialogue between Paul and Olga from the introduction. A weakly relevant version of this dialogue is when Olga moves her second argument for 'not safe' directly after her first in the same turn, after which Paul must attack both of them in his next turn to change the status of his main claim.

4.3.6 Respecting commitments

A further means to promote coherence of dialogues is by using the players' commitments in regulating move legality. Players can, for instance, be required to keep their own commitments consistent or restore consistency upon demand, or not to challenge their own commitments. In this section the addition will be studied of protocol rules referring to commitments to the protocols discussed so far. However, since commitments are a topic of their own, the discussion will be restricted to some simple rules and a more advanced treatment of commitments will be left for future research.

If $m \in Pr(d)$ and $pl(m) = p$, then:

- R_9: $C_p(d, m)$ is consistent.
- R_{10}: If $s(m) = \textit{why } \varphi$, then $C_p(d) \not\vdash \varphi$.

R_9 ensures logical consistency of the players' commitments and R_{10} prevents players from challenging a proposition to which they are themselves committed.

It should be noted that in the preset stup there is a tension between the effects of moves as replies to targets, which are local, and their effects as operations on commitment sets, which are global. As a reply to a target, a move's direct effect is on the target's dialogical status. As an operation on commitments, a move's direct effect is on the speaker's commitment set, which is global to the dialogue. The dialogue in Figure 4.4 illustrates this tension. After P_8 the protocols of this study allow opponent to continue with challenging s in P_5 and proponent to reply to this challenge with an argument for s. This dialogue is not very coherent since proponent already retracted s. Additional conventions

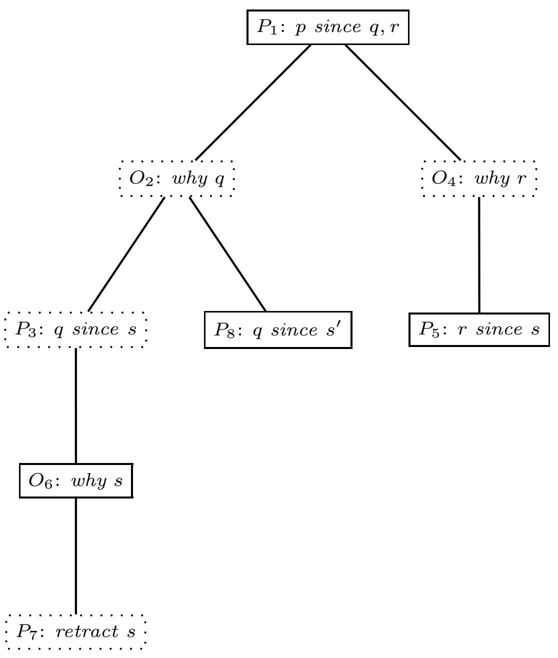

Figure 4.4: Replies vs. commitments.

could be added that reflect the global nature of commitments, for example, that a proposition may only be supported with an argument if the speaker is still committed to the proposition, or that once a proposition is retracted, no new commitment to the same proposition may be incurred by the same player.

4.4 Formalising dialogue games with the Event Calculus

Logical specification of dialogue systems could benefit both the formal investigation of such protocols and their implementation in declarative programming languages. Yet so far few persuasion systems have been fully formally specified in a declarative way. Two notable exceptions are [12] and [19]. However, the argumentation protocols formalised in these papers are rather simple and the logical specification of more complex protocols like the ones just mentioned still awaits further investigation. This section therefore studies the logical specification of the persuasion dialogue games discussed above. These games will first be formalised in a variant of the Event Calculus proposed by [75] and then implemented as a Prolog program (in Section 4.7).

4.4.1 The Event Calculus

The Event Calculus (EC) is a theory specified in first-order logic about events and their effects on states-of-affair in the world (called 'fluents' in EC). EC was originally developed by Kowalski and Sergot in 1986 [46]; in this deliverable we use a later variant, due to Shanahan [75], called the 'Full Event Calculus'. Its axioms express principles like 'If an event happens at time T that initiates a some fluent then that fluent starts to hold at T' and 'If a fluent holds at T and nothing terminates it then it also holds at $T+1$'. The latter is commonly called the 'law of inertia'; it can be overruled by axioms expressing when a fluent is terminated. In applications of EC its general axioms must be supplemented with domain-specific axioms.

Our use of EC for the specification of dialogue protocols is motivated by the fact that a dialogue game can be seen as a dynamic system where dialogue utterances are events that initiate and terminate various aspects of the 'dialogical world', such as a player being the player-to-move or not, player being committed to a certain proposition or not and a move being legal or not. Such aspects will be modelled as fluents, the value of which can change as an effect of utterances made during the dialogue.

A particularly attractive feature of EC is that it can easily be implemented as a Prolog program. This allows the modelling of temporal persistence of fluents through negation as failure: if termination of a fluent cannot be derived, it can be assumed to persist. Thus, for instance, a proposition added to a player's commitments can be assumed to remain a commitment until this is explicitly terminated. Also, it can be elegantly modelled that the present protocol allows replies to any earlier move in the dialogue and not only to the last move. This is modelled by the fact that the legality of a reply persists until it is explicitly terminated.

To be able to reason about fluents, they are *reified* in EC. Reification means that the fluents are treated as first-class objects so that they can be used as arguments of predicates. For example, the sentence "at time point 2 it is the turn of participant P" can be represented as follows:

$$HoldsAt(Turn(P), 2)$$

Here the statement *Turn(P)* is reified as an object to allow it to an be argument of the predicate *HoldsAt*.

The axioms of the Full EC make use of a number of special predicates. We now describe their informal meaning and indicate how they can be used in the specification of dialogue games. The first two predicates concern the effects of an action on the value of a fluent.

Initiates(α, β, τ) means that fluent β starts to hold after action α at time τ. This formula will be used in the following ways. Firstly, it will be used to express the addition of a statement to a player's commitment set in effect of an utterance. For instance,

$$Initiates(move(1, P, claim\ q, 0), CS(P, q), 1)$$

says that proponent's claim in his first move of q adds q to his commitments (the move identifier following the speech act is the move's target, in this case a dummy value to express that the claim is the dialogue's first move). In a similar way it can be expressed that in effect of an utterance another move becomes legal. For instance,

$$\begin{array}{l} \mathit{Initiates}(\\ \mathit{move}(1, P, \mathit{claim}\ q, 0), \\ \mathit{Legal}(\mathit{move}(\mathit{id}, O, \mathit{why}\ q, \mathit{id}), t)) \end{array}$$

says that proponent's claiming of q in his first move initiates the legality of a challenge of q by opponent. Finally, the fact that a move makes a player the player-to-move can be expressed in a similar way.

Terminates(α, β, τ**)** means that fluent β ceases to hold after action α at time τ. This predicate is the 'mirror predicate' of *Initiates*: in can be used in an analogous way as that predicate for expressing the deletion of a commitment, the termination of legality of a move and the termination of a player being the one to move.

At the beginning of a dialogue certain fluents will hold and certain fluents will not hold. The following two predicates can be used to express the begin situation of a dialogue.

Initially$_P(\beta)$ means that fluent β holds at the beginning of the dialogue. This formula will be used to define the commitment set of the players at the beginning of the dialogue and for defining the legal moves at the beginning of the dialogue.

Initially$_N(\beta)$ means that fluent β does not hold at the beginning of the dialogue. The formula will be used to define which moves are not legal at the beginning of the dialogue, which participant is not allowed to make a move and which propositions a participant is not committed to.

To express that a fluent holds at a certain time point the following predicate is used.

HoldsAt(β, τ**)** means that fluent β holds at time point τ. For instance, the formula

$$\mathit{Holdsat}(\mathit{Legal}(\mathit{move}(4, O, \mathit{why}\ q, 1), 4))$$

expresses that a challenge of q by opponent is legal at move 4 as a reply to move 1. Such formulas will (often with variables) be used in the conditions of the rules for move legality, turntaking and termination.

A dialogue is a sequence of events that happen. To express that a move is made the following predicate can be used.

Happens$(\alpha, \tau 1, \tau 2)$ means that action α starts at time point $\tau 1$ and ends at time point $\tau 2$. This predicate will be used to express all moves made during the dialogue. Since a dialogue move is assumed to have no duration, it will always hold that $\tau 1 = \tau 2$.

Besides persistence of a fluent, it must be possible to express *termination* and *initiation* of fluents at certain time points. This can be done with the predicates *Clipped* and *Declipped*; they are used in the general axioms of EC but we will not use them in our domain-specific axioms.

Clipped$(\tau 1, \beta, \tau 2)$ means that fluent β is terminated between times $\tau 1$ and $\tau 2$.

Declipped$(\tau 1, \beta, \tau 2)$ means that fluent β is initiated between times $\tau 1$ and $\tau 2$.

We next list the general axioms of the full Event Calculus. Following the usual conventions, variables are assumed to be implicilty universally quantified. The unique-name axioms, which are part of the Event Calculus, are left implicit, as well as the usual definitions of (in)equality. The first two axioms concern the conditions that should be met in order for a fluent to *persist*. The third axiom concerns the conditions for a fluent to *terminate to hold*. Then three axioms are presented which express exactly the opposite. The fourth axiom expresses when a fluent does *not persist* and the fifth and sixth axiom express what conditions should be met for a fluent to *start to hold*. The last axiom ensures that an event takes a non-negative amount of time. Note that in our Prolog implementation the occurrences of classical negation $\neg$ will be implemented as negation-as-failure, to capture the law of inertia.

1. $\mathit{HoldsAt}(f, t) \leftarrow \mathit{Initially}_P(f) \wedge \neg \mathit{Clipped}(0, f, t)$
   This axiom states that if a fluent initially holds and is not terminated between time point 0 and time point t then the fluent still holds at time point t.

2. $\mathit{HoldsAt}(f, t3) \leftarrow \mathit{Happens}(a, t1, t2) \wedge \mathit{Initiates}(a, f, t1) \wedge (t2 < t3) \wedge \neg \mathit{Clipped}(t1, f, t3)$
   This axiom states that if event a which initiates fluent f occurs then fluent f starts to hold until fluent f is terminated.

3. $\mathit{Clipped}(t1, f, t4) \leftrightarrow \exists a, t2, t3(\mathit{Happens}(a, t2, t3) \wedge (t1 < t3) \wedge (t2 < t4) \wedge \mathit{Terminates}(a, f, t2))$
   This axiom states that if and only if there exists an event a which occurs and terminates fluent f then fluent f is said to be *clipped*.

4. $\neg \mathit{HoldsAt}(f, t) \leftarrow \mathit{Initially}_N(f) \wedge \neg \mathit{Declipped}(0, f, t)$
   This axiom states that if a fluent did not initially hold and was not initiated between time point 0 and time point t then the fluent does not hold at time point t.

5. $\neg HoldsAt(f, t3) \leftarrow Happens(a, t1, t2) \wedge Terminates(a, f, t1) \wedge (t2 < t3) \wedge \neg Declipped(t1, f, t3)$
   This axiom states that if event a which terminates fluent f occurs then fluent f does not hold as long as fluent f is not initiated.

6. $Declipped(t1, f, t4) \leftrightarrow \exists a, t2, t3(Happens(a, t2, t3) \wedge (t1 < t3) \wedge (t2 < t4) \wedge Initiates(a, f, t2))$
   This axiom states that if and only if there exists an event a which occurs and initiates fluent f then fluent f is said to be declipped.

7. $Happens(a, t1, t2) \rightarrow (t1 \leq t2)$
   This axiom ensures that the time an event takes can never be negative.

4.4.2 Formalisation of a persuasion game

In this subsection the liberal persuasion game of Section 4.3.3 will be formalised.

4.4.2.1 Additional fluents

The following additional fluents will be used.

- $move(id, p, s, tr)$ This fluent states that this is the id^{th} move where participant p states locution s targeted at tr. Just as in Prakken's Framework, a quadruple is used for defining a move. The id is the identifier of a move and stands for the id^{th} move in the dialogue. It seems reasonable to unify the time points with the identifier of the move because every move is an event. However, this becomes problematic when a participant states an illegal move, this move will cause no changes to the fluents but will move time with one unit. So the time point is raised while the identifier of the move is not. For this reason the time points and the identifiers are not unified. The p is the player of the move, either the proponent or the opponent. The s is the speech act made in the dialogue and the tr is the target of the move, this is the identifier of the move it is targeted at.

- $Legal(\varphi)$ This fluent expresses that it is legal to make move φ where φ represents a tuple $move(id, p, s, tr)$. This fluent is initiated when move φ is legal to make. In Prakken's Framework as well as in the PWA System the only legal moves are the ones that serve as a reply to another move. So every move initiates one or more legal fluents as a reply to that move.

- $CS(p, \varphi)$ This fluent represents that participant p is committed to proposition φ. CS stands for commitment set and is initiated when participant p becomes committed to proposition φ.

- $Turn(p)$ is a fluent which is initiated when it becomes the turn of participant p.

- P stands for proponent.

- O stands for opponent.
- p and $\bar{p}$ are defined as: $\bar{p} = O$ if and only if $p = P$ and $\bar{p} = P$ if and only if $p = O$.

In the Event Calculus the predicate *Happens* is used for uttering a statement. This predicate is to be instantiated with a fluent (id, p, s, tr) and holds without conditions. A typical locution looks like this: $Happens(1, P, claim(a), 0)$. This locution states a first move by the proponent, claiming that a is the case.

4.4.2.2 The protocol

The formalisation of the **protocol** first specifies which initial moves are legal. Initially the only legal moves are a *claim* or an *argue* move by the proponent. After his first move the legality of these moves terminates.

$$\begin{array}{l} Initially_P(Legal(move(1, P, s, 0))) \quad \leftarrow \\ \quad s = claim\ \varphi \lor s = argue\ A \end{array}$$

$$\begin{array}{ll} Initially_N(Legal(move(id, p, s, tr))) & \leftarrow \\ \quad p = O & \lor \\ \quad (id \neq 1) & \lor \\ \quad (s \neq claim\ \varphi \land s \neq argue\ A) & \lor \\ \quad (t \neq 0) & \end{array}$$

$$\begin{array}{l} Terminates(move(1, P, s_1, 0), Legal(move(1, P, s_2, 0)), t) \quad \leftarrow \\ \quad HoldsAt(Legal(move(1, P, s_1, 0)), t) \end{array}$$

The next formulas specify how the legality of non-initial moves is initiated, capturing rules R$_2$, R$_3$, R$_4$ and R$_7$ of the protocol. $Defeats(B, A)$ means 'argument B defeats argument A'. Note that the rules for *argue* moves assume that the well-formedness of arguments and their defeat relations are determined by external means. The general format of the legality-initiating rules is as follows:

> m_1 initiates the legality of m_2 if
> m_1 was moved legally, and
> $pl(m_2)$ is the player-to-move, and
> $s(m_2)$ is a well-formed reply to $s(m_1)$, and
> the specific conditions for m_2, if any, are satisfied.

But the rule for replies to initial moves can be simpler:

$$\begin{array}{l} Initiates(move(1, P, claim\ \varphi, 0), Legal(move(id, O, s, 1)), t) \quad \leftarrow \\ \quad s = why\ \varphi \lor s = concede\ \varphi \end{array}$$

$$\begin{array}{ll} Initiates(move(id, p, why\ \varphi, tr), Legal(move(id_2, \bar{p}, s, id)), t) & \leftarrow \\ \quad HoldsAt(Legal(move(id, p, why\ \varphi, tr)), t) & \land \\ \quad HoldsAt(Turn(p), t) & \land \\ \quad ((s = argue\ A \land conc(A) = \varphi) \lor (s = retract\ \varphi)) & \end{array}$$

$$\begin{array}{ll}
\mathit{Initiates}(\mathit{move}(\mathit{id}, p, \mathit{argue}\ A, \mathit{tr}), \mathit{Legal}(\mathit{move}(\mathit{id}_2, \bar{p}, s, \mathit{id})), t) & \\
\quad \leftarrow & \\
\quad \mathit{HoldsAt}(\mathit{Legal}(\mathit{move}(\mathit{id}, p, \mathit{argue}\ A, \mathit{tr})), t) & \wedge \\
\quad \mathit{HoldsAt}(\mathit{Turn}(p), t) & \wedge \\
\quad ((s = \mathit{why}\ \varphi \wedge \varphi \in \mathit{prem}(A)) & \\
\quad \vee & \\
\quad (s = \mathit{argue}\ B \wedge \mathit{Defeats}(B, A)) & \\
\quad \vee & \\
\quad (s = \mathit{concede}\ \varphi \wedge \varphi = \mathit{conc}(A) \wedge & \\
\quad\quad \neg(\mathit{Happens}(\mathit{move}(\mathit{tr}, \bar{p}, \mathit{why}\ \varphi, \mathit{tr}_2), t_2) \wedge t_2 < t) & \\
\quad \vee & \\
\quad (s = \mathit{concede}\ \varphi \wedge \varphi \in \mathit{prem}(A))) &
\end{array}$$

Next the conditions are specified under which the legality of non-initial moves terminates. The first rule below says that after a move with a specific content is made, it terminates to hold as a legal move with that specific content and the same target.

$$\begin{array}{ll}
\mathit{Terminates}(\mathit{move}(\mathit{id}, p, s, \mathit{tr}), \mathit{Legal}(\mathit{move}(\mathit{id}_2, p, s, \mathit{tr})), t) & \leftarrow \\
\quad \mathit{HoldsAt}(\mathit{Turn}(p), t) & \wedge \\
\quad \mathit{HoldsAt}(\mathit{Legal}(\mathit{move}(\mathit{id}, p, s, \mathit{tr})), t) &
\end{array}$$

The second rule captures protocol rule R_5 and states that when a move is a surrendering move to a target the attacking counterpart of this move at the same target terminates to be legal.

$$\begin{array}{ll}
\mathit{Terminates}(\mathit{move}(\mathit{id}, p, s_1, \mathit{tr}), \mathit{Legal}(\mathit{move}(\mathit{id}_2, p, s_2, \mathit{tr})), t) & \leftarrow \\
\quad \mathit{HoldsAt}(\mathit{Legal}(\mathit{move}(\mathit{id}, p, s_1, \mathit{tr})), t) & \wedge \\
\quad \mathit{HoldsAt}(\mathit{Legal}(\mathit{move}(\mathit{id}_2, p, s_2, \mathit{tr})), t) & \wedge \\
\quad \mathit{HoldsAt}(\mathit{Turn}(p), t) & \wedge \\
\quad ((s_1 = \mathit{concede}\ \varphi \wedge s_2 = \mathit{why}\ \varphi) & \\
\quad \vee & \\
\quad (s_1 = \mathit{concede}\ \varphi \wedge s_2 = \mathit{argue}\ A \wedge \neg\varphi = \mathit{conc}(A)) & \\
\quad \vee & \\
\quad (s_1 = \mathit{retract}\ \varphi \wedge s_2 = \mathit{argue}\ A \wedge \varphi = \mathit{conc}(A))) &
\end{array}$$

Summarising, EC's 'law of inertia' is used in this formalisation as follows. Initially, only a *claim* and *argue* move are legal and all other moves are illegal. After a move is made its legality is terminated (but only with that specific content) and the legality of well-formed replies to that move's speech act is initiated. Such legality persists until the move is made. The illegality of moves persists until its legality is initiated as just described.

4.4.2.3 Commitment set

Initially, the commitment set of both players is empty as stated. When performing a move, the commitment set of the participant performing the locution is affected. It is always checked whether the performing participant uses a legal move initiated as a reply to a previous locution. It is also checked whether it is the turn of the participant. The following four rules ensure the altering of the commitment set during the dialogue.

$Initially_N(CS(p, \varphi))$

$$\begin{array}{ll} Initiates(move(id, p, s, tr), CS(p, \varphi), t) \quad \leftarrow & \\ \quad HoldsAt(Legal(move(id, p, s, tr)), t) & \wedge \\ \quad HoldsAt(Turn(p), t) & \wedge \\ \quad (s = claim\ \varphi \vee s = concede\ \varphi) & \end{array}$$

$$\begin{array}{ll} Terminates(move(id, p, retract\ \varphi, tr), CS(p, \varphi), t) \quad \leftarrow & \\ \quad HoldsAt(Legal(move(id, p, retract\ \varphi, tr)), t) & \wedge \\ \quad HoldsAt(Turn(p), t) & \end{array}$$

$$\begin{array}{ll} Initiates(move(id, p, argue\ A, tr), CS(p, \varphi), t) \quad \leftarrow & \\ \quad HoldsAt(Legal(move(id, p, argue\ A, tr)), t) & \wedge \\ \quad HoldsAt(Turn(p), t) & \wedge \\ \quad (\varphi = conc(A) \vee \varphi = prem(A)) & \end{array}$$

4.4.2.4 Turntaking

In the last four rules the turntaking in the dialogue system is realised. The first move is always made by the proponent after which the turn switches. This means that after the first move, the turn of the proponent terminates to hold and the turn of the opponent is initiated. After the second move, both participants can participate in the dialogue so now also the turn of the proponent is initiated.

$Initially_P(Turn(P))$

$Initially_N(Turn(O))$

$$\begin{array}{l} Terminates(move(1, P, s, 0), Turn(P), t) \quad \leftarrow \\ \quad HoldsAt(Legal(move(1, P, s, 0)), t) \end{array}$$

$$\begin{array}{ll} Initiates(move(id, p_1, s, tr), Turn(p_2), t) \quad \leftarrow & \\ \quad HoldsAt(Legal(move(id, p1, s, tr)), t) & \wedge \\ \quad ((id = 1 \wedge p_1 = P \wedge tr = 0 \wedge t = 1 \wedge p_2 = O) & \\ \quad \vee & \\ \quad (id = 2 \wedge p_1 = O \wedge tr = 1 \wedge t = 2 \wedge p_2 = P)) & \end{array}$$

This completes the specification of the liberal persuasion protocol of Section 4.3.3.

4.5 Another illustration of our approach

We now apply the above approach to formalisation and implementation of protocols to a second dialogue system for persuasion, that of Parsons, Wooldridge and Amgoud [62], which we refer to as *PWA* or the *PWA System.* The reason for a second application is to illustrate the generality of our approach by applying it to a pre-existing protocol. The logic used in PWA is that of Amgoud and Cayrol [8], a nonmonotonic argumentation system instantiating the grounded semantics of Dung [33]. The notational conventions used by Parsons, Wooldridge and Amgoud in [62] will be unified below with the notation we used above.

4.5.1 The PWA system

We first describe the PWA System as presented in [62].

4.5.1.1 Communication Language

The following table compares the names for the locutions used by Parsons, Wooldridge and Amgoud [62] and those used here.

PWA System	Present Framework
assert	claim
accept	concede
challenge	why

As just noted, we will adjust the names used in the PWA System to ours. Their communication language then consists of the locutions in Table 4.2.

- *claim* φ, a participant claims proposition φ
- *claim* S, a participant claims the set of propositions S
- *concede* φ, a participant concedes proposition φ
- *concede* S, a participant concedes the set of propositions S
- *why* φ, a participant challenges the other participant to state an argument to support proposition φ

Table 4.2: Locutions

4.5.1.2 Protocol

Protocol rules

The protocol is unique-move, which means that the turn shifts after each move. Also, except for replies to claim moves it is unique-reply, which means that alternative replies to the same move are not allowed. The exception is when a participant claims a set of propositions as reply to a *why* move: then the other participant can give a reply to *each* proposition of this set. The possible replies to a locution have a preferred order: if the first, most preferred response is not a legal one, the next best option is considered. The legality of a move partly depends on the *attitude* of the agent which is explained below. The first move is always made by the proponent, agent A, who claims proposition p after which the opponent, agent B, makes a move.

1. A *claims* p.
2. B *concedes* p if it is a legal according to its concede-attitude,
   If it is not legal to *concede* p, B *claims* $\neg p$ if it is legal according to its claim-attitude,
   Otherwise B *challenges* p with a *why* move.
3. If B *claimed* $\neg p$ then
   goto 2 with roles of the agents reversed and $\neg p$ in place of p.
4. If B *challenged* p with a *why* move then
   A *claims* S, the support for p;
   Goto 2 for each $s \in S$ in turn.

Table 4.3: Protocol

Attitudes

Whether it is legal to claim or concede a proposition also depends on the ability of the agent of finding an argument of a certain kind to support the proposition. Every participating agent has one *claim-attitude* and one *concede-attitude*. This attitude is fixed for the agent during the entire dialogue.

Definition 67 *Claim-attitude*

- *A* confident *agent can claim any proposition* p *for which it can construct an argument* A *where* $p = conc(A)$.
- *A* careful *agent can claim any proposition* p *if it is not able to construct a stronger argument* A *for proposition* $\neg p$ *where* $\neg p = conc(A)$.
- *A* thoughtful *agent can claim any proposition* p *for which it can construct a justified argument* A *where* $p = conc(A)$.

Definition 68 *Concede-attitude*

- *A* credulous *agent can concede to any proposition* p *if it is able to construct an argument* A *where* $p = conc(A)$.

- *A* cautious *agent can concede to any proposition p if it is not able to construct a stronger argument A where $\neg p = conc(A)$.*
- *A* skeptical *agent can concede to any proposition p if it is able to construct a justified argument A where $p = conc(A)$.*

In these definitions a participant is supposed to construct arguments on the basis of its own internal knowledge base plus the commitments of the other participant.

4.5.1.3 Effect Rules

The effect rules are essentially the same as in our framework, with the obvious addition that conceding or claiming a set of propositions commits to the entire set.

4.5.1.4 Turntaking

Generally, the protocol is *unique-move* so the turn switches after every move, this is regulated directly through the structure of the protocol. Only when a *set* of propositions is claimed the turn does *not* need to switch after every move. When a reply move is made by the opponent to one proposition of the set propositions the proponent claimed and this proposition is eventually conceded by the opponent, the turn does not switch and the opponent can reply to another proposition of the set of propositions claimed by the proponent.

4.5.1.5 Termination Rules

The dialogue terminates when there are no more moves. This happens when there are no more propositions the agents do not agree upon. Besides the termination rules there is also an *outcome* rule. If an agent is not able to make a move when it is his turn, he has to *concede* the dialogue game. If the proponent has to concede the dialogue game, he has not succeeded in persuading the opponent. If the opponent has to concede the dialogue game then the proponent is succeeded in persuading his opponent.

4.5.2 Formalisation of PWA in Event calculus

We next formalise the PWA Persuasion protocol, using the same approach as above.

4.5.2.1 Additional fluents

To formalise the PWA System, further fluents are used in addition to those used to formalise the system of Section 4.3.3 above.

- *ConcedeAttitude*(p, c) means that the concede-attitude of participant p is the attitude c this fluent is used to define the concede-attitude of the participants at the beginning of the dialogue. The attitude remains static throughout the dialogue.

- *ClaimAttitude*(p, c) means that the claim-attitude of participant p is the attitude c. This fluent functions the same as the *ConcedeAttitude*(p, c).

- *Allows*(s, p) is a fluent which holds if the attitude of participant p allows move s. This fluent is initiated after every move if the formal definition of the attitudes as discussed in Tables 67 and 68 allows the proposition used in the move.

- *ConcedeGame*(p) is a fluent that defines that participant p concedes to the dialogue. A participant should concede to the dialogue if the participant is not able to make the indicated move. The dialogue ends when a participant concedes to the dialogue.

- *Retain*(φ) is a fluent used for storing move φ. After claiming a set of propositions the other participant can reply to each proposition. The propositions which are not replied to directly are stored in the *Retain*(φ) fluent.

- *Formula*(A, id) is a fluent used for expressing that formula A has preference order id. In the PWA System the preference of arguments and propositions is agreed upon before the dialogue starts. In the formalisation this fluent will be used at the beginning of the dialogue to initiate all arguments with their preference order available in the "world" of the agents.

- *Acceptable*(A, p) is a fluent used to express that argument A is an acceptable argument for participant p. In this formalisation it is not specified when an argument holds as acceptable, acceptable arguments are simply initiated.

- *KB*(A, p) is a fluent which denotes that argument or proposition A is an element of the knowledge base *KB* of participant p. This fluent is used to express that an agent has access to an argument or proposition in the dialogue.

4.5.2.2 Begin situation

Due to the closed world assumption of the Event Calculus all fluents have to be known initially and therefore should be known as to initially hold or not hold. In the PWA System there are several fluents whose initial value depends on the dialogue. These fluents are *Acceptable*(a, p), *Formula*(a, id), *KB*(a, p), *Allows*(s, p), *ClaimAttitude*(a, b) and *ConcedeAttitude*(a, b). The fluents with values that hold at the beginning of the dialogue should be added to the formalisation before starting the dialogue by the agents or the user of the system.

These clauses will, for example, have the following form.

$Initially_P(Acceptable(since(P, Q)))$

The user of the system should express which arguments are *acceptable* for each participant at the beginning of the dialogue, what the preference order of each *formula* is during this dialogue, which propositions and arguments are in the *knowledge base* of the participants, which propositions and arguments are *allowed* for the proponent to use according to its attitude at the beginning of the dialogue and also the *claim and concede attitudes* should be added to the formalisation. All other values of these fluents are assumed not to hold at the beginning of the dialogue. This is covered by the following six rules where the clause $Initially_N$ holds if $Initially_P$ does not hold. The first three rules cover the available propositions and arguments during the dialogue and the last three rules deal with the attitudes of the participants and the suitability of arguments according to these attitudes.

BS 1 $Initially_N(Acceptable(a, p)) \quad \leftarrow$
$\neg Initially_P(Acceptable(a, p))$

BS 2 $Initially_N(Formula(a, id)) \quad \leftarrow$
$\neg Initially_P(Formula(a, id))$

BS 3 $Initially_N(KB(a, p)) \quad \leftarrow$
$\neg Initially_P(KB(a, p))$

BS 4 $Initially_N(Allows(s, p)) \quad \leftarrow$
$\neg Initially_P(Allows(s, p))$

BS 5 $Initially_N(ClaimAttitude(a, b)) \quad \leftarrow$
$\neg Initially_P(ClaimAttitude(a, b))$

BS 6 $Initially_N(ConcedeAttitude(a, b)) \quad \leftarrow$
$\neg Initially_P(ConcedeAttitude(a, b))$

In the PWA System it is the turn of the proponent at the start of the dialogue and his only legal move is making a claim.

BS 7 $Initially_P(Turn(P))$

BS 8 $Initially_P(Legal(move(1, P, claim\ \varphi, 0)))$

It is necessary to specify that initially it is not the turn of the opponent. This is specified by BS 9. BS 10 specifies which moves do not hold as legal at the beginning of the dialogue. This rule states that no move of the opponent holds as legal and that except for a *claim* move no move holds at the beginning of the dialogue.

BS 9 $Initially_N(Turn(O))$

BS 10 $Initially_N(Legal(move(id, p, s, 0))) \quad \leftarrow$
$$\begin{aligned} & p = O && \vee \\ & \neg(id = 1) && \vee \\ & \neg(s = claim\ \varphi) && \vee \\ & \neg(t = 0) \end{aligned}$$

Initially the commitment set of both participants is empty. This means that there is no fluent $CS(p, \varphi)$ that initially holds. Also the fluent $ConcedeGame(P)$ does not hold at the beginning of the dialogue and there are no retained moves so the fluent $Retain(s)$ does not hold initially.

BS 11 $Initially_N(CS(p, \varphi))$

BS 12 $Initially_N(ConcedeGame(p))$

BS 13 $Initially_N(Retain(s))$

4.5.2.3 Legal moves

This subsection deals with the PWA System Protocol as stated in Table 4.3. In the PWA System a reply can be given to a *claim* and to a *why* move. Only one locution is *legal* as a reply to a move. It might be possible however that there are several propositions to use in the locution as a reply. There are three possible replies to a *claim* move; which one is legal depends on the *attitude* of the agent. A *claim* move can be made for a set of propositions or just for one proposition. The following formula, LE 1, deals with initiating the three possible replies to a *claim* move for a single proposition. The formula after that, LE 2, deals with initiating the three possible replies to a *claim* move for a *set* of propositions. The third formula, LE 3, deals with the reply to a *why* move. To a *why* move only one reply is possible. The last formula of this subsection, LE 4, deals with the termination of the legality of moves.

The reply move to a claim is a *concede* move if the concede-attitude allows it. If the concede-attitude does not allow a *concede* move, a *claim* move is initiated as legal if the claim-attitude allows it. When the *concede* as well as the *claim* move are not allowed according to the attitudes, a *why* move is initiated as legal. This is described in the third part of the formula. It is checked whether it is the turn of the moving participant and whether the move is legal. The conditions under which fluent $Allows(s, p)$ holds is described in Subsection 4.5.2.6 and indicates whether the attitude of participant p allows the statement of the locution.

LE 1 $Initiates(move(id, p, claim\ \varphi, tr), Legal(move(id2, \bar{p}, s, id)), t) \quad \leftarrow$

$HoldsAt(Legal(move(id, p, claim\ \varphi, tr)), t) \quad \wedge$
$HoldsAt(Turn(p), t) \quad \wedge$
$((s = concede\ \varphi \quad \wedge$
$HoldsAt(Allows(s, \bar{p}), t))$
$\vee$
$(s = claim\ \neg\varphi \quad \wedge$
$\neg HoldsAt(Allows(concede\ \varphi, \bar{p}), t) \quad \wedge$
$HoldsAt(Allows(s, \bar{p}), t))$
$\vee$
$(s = why\ \varphi \quad \wedge$
$\neg HoldsAt(Allows(concede\ \varphi, \bar{p}), t) \quad \wedge$
$\neg HoldsAt(Allows(claim\ \neg\varphi, \bar{p}), t)))$

The following formula deals with the initiation of legal replies to a claim with a *set* of propositions in the same way as the previous formula describes the initiation of legal replies to a claim with a single proposition.

LE 2
$Initiates(move(id, p, claim\ S, tr), Legal(move(id2, \bar{p}, s, id)), t) \quad \leftarrow$
$HoldsAt(Legal(move(id, p, claim\ S, tr)), t) \quad \wedge$
$HoldsAt(Turn(p), t) \quad \wedge$
$((s = concede\ \varphi \quad \wedge$
$HoldsAt(Allows(s, \bar{p}), t) \quad \wedge$
$\varphi \in S)$
$\vee$
$(s = claim\ \varphi \quad \wedge$
$\neg HoldsAt(Allows(concede\ \neg\varphi, \bar{p}), t) \quad \wedge$
$HoldsAt(Allows(s, \bar{p}), t) \quad \wedge$
$\neg\varphi \in S)$
$\vee$
$(s = why\ \varphi \quad \wedge$
$\neg HoldsAt(Allows(concede\ \varphi, \bar{p}), t) \quad \wedge$
$\neg HoldsAt(Allows(claim\ \neg\varphi, \bar{p}), t) \quad \wedge$
$\varphi \in S))$

The next formula deals with the initiation of a legal move as reply to a *why* move. The only legal reply to a *why* move is claiming a set of propositions. This set of propositions, S, must be support for proposition φ challenged by the other participant. This means that there must be an argument A available to participant $\bar{p}$ with conclusion φ and premises S in its knowledge base. The premises S must be available in its own knowledge base or in the commitment set of the other participant.

LE 3 $Initiates(move(id, p, why\ \varphi, tr), Legal(move(id2, \bar{p}, claim\ S, id)), t) \quad \leftarrow$

$HoldsAt(Legal(move(id, p, why\ \varphi, tr)), t) \quad \wedge$
$HoldsAt(Turn(p), t) \quad \wedge$
$conc(A) = \varphi \quad \wedge$
$prem(A) = S \quad \wedge$
$HoldsAt(KB(A, \bar{p}), t)$
$(x \in S \wedge (HoldsAt(KB(x, \bar{p}), t) \vee HoldsAt(CS(x, p), t)))$

After making a move, the legality of the move is terminated. When a *claim* move for a set of propositions is made all propositions of this set can be replied to and will be initiated as legal by formula LE 2. When participant p makes one of the initiated moves, all legal moves, including the move participant p made, terminate to hold as legal. In the dialogue it is possible to reply to *each* proposition of this set propositions, this is described in Subsection 4.5.2.4. All legal moves of participant p with target tr terminate to hold as legal after making a move. It is important to notice that this construction does not prevent the ability of making the same move twice. The protocol as stated in [62] does not prohibit making the same move twice and therefore this will also be allowed in this formalisation.

LE 4 $Terminates(move(id, p, s, tr), Legal(move(id2, p, s2, tr)), t) \quad \leftarrow$
$HoldsAt(Legal(move(id, p, s, tr)), t) \quad \wedge$
$HoldsAt(Turn(p), t)$

4.5.2.4 Claiming a set of propositions

When a participant claims a set of propositions to support another proposition, the other participant can reply to each proposition of this claim separately. In the protocol of [62] it is not specified what the order is in which propositions should be dealt with when there is more than one retained move. There are two ways of dealing with this situation. The first option is to deal first with the first retained move and when these propositions are all addressed then start looking at other retained moves. This is a *First-In, First-Out* system. The second option is to treat the retained moves as a stack where one proposition is completely dealt with first before starting with another. Both options are illustrated in the following examples where in the first example after the *concede* move of $P1$ the dialogue continues with the first retained move and in the second example continues the dialogue with the most recent retained move.

First option: first in first out.
$P1 : claim\ S$
    $P2 : why\ q, q \in S$
    ...
    $P2 : claim\ T$
        $P1 : why\ r, r \in T$
        ...
        $P1 : concede\ r$

$P2 : \textit{why}\ v, v \in S$
$\ldots$

Second option: last in first out.
$P1 : \textit{claim}\ S$
$P2 : \textit{why}\ q, q \in S$
$\ldots$
$P2 : \textit{claim}\ T$
$P1 : \textit{why}\ r, r \in T$
$\ldots$
$P1 : \textit{concede}\ r$
$P1 : \textit{why}\ s, s \in T$
$\ldots$

Here the second option is chosen. This means that when the replying participant has chosen a proposition to reply to, the rest of the claim should be retained to serve as a basis for a next reply to that retained move when the discussion about the first proposition has ended. The fluent used for this purpose is *Retain.* As soon as a set of propositions is claimed, the fluent *Retain* for this move is initiated.

SP 1 $\textit{Initiates}(\textit{move}(id, p, \textit{claim}\ S, tr), \textit{Retain}(\textit{move}(id, p, \textit{claim}\ S, tr)), t) \quad \leftarrow$
$\textit{HoldsAt}(\textit{Legal}(\textit{move}(id, p, \textit{claim}\ S, tr)), t) \quad \wedge$
$\textit{HoldsAt}(\textit{Turn}(p), t)$

Every reply to a claim of a set of propositions initiates the *Retain* fluent for that *claim* move *without* the proposition it replies to. Formula SP 2 ensures this for all three reply moves; *concede, claim* and ask *why.* First it is checked whether it is the participant's turn en whether the move is legal. The second condition is that there must be a *Retain* move to which the move replies. The last condition checks whether the remaining set of propositions S is not empty.

SP 2 $\textit{Initiates}(\textit{move}(id, p, s, id2), \textit{Retain}(\textit{move}(id2, \bar{p}, \textit{claim}\ S, tr2)), t) \quad \leftarrow$
$\textit{HoldsAt}(\textit{Legal}(\textit{move}(id, p, s, id2)), t) \quad \wedge$
$\textit{HoldsAt}(\textit{Turn}(p), t) \quad \wedge$
$\textit{HoldsAt}(\textit{Retain}(\textit{move}(id2, \bar{p}, \textit{claim}\ S2, tr2)), t) \quad \wedge$
$S \neq \emptyset \quad \wedge$
$((s = \textit{concede}\ \varphi \wedge S = S2/\varphi)$
$\vee$
$(s = \textit{claim}\ \varphi \wedge S = S2/\neg\varphi)$
$\vee$
$(s = \textit{why}\ \varphi \wedge S = S2/\varphi))$

The original *Retain* fluent, the one *with* the proposition it replies to, is terminated at the same time. The following formula formalises this termination.

$$\begin{array}{ll}\textbf{SP 3}\ \mathit{Terminates}(\mathit{move}(\mathit{id}, p, s, \mathit{id}2), \mathit{Retain}(\mathit{move}(\mathit{id}2, \bar{p}, \mathit{claim}\ S, \mathit{tr}2)), t) & \leftarrow \\ \quad \mathit{HoldsAt}(\mathit{Legal}(\mathit{move}(\mathit{id}, p, s, \mathit{id}2)), t) & \wedge \\ \quad \mathit{HoldsAt}(\mathit{Turn}(p), t) & \wedge \\ \quad \mathit{HoldsAt}(\mathit{Retain}(\mathit{move}(\mathit{id}2, \bar{p}, \mathit{claim}\ S, \mathit{tr}2)), t) & \end{array}$$

The fluent *Retain* results in additional rules for initiating legal moves and only plays a role when the dialogue threatens to end because there are no more possible moves. This can happen if a participant concedes a proposition or when a participant is not able to support a proposition with a *claim S* move. Formula SP 4 formalises the first case and formula SP 5 formalises the second case.

In the first case it now becomes possible to proceed the dialogue after a *concede* move because there still may be a retained move. This means that instead of ending the dialogue after a concede move, it is first tried to initiate a reply from the retained set. Participant *p2* stating the *concede* move can be the participant who made the retained *claim S* move, namely $\bar{p}$, as well as the participant who now can reply to this retained *claim S* move, namely participant *p*. Just as when a reply is made to a normal *claim* move, there are three possible replies whose legality depends on the attitude of participant *p*. There are several conditions to be checked. First the move must be legal and it must be the turn of participant *p2*. There must also be a retained *claim* move to which the move, that needs to be initiated as legal, replies to. The attitudes concerning the legality of a move according to the protocol stated in Subsection 4.5.1.2 must hold and the proposition ψ to be stated should be an element of the set S of the *claim* move. To ensure that the retained moves are dealt with as in a stack, there cannot be a retained move with a higher identifier in case of more than one retained move, meaning that there cannot be a move that was retained after the retained move that is going to supply the proposition to reply to.

SP 4

$$\begin{array}{ll}\mathit{Initiates}(\mathit{move}(\mathit{id}, p2, \mathit{concede}\ \varphi, \mathit{tr}), \mathit{Legal}(\mathit{move}(\mathit{id}2, p, s, \mathit{id}3)), t) & \leftarrow \\ \quad \mathit{HoldsAt}(\mathit{Turn}(p2), t) & \wedge \\ \quad \mathit{HoldsAt}(\mathit{Legal}(\mathit{move}(\mathit{id}, p2, \mathit{concede}\ \varphi, \mathit{tr})), t) & \wedge \\ \quad \mathit{HoldsAt}(\mathit{Retain}(\mathit{move}(\mathit{id}3, \bar{p}, \mathit{claim}\ S, \mathit{tr}2)), t)) & \wedge \\ \quad \neg(\mathit{HoldsAt}(\mathit{Retain}(\mathit{move}(x, p3, s, \mathit{tr}3), t) \wedge x > \mathit{id}3) & \wedge \\ \quad ((s = \mathit{concede}\ \psi & \wedge \\ \quad \mathit{HoldsAt}(\mathit{Allows}(s, p), t) & \wedge \\ \quad \psi \in S) & \\ \quad \vee & \\ \quad (s = \mathit{claim}\ \psi & \wedge \\ \quad \neg \mathit{HoldsAt}(\mathit{Allows}(\mathit{concede}\ \neg\psi, p), t) & \wedge \\ \quad \mathit{HoldsAt}(\mathit{Allows}(s, p), t) & \wedge \\ \quad \neg\psi \in S) & \\ \quad \vee & \\ \quad (s = \mathit{why}\ \psi & \wedge \\ \quad \neg \mathit{HoldsAt}(\mathit{Allows}(\mathit{concede}\ \psi, p), t) & \wedge \end{array}$$

$\neg HoldsAt(Allows(claim \neg\psi, p), t) \quad \wedge$
$\psi \in S))$

The other situation in which the fluent *Retain* should be checked is when the dialogue threatens to end because a participant is not able to construct support for a proposition and therefore is not able to make the indicated *claim* move. Also in this case it should be checked whether there is a retained *claim* move. This should be done in the last possible move which is the move *preceding* the impossible *claim* move. The only move preceding a *claim* move is a *why* move. So after a *why* move, it should be checked whether the other participant is able to construct a legal move and whether there is a retained move available. The conditions are like the conditions of formula SP 4.

SP 5 $Initiates(move(id, p2, why\ \varphi, tr), Legal(move(id2, p, concede\ \psi, id3)), t) \quad \leftarrow$
$HoldsAt(Retain(move(id3, \bar{p}, claim\ S, tr2)), t)) \quad \wedge$
$\neg HoldsAt(Legal(move(id4, \bar{p2}, claim\ S2, id))) \quad \wedge$
$HoldsAt(Turn(p2), t) \quad \wedge$
$HoldsAt(Legal(move(id, p2, why\ \varphi, tr)), t) \quad \wedge$
$HoldsAt(Allows(concede\ \psi, p), t) \quad \wedge$
$\psi \in S \quad \wedge$
$\neg(HoldsAt(Retain(move(x, p3, s, tr3))) \wedge x > id3)$

4.5.2.5 Attitudes

As noted above, in the PWA System the assertion and acceptance attitudes of the agent partly determine which speech acts are allowed at which point in the dialogue, where the agents are assumed to reason with the commitment sets of both agents and their own private knowledge base. After a *claim* as well as a *concede* move the commitment set of the moving participant is altered according to the effect rules. This means that after a *concede* or *claim* move the set of speech acts which are allowed according to the attitude of the agent might change. To ensure the flexibility of the formalisation all fluents $Allows(s, p)$, indicating which speech acts s where allowed for participant p at the time of the move, are terminated. After terminating all fluents of the form $Allows(s, p)$ it is checked which fluents $Allows(s, p)$ start to hold by making use of the attitudes of the participant. This attitude is activated at the beginning of the dialogue with an $Initially_p(ConcedeAttitude(p, c))$ statement where c indicates which attitude the participant adopts throughout the dialogue.

AT 1 $Terminates(move(id, p, s, tr), Allows(\varphi, p), t) \quad \leftarrow$
$HoldsAt(Turn(p), t) \quad \wedge$
$HoldsAt(Legal(move(id, p, s, tr)), t) \quad \wedge$
$(s = claim\ \psi \vee s = concede\ \psi)$

AT 2
$Initiates(move(id, p, s, tr), Allows(claim\ \varphi, p), t) \quad \leftarrow$

$HoldsAt(Turn(p), t) \quad \wedge$
$HoldsAt(Legal(move(id, p, s, tr)), t) \quad \wedge$
$(s = claim\ \psi \vee s = concede\ \psi) \quad \wedge$
$HoldsAt(ClaimAttitude(p, c), t) \quad \wedge$
$((c = confident \wedge HoldsAt(KB(A, p), t) \wedge \varphi = conc(A) \quad \wedge$
$\quad S = prem(A) \wedge x \in S \wedge (HoldsAt(KB(x, p), t) \vee HoldsAt(CS(x, \bar{p}), t)))$
$\vee$
$(c = careful \wedge \neg(HoldsAt(KB(A, p), t) \wedge \varphi = conc(A) \quad \wedge$
$\quad S = prem(A) \wedge x \in S \wedge (HoldsAt(KB(x, p), t) \vee HoldsAt(CS(x, \bar{p}), t)) \quad \wedge$
$\quad HoldsAt(Formula(A, id3), t) \wedge HoldsAt(KB(B, p), t) \quad \wedge$
$\quad R = prem(B) \wedge y \in R \wedge (HoldsAt(KB(y, p), t) \vee HoldsAt(CS(y, \bar{p}), t)) \quad \wedge$
$\quad \neg\varphi = conc(B) \wedge HoldsAt(Formula(B, id4), t) \wedge id4 > id3))$
$\vee$
$(c = thoughtful \wedge HoldsAt(KB(A, p), t) \wedge \varphi = conc(A) \quad \wedge$
$\quad S = prem(A) \wedge x \in S \wedge (HoldsAt(KB(x, p), t) \vee HoldsAt(CS(x, \bar{p}), t)) \quad \wedge$
$\quad HoldsAt(Acceptable(A, p), t)))$

AT 3
$Initiates(move(id, p, s, tr), Allows(concede\ \varphi, p), t) \quad \leftarrow$
$HoldsAt(Turn(p), t) \quad \wedge$
$HoldsAt(Legal(move(id, p, s, tr)), t) \quad \wedge$
$(s = claim\ \psi \vee s = concede\ \psi) \quad \wedge$
$HoldsAt(ConcedeAttitude(p, c), t) \quad \wedge$
$((c = credulous \wedge HoldsAt(KB(A, p), t) \wedge \varphi = conc(A)$
$\quad S = prem(A) \wedge x \in S \wedge (HoldsAt(KB(x, p), t) \vee HoldsAt(CS(x, \bar{p}), t)))$
$\vee$
$(c = cautious \wedge \neg(HoldsAt(KB(A, p), t) \wedge \varphi = conc(A) \quad \wedge$
$\quad S = prem(A) \wedge x \in S \wedge (HoldsAt(KB(x, p), t) \vee HoldsAt(CS(x, \bar{p}), t)) \quad \wedge$
$\quad HoldsAt(Formula(A, id3), t) \wedge HoldsAt(KB(B, p), t) \quad \wedge$
$\quad R = prem(B) \wedge y \in R \wedge (HoldsAt(KB(y, p), t) \vee HoldsAt(CS(y, \bar{p}), t)) \quad \wedge$
$\quad \neg\varphi = conc(B) \wedge HoldsAt(Formula(B, id4), t) \wedge id4 > id3))$
$\vee$
$(c = skeptical \wedge HoldsAt(KB(A, p), t) \wedge \varphi = conc(A) \quad \wedge$
$\quad S = prem(A) \wedge x \in S \wedge (HoldsAt(KB(x, p), t) \vee HoldsAt(CS(x, \bar{p}), t)) \quad \wedge$
$\quad HoldsAt(Acceptable(A, p), t)))$

4.5.2.6 Turntaking

Generally, after every move the turn switches. Of course a move should be a legal move and it has to be the participant's turn in order for a move to change the turn. The termination and initiation of the turn of the participants is specified in the first two formulas, TU 1 and TU 2. There is one case in which the turn does not need to switch, namely when a *concede* move is done. There is no legal reply to a *concede* move so in general the dialogue ends but when the *concede* move is made as a reply move to one proposition of a *claim* move with a set of propositions, the dialogue may then proceed. This is described in the third and fourth formula, TU 3 and TU 4 of this subsection.

TU 1 $\mathit{Terminates}(\mathit{move}(id, p, s, tr), \mathit{Turn}(p), t) \quad \leftarrow$
$\mathit{HoldsAt}(\mathit{Legal}(\mathit{move}(id, p, s, tr)), t) \quad \wedge$
$\mathit{HoldsAt}(\mathit{Turn}(p), t) \quad \wedge$
$s \neq \mathit{concede}\ \varphi$

TU 2 $\mathit{Initiates}(\mathit{move}(id, p, s, tr), \mathit{Turn}(\bar{p}), t) \quad \leftarrow$
$\mathit{HoldsAt}(\mathit{Legal}(\mathit{move}(id, p, s, tr)), t) \quad \wedge$
$\mathit{HoldsAt}(\mathit{Turn}(p), t) \quad \wedge$
$s \neq \mathit{concede}\ \varphi$

When the dialogue does not end after a *concede* move because there are still propositions left to reply to, it can be the turn of both players. This depends on the original *claim* move with a set of propositions. If this move is made by the proponent then the opponent may reply to the remaining propositions and it will be the turn of the opponent. This can also be the other way around in which case it will be the turn of the proponent. This means that after a *concede* move the turn *may or may not* change. The formulas dealing with retaining the *claim* move can be found in Subsection 4.5.2.4.

The next formula ensures that after the *concede* move the participant whose turn it is starts to hold. Of course the initiating *concede* move should be legal and it must be the turn of participant *p*. The turn only switches if the participant *p* who states the *concede* φ move also made the retained *claim S* move. If there are more than one retained *claim* moves the one with the highest identifier is dealt with first. This can be looked at as being a stack; first in last out.

TU 3 $\mathit{Initiates}(\mathit{move}(id, p, \mathit{concede}\ \varphi, tr), \mathit{Turn}(\bar{p}), t) \quad \leftarrow$
$\mathit{HoldsAt}(\mathit{Legal}(\mathit{move}(id, p, \mathit{concede}\ \varphi, tr)), t) \quad \wedge$
$\mathit{HoldsAt}(\mathit{Turn}(p), t) \quad \wedge$
$\mathit{HoldsAt}(\mathit{Retain}(\mathit{move}(id3, p, \mathit{claim}\ S, tr2)), t) \quad \wedge$
$\neg(\mathit{HoldsAt}(\mathit{Retain}(\mathit{move}(x, p2, s, tr3)), t) \wedge x > id3)$

If the turn switches after the concede move it terminates to be the turn of participant p. The conditions of the following formula are therefore the same as of the preceding ones.

TU 4 $\mathit{Terminates}(\mathit{move}(id, p, \mathit{concede}\ \varphi, tr), \mathit{Turn}(p), t) \quad \leftarrow$
$\mathit{HoldsAt}(\mathit{Legal}(\mathit{move}(id, p, \mathit{concede}\ \varphi, tr)), t) \quad \wedge$
$\mathit{HoldsAt}(\mathit{Turn}(p), t) \quad \wedge$
$\mathit{HoldsAt}(\mathit{Retain}(\mathit{move}(id3, p, \mathit{claim}\ S, tr2)), t)) \quad \wedge$
$\neg(\mathit{HoldsAt}(\mathit{Retain}(\mathit{move}(x, p2, s, tr3)), t) \wedge x > id3)$

4.5.2.7 Effect of a move on the commitment set

The commitment set of the participants changes after performing a move. When a move is made by a participant the commitment set is updated according to the effect rules.

ER 1 $Initiates(move(id,p,s,tr), CS(p,\varphi),t) \quad \leftarrow$
$HoldsAt(Legal(move(id,p,s,tr)),t) \quad \wedge$
$HoldsAt(Turn(p),t) \quad \wedge$
$(s = claim\ \varphi \vee s = concede\ \varphi)$

4.5.2.8 Ending the dialogue

The dialogue ends when a participant is not able to make the indicated move. This can also be found in the Subsection 4.5.1.5. There are two cases in which a participant is not able to make the indicated move. The first is when a participant has to construct an argument to support a proposition to make a *claim* move in the fourth step of the protocol described in Subsection 4.5.1.2. When the participant is unable to find this support for the proposition the participant has to "concede" the game. It should be checked whether the participant does not have any retained moves left. If there are retained moves for this participant then those should be dealt with first. Retained moves for the other participant are not relevant in this case because this will not help the case of the participant who has to concede to the game. This is represented in the following formula.

EN 1
$Initiates(move(id,p,why\ \varphi,tr), ConcedeGame(\bar{p}),t) \quad \leftarrow$
$HoldsAt(Legal(move(id,p,why\ \varphi,tr)),t) \quad \wedge$
$HoldsAt(Turn(p)) \quad \wedge$
$\neg(HoldsAt(KB(since(\varphi,S),\bar{p}),t) \wedge$
$(x \in S \wedge (HoldsAt(KB(x,\bar{p}),t) \vee HoldsAt(CS(x,p),t)))) \quad \wedge$
$\neg HoldsAt(Retain(move(id2,\bar{p},s,tr2)),t)$

The other way to end a dialogue is after a *concede* move when there are no more legal moves and there is no move retained. In this case the dialogue will just not proceed because there are no more legal moves. The dialogue will not be actively ended by the formalisation.

4.6 Argument-based negotiation

The need for negotiation protocols with embedded argumentation was identified in several preceding ASPIC deliverables. To summarise, the idea is that if negotiating agents exchange reasons for their proposals and rejections, the negotiation process may be more efficient and the negotiation outcome may be of higher quality. This section especially focuses on reasons given for rejections of proposals. If an agent explains why he rejects a proposal, the other agent may be able to infer which of her future proposals will be rejected and thus, if she is rational, she will not waste effort making such proposals. Thus efficiency is promoted. In such exchanges, reasons are not only exchanged, they can also become the subject of debate. Suppose a car seller offers a Peugeot to the customer but the customer rejects the offer on the grounds that French cars are not safe. The car seller might then try to persuade the customer that he is mistaken about the safety of French cars. If she succeeds in persuading the customer that he was wrong, she can still offer her Peugeot. Thus the quality of the negotiation is promoted, since the buyer has revised his preferences to bring them in agreement with reality.

This example illustrates that a negotiation dialogue (where the aim is to reach a deal) may contain an embedded persuasion dialogue (where the aim is to resolve a conflict of opinion). The aim of this section is to formulate a protocol for negotiation with embedded persuasion dialogues about the reasons for rejecting a proposal. The key idea is that the propositional commitments incurred by the agents in the embedded persuasion dialogue constrain their behaviour in the surrounding negotiation dialogue. The question of interactions of commitments between agents engaged in different connected dialogues is considered in greater detail in Section 4.8.

Our proposal here will be stated in a dialogue game form. It will combine a negotiation protocol and language of Walton and Krabbe [83] with the dialogue system for relevant dialogues of Section 4.3.4 above, with the additional requirement of Section 4.3.6 that moves leave the players' commitments consistent. The combined protocol will satisfy the general format of Section 4.3, so that formalisation with the Event Calculus presented in Section 4.4 above and the implementation presented in Section 4.7 below are both also possible for this combined protocol.

4.6.1 A language and protocol for multi-attribute negotiation

The negotiation system we will use is that of Wooldridge and Parsons [83]. The **negotiation topic language** L_t^n of this system assumes that in a negotiation agents try to reach agreement over the values of a finite set $V = \{v_1, ..., v_m\}$ of *negotiation issues*. Each issue v can be assigned at most one value from a range $C(v)$ of values. An *outcome* of a negotiation is an assignment of values to a subset of V. A proposal is expressed in a subset of the language of first-order logic as a conjunction of expressions of the form vRc, where $v \in V$ and $c \in C(v)$ or $c =?$, (where ? technically is a free variable, capturing that the issue has not been assigned a value) and R denotes one of the relations $=, <, >, \leq$ or $\geq$.

The **negotiation communication language** L_c^n can be used to talk about proposals. The left column of table 4.4 shows the speech acts that agents can perform and the right column their possible replies. The formulas φ and φ' are elements of L_t^n. *Request(φ)* is a request for an offer. Here φ typically is wholly or partially uninstantiated (i.e., it may contains occurrences of ?): the speech act *request*($price$ =? $\wedge$ $warranty$ = 12) can be read as "What is the price if I want a 12 months warranty?". The speech act *offer*(φ) makes a fully instantiated proposal φ, and with *accept*(φ) an agent accepts an offer φ made by another agent. With *reject*(φ) such an offer is rejected. With *withdraw* an agent withdraws from the negotiation.

We next outline the **negotiation protocol** of [83] for this language, with notation slightly adapted to our purposes. A negotiation takes place between two agents, one of whom starts with either an *offer* or a *request*. The agents then take turns after each utterance, selecting their replies from Table 4.4. As the table indicates, a negotiation terminates when an agent accepts an offer or withdraws from the negotiation. Finally, moves may not be repeated by the same player.

Table 4.4: Speech acts and replies in L_c^n

Acts	Replies:
request(φ)	*offer*(φ')
offer(φ)	*offer*(φ') or *accept*(φ) or *reject*(φ) or *withdraw*
reject(φ)	*offer*(φ') or *withdraw*
accept(φ)	end of negotiation
withdraw	end of negotiation

($\varphi \neq \varphi'$)

To ensure that the offers exchanged during a negotiation and its outcome are related to an initial request, we add the following rule to the protocol of [83]:

- If *request*(φ) is the initial request of a dialogue then for any move *offer*(ψ) in the dialogue:
  - ψ is logically consistent with φ; and
  - ψ contains at least the same issues as φ.

Since issues have at most one value, this rule implies that an instantiated part of a request cannot be changed by an offer (but the offer may contain more issues than the request). Therefore:

Proposition 13 *If a negotiation that starts with a request terminates with acceptance of an offer, that offer is consistent with and fully instantiates the request.*

We illustrate the system with an example in which two agents, Paul (P) and Olga (O), negotiate over the sale of a car. The dialogue starts when Paul requests to buy a car, and shows that he is interested in the brand and the price.

P_1: $\mathit{request}(\mathit{brand} = ? \wedge \mathit{price} = ?)$
O_2: $\mathit{offer}(\mathit{brand} = \mathit{peugeot} \wedge \mathit{price} = 10000)$
P_3: $\mathit{reject}\ (\mathit{brand} = \mathit{peugeot} \wedge \mathit{price} = 10000)$
(Olga has offered a Peugeot for 10000, but Paul has rejected the offer. Olga makes him another offer.)
O_4: $\mathit{offer}(\mathit{brand} = \mathit{renault} \wedge \mathit{price} = 8000 \wedge \mathit{stereo}= \mathit{yes})$
P_5: $\mathit{reject}(\mathit{brand} = \mathit{renault} \wedge \mathit{price} = 8000 \wedge \mathit{stereo}= \mathit{yes})$

O_6: *offer*$(brand = audi \wedge price = 10000)$
P_7: *accept*$(brand = audi \wedge price = 10000)$
(Olga offers a Renault with stereo for 8000. Paul again rejects after which Olga offers a non-French car for 10000. Paul accepts and the dialogue terminates. Move O_4 illustrates that an offer may introduce additional issues, for instance, to make an offer more attractive or to make a trade-off possible.)

4.6.2 The combination

We now combine the negotiation and persuasion systems in a way that allows persuasion dialogues to be embedded in negotiation dialogues. In a negotiation dialogue it is the reject move that shows that there is a conflict between the preferences of an agent and the offer that it receives. By starting a persuasion dialogue, the offerer can question the reasons that the offeree has for rejecting. Statements made during persuasion invoke commitments that reflect the preferences of the agents. These commitments are used to restrict further negotiations.

In formally realising the combination of the two dialogue systems, the key idea is to reformulate the negotiation system in the format of the persuasion system so that the mechanisms of relevance and dialogical status can also be applied to the negotiation part of a dialogue. These mechanisms will then be used to ensure that as long as a persuasion move is legal, no negotiation move can be made: thus the protocol will capture the idea of embedding persuasion in negotiation. As for notation, the above notations for the persuasion topic and communication languages and protocol are now assumed to have a superscript p.

First the combined communication language L_c is defined in Table 4.5. As can be seen, the negotiation language is reformulated in the format of the framework of Section 4.3 by dividing the "Replies" of Table 4.4 into surrendering replies (*accept*(φ)) and attacking replies (all other replies). Next a new attacking reply is added, viz. *why-reject*(φ) as a reply to *reject*(φ). The only possible reply to this new locution other than a withdrawal is with a *claim* move from ${L_c}^p$ of which the content negates the conjunction of one or more elements of the rejected offer. Thus the player who rejected the offer can indicate which elements of the offer made him reject it. The use of this reply induces a shift from a negotiation to a persuasion subdialogue.

Next, in order to specify the combined protocol, the notion of negotiation moves must be adapted to fit the format of Definition 53 (which we leave implicit). The combined protocol is then defined as follows.

Definition 69 *(The protocol Pr for L_c.) For all dialogues d and moves m it holds that $m \in Pr(d)$ if and only if m satisfies all of the following rules.*

- *R_1: m satisfies $R_1 - R_9$ of protocols for relevant dialogues with consistent commitments, but where in R_2, L_c^p is replaced by L_c and R_6 now says that if $d =$, then $s(m)$ is of the form request(φ);*

Table 4.5: Speech acts and replies in L_c.

Acts	Attacks	Surrenders
negotiation		
request(φ)	*offer*(φ') *withdraw*	
offer(φ)	*offer*$(\varphi'))$ $(\varphi \neq \varphi')$ *reject*(φ) *withdraw*	*accept*(φ)
reject(φ)	*offer*(φ') $(\varphi \neq \varphi')$ *why-reject*(φ) *withdraw*	
accept(φ)		
why-reject$(\varphi_1 \wedge \ldots \wedge \varphi_n)$	*claim*$(\neg(\varphi_i \wedge \ldots \wedge \varphi_j))$ $(1 \leq i \leq j \leq n)$ *withdraw*	
withdraw		
persuasion		
claim(φ)	*why*(φ)	*concede*(φ)
why(φ)	*argue*(A) $(conc(A) = \varphi)$	*retract*(φ)
argue(A)	*why*(φ) $(\varphi \in prem(A))$ *argue*(B) $(B$ defeats $A)$	*concede*(φ) $(\varphi \in prem(A)$ or $\varphi = conc(A))$
concede(φ)		
retract(φ)		

- R_2: *If* $s(m) =$ *offer*(φ) *and* $s(m_1) =$ *request*(φ') *then* $\{\varphi, \varphi'\}$ *is consistent and* φ *contains at least the same issues as* φ'*;*
- R_3: *If* $s(m) =$ *offer*(φ) *then of no* $m' \in d$, $s(m') =$ *offer*(φ)*;*
- R_4: *If* $s(m) =$ *accept*(φ) *then* φ *contains no variables;*
- R_5: *If* m *is a negotiation locution then* m *replies to the most recent target to which a reply is legal;*
- R_6: *If* m *is a negotiation locution then there is no move* $m' \in Pr(d)$ *such that* $s(m')$ *is a persuasion locution;*
- R_7: *If* $s(m) =$ *offer*(φ) *then* $C_s(d) \cup \{\varphi\}$ *and* $C_{\overline{s}}(d) \cup \{\varphi\}$ *are consistent.*

Rule R_1 generalises the general structure of the persuasion protocol to the combined protocol and says that each combined dialogue starts with a request for an offer. Rules $R_2 - R_4$ formalise the negotiation protocol rules of [83] that are not implied by R_1 (see also below). Rule R_5 prevents unnecessary negotiation backtracking moves. Finally, rules R_6 and R_7 perform a key role in the embedding of persuasion in negotiation. R_6 enforces that the relation between the negotiation and persuasion parts of dialogues is one of embedding of the latter in the former (cf. [54]): as long as a persuasion move is legal, no negotiation move is legal. And R_7 formalises the intuition that offers need to respect the reasons for rejection given by the other party when these reasons have been successfully defended in an embedded persuasion dialogue.

Rule R_7 is justified by the following property of the persuasion protocol of Section 4.3.4: under some plausible assumptions on the contents of arguments a *retract*(t) move in reply to a challenge of the initial claim is always legal. Then by R_6 of the persuasion protocol, which requires retractions to be successful, a player who has defended a rejection with a *claim*(t) move in a terminated persuasion dialogue is committed to t only if he has won the persuasion dialogue about t.

The turntaking rule of the combined system is the same as for persuasion. Given L_c, this rule implies that just as in Section 4.6.1 the turn shifts after each negotiation move except after an *accept* move, which terminates a dialogue.

Finally, the new commitment rules need to be defined. In fact, they are the same as for persuasion moves in Section 4.3.3. The effects that negotiation moves have on the players' commitments are irrelevant as long as a dialogue has not terminated, since an offer commits the offeree to an action only after the offer has been accepted: so checking compliance with negotiation commitments lies outside the negotiation dialogue in which the commitment was incurred.

Note that the new system completely preserves the original persuasion system and as much as possible preserves the original negotiation system. Above we already noted that turntaking in the negotiation part is still the same. Furthermore, backtracking from negotiation moves (which was impossible in the original system) is legal in two cases only: if the one who challenges a rejection loses the resulting persuasion dialogue, s/he must move an alternative reply to the rejection, and if the other party loses such a persuasion dialogue, s/he must move a counteroffer or withdrawal in reply to the rejected offer.

4.6.3 Properties of the combined protocol

The main property of the new protocol is about the maximum number of negotiation moves needed to reach a certain agreement.

Proposition 14 *For any proposal φ the maximum length of a negotiation dialogue to end with acceptance of φ is never higher and sometimes lower in the system of Section 4.6.2 than in the system of Section 4.6.1.*

proof: This follows from the fact that the only effect of a terminated persuasion dialogue on an embedding negotiation dialogue is that it may make offers illegal since they do not respect the commitments of the other agent. Thus the number of legal offers in a negotiation according to Section 4.6.2 is never higher and sometimes lower than in a negotiation according to Section 4.6.1.

Since our persuasion protocol is not guaranteed to terminate, the same holds for our combined protocol. However, on the assumption that a persuasion dialogue always terminates, Proposition 14 implies that the 'success' result on the negotiation protocol proven by [83] still holds for our combined protocol: if the set of possible outcomes is finite then any negotiation is guaranteed to terminate with a withdraw or an accept.

4.6.4 An example

We next illustrate our new protocol by extending our example from Section 4.6.1 with an embedded persuasion dialogue. For simplicity we paraphrase the contents of the arguments. To illustrate the use of defeasible inference rules, some arguments are assumed to be constructed with presumptive argumentation schemes from [80]. In [18] it is discussed how such schemes can be formalised as defeasible inference rules and their critical questions as pointers to undercutters. Elementary inferences within arguments are paraphrased as *conclusion since premises*. All moves in the dialogue except proponent's last four moves reply to their immediate predecessor.

P_1: *request*(*brand* = ? $\wedge$ *price* = ?)
O_2: *offer*(*brand* = *peugeot* $\wedge$ *price* = 10000)
P_3: *reject* (*brand* = *peugeot* $\wedge$ *price* = 10000)
Olga now exploits the additional features of the protocol by asking Paul why he rejected the offer.
O_4: *why-reject*(*brand* = *peugeot* $\wedge$ *price* = 10000)
Paul now meets Olga's challenge of his rejection so that the negotiation shifts into a persuasion. All persuasion moves below until P_{14} reply to their immediate predecessor.
P_5: *claim*($\neg$ *brand* = *peugeot*)
Paul says that he rejected the offer since he does not want a Peugeot. He is now committed to the content of his claim.
O_6: *why* ($\neg$ *brand* = *peugeot*)
P_7: *argue* ($\neg$ *brand* = *peugeot since brand* = *peugeot* $\rightarrow$ *brand* = *french and* $\neg$ *brand* = *french*)
It turns out that Paul rejected the offer since a Peugeot is a French car and he does not want French cars. If Olga now simply concedes Paul's claim as an alternative reply to P_7, the persuasion dialogue terminates and the negotiation is resumed. Then Olga cannot reply to P_3 in the same way as in section 4.6.1 by offering another french car. Olga could offer a non-French car (as in O_6 in section 4.6.1) but she chooses to try to persuade Paul that he is wrong in not wanting a French car and she therefore challenges Paul's second premise.
O_8: *why* ($\neg$ *brand* = *french*)
P_9: *argue* ($\neg$ *brand* = *french since having french cars is bad; this is so since french cares are unsafe and having an unsafe care usually is bad.*)
Paul defends his second premise with an argument from (bad) consequences.
O_{10}: *why* (*french cars are unsafe*)

P_{11}: *argue* (*french cars are unsafe since car magazine mycar says so and mycar are experts about cars*)
This is a defeasible argument based on the argumentation scheme from expert opinion: "what experts say is normally true".
O_{12}: *argue* (*magazine mycar is biased since magazine mycar is german and german car magazines are usually biased against french cars*)
Using a default rule, Olga constructs an undercutter of the argument from expert opinion, namely that this expert is biased.
P_{13}: *concede* (*magazine mycar is german*)
P_{14}: *concede* (*german car magazines are usually biased against french cars*)
Even though Paul has conceded the premises of Olga's undercutter, he can still move a counterargument, since the argument is defeasible because it uses a default rule. Paul chooses to rebut the undercutter, using another default rule.
P_{15}: *argue* ($\neg$ *magazine mycar is biased since magazine mycar has a very high reputation and car magazines with high reputation usually are not biased*)
Note that P_{14} is a second and P_{15} a third reply to O_{12}.
O_{16}: *why* (*magazine mycar has a very high reputation*)
Let us assume that Paul now realises that he has no plausible way to defend his premise that the car magazine has a high reputation. At this point, all of Olga's persuasion moves are relevant targets for Paul. He could, for instance, move another rebuttal of Olga's undercutter, or another argument why French cars are not safe or why he does not want french cars. But suppose that Paul sees no plausible way of doing so and instead retracts that he does not want French cars by moving an alternative reply to O_8 and then retracting his main claim as a second reply to O_6.
P_{17}: *retract* ($\neg$ *brand* = *french*)
P_{18}: *retract* ($\neg$ *brand* = *peugeot*)
Now Paul has no legal persuasion moves any more since all targets have become irrelevant: since Paul has surrendered to O_6, his main claim P_5 cannot be changed from *out* to *in*. So the persuasion dialogue terminates and the negotiation resumes with Olga to move after P_2. Since with P_{18} Paul has ended his commitment to his main claim, Olga is now allowed to offer another French car, perhaps even a Peugeot for a lower price. The negotiation could now continue as in Section 4.6.1 with move O_4.

It is instructive to construct the dialectical graph of arguments and counterarguments exchanged by Paul and Olga during the persuasion dialogue ($p \rightsquigarrow q$ reads as "if p then usually q").

The graph contains a simple argument game according to the proof theory of the underlying logic. Since on the basis of the information exchanged during the persuasion dialogue no other counterarguments to one of these three arguments can be constructed, the graph is actually a proof that, on the basis of this information, the proposition $\neg$ *peugeot* is justified. However, the last argument in the graph has one challenged premise, viz. *highrep*, so this argument is not defended (indicated by the dotted box). The defended part of the graph is instead a proof that $\neg$ *peugeot* is not justified on the basis of all *defended* information.

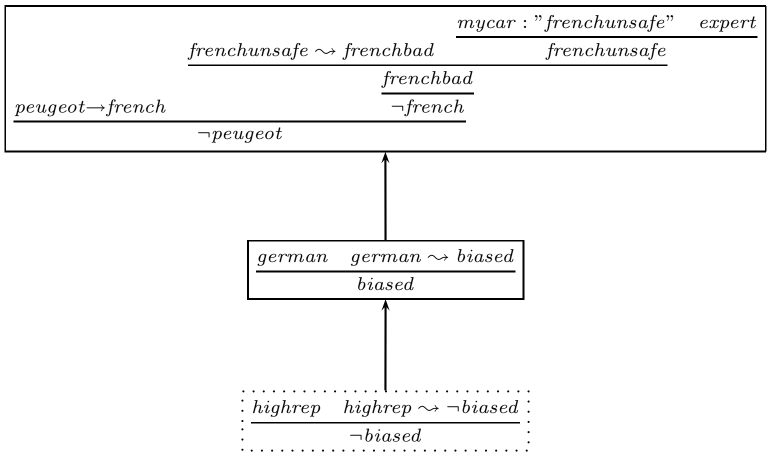

Figure 4.5: The dialectical graph

4.6.5 Adding dialogue policies for the e-Consent Scenario

We next outline a way in which dialogical agents could be designed that interact according to the combined negotiation-persuasion protocol, in the context of the *e-Consent Scenario* of Section 6 of ASPIC Deliverable D2.3 [10]. In this scenario, a requesting agent requests information about a patient and a responding agent must decide about this request. Here, we model this as a negotiation dialogue, since a request here for information is a request to another agent to perform an action (i.e., retrieve some data and transfer it) which will likely involve resource-use. This is unlike a dialogue where some the requesting agent simply seeks to determine whether or not a proposition is true. The ideas in this subsection were developed in collaboration with and are also reported in [29] (where these ideas are applied to information exchange between police forces in crime investigations).

Our idea is to specify policies for what an agent will choose to do at various points in a dialogue. Two main kinds of dialogue policies need to be specified, viz. for negotiation and for persuasion. One of the first proposals for formal dialogue policies for argumentation dialogues was that of Parsons, Wooldridge and Amgoud in [61], who called them *"agent attitudes"*. (See also Section 4.5.1.2 above.)[3] That paper only defined policies for persuasion; because here we are particularly interested in policies for negotiation, we largely adopt their persuasion policies. An important difference with [61] is that when a policy requires an agent to construct arguments, in our case the agent only reasons with his initial knowledge base plus the propositions that he has explicitly conceded in the persuasion dialogue, while in [61] the agent must also reason with everything the other agent has said, regardless of whether he has conceded to this or not.

[3]Another early proposal was contained in [74].

We regard the latter assumption as less realistic than the one we have adopted.

Our negotiation policies consider two issues: the normative issue of whether accepting an offer is obligatory, permitted or forbidden, and the teleological issue of whether accepting an offer is in the agent's interest. Accordingly, we assume that the agents' internal knowledge bases contain relevant domain knowledge about their obligations and permissions and about the likely consequences of their communicative acts for the achievement of their internal goals and desires. We also assume that they have the appropriate mechanisms to reason about these issues. Reasoning about desires and goals can, for instance, be modelled as proposed in Chapter 3 above.

Of course the dialogue policies can be different for the requesting and the responding agent. Also, different responding agents may have different policies. One agent, for example, might easily be persuaded to give information and thus agree with every request he receives, while another agent may guard his secrets more closely, adding extra conditions to the information exchange to make sure the information does not fall into the wrong hands. For simplicity we have chosen to specify just one set of policies which we think best suits our domain (although we will comment on other design choices when relevant).

4.6.5.1 Notation

The following notation will be used in the policies. Recall first that L_t^n is the negotiation topic language and L_t^p is the persuasion topic language. The content of an offer is a conjunction of literals from L_t^n. For any formula p of L_t^n and agent a the notation p^a denotes the conjunction of all conjuncts in p that describe actions performed by a. Also, $KB_d(a) \subseteq L_t^p$ denotes the set of internal beliefs of agent a at dialogue stage d. Note that these beliefs may change during persuasion if the agent concedes or retracts propositions and then brings his internal beliefs in agreement with his (public) commitments. Finally, we assume that each offer content p is divided into two conjunctive parts $q \wedge c$, where q are the essentials of an offer, which the offeror does not want to be changed in counteroffers, and where c are the conditions of the offer, which the offeror does not mind to be changed in counteroffers. For instance, an offer "send me a file containing all you know about patient Jones" can be decomposed into an essential element "inform me about all you know about patient Jones" and a condition "send me the information in a file". The offeror thus indicates that he does not want to receive counteroffers "I will tell you how to find out everything about patient Jones" but that he does not mind receiving a counteroffer "I will tell you everything I know about patient Jones over the phone but I will not send you a file".

We now specify the dialogue policies, starting with the negotiation policies.

4.6.5.2 Negotiation policies

Responding to an offer We first specify the policy agent a should apply in responding to an offer in a dialogue. Note that the offer can be the initial one (i.e., a request) or a counteroffer.

When agent a receives an offer $p = q \wedge c$, where q are the essentials of the offer and c the conditions, he should perform the following actions:

- Determine whether $KB_d(a)$ supports a (justified/defensible) argument for the conclusion that p^a is obligatory. If there is such an argument, then accept the offer. Otherwise,
- Determine whether $KB_d(a)$ supports a (justified/defensible) argument for the conclusion that p^a is forbidden. If there is such an argument, then reject the offer. Otherwise,
- Determine whether $KB_d(a)$ supports a (justified/defensible) argument for the conclusion that p^a violates a's interests. If there no such argument, then accept the offer. Otherwise:
  - Find a subset-minimal set $c' \subseteq L_t^n$ such that $KB_d(a) \cup c'$ supports a (justified/defensible) argument for the conclusion that p does not violate a's interests and $KB_d(a)$ does not support a (justified/defensible) argument for the conclusion that $q \wedge c'$ is forbidden.
  - If there is one such set c', then make a counteroffer $q \wedge c'$.
  - If there are more than one such sets, make a counteroffer $q \wedge c'$ where c' is a subset-minimal set that has a maximal number of elements in common with c.
  - Otherwise reject the offer p.

This negotiation policy contains a number of design choices. It is especially suitable for those multi-agent systems where the overall goal of the system is to exchange as much information as possible while respecting the relevant regulations. Of course, in other domains a choice for other types of agents can be made. For example, the agent could also reject the request for information if there is no justified argument which says he is obliged to give the information. Furthermore, in our design of the policy we have not yet specified whether the agent needs justified or defensible arguments for his conclusions. This can have a significant impact on the behaviour of the agent. For example, an agent who accepts defensible arguments that say that it is not forbidden to give a certain piece of information will be more readily persuaded than an agent who only accepts justified arguments for the conclusion that giving the information is not forbidden. So the first agent, who only needs a defensible argument, will not be as protective of his information as the second agent, who needs a justified argument. Different policies are also possible with respect to making a counteroffer. We have described a policy where having as few extra conditions as possible is more important than making a counteroffer which includes the conditions offered by the opponent. In our policy it is thus possible to delete some or all conditions of the opponent's offer. Another policy would be one where an agent can only add conditions to the original offer. Another choice option is in how an agent should internally reason about whether an offer is in the agent's interests. Is the reasoning only about his own actions or also about the actions of the other agent? The different design options mentioned here may be mostly empirical questions, to be answered by domain analysis.

Responding to a rejection The next policy describes responding to a rejection.

- First respond with a *why-reject* move. If the resulting persuasion subdialogue is won, then it is the other agent's turn. Otherwise,
- If the *reject* move responded to the initial offer then reply with a *withdraw*, while
- If the *reject* move responded to a counteroffer, backtrack to the target of the *reject* move.
  - If an alternative counteroffer exists that satisfies the policy for responding to an offer then make it.
  - Otherwise reply with a *withdraw*.

4.6.5.3 Persuasion policies

We now turn to a less formal specification of persuasion policies, which determine how an agent should respond to *argue* and *why* moves. Other persuasion policies can be developed similarly. We first specify how an agent can respond to an argument.

Responding to arguments Can the agent construct a (justified/defensible) counterargument?

- If the agent can construct such an argument, it should be moved in the dialogue.
- If the agent cannot construct such an argument and there is a premise p of the opponent's argument for which the agent has no (justified/defensible) argument, then the agent should ask a *why* p question.
- If the agent cannot construct such an argument and for all of the premises of the opponent's argument has a justified argument then concede to the conclusion of the opponent's argument.

Responding to why moves Suppose that the requesting agent asks a *why* p question in response to a *claim* or *argue* move by the responding agent. Can the responding agent construct a (justified/defensible) argument for p?

- If the agent can construct such an argument, it should be moved in the dialogue.

- If the agent cannot construct such an argument, he should retract his claim p or the conclusion of the argument of which p is a premise.

There are several points to note regarding these persuasion policies. Firstly, as in the negotiation policies, we have not specified what kind of arguments (justified or defensible) an agent needs for his decisions. Secondly, the agent is cooperative in that he only asks *why p* questions if he does not have an argument for p. Clearly, several other policies are possible. One option we want to explore in future research is to make policies partly domain-specific. For example, the second part of the policy for responding to arguments could be refined such that premises are never challenged when they are about subject X and/or when they are claimed by person Y, who is considered to be a reliable source of information concerning X.

4.7 Implementation

In this section, two software implementations of dialogue protocols are reported. The first is an implementation in Prolog of the formalization using the Event Calculus of the Persuasion Protocol presented above in Sections 4.3 and 4.4. This first implementation is included to show the support which the Event Calculus formalization provides for implementation, a feature which strengthens the case for such a formalization. The second is an implementation using Tuple Spaces of the protocol for Information-Seeking-dialogues-with-Permissions presented in Chapter 6 of ASPIC Deliverable D2.3 [10]. This second implementation is included in addition because the Tuple Space semantics of Deliverable D2.3 has guided implementation of the prototype dialogue manager of ASPIC Workpackage 4, and may also guide development of the large-scale Demonstrator of ASPIC Workpackage 5. The Information-Seeking-with-Permissions dialogue protocol was applied to the *e-Consent Scenario* in Deliverable D2.3.

4.7.1 Persuasion protocol implementation

In this Section, we discuss an implementation in Prolog of the Persuasion dialogue protocol of Section 4.3.3. The source code for the full implementation is available on the web at `www.ewi.utwente.nl/∼bodenstaffl`.

4.7.1.1 The basics

A Prolog program consists of a database of clauses that specify *which* facts can be computed and not *how* they are computed. Variables in Prolog are written as capitals and anonymous variables are written as underscores (_). Constants are denoted by lowercase letters or numbers and every clause ends with a period. These clauses can be *facts* as well as *rules*. To illustrate the basics of Prolog a simple example with two facts and one rule is used.

```
holds_at(turn(p), 2).
holds_at(legal(move(Id, P, S, Tr)), 2).
```

The first fact states that `holds_at(turn(p), 2)` is true and thus that at time point 2 it is the turn of participant `p`. The second fact states that `holds_at(legal(move(Id, P, S, Tr)), 2)` is true and therefore that at time point 2 all moves are legal. A typical rule is of the form `A :- B, C`. This rule means that *if* B and C are both true *then* A is true. Logical "or" is written as ";". Next, the rule of this example is stated.

```
initiates(X, turn(P), T) :-
   holds_at(legal(X), T),
   holds_at(turn(Q), T),
   P \== Q
   ;
   holds_at(turn(o), T).
```

This rule states that *if* the predicates `holds_at(legal(move(X)), T)`, `holds_at(turn(Q), T)` and `P` and `Q` are not equal are all true *or* if `holds_at(turn(o), T)` is true *then* `initiates(move(X), turn(P), T)` is also true. After consulting a database consisting of these clauses in a Prolog interpreter, queries can be entered in the interpreter about the database. These queries are to be entered after the prompt (?−) as in the next example.

```
   ?- initiates(move(1, p, claim q, 0), turn(p), 2).
```

Prolog will try to match facts in its database with the query. Here there is no match because the only two facts in the database are `initially_P(turn(p))` and `holds_at(legal(move(Id, P, S, Tr)), 2)`. When this attempt fails Prolog will try to find rules where the conclusion matches with the query. This will succeed because the only rule in this example has a matching conclusion. Prolog will assign values to the variables in the rule. Prolog will bind `move(1, p, claim q, 0)` to `X`, bind `p` to `P` and bind 2 to `T`. Next Prolog will try to satisfy all variables in the premises. In this case Prolog will answer "no" to the query because `p` will bind to `Q` and this results in the same constant assignment to `P` and `Q` therefore the third condition will fail; `P \== Q` is false. Also the last condition, `holds_at(turn(o), T)`, is false because there is no matching fact.

Besides queries where Prolog is only able to answer "yes" or "no" it is also possible to leave anonymous variables in the query. Now Prolog will return all possible values of this variable if there are any. Another example of a query is as follows.

```
   ?- initiates(move(1, R, claim q, 0), turn(p), 2).
```

Now Prolog will answer: "`R = o`" because with the assignment of `o` to `R` the rule is satisfied. After this answer the user of the interpreter enters a semicolon after which Prolog will give another assignment if there is one. If not Prolog will return "no". This feature will prove to be very useful for the implementation of the persuasion game. Although this is a very concise introduction to Prolog it is sufficient to understand the implementation described in this thesis.

4.7.1.2 Implementation

As for the implementation, here only the important implementation choices, the specific program characteristics and an example run are surveyed.

Axioms

Largely, the implementation of the axioms of the Event Calculus resembles the format of the above formalisation of the Event Calculus, except for the treatemnt of negation. Since Prolog, interprets negation as 'negation as failure' while our EC formalisation contains classical negations, some care is needed in the transformation to Prolog clauses. The negations in the conditions of the general EC axioms can safely be translated as negation-as-failure to capture the law of inertia. Furthermore, the negations in the conditions of our specific axioms can be translated as negation-as-failure since in the current application the closed-world assumption can safely be made: a dialogue state can be completely specified, so what is not specified can be assumed false. However, for classical negations in the consequents of axioms a special predicate `not_holds_at` has to be introduced and the program has to be designed such that for no fluent both `holds_at(F,T)` and `not_holds_at(F,T)` can be derived (in our program this was straightforward). The implementation of, for example, Axiom 4 now is as follows.

```
% Axiom 4
not_holds_at(F, T) :-
            initially_n(F),
            \+ declipped(0, F, T).
```

Throughout the entire implementation this new predicate is used instead of the negation of `holds_at`. Another difference is that Axiom 7 has not been implemented. Axiom 7 ensures that an event takes a positive amount of time. In this thesis events have no duration but happen at one time point, recall that the predicate `happens` is binary in this thesis and not ternary, and therefore it is not necessary to specify that the duration of an event has to take a positive amount of time. The last difference is that in the implementation a ternary predicate `happens3` is used although the participants are able to enter the binary predicate `happens` to add events to the database. As soon as a binary `happens` clause is added to the database the program deals with this according to the following rule where the binary predicate can be read as ternary by duplicating the time point.

```
happens3(A, T1, T1) :- happens(A, T1).
```

Rules

Moreover, the implementation of the rules resembles the formalisation in the Event Calculus for the most part. Some rules may seem quite different at first glance because Prolog requires the conditions to be in Horn clause format and therefore many of the rules had to be rewritten. Rule I7 is a good example of this rewriting. First the rule as stated in the Event Calculus and after that the rule as implemented in Prolog is denoted.

I7

$$\begin{aligned}&\mathit{Initiates}(\mathit{move}(\mathit{id}, p1, s, \mathit{tr}), \mathit{Turn}(p2), t) && \leftarrow\\ &\quad \mathit{HoldsAt}(\mathit{Legal}(\mathit{move}(\mathit{id}, p1, s, \mathit{tr})), t) && \wedge\\ &\quad ((\mathit{id} = 1 \wedge p1 = P \wedge \mathit{tr} = 0 \wedge t = 1 \wedge p2 = O)\\ &\quad \vee\\ &\quad (\mathit{id} = 2 \wedge p1 = O \wedge \mathit{tr} = 1 \wedge t = 2 \wedge p2 = P))\end{aligned}$$

```
% I7
initiates(move(Id, P1, S, Tr), turn(P2), T) :-
                       Id = 1,
                       P1 = p,
                       Tr = 0,
                       P2 = o,
                       holds_at(legal(move(Id, P1, S, Tr)), T)
                       ;
                       Id = 2,
                       P1 = o,
                       Tr = 1,
                       P2 = p,
                       holds_at(legal(move(Id, P1, S, Tr)), T).
```

In addition to rewriting the rules in Horn clauses, the evaluation of arguments is also different from the way they are formalised in the Event Calculus. An argument in the implementation is represented as `since(Phi, Psi)` and should be read as "`Phi` since `Psi`". The support for proposition `Phi`, namely `Psi`, should be entered as a list. In Prolog a list is of the form: `[p, q, r]`. This list structure is necessary for the implementation. In several rules one of the conditions to be checked is whether a proposition is an element of the premises of an argument. Rule E3 is a good example of the use of this list structure. The first part of the "or" clause is deals with the conclusion of the argument. The second part of the "or" clause deals with the premises of the argument. Now the predicate `member_(X, XS)` makes sure that the only propositions added to the commitment set are elements of the premises of the argument.

```
% E3
```

```
initiates(move(Id, P, argue(since(Phi,Psi)), Tr), cs(P,Chi), T) :-
                    Phi = Chi,
                    holds_at(legal(move(Id, P, argue(since(Phi,Psi)), Tr)), T),
                    holds_at(turn(P), T)
                    ;
                    member_(Chi,Psi),
                    holds_at(legal(move(Id, P, argue(since(Phi,Psi)), Tr)), T),
                    holds_at(turn(P), T).
```

4.7.1.3 Running the program

The implementation has been tested extensively. In this subsection the method of testing is explained. It should first be noted that in the test runs no defeat relations where used. This relation should be used as a condition in L3 where the legality of an *argue* move after another *argue* move depends on the defeat relations between those two arguments. It is possible to use the implementation including the defeat conditions. The user should then add $\texttt{Initially}_\texttt{p}$ clauses for those defeat relations at the beginning of the dialogue. It is also important to realise that it is assumed that the available arguments and propositions of the knowledge base of the participants are part of the internal representation and thus are not denoted in the implementation.

After loading the program in a Prolog interpreter the user is able to state queries about fluents at certain time points. The following queries are useful for planning the first move and should be entered one by one.

```
holds_at(turn(P), 0). holds_at(cs(P,A),0).
holds_at(legal(move(1,P,S,0)), 0).
```

Prolog will show whose turn it is at time point 0, what the commitments of the participants are at time point 0 and which moves are legal at time point 0. When the dialogue proceeds these queries can be used with different time points to plan the following move and to check whether previous moves had the expected effect on, for example, the commitment sets. These queries can also be entered with constants and in negated form. Consider the following example queries.

```
not_holds_at(turn(o), 0). holds_at(cs(p,q),3).
not_holds_at(legal(move(3,o,claim q,2)), 4).
```

When the user has decided on the move he wants to make he enters this move with an `assert` command. `Assert` is a built-in predicate that adds its argument as a fact to the program that is loaded at that time. This means that by entering the next clause in the Prolog window that `happens(move(1, p, claim(a), 0), 1)` becomes a fact in the program. Asserting a clause always succeeds in Prolog.

```
assert(happens(move(1,p,claim(a),0),1)).
```

By asserting several `happens` clauses a dialogue is simulated. In the next subsection a test run is presented.

Example run

We now give an example run of the program. A user wants to simulate the following dialogue in the protocol for liberal dialogues to find out whether this is a legal dialogue according to the protocol. The knowledge base of the proponent is $\{z; q; z, q \to a\}$ and the one of the opponent is $\{d; d \to c\}$.

Time point	Move	CS_P	CS_O
T_1	$move(1, P, claim\ a, 0)$	$\{a\}$	
T_2	$move(2, O, why\ a, 1)$	$\{a\}$	
T_3	$move(3, P, argue\ A, 2)$ $conc(A) = a$, $prem(A) = z, q$	$\{a, z, q\}$	
T_4	$move(4, O, argue\ B, 3)$ $conc(B) = c$, $prem(B) = d$	$\{a, z, q\}$	$\{c, d\}$
T_5	$move(5, P, why\ d, 4)$	$\{a, z, q\}$	$\{c, d\}$
T_6	$move(6, O, retract\ d, 5)$	$\{a, z, q\}$	$\{c, d\}$

After loading the program the user enters the fist move by the proponent. This move is a *claim a* move and is entered by the `assert` command.

```
?- assert(happens(move(1,p,claim(a),0),1)).

Yes ?-
```

The user now tries to plan his next move and enters the following query to find out which participant is allowed to utter which locution.

```
?- holds_at(legal(move(2,P,S,1)), 2). P = o S = why(a) ;

P = o S = concede(a) ;

No ?-
```

The user finds out that only the opponent is allowed to make a move and that its only legal locutions are `why a` and `concede a`. The user of the system now proceeds his original plan and asserts the following move. After that also the remaining part of the dialogue is given as it is simulated in Prolog. After the planned actions the user wishes to know which legal moves are left.

```
?- holds_at(legal(move(3,P,A,2)), 3).
```

```
P = p A = argue(since(a, _G511)) ;

P = p A = retract_(a) ;

No ?- assert(happens(move(3,p,argue(since(a,[z,q])),2),3)).

Yes ?- holds_at(legal(move(4,P,A,3)), 4).

P = o A = why(z) ;

P = o A = why(q) ;

P = o A = argue(_G481) ;

No ?- assert(happens(move(4,o,argue(since(c,[d])),3),4)).

Yes ?- holds_at(legal(move(5,P,S,4)), 5).

P = p S = concede(d) ;

P = p S = why(d) ;

P = p S = argue(_G481) ;

P = p S = concede(c) ;

No ?- assert(happens(move(5,p,why(d),4),5)).

Yes ?- holds_at(legal(move(6,P,S,5)), 6).

P = o S = argue(since(d, _G660)) ;

P = o S = retract_(d) ;

No ?- assert(happens(move(6,o,retract_(d),5),6)).

Yes ?- holds_at(legal(move(7,P,A,N)), 7).


P = o A = concede(a) N = 1 ;

P = p A = retract_(a) N = 2 ;

P = o A = why(z) N = 3 ;

P = o A = why(q) N = 3 ;
```

```
P = o A = concede(z) N = 3 ;

P = o A = concede(q) N = 3 ;

P = p A = argue(_G500) N = 4 ;

P = p A = concede(c) N = 4 ;

P = p A = concede(d) N = 4 ;

No ?-
```

It should be noted that two other moves are legal at time point 7 but are not returned by Prolog in the last question `holds_at(legal(move(7,P,A,N)),7)`. These moves are `move(7,p,argue(since(a,Phi)),2)` where `Phi` cannot be `[z,q]` and `move(7,o,argue(Psi),3)` where `Psi` cannot be `[d]` because the moves with these propositions already happened at time point 3 and 4 respectively. Prolog fails to return these two clauses because not *all* available propositions are legal to use in these *argue* moves. However, when the user asks whether a *specific* argue move is legal, for example, `holds_at(legal(move(7,p,argue(since(a,b)),2)),7)` Prolog will answer affirmatively.

4.7.2 Information-seeking protocol implementation

In Section 6 of ASPIC Deliverable D2.3 [10], we presented the syntax and a semantics for an Information-seeking dialogue protocol in which participants could seek, and present, arguments to justify their rights to receive requested information. This protocol was applied to the *e-Consent Scenario*, in which an agent seeks medical information from a patient record database outside the requesting agent's normal jurisdiction, and thus is required to provide, to an agent controlling access to the database, a justification for the information sought. The agent dialogue can therefore be considered as an Information-Seeking dialogue-with-Permissions. Such dialogues had been considered once before, in [17], which presents an axiomatic semantics for these dialogues.[4] However, this earlier formalization did not include the possibility of arguments for or against permissions.

Accordingly, Deliverable D2.3 presented a novel syntax and a denotational semantics, in which utterances in a dialogue under the protocol are translated into operations on partitioned *Tuple Spaces* [35, 39]. These spaces, originally developed as models of distributed computation, are databases or repositories whose access is shared between the dialogue participants. An implementation of such a semantics may be viewed as an instantiation of an abstract *Co-ordination Artefact* between agents, a formal theory of which is now emerging [57, 78].

[4]See Section 2 of ASPIC Deliverable D2.3 [10] for definitions of different types of semantics for agent communications languages and protocols.

Subsequent to the completion of Deliverable D2.3, the research underlying our work was reported in [30, 31].

One of the reasons for considering a semantics using Tuple Spaces was a belief that this should facilitate implementation of the protocol. In order to assess this belief, we undertook an implementation of the protocol presented in Section 6 of ASPIC Deliverable D2.3 [10]. The implementation was executed using the *TuCSoN* software platform for tuple center applications [26], developed at the Alma Mater Studiorum – Università di Bologna, in Cesena, Italy.[5] Tuple Centers are an extension of the notion of Tuple Spaces in which the shared spaces are not merely passive, but may be programmed to act in anticipation of or react in response to events such as the deposit in them of particular tuples [56]. In the case of the *TuCSoN* framework, programming of the shared spaces is undertaken in a Prolog-like language. Because this implementation was not intended for production use, only the protocol itself (and the supporting tuple space semantic framework) was implemented, and no agents capable of interacting via the protocol were created. The selection of locutions, within those legally-permitted, at each step in a dialog and the creation of content for these locutions, was therefore left to human participants. The implementation was undertaken on a standard desktop PC running linux, with simulation of the client (requesting access to some information) and server (controlling access to that information) enabled through *TuCSoN*. The implementation is available from: `www.csc.liv.ac.uk/research/techreports/tr2005/tr05010abs.html`.

The key outcome of this exercise was to demonstrate how readily the protocol could be successfully implemented. Prior experience with developing a multi-agent co-ordination application using *TuCSoN* meant that no learning of the platform was required, which no doubt eased implementation. One issue that did arise in the implementation concerned the partitioned nature of the tuple space. The semantic framework described in Section 6.2 of Deliverable D2.3 [10] partitions the tuple space for a dialog into four subspaces for each client-server pair, with potentially-different access for different clients to these subspaces. During the implementation, it was found that the current available version of *TuCSoN* does not permit a space to be partitioned in this way explicitly, and so we developed a virtual, on-the-fly, partitioning of the tuple center. This was achieved through the use of client identifiers in the names of output tuples created in response to successful requests for information. In this way, clients without the appropriate identifier would not be able to read the particular tuples, thus maintaining information security. Of course, this is not an ideal solution, since in real-world applications client identities may be emulated by malevolent others, but is the best possible given the current architecture of *TuCSoN*.[6]

[5]Available from: `http://lia.deis.unibo.it/research/tucson/`

[6]We understand that the ability to partition Tuple Centres may be added as a feature to *TuCSoN* in a near-term release.

4.8 Combining Dialogues

In Section 4.6, we presented a protocol for negotiation dialogues in which persuasion dialogues may be embedded. Embedding is one way in which different dialogues may be combined, and it has received some attention in the philosophy of argumentation literature, for example in [81]. Real-world dialogues, whether between humans or software agents or both, may combine different types of dialogues in other ways, for instance, sequentially, or in parallel. To represent such different possible combinations, Reed [70] proposed a formalism for combinations of dialogues of different types, with no restrictions on the types of dialogues which can be represented. Reed's formalism, however, does not permit parallel combination of dialogues and it is descriptive rather than generative: in other words, it could not (on its own) be used to create automated dialogues between agents.

In response to Reed's framework, McBurney and Parsons proposed in [54] a generative formalism which permitted several different forms of combinations of atomic dialogue types between the same group of participants, including: *iteration* (repeated occurrence of dialogues of the same type); *sequential combination* of two or more dialogues of any types; *parallel combination* of two or more dialogues of any types; and *embedding* of one dialogue inside another dialogue, both of any type. The formalism was represented in a modal logical formalism similar to that of Propositional Dynamic Logic [43] and inspired by Parikh's *Game Logic* [58].[7] In addition, the framework proposed a Control Layer, above the layer at which dialogue combinations were represented, to allow participants to agree to initiate a dialogue or a specified combination of dialogues, of a given type (or types) on a given topic (or topics).

One issue of importance in considering combinations of dialogues is that of the interaction of commitments made by the participants in different dialogues. There are no universally-applicable rules for how commitments should interact, as the following example illustrates (taken from [54]). For dialogues conducted sequentially, one opinion may be that the commitments incurred in earlier dialogues should take precedence over those from later ones, as is usually the case in legal and contractual domains. Alternatively, another opinion may be that the commitments incurred in later dialogues should take precedence over those from earlier ones, as is usually the case with party political promises, or edicts issued by religious authorities. It is clear that the rules for commitment-interaction adopted in any specific combination of dialogue types will depend on the wider context of the dialogues and, perhaps, on the preferences and goals of the participants.

The formalism of McBurney and Parsons [54] distinguished between dialogical commitments, which only concern matters inside a given dialogue (such as a responsibility to respond to a question), and semantic commitments, which create an obligation on a participant in the external world beyond the dialogue (such as a responsibility to pay for an agreed purchase). That formalism also

[7]The formalism of [54] did not expressly represent interleaving of two or more dialogues, but this should also be possible.

allowed participants to agree at the Control Layer how commitment-interaction in specific dialogue combinations should be addressed. In other words, the rules for commitment-interaction in that framework were decided at dialogue-run-time, by the participants, rather than at protocol-design-time, by the protocol designer. The latter approach, of course, was adopted in the negotiation-with-embedded-persuasion dialogue combination presented above in Section 4.6. As with the commitment-interaction rules themselves, which approach is more applicable will depend on the nature and wider context of the application and on the preferences and goals of the participants or the designer.

Further research is needed to make use of dialogue-combination frameworks such as that of Reed and that of McBurney and Parsons within the ASPIC argumentation framework. This work has commenced [1], and will continue between the ASPIC partners.

Chapter 5

Conclusion

Argumentation theory is seen as a foundation for reasoning systems. Consequently, an increasing number of argumentation systems have been proposed. While, these systems use generally the same acceptability semantics, they differ in the way they define their logical language, the notion of argument and the defeasibility relation. These last are defined in *ad hoc* way and this leads the systems to encounter some problems such as returning counter-intuitive results.

In order to avoid such problems, in a previous deliverable we have defined some postulates or axioms that any argumentation system should satisfy. These postulates govern the well definition of an argumentation system and guarantee the safety of its outputs. We have focused on four important postulates: the *closeness*, the *direct consistency*, the *indirect consistency* of the results of a system and *non-contamination*. These last are violated by several argumentation systems such as [38, 42, 68]. Then we have presented different solutions that warrant those postulates. In particular, we have proposed two closure operators that allow to make more explicit some implicit information. These closure operators solve the problems encountered by the argumentation systems defined in in [68, 42, 38].

This deliverable reports the solution chosen to be implemented by the ASPIC consortium. It presents first a general argumentation-based framework for inference, then this last is extended for making decisions. This offers for the first time a coherent setting for argumentation-based inference and decision. The properties of the framework are deeply studied.

Concerning dialogue, a formal framework which captures different types of dialogues has been proposed. Moreover, different protocols for persuasion dialogues have been analyzed, and two small-scale implementations reported.

Bibliography

[1] L. Amgoud. Towards an abstract dialogue framework. Technical report, Paul Sabatier University, 2005.

[2] L. Amgoud. A general argumentation framework for inference and decision making. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence, UAI'2005*, 2005, to appear.

[3] L. Amgoud, J-F. Bonnefon, and H. Prade. An argumentation-based approach to multiple criteria decision. In *Proceedings of the 8th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, ECSQARU'2005*, pages 269–280, 2005.

[4] L. Amgoud, M. Caminada, C. Cayrol, S. Doutre, MC. Lagasquie, H. Prakken, and G. Vreeswijk. Draft formal semantics for inference and decision-making. Technical report, Deliverable D2.2 of ASPIC project, 2004.

[5] L. Amgoud, M. Caminada, C. Cayrol, S. Doutre, MC. Lagasquie, H. Prakken, and G. Vreeswijk. Theoretical framework for argumentation. Technical report, Deliverable D2.1 of ASPIC project, 2004.

[6] L. Amgoud, M. Caminada, S. Doutre, H. Prakken, and G. Vreeswijk. Draft formal semantics for aspic systems. Technical report, Deliverable D2.5 of ASPIC project, 2005.

[7] L. Amgoud and C. Cayrol. Inferring from inconsistency in preference-based argumentation frameworks. *International Journal of Automated Reasoning*, Volume 29 (2):125–169, 2002.

[8] L. Amgoud and C. Cayrol. A reasoning model based on the production of acceptable arguments. *Annals of Mathematics and Artificial Intelligence*, 34:197–216, 2002.

[9] L. Amgoud, C. Cayrol, and M.-C. Lagasquie-Schiex. On the bipolarity in argumentation frameworks. In J. Delgrande and T. Schaub, editors, *Proc. of the 10th NMR workshop (Non Monotonic Reasoning), Uncertainty Framework subworkshop*, pages 1–9, Whistler, BC, Canada, 2004.

[10] L. Amgoud, S. Doutre, P. McBurney, and S. Parsons. Draft formal semantics for communication, negotiation and dispute resolution. Technical report, Deliverable D2.3 of ASPIC project, 2004.

[11] L. Amgoud and H. Prade. Using arguments for making decisions. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pages 10–17, 2004.

[12] A. Artikis, M. J. Sergot, and J. Pitt. An executable specification of an argumentation protocol. In *Proceedings of the Ninth International Conference on Artificial Intelligence and Law (ICAIL-03)*, pages 1–11, New York, 2003. ACM Press.

[13] ASPIC. Theoretical framework for argumentation. Technical report, Deliverable D2.4 of ASPIC project, 2004.

[14] K. Atkinson, T. Bench-Capon, and P. McBurney. Computational representation of practical argument. *Synthese: Knowledge, Rationality and Action*, 2005. *In press.*

[15] T. Ballmer and W. Brennenstuhl. *Speech Act Classification. A Study in the Lexical Classification of English Speech Activity Verbs.* Springer Verlag, Berlin, 1981.

[16] J. Barwise and L. Moss. *Vicious Circles.* Number 60 in CSLI Lecture Notes. CSLI Publications, Stanford, CA, 1996.

[17] T. J. M. Bench-Capon. Specifying the interaction between information sources. In G. Quirchmayr, E. Schweighofer, and T. J. M. Bench-Capon, editors, *DEXA*, volume 1460 of *Lecture Notes in Computer Science*, pages 425–434. Springer, 1998.

[18] F. J. Bex, H. Prakken, C. Reed, and D. N. Walton. Towards a formal account of reasoning about evidence: argumentation schemes and generalisations. *Artificial Intelligence and Law*, 12, 2004. *In press.*

[19] G. Brewka. Dynamic argument systems: a formal model of argumentation processes based on situation calculus. *Journal of Logic and Computation*, 11:257–282, 2001.

[20] M. Caminada. *For the sake of the Argument. Explorations into argument-based reasoning.* Doctoral dissertation Free University Amsterdam, 2004.

[21] M. Caminada and L. Amgoud. An axiomatic account of formal argumentation. In *Proceedings of the 20th National Conference on Artificial Intelligence, AAAI'2005*, pages 608–613, 2005.

[22] M. Caminada and L. Amgoud. On the evaluation of argumentation formalisms. In *IRIT Technical report*, 2006.

[23] C. Cayrol, S. Doutre, and J. Mengin. Dialectical Proof Theories for the Credulous Preferred Semantics of Argumentation Frameworks. In *ECSQARU 2001*, volume 2143 of *LNAI*, pages 668–679. Springer-Verlag, 2001.

[24] C. Cayrol, S. Doutre, and J. Mengin. On Decision Problems related to the preferred semantics for argumentation frameworks. *Journal of Logic and Computation*, 13(3):377–403, 2003.

[25] E. Cogan, S. Parsons, and P. McBurney. What kind of arguments are we going to have today? In F. Dignum, V. Dignum, S. Koenig, S. Kraus, M. P. Singh, and M. Wooldridge, editors, *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2005), Utrecht, The Netherlands*, pages 544–551, New York City, NY, USA, 2005. ACM Press.

[26] DEIS. *TuCSoN Guide: TuCSoN Version 1.4.0.* DEIS, Universit'a di Bologna, Bologna, Italy, document revision 003 edition, 2002. Last changes 16.12.2004.

[27] F. Dignum, B. Dunin-Kęplicz, and R. Verbrugge. Agent theory for team formation by dialogue. In C. Castelfranchi and Y. Lespérance, editors, *Seventh Workshop on Agent Theories, Architectures, and Languages*, pages 141–156, Boston, USA, 2000.

[28] F. Dignum, B. Dunin-Kęplicz, and R. Verbrugge. Creating collective intention through dialogue. In J. Cunningham and D. Gabbay, editors, *Proceedings of the International Conference on Formal and Applied Practical Reasoning*, pages 145–158, London, UK, 2000. Department of Computing, Imperial College, University of London.

[29] P. Dijkstra, F.J. Bex, H. Prakken, and C.N.J. De Vey Mestdagh. Towards a multi-agent system for regulated information exchange in crime investigations. *Artificial Intelligence and Law*, 13, 2005. *In press.*

[30] S. Doutre, P. McBurney, and M. Wooldridge. Law-governed linda as a semantics for agent interaction protocols. In F. Dignum, V. Dignum, S. Koenig, S. Kraus, M. P. Singh, and M. Wooldridge, editors, *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2005), Utrecht, The Netherlands*, pages 1257–1258, New York City, NY, USA, 2005. ACM Press.

[31] S. Doutre, P. McBurney, M. Wooldridge, and W. Barden. Information-seeking agent dialogs with permissions and arguments. Technical Report ULCS-05-010, Department of Computer Science, University of Liverpool, Liverpool, UK, 2005. Available from: `www.csc.liv.ac.uk/research/techreports/tr2005/tr05010abs.html`.

[32] S. Doutre and J. Mengin. On sceptical versus credulous acceptance for abstract argument systems. In *Ninth European Conference on Logics in Artificial Intelligence (JELIA 2004)*, pages 462–473, September 2004.

[33] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77:321–357, 1995.

[34] P. E. Dunne and T. J. M. Bench-Capon. Two party immediate response dispute: Properties and efficiency. *Artificial Intelligence*, 149:221–250, 2003.

[35] E. Freeman, S. Hupfer, and K. Arnold. *JavaSpaces Principles, Patterns and Practice*. Addison-Wesley, USA, 1999.

[36] D. M. Gabbay and J. Woods. More on non-cooperation in Dialogue Logic. *Logic Journal of the IGPL*, 9(2):321–339, 2001.

[37] D. M. Gabbay and J. Woods. Non-cooperation in dialogue logic. *Synthese*, 127(1-2):161–186, 2001.

[38] A.J. García and G.R. Simari. Defeasible logic programming: an argumentative approach. *Theory and Practice of Logic Programming*, 4(1):95–138, 2004.

[39] D. Gelernter. Generative communication in Linda. *ACM Transactions on Programming Languages and Systems*, 7(1):80–112, 1985.

[40] R. Girle. Commands in Dialogue Logic. In D. M. Gabbay and H. J. Ohlbach, editors, *Practical Reasoning: Proceedings of the First International Conference on Formal and Applied Practical Reasoning (FAPR 1996), Bonn, Germany*, Lecture Notes in Artificial Intelligence 1085, pages 246–260, Berlin, Germany, 1996. Springer.

[41] T. F. Gordon. *The Pleadings Game. An Artificial Intelligence Model of Procedural Justice.* Kluwer Academic Publishers, Dordrecht/Boston/London, 1995.

[42] G. Governatori, M.J. Maher, G. Antoniou, and D. Billington. Argumentation semantics for defeasible logic. *Journal of Logic and Computation*, 14(5):675–702, 2004.

[43] D. Harel. Dynamic logic. In D. M. Gabbay and F. Guenther, editors, *Handbook of Philosophical Logic, Volume II: Extensions of Classical Logic*, pages 497–604. D. Reidel, Dordrecht, The Netherlands, 1984.

[44] H. Jakobovits and D. Vermeir. Dialectic semantics for argumentation frameworks. In *Proceedings of the Seventh International Conference on Artificial Intelligence and Law (ICAIL-99)*, pages 53–62, New York, 1999. ACM Press.

[45] P. Kirschner, S. Buckingham Shum, and C. Carr, editors. *Visualizing Argumentation: Software Tools for Collaborative and Educational Sense-Making*. Springer-Verlag: London, 2003.

[46] R. Kowalski and M. Sergot. A logic-based calculus of events. *New Generation Computing*, 4(1):67–96, 1986.

[47] S. Kraus, K. Sycara, and A. Evenchik. *Reaching agreements through argumentation: a logical model and implementation*, volume 104. Journal of Artificial Intelligence, 1998.

[48] R.P. Loui. Process and policy: resource-bounded non-demonstrative reasoning. *Computational Intelligence*, 14:1–38, 1998.

[49] M. Luck, P. McBurney, O. Shehory, and S. Willmott. *Agent Technology: Computing as Interaction. A Roadmap for Agent Based Computing.* AgentLink III, the European Co-ordination Action for Agent-Based Computing, Southampton, UK, 2005.

[50] J. D. MacKenzie. Question-begging in non-cumulative systems. *Journal of Philosophical Logic*, 8:117–133, 1979.

[51] P. McBurney, R. M. van Eijk, S. Parsons, and L. Amgoud. A dialogue-game protocol for agent purchase negotiations. *Journal of Autonomous Agents and Multi-Agent Systems*, 7(3):235–273, 2003.

[52] P. McBurney, D. Hitchcock, and S. Parsons. The eightfold way of deliberation dialogue. *International Journal of Intelligent Systems*, 2005. *In press.*

[53] P. McBurney and S. Parsons. Representing epistemic uncertainty by means of dialectical argumentation. *Annals of Mathematics and Artificial Intelligence*, 32(1–4):125–169, 2001.

[54] P. McBurney and S. Parsons. Games that agents play: A formal framework for dialogues between autonomous agents. *Journal of Logic, Language and Information*, 13:315–343, 2002.

[55] R. McConachy and I. Zukerman. Dialogue requirements for argumentation systems. In *Proceedings of IJCAI'99 Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, 1999.

[56] A. Omicini and E. Denti. From tuple spaces to tuple centres. *Science of Computer Programming*, 41(3):277–294, 2001.

[57] A. Omicini, A. Ricci, and M. Viroli. *Agens Faber:* towards a theory of artefacts for MAS. *Electronic Notes in Theoretical Computer Science*, 2006. *In press.*

[58] R. Parikh. The logic of games and its applications. *Annals of Discrete Mathematics*, 24:111–140, 1985.

[59] S. Parsons and N. R. Jennings. Negotiation through argumentation—a preliminary report. In *Proceedings of the 2nd International Conference on Multi Agent Systems*, pages 267–274, 1996.

[60] S. Parsons, C. Sierra, and N. R. Jennings. Agents that reason and negotiate by arguing. *Journal of Logic and Computation*, 8(3):261–292, 1998.

[61] S. Parsons, M. Wooldridge, and L. Amgoud. An analysis of formal inter-agent dialogues. In *Proceedings of the First International Conference on Autonomous Agents and Multiagent Systems (AAMAS-02)*, pages 394–401, 2002.

[62] S. Parsons, M. Wooldridge, and L. Amgoud. Properties and complexity of formal inter-agent dialogues. *Journal of Logic and Computation*, 13(3):347–376, 2003.

[63] M. Pechoucek, D. Steiner, and S. Thompson, editors. *Proceedings of the Industry Track of the Fourth International Joint Conference on Autonomous Agents and Multi Agent Systems*, New York City, NY, USA, 2005. ACM Press. Universiteit Utrecht, The Netherlands, 25–29 July 2005.

[64] J. L. Pollock. Defeasible reasoning. *Cognitive Science*, 11:481–518, 1987.

[65] J. L. Pollock. *Cognitive Carpentry. A Blueprint for How to Build a Person.* MIT Press, Cambridge, MA, 1995.

[66] H. Prakken. On dialogue systems with speech acts, arguments, and counterarguments. In *Proceedings of the 7th European Workshop on Logic for Artificial Intelligence (JELIA-00)*, number 1919 in Springer Lecture Notes in Artificial Intelligence, pages 224–238, Berlin, 2000. Springer Verlag.

[67] H. Prakken. Relating protocols for dynamic dispute with logics for defeasible argumentation. *Synthese*, 127:187–219, 2001.

[68] H. Prakken and G. Sartor. Argument-based extended logic programming with defeasible priorities. *Journal of Applied Non-Classical Logics*, 7:25–75, 1997.

[69] H. Prakken and G.A.W. Vreeswijk. Logics for defeasible argumentation. In D. Gabbay and F. Günthner, editors, *Handbook of Philosophical Logic*, volume 4, pages 219–318. Kluwer Academic Publishers, Dordrecht/Boston/London, second edition, 2002.

[70] C. Reed. Dialogue frames in agent communication. In *Proceedings of the 3rd International Conference on Multi Agent Systems*, pages 246–253, 1998.

[71] C. Reed and T. J. Norman, editors. *Argumentation Machines: New Frontiers in Argument and Computation.* Kluwer Academic Publishers, 2004.

[72] N. Rescher. *Dialectics: A Controversy-Oriented Approach to the Theory of Knowledge.* State University of New York Press, Albany, NY, 1977.

[73] F. Sadri, F. Toni, and P. Torroni. Dialogues for negotiation: Agent varieties and dialogue sequences. In *Proceedings of the International Workshop on Agent Theories, Architectures and Languages, ATAL*, 2001.

[74] F. Sadri, F. Toni, and P. Torroni. Logic agents, dialogues and negotiation: an abductive approach. In M. Schroeder and K. Stathis, editors, *Proceedings of the Symposium on Information Agents for E-Commerce, Artificial Intelligence and the Simulation of Behaviour Conference (AISB-2001)*, York, UK, 2001. AISB.

[75] M. Shanahan. The event calculus explained. *Lecture Notes in Computer Science*, 1600:409–430, 1999.

[76] Y. Tang and S. Parsons. Argumentation-based dialogues for deliberation. In F. Dignum, V. Dignum, S. Koenig, S. Kraus, M. P. Singh, and M. Wooldridge, editors, *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2005), Utrecht, The Netherlands*, pages 552–559, New York City, NY, USA, 2005. ACM Press.

[77] F. Tohmé. Negotiation and defeasible reasons for choice. In *Proceedings of the Stanford Spring Symposium on Qualitative Preferences in Deliberation and Practical Reasoning*, pages 95–102, 1997.

[78] M. Viroli and A. Ricci. Instructions-based semantics of agent-mediated interaction. In N. R. Jennings, C. Sierra, E. Sonenberg, and M. Tambe, editors, *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2004), New York City, NY, USA*, pages 102–109, New York City, NY, USA, 2004. ACM Press.

[79] G. A. W. Vreeswijk and H. Prakken. Credulous and sceptical argument games for preferred semantics. In *Proceedings of the 7th European Workshop on Logic for Artificial Intelligence (JELIA-00)*, number 1919 in Springer Lecture Notes in AI, pages 239–253, Berlin, 2000. Springer Verlag.

[80] D. N. Walton. *Argument Schemes for Presumptive Reasoning*. Lawrence Erlbaum Associates, Mahwah, NJ, USA, 1996.

[81] D. N. Walton and E. C. W. Krabbe. *Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning*. SUNY Series in Logic and Language. State University of New York Press, Albany, NY, USA, 1995.

[82] M. J. Wooldridge. *Introduction to Multiagent Systems*. John Wiley and Sons, New York, NY, USA, 2002.

[83] M. J. Wooldridge and S. Parsons. Languages for negotiation. In W. Horn, editor, *Proceedings of the Fourteenth European Conference on Artificial Intelligence (ECAI 2000)*, pages 393–397, Berlin, Germany, 2000. IOS Press.

[84] S. Zabala, I. Lara, and H. Geffner. Beliefs, reasons and moves in a model for argumentation dialogues. In *Proceedings of the Latino-American Conference on Computer Science*, 1999.