Imperial College London
Department of Computing

Negotiating Socially Optimal Allocations of Resources with Argumentation

Adil Hussain

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy in Computing of Imperial College London and
the Diploma of Imperial College, August 2010

Abstract

The resource allocation problem of multi-agent systems is the problem of deciding how to allocate resources, controlled by agents, to agents within a given system. Agents typically have preferences over alternative allocations of resources. These preferences may be derived from the agents' goals, which can be fulfilled by different plans (sets of resources). The problem arises because agents may not be able to fulfil their goals without being re-allocated resources controlled by other agents and agents may have conflicting preferences over allocations. Examples of the resource allocation problem include electronic commerce (where resources are commodities equipped with prices), the grid (where resources are computational entities equipped with computational power), and scheduling and timetabling (where resources may be tasks with costs).

The focus in this thesis is distributed decision-making amongst agents, whereby agents actively participate in computing re-allocations, starting from initial allocations which may or may not fulfil their goals. A re-allocation is arrived at by means of local negotiation steps wherein resources change hands between the agents involved in the negotiations. The negotiation method of choice in this thesis is argumentation-based negotiation supported by assumption-based argumentation. This method allows agents to work towards their goals despite incomplete information regarding the goals of and resources allocated to other agents, to share knowledge, thereby eliminating unknowns, and to resolve conflicts within themselves and between one another which may arise because of inconsistent information.

Solutions generated by a resource allocation mechanism may be ranked according to how they affect the individual welfare of the agents as well as the overall social welfare of the agent society, according to different notions of social welfare borrowed from economics. The argumentation-based negotiation mechanism we propose guarantees, for the problem domain of interest in this thesis, that negotiations between agents always terminate converging to a solution. Moreover, the mechanism guarantees that solutions reached optimise the welfare of the individual agents as well as the agent society as a whole according to Pareto optimal and utilitarian notions of social welfare.

Acknowledgements

I would like to express my heartfelt gratitude to my supervisor Dr Francesca Toni, Reader in Computational Logic in the Department of Computing at Imperial College London, for her support throughout this doctorate, and for everything she has taught me along the way. Thereafter I would like to express my thanks to everyone who I have been fortunate to exchange discussion with over the past four years. It has been a pleasure.

Dedication

In appreciation of my loving parents and family who I am most blessed to have in my life and without whom I would be incomplete :)>

‘In the Name of God, Most Gracious, Most Merciful.’

Contents

Acknowledgements **iii**

1 Introduction **1**

1.1 Multi-agent Systems for Resource Allocation 1

1.2 Motivation and Objectives 3

1.3 Structure of Thesis 5

1.4 Contributions 6

1.5 Statement of Originality 8

1.6 Publications 8

2 Background and Preliminaries **9**

2.1 Resource Reallocation Problem 9

2.1.1 Resources 10

2.1.2 Allocations 11

2.1.3 Agent Preferences 12

2.1.4 Social Welfare 13

2.1.5 Centralised versus Distributed Allocation Procedures 16

2.1.6 Deals 17

2.2 Dialogues in Multi-agent Systems 21

2.2.1 Agent Communication Languages 24

2.2.2 Negotiation 28

2.2.3 Logic-based Negotiation 31

2.3 Argumentation-Based Negotiation 34

2.3.1 Introduction to Argumentation 34

2.3.2 Argumentation for Negotiation 37

2.4 Computational Argumentation 39

2.4.1 Dung's Abstract Argumentation Framework 40

2.4.2 Assumption-Based Argumentation 42

2.5 Summary 49

3 Agent Systems for the Resource Reallocation Problem 50

3.1 Agent System 51

3.2 Example Application Domain 54

3.3 Solving the Resource Reallocation Problem 56

3.4 Individual and Social Welfare 58

3.5 Discussion 63

3.6 Conclusion 64

4 Minimally Cooperative Rational Deals for the Resource Reallocation Problem 65

4.1 Deals and Cooperative Rationality 66

4.2 Minimally Cooperative Rational Deals 71

4.3 Maximising Utilitarian Social Welfare . . . 73
4.4 Properties of Min CR deals . . . 75
4.5 Completeness of Min CR deals . . . 83
4.6 Discussion . . . 87
4.7 Conclusion . . . 88

5 Dialogues for Solving the Resource Reallocation Problem 89

5.1 Introduction . . . 89
5.2 Communication Language and Protocols for Dialogues . . . 91
5.3 Communication Rules . . . 100
5.4 Procedures . . . 109
5.5 Examples of Generated Dialogue Instances . . . 118
5.6 Properties of Generated Dialogue Instances . . . 122
5.7 Discussion . . . 127
5.8 Conclusion . . . 128

6 Assumption-Based Argumentation for Communicating Agents 130

6.1 Introduction . . . 130
6.2 Agent ABA Framework . . . 132
6.2.1 Example — Exchanging appointments between agents . . . 133
6.3 Fact-supported Arguments . . . 135
6.4 Computation of Admissible Arguments . . . 139
6.4.1 Example — Arguments in Dialogue . . . 144
6.5 Theoretical Results for AABA . . . 146

6.6 Discussion . . . 149

6.7 Conclusion . . . 150

7 AABA Negotiation Policy for Socially Optimal Resource Allocation 151

7.1 Introduction . . . 152

7.2 Resource Reallocation Agents as AABA frameworks . . . 153

7.3 Consistency Properties of an AABA agent . . . 163

7.4 Procedures defined using AABA . . . 166

7.5 Example of Dialogue Instances Generated by the AABA Agents . . . 173

7.6 Discussion . . . 176

7.7 Conclusion . . . 179

8 Conclusion 180

8.1 Summary of Thesis Achievements . . . 180

8.2 Future Work . . . 181

8.3 Final remarks . . . 183

A Additional properties of the CR and Min CR deals of Chapter 4 185

A.1 Number of resources agents receive in Min CR deals . . . 185

A.2 Number of resources agents give away in Min CR deals . . . 186

A.3 Changes in utility of agents in Min CR deals . . . 187

B Example Definition of the Control Procedures of Chapter 5 190

C Conditions for Inferences of Chapter 5 196

D Proofs of Lemmas and Theorems pertaining to AABA (in Chapter 6) 199

D.1 Proofs pertaining to backward deductions of AABA 199

D.1.1 Proof of Lemma 6.1 199

D.1.2 Proof of Lemma 6.2 200

D.2 Proofs pertaining to fact-inclusive AB-dispute derivations of AABA 203

D.2.1 Proof of Theorem 6.1 203

D.2.2 Proof of Theorem 6.2 205

E Definition of the Control Procedures of Chapter 5 by means of AABA 209

E.1 Dialogical Status as AABA framework 209

E.2 Control Procedures 212

Bibliography 214

List of Tables

2.1 Types of Dialogue 23

6.1 Backward deduction for $l(x, app2)$ according to $\mathcal{AF}_x$ where $\mathcal{AF}_x$ is as defined in Section 6.2.1. The underlined atoms are picked by the selection function. 138

6.2 Fdd for $sa(x, y, app1, app2)$ according to $\mathcal{AF}_x$ where $\mathcal{AF}_x$ is as defined in Section 6.2.1. $S = \{h(x, app1),\ r(x, seeDrAli),\ fr(app2, seeDrAli),\ \neg l(x, app1),\ cs(y, app2, app1)\}$. The underlined atoms are picked by f. 143

6.3 Attempt to build a fdd for $sa(x, y, app1, app2)$ according to $\mathcal{AF}_y$ where $\mathcal{AF}_y$ is as defined in Section 6.2.1. S is as defined in Table 6.2. $S' = \{h(y, app2), r(y, fridayApp), fr(app2, fridayApp), \neg l(y, app1)\}$. The underlined elements are picked by f and g. 144

6.4 Arguments exchanged between x and y regarding the acceptability of $sa(x, y, app1, app2)$, and respective changes to each agent's personal facts. S and S' are as defined in Tables 6.2 and 6.3 (respectively). $S'' = \{r(y, fridayApp),\ fr(app1, fridayApp)\}$. $S''' = S \cup S''$. 146

List of Figures

3.1 Plans Pl in Example 3.1. 54

3.2 Plans in Example 3.3. 61

3.3 Plans in Example 3.4. 63

4.1 Plans in Example 4.3 (and as in Example 3.4). 72

4.2 Plans in Examples 4.4–4.6 (and as in Example 3.1). 79

5.1 Request Dialogue Instance as Finite State Automaton 94

5.2 Proposal Dialogue Instance as Finite State Automaton 95

5.3 Plans Pl in Example 5.5 (and as in Example 3.1). 120

6.1 A f-argument for $l(x, app2)$ with support $\{r(x, seeDrAli),\ fr(app2, seeDrAli)\}$ wrt $\mathcal{AF}_x$ as defined in Section 6.2.1. 136

7.1 Plans Pl in Example 7.10. 173

Chapter 1

Introduction

1.1 Multi-agent Systems for Resource Allocation

Central to this thesis is the area of multi-agent systems, one of the most vibrant and important areas of research and development to have emerged in information technology in recent years. As the computing landscape moves from a focus on the individual standalone computer system to a situation in which the real power of computers is realised through distributed, open and dynamic systems, we are faced with new technological challenges and new opportunities [LMSW05]. The multi-agent paradigm for software development offers a new and often more appropriate route to the development of complex systems [Jen01], especially in open and dynamic environments [LMSW05, LM08].

In agent-based systems, the general idea is that decision-making is delegated to programs which are (in some sense) intelligent. Multi-agent systems in particular can be understood as consisting of two closely interwoven strands of work [Woo09]. The first is concerned with *individual* agents, while the second is concerned with *collections* of these agents. An individual agent can be seen as an autonomous computational entity, situated within an environment, that is capable of both perceiving and actuating changes to that environment in order to meet its delegated objectives [Woo09, Woo00]. Whilst there is no universally accepted definition of the term "agent", an *intelligent* agent can be expected to contain the following capabilities [WJ95]:

- Autonomy: Intelligent agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state.
- Social ability: Intelligent agents are capable of interacting with other agents (and possibly humans) in order to satisfy their design objectives.
- Reactivity: Intelligent agents are able to perceive their environment, and respond in a timely fashion to changes that occur in it in order to satisfy their design objectives.
- Proactiveness: Intelligent agents are able to exhibit goal-directed behaviour by *taking the initiative* in order to satisfy their design objectives.

A multi-agent system then is a system composed of multiple agents, which *interact* with users, customers, system resources and each other. In the most general case, the agents in a multi-agent system represent or act on behalf of users or owners with very different goals and motivations. In order to successfully interact, agents are required to *cooperate*, *coordinate*, *negotiate* and so on with each other [Woo09, LMSW05]. This interaction typically occurs by exchanging messages through some computer network infrastructure [Woo09]. The above classification of an agent is of course not the only classification but does capture somewhat the general idea. Some further discussion regarding other possible definitions and capabilities of an agent can be found in [FG97, Cas97].

Our area of interest in this work are multi-agent systems as mechanisms for resource allocation where the notion of a resource is general but resources would typically be scarce and desired by more than one agent. A well-known type of resource allocation mechanism are auctions, which have achieved particular prominence in computer science [CSS10, Kri09]. Our focus in this thesis is not resource allocation where the allocation of resources is computed (decided) by means of a central controller agent as in an auction but, rather, computing allocations of resources by means of the participating agents themselves, i.e. the agents which have preferences over alternative allocations of resources and perhaps even control some of the resources to be (re-)allocated. Such a distributed approach is essential in cases where agents do not wish to disclose their preferences to a central controller, or where a central controller does not

exist. The distributed computation which we consider could be seen as arising by means of agent interactions (cooperation, coordination or negotiation as the case may be). Indeed such multi-agent interactions are an important area of research in artificial intelligence and computer science [RZ94, Kra01, San99] and are expected to become a key part of next generation electronic commerce and supply chain systems, as well as other application areas [LMSW05].

In reaching agreement regarding the allocation of scarce resources in the presence of conflicting goals and preferences, a multi-agent system can be thought of as an artificial society of autonomous software agents to which concepts borrowed from welfare economics and social choice theory can be applied to assess the social welfare of the multi-agent system understood as an agent society [EMST06, CDE+06]. Our interest then is for agents to arrive at allocations of resources, possibly by means of a number of local 'negotiation' steps (involving proposals and acceptances of proposals being exchanged), such that the agents themselves benefit and the agent society (the collection of agents that make up the multi-agent system) as a whole benefits. Moreover, our interest is with agents that make use of argumentation, allowing participants to state justficiation for their negotiation stances, and not just simply propose, accept and reject proposals. The main distinguishing feature of such an approach is that it allows for more sophisticated forms of interaction than otherwise possible by means of, for example, game theoretic [RZ94, Kra01] and heuristic [Far00, FWJ02] approaches [RRJ+03].

1.2 Motivation and Objectives

The overall aim of this thesis is to present a generative framework wherein agents can reason and interact using argumentation, in order to reach agreements for actions to be performed. The actions of interest to us are exchanges of resources aimed at solving resource reallocation problems in the context of multi-agent systems where it is difficult or impossible to organise a central decision-making "agent" which has full knowledge of the agents and resources involved. Examples of problems which are appropriate include the following:

- Grid computing: agencies, such as research laboratories and universities which have com-

putational power that is at times insufficient and at times greater than the amount of processing required for their respective workloads, interact to share the resources (computers etc) under their control to better fulfil their individual goals than they would be able to otherwise and in a manner that the disclosure of information and decision-making is left up to the agents themselves [MBP05, CGG+06, Ton07, TGK+08, JDC08];

- Scheduling and timetabling: in dynamic environments such as air traffic control and task/appointment allocation where resources are allocated to individuals as they enter the system or their individual circumstances change, it can often be worthwhile or necessary for agents to refine their allocated share taking these changes into account. Interaction (co-operation, negotiation) is the most significant characteristic in such systems and, as the number of agents increases, delegation of interaction and decision-making to the agents themselves, which operate autonomously and represent their best interests (goals), releases the burden and complexity required of a single central decision-making agent [MHVB00, BHS03, AF03, WVI04, Hal07, GIRS09];

- Disaster response: agencies, such as governmental and non-governmental organisations which operate independently in their response to and alleviation of disaster, communicating and exchanging resources under their control as necessary to better fulfil their individual goals and to better overcome the overall disaster at hand [FB07, KT01, RFM+09].

We present our generative argumentation-based negotiation framework for resource allocation by bridging the gap between three different areas of multi-agent systems research, namely: resource allocation [EMST06, CDE+06], argumentation-based negotiation [JFL+01, RRJ+03] and assumption-based argumentation [DKT09]. We briefly mention here the scope of existing work in each subject area in turn and discuss how each connects and fills the gaps in the others (in this thesis). Firstly, let us consider the resource allocation problem. We find that existing work [EMST06] identifies the kinds of deals (exchanges of resources) required for different classes of resource allocation problems, where agents specify utilities over resources and the task of the resource allocation mechanism is to optimise some notion of social welfare [HP02, CDE+06]. We find, however, that such work does not specify exactly how such

deals can be found by the agents themselves. This is from where we pick up this work and this takes us to our second area of interest, argumentation-based negotiation. Argumentation-based negotiation offers much promise and advantages over standard negotiation for allowing agents to arrive at decisions, or, more specifically, deals with one another [RRJ+03]. Much work exists discussing how allowing agents to argue as part of the negotiation process can not only increase the speed at which decisions are reached, but also improve and expand upon the decisions that can be reached [RPSD07, RSD03, SJNP97, PSJ98]. We find, however, that the frameworks that exist are not fully specified, and it is not entirely obvious how the agents' reasoning and decision making models could be concretely specified to generate moves during the negotiation process, and to receive and respond to moves by other agents. This takes us to our third area of interest, assumption-based argumentation, which is a general-purpose argumentation framework for representing the knowledge and reasoning of agents, as well as for allowing agents to decide what beliefs and decisions are acceptable, according to various acceptability criteria [DKT06, DMT07]. Our motivation for this thesis then was to bring these three areas together to present concretely how agents could negotiate socially optimal allocations of resources supported by argumentation, by means of a concrete but general-purpose argumentation framework such as assumption-based argumentation.

1.3 Structure of Thesis

In Chapter 2 we present background on the resource reallocation problem of interest to us in this thesis, as well as the concepts of social welfare, agent communication (negotiation dialogues) and computational argumentation which we consider.

In Chapter 3 we define the class of resource reallocation problem to be solved in this thesis. Moreover, we define the key components of our problem domain, including the notions of agents and agent systems which we consider in this thesis, as well as the notions of individual and social welfare.

In Chapter 4 we define the kinds of deals necessary and sufficient for reaching a socially optimal

allocation of resources for the particular class of resource reallocation problem as defined in Chapter 3. We prove that any sequence of such deals leads to an optimal allocation.

In Chapter 5 we present a communication language and protocol for argumentation-based negotiation, which allows agents to negotiate deals (exchanges of resources) and share arguments in the process. We present also a means of defining agent policies/strategies that allow agents to generate moves conforming to the protocol, and we discuss how such an agent policy would need to be defined to negotiate the kinds of deals described in Chapter 4 for solving the resource reallocation problem of Chapter 3. A concretely defined agent policy is such that an agent can decide what move to make in any situation it finds itself in, as well as knowing what to do with what it is told in terms of updating its internal state (beliefs etc) and responding accordingly.

In Chapter 6 we extend the Assumption-Based Argumentation (ABA) framework for multi-agent systems, so that agents can build acceptable arguments according to a particular notion of acceptabilty, but which are also communicable and can be understood by other agents in the agent system. We term our new framework "Agent ABA" (AABA).

In Chapter 7 we instantiate the agent policy described in Chapter 5 for solving the resource reallocation problem of Chapter 3 by means of AABA defined in Chapter 6. In particular, AABA realises the reasoning of agents, allowing each agent to deal with inconsistency and incompleteness in its beliefs, and in deciding what actions to undertake.

Lastly, in Chapter 8 we summarise achievements of the thesis and conclude discussing related and future work.

1.4 Contributions

The contributions of the thesis are as follows:

- Presentation of a means of capturing resource reallocation problems in terms of the agents, goals and resources involved (Chapter 3);

- Definition of a resource exchange (deal) between agents, termed "minimally cooperatively rational deals", sufficient for solving the class of resource reallocation problems of interest in this thesis (Chapter 4). Given a resource reallocation problem (the agents, resources and goals involved), these deals determine the smallest incremental steps required for reaching an optimal solution. Although neutral to a centralised or distributed approach and also any mechanism for generating the deals, the "minimal" nature of these deals lends itself to a distributed solution where the successive deals required could be captured by the relevant agents;

- Definition of an (argumentation-based) negotiation protocol which allows for multiple agents to agree (or not) to exchange any number of resources (Chapter 5). These protocols are independent of the particular class of resource reallocation problems of interest to us in this thesis and allow for capturing exchanges of resources (and arguments) involving any number of agents and any number of resources;

- Representation of a generative argumentation-based negotiation policy for the exchange of resources between agents (Chapter 5). This presents a means for agents in an agent system to generate the ("minimally cooperative rational") deals required for solving the resource reallocation problem defined according to the agents, resources and goals involved;

- Extension of the Assumption-Based Argumentation (ABA) framework to Agent ABA (AABA), for multi-agent system settings where agents need to communicate arguments between one another (Chapter 6). This is useful in the context of negotiation settings in particular, as is the focus in this thesis, where agents are required to generate and exchange arguments as part of the negotiation process;

- Instantiation of the argumentation-based negotiation policy of Chapter 5 by means of AABA (Chapter 7). The instantiation demonstrates how a general-purpose argumentation framework can be used to complete our negotiation policy allowing agents to reason internally in the absence of information and to share their reasoning in the form of arguments in generating exchanges of resources.

1.5 Statement of Originality

I declare that this thesis was composed by myself and that the work it presents is my own, except where otherwise stated.

1.6 Publications

The work in this thesis brings together and builds upon work published as follows:

- The representation of an agent's negotiation policy for generating dialogues by means of Communication Rules (Action Rules) in Chapter 5 is as published in 'Bilateral Agent Negotiation with Information-Seeking' at The Fifth European Workshop on Multi-Agent Systems, 2007 [HT07]. There we defined a negotiation policy for a simpler class of resource allocation problem than is addressed in this thesis and using instead information-seeking dialogues for the exchange of information between agents and not arguments as part of the negotiation dialogues as in this thesis.

- The definitions of resource reallocation problem and agent system in Chapter 3 as well as the definitions of dialogue instances and dialogue sequences in Chapter 5 build upon those published in 'On the Benefits of Argumentation for Negotiation' at The Sixth European Workshop on Multi-Agent Systems, 2008 [HT08]. Moreover, there we discussed the use of a general-purpose argumentation framework such as Assumption-Based Argumentation for realising an agent's negotiation policy and how this could be achieved. This idea is developed and built upon in Chapter 7.

- The extension of Assumption-Based Argumentation for multi-agent system settings in Chapter 6 is as published in 'Assumption-Based Argumentation for Communicating Agents' at The Uses of Computational Argumentation Symposia, as part of the AAAI 2009 Fall Symposium Series [HT09].

Chapter 2

Background and Preliminaries

This chapter introduces background knowledge on the key subject areas underpinning this thesis. Section 2.1 introduces the resource reallocation problem in the context of multi-agent systems, as well the concepts of social welfare and deals. Subsequently, the issues involved with agent communication and interaction for solving multi-agent problems such as the resource reallocation problem by means of agent dialogues (negotiation) is discussed in Section 2.2. Section 2.3 provides a general introduction to argumentation as well as discussing argumentation-based negotiation, the type of dialogue we focus on in this thesis, and its usefulness for distributed decision-making. Following on from this, Section 2.4 introduces an abstract theory of computational argumentation and then, in particular, assumption-based argumentation, a general-purpose argumentation framework for representing the reasoning of agents in logic, which we adopt and extend in this thesis.

2.1 Resource Reallocation Problem

Resource allocation in agent societies is the problem of deciding how to distribute a number of resources amongst a number of agents. The agents are autonomous entities requiring (requesting) and receiving the resources. Both resources and agents vary in nature. This is a major factor which complicates the allocation problem. Thus, the properties of both resources

and agents must be determined before the problem can be solved. Resources may be shareable, and/or used up with time, and/or divisible. These properties as well as others need to be determined. Agents, on the other hand, may have preference of some resources over others, they may not want to be given some resources, or they may require multiple resources simultaneously.

Given the properties of the resources and agents, it may well be that agents are competing for the very same (limited) resources. Thus, it may not be possible for every agent to receive the set of resources that it desires most, or worse yet, it may be that some agents do not receive any resources at all. The problem then, given an agent system, is to determine which of the agents receive which of the resources, possibly taking into account the resources that agents have initially and that agents are autonomous and rational.

Before explaining the kinds of transitions from initial allocation of resources to new allocation required for solving an allocation problem, another important factor needs to be introduced: social welfare. The definition of social welfare can vary and, again, it depends on the properties of the resources and agents. An example of a social welfare consideration for an allocation may be to guarantee that all agents have at least a certain level of "happiness" defined in terms of the resources allocated to them, or for at least one agent to receive its most-desired set of resources. We discuss each of the issues touched upon above in greater detail in this section.

2.1.1 Resources

These are the items being distributed. The nature of resources is a central parameter in any resource allocation problem and resources can vary. Some properties, as characterised by Chevaleyre et al. in [CDE$^+$06], are as follows:

- *Divisible* or *indivisible*: Whether an agent can receive a fraction of a resource. For example, a fruit which can be split and its consituent parts allocated to different agents is said to be divisible, whereas a (piece of a) fruit which can only be allocated wholly is said to be indivisible;

- *Sharable* or *non-sharable*: Whether the resource can be allocated to a number of different agents at the same time. For example, it may be possible (or not) that multiple patients (the agents in this case) who are family members may make a joint appointment to see a doctor in which case appointments (the resources in this case) are sharable (or non-sharable respectively);

- *Static* or not: A static resource is one whose properties do not change during the allocation process. On the flip side are resources which, for instance, are *consumable* and can be used up by the agent holding it (e.g. fuel), or *perishable* and lose their value over time (e.g. food);

- *Single-unit* or *Multi-unit*: In a multi-unit setting it is possible to have many resources of the same type and to refer to these resources using the same name. In a single-unit setting, every resource to be allocated is distinguishable from the other resources and has a unique name. For example, if there are a number of bottles of drink to be allocated, but agents cannot distinguish between the bottles, then the bottles are multi-unit. In a single-unit setting, on the other hand, every bottle would be distinguishable from the others and would have a unique name.

More details on the nature of resources can be found in [CDE+06].

In this thesis, the focus is on indivisible, non-sharable, static, single-unit resources.

An example of the kinds of resources we consider in this thesis are rooms or appointments to be allocated to booking or patient agents as part of a scheduling system. Indeed in such a setting each time slot would be uniquely distinguishable, allocated wholly to a single agent and its properties would not change over time.

2.1.2 Allocations

Given a set of agents $\mathcal{A}$ and a set of resources $\mathcal{R}$, an allocation, as defined similarly by Endriss et al. in [EMST06], is a partitioning of $\mathcal{R}$ amongst the agents in $\mathcal{A}$, as follows:

Definition 2.1 (Allocations of resources [EMST06]) *An* allocation of resources P *is a function from* $\mathcal{A}$ *to subsets of* $\mathcal{R}$ *such that* $P(i) \cap P(j) = \emptyset$ *for* $i, j \in \mathcal{A}$, $i \neq j$ *and* $\bigcup_{i \in \mathcal{A}} P(i) = \mathcal{R}$. *For agent* $i \in \mathcal{A}$, $P(i)$ *is the bundle of resources held by agent* i *in allocation* P.

An allocation then is a distribution of resources such that a (possibly empty) *bundle* of resources is assigned to each agent. The above definition makes sense in particular in the case where resources are assumed to be non-sharable and indivisible, as in this thesis.

In this thesis, an allocation is such that each and every resource is allocated to one and only one agent, and agents only receive resources that exist (are available for distribution) in the system.

2.1.3 Agent Preferences

Agents have preferences over alternative allocations of resources, and, in particular, the bundle of resources they receive in these allocations. This preference expresses the relative or absolute satisfaction of an agent when faced with a choice between alternatives. A preference structure represents an agent's preferences over a set of alternatives, and this preference structure can take various forms (as further discussed by Chevaleyre et al. in [CDE+06]):

- *Cardinal*: Consists of a utility function where a value is assigned to alternative allocations (or bundles);
- *Ordinal*: Consists of a binary relation on alternative allocations (or bundles);
- *Binary*: Consists of a partition of alternative allocations (or bundles) into a set of good and a set of bad allocations (bundles respectively);
- *Fuzzy*: Consists of a fuzzy relation such that, given two alternatives x and y, the fuzzy relation is the degree (between 0 and 1 inclusive) to which x is preferred over y.

Ordinal preferences allow only for comparing the satisfaction of a given agent for different alternatives, but cannot express preference intensity and do not allow for interpersonal comparison

of preferences (e.g. "agent i is happier with x than agent j with y"). Any cardinal preference induces an ordinal preference, since an allocation or bundle x is ordinally preferred to y if a better utility value is assigned to x than y. Fuzzy relations, on the other hand, are more general than both ordinal and cardinal preferences. A disadvantage of cardinal and fuzzy preferences is that it can be difficult to accurately specify utility values or the degree of preference respectively to different alternatives, and alternatives can be exponentially many. Lastly, a binary preference structure is a special case of both cardinal and ordinal preferences since it is possible to translate cardinal or ordinal preferences into a partioning of good and bad alternatives. Despite the apparent simplicity of a binary preference structure, this is particularly interesting in problems where agents have goals with multiple plans, as is the focus in this thesis.

In this thesis, we will adopt a cardinal utility function $u_i : 2^{\mathcal{R}} \to \{0, 1\}$ where $i \in \mathcal{A}$.

We can translate the cardinal utility function u_i as above to an ordinal preference relation $\preceq_i$ and write $x \preceq_i y$ iff $u_i(x) \leq u_i(y)$ $(x, y \in 2^{\mathcal{R}})$. We say that u_i as above is *dichotomous* in the sense that utility values are either 0 or 1. Moreover, we can translate u_i to a binary partioning, for example, by splitting bundles into a set of bad and good bundles depending on their utility value — 0 and 1 respectively. Lastly:

In this thesis, agents assign utility to an allocation P solely according to the bundle they receive in P, i.e. $u_i(P) = u_i(P(i))$ for all $i \in \mathcal{A}$.

As a consequence of the above choice, an agent i prefers an allocation P over an allocation Q iff i prefers the resources it receives in P over the resources it receives in Q, i.e. $Q \preceq_i P$ iff $Q(i) \preceq_i P(i)$.

2.1.4 Social Welfare

Multi-agent systems can sometimes be seen as a society of agents, and allocations may be ranked according to how they affect the social welfare of the system, depending on the preferences of the individual agents in the system and according to different notions of social welfare borrowed from economics [EMST06, CDE+06]. There are various measures of social welfare

over alternative allocations of resources given some utility function u_i or preference relation $\preceq_i$ (for each $i \in \mathcal{A}$) assessing the overall quality of an allocation for the agent system. Some of these measures, as found in the works of Herreiner and Puppe [HP02] and Chevaleyre et al. [CDE+06], are as follows:

- *Pareto Optimality* [HP02, CDE+06]: An allocation P is Pareto-optimal if there does not exist an allocation Q that is unanimously preferred to P by all agents with a strict preference for at least one individual agent. Formally, there is no allocation Q such that:
  - $P \preceq_i Q$ for all agents $i \in \mathcal{A}$; and
  - $P \prec_i Q$ for at least one agent $i \in \mathcal{A}$.

  The concept of Pareto optimality is purely ordinal, and the definition does not take into account the preferences agents might have on resources allocated to other agents.

- *Utilitarian Social Welfare* [CDE+06]: The utilitarian social welfare of an allocation P is the sum of the utilities each agent assigns to P. Formally,

$$sw_u(P) = \sum_{i \in \mathcal{A}} u_i(P)$$

  This provides a meaningful metric for overall (as well as average) utility of an allocation. The optimal utilitarian social welfare value is given by the allocation that maximises the utilitarian social welfare of the agent system.

- *Nash Product* [CDE+06]: The Nash product of an allocation P is the product of the utilities each agent assigns to P. Formally,

$$sw_N(P) = \prod_{i \in \mathcal{A}} u_i(P)$$

  Note that this only provides a meaningful metric if utilities are positive (non-zero). The optimal Nash product value is given by the allocation that maximises the Nash product of the agent system.

- *Egalitarian Social Welfare* [CDE+06]: The egalitarian social welfare of an allocation P is given by the utility of the agent that assigns the worst utility to P. Formally,

$$sw_e(P) = min \{ u_i(P) \mid i \in \mathcal{A} \}$$

  This offers a level of fairness and may be suitable when the minimal needs of agents needs to be satisfied. The optimal egalitarian social welfare value is given by the allocation that maximises the egalitarian social welfare of the agent system.

- *Elitist Social Welfare* [CDE+06]: The elitist social welfare of an allocation P is given by the utility of the agent that assigns the best utility to P. Formally,

$$sw_{el}(P) = max \{ u_i(P) \mid i \in \mathcal{A} \}$$

  This is clearly not a *fair* measure for social welfare, but can be useful where it is sufficient for one agent to achieve its goals. The optimal elitist social welfare value is given by the allocation that maximises the elitist social welfare of the agent system.

- *Envy-freeness* [HP02, CDE+06]: An allocation P is envy-free if and only if each agent prefers the bundle of resources allocated to it in P over the bundles assigned to the other agents in P, i.e. for each agent $i \in \mathcal{A}$, $P(j) \preceq_i P(i)$ holds for all agents $j \in \mathcal{A}$. This measure does not require the intercomparability of the utilities of different agents (i.e. it is defined solely on the basis of agents' private preferences). A concept of optimality could be defined, for example, in terms of minimising the number of envious agents in the agent system.

Further details of the above-mentioned measures of social measure and other measures can be found in [HP02, CDE+06]. Indeed it is necessary to choose appropriate social welfare orderings for a given application of multi-agent resource allocation, for example, the elitist measure of social welfare where the interest is in achieving the objective of one agent as quickly as possible.

In this thesis the focus is on Pareto optimal and utilitarian notions of social welfare.

Once the social welfare measures of interest are decided upon, a choice is to be made as to how such an optimal allocation is to be computed. We present next the distinction between centralised and distributed approaches to allocation, and then subsequently the kinds of transitions (*deals*) that can take an agent system from one allocation to another.

2.1.5 Centralised versus Distributed Allocation Procedures

The objective of a resource allocation procedure is to find an allocation that is optimal in some sense (according to some measure of social welfare). Generally speaking, the allocation procedure used to find a suitable allocation of resources may be somewhere between a centralised or distributed approach. As outlined in the works of Herreiner and Puppe [HP02] and Chevaleyre et al. [CDE+06], central or distributed approaches are categorised as follows:

- *Centralised*: A single entity (the central host agent) decides on the final allocation of resources amongst agents, possibly after having elicited the agents' preferences over alternative allocations.
- *Distributed*: The computational burden of finding an allocation is shared amongst several agents and allocations emerge as a result of a sequence of local negotiation steps. Such local negotiation is often restricted to bilateral trading (exchanges of resources between two agents) but this is not necessary.

Both the centralised and the distributed approach to multi-agent resource allocation have their advantages and disadvantages, as follows:

- Centralised procedures are limited computationally because of the burden put on one (central) agent. As the number of agents increases with a wider number of resources, central allocation is likely to become less viable. Possibly the most important arguments in favour of the centralised approach concerns the simplicity of the communication required to implement such procedures, as well as the fact that it can deliver optimal allocations. However, agents might not want to disclose their preferences to a central controller and,

as the number of agents increase, the requirement of communication with a single central controller can lead to bottlenecks. In such scenarios a centralised approach is not viable.

- Distributed allocation of resources will almost certainly scale, since processing is shared amongst agents. A drawback of the distributed approach is that results are much less likely to be as good, but it does seem more natural in cases where finding an optimal allocation may be (computationally) infeasible, and even small improvements over an initial allocation of resources would be considered a success.

In this thesis our focus is on the distributed approach where the agents to be allocated resources actively participate in computing the allocation by means of local negotiation steps where each negotiation step involves two or more of the agents.

We discuss next the kinds of exchanges of resources (deals) that typically result from local negotiation steps in the distributed approach and then, in the subsequent section, how such deals can be identified (using agent dialogues).

2.1.6 Deals

Given a particular allocation of resources, agents may agree on a (multilateral) *deal* to exchange some of the resources they currently hold. We define below a *deal*, the agents *involved* in a deal and a *composition* of deals as in the works of Endriss et al. [EMST06] and Dunne and Chevaleyre [DC08]. In the most general case, any numbers of agents and resources could be involved in a single deal. From an abstract point of view, a deal takes us from one allocation of resources to the next:

Definition 2.2 (deals [EMST06, DC08]) *A deal is a pair $\delta = (P, Q)$ where P and Q are allocations of resources and $P \neq Q$.*

The intended interpretation of the above definition is that the deal $\delta = (P, Q)$ is only applicable in situation P and will result in situation Q. It specifies for each resource in the system whether

it is to remain with the same agent or to which agent it is to be moved to. The agents between which resources "change hands" in a deal are the agents *involved* in the deal, as follows:

Definition 2.3 (agents involved in a deal [EMST06, DC08]) *The set of agents* involved *in a deal* $\delta = (P, Q)$ *is given by* $\mathcal{A}^\delta = \{\ i \in \mathcal{A} \mid P(i) \neq Q(i)\ \}$.

The agents involved in a deal are exactly those that hold a different bundle of resources after the deal has been implemented. Moreover, two deals may be *composed* to give a single deal, as follows:

Definition 2.4 (composition of two deals [EMST06]) *The* composition *of two deals* $\delta_1 = (P, P')$ *and* $\delta_2 = (P', Q)$ *is given by* $\delta_1 \circ \delta_2 = (P, Q)$.

A deal that follows another in *sequence* can be composed with the deal or composition of deals that it follows to give a single deal, as follows:

Definition 2.5 (sequence of deals) *A sequence of* n *deals* $\delta_1 \circ \delta_2 \circ \ldots \circ \delta_n$ *is a composition of deals* δ_1 *and* δ_2 *composed with deal* δ_3 *and so on, as follows:* $(\ldots((\delta_1 \circ \delta_2) \circ \delta_3) \circ \ldots) \circ \delta_n$.

Note that a sequence of deals as above is identical to a *contract path* as defined by Dunne and Chevaleyre in [DC08] except that it is defined here in terms of *compositions* of deals. Sometimes, as for deals found in the works of Sandholm [San98] and Endriss et al. [EMST06], a deal may be accompanied by a number of monetary *side payments* to compensate some of the agents involved for accepting a loss in utility. However, in this thesis, we consider only deals where the resources are exchanged and the deals are not accompanied by monetary side payments.

An agent may or may not find a particular deal δ acceptable. As discussed by Endriss et al. in [EMST06], whether or not an agent will accept a given deal depends on the *rationality criterion* it applies when evaluating deals. We provide below a few examples of rationality criteria as similarly found in the work of Endriss et al. [EMST06]:

- *Individually rational deals*: A deal $\delta = (P, Q)$ is called individually rational iff $u_i(P) < u_i(Q)$ for all $i \in \mathcal{A}$, except possibly $u_i(P) = u_i(Q)$ for all $i \notin \mathcal{A}^\delta$. In a negotiation model based on *individual rationality* agents would only propose or accept deals (i.e. be *involved* in deals) that strictly increase their own welfare. This would make sense in the case of self-interested (selfish) agents.

- *Cooperatively rational deals*: A deal $\delta = (P, Q)$ is called cooperatively rational iff $u_i(P) \leq u_i(Q)$ for all $i \in \mathcal{A}$ and there is an agent $j \in \mathcal{A}$ such that $u_j(P) < u_j(Q)$. This rationality criterion maintains the utility of all agents whilst strictly increasing the utility of at least one agent. The agent increasing in utility could, for instance, be the agent proposing the deal in question.

- *Pigou-Dalton transfers*: A deal $\delta = (P, Q)$ is called individually rational iff it satisfies the following criteria:

  - Only two agents i and j are involved in the deal: $\mathcal{A}^\delta = \{i, j\}$.
  - The deal is mean-preserving: $u_i(P) + u_j(P) = u_i(Q) + u_j(Q)$.
  - The deal reduces inequality: $|u_i(Q) - u_j(Q)| < |u_i(P) - u_j(P)|$.

  The Pigou-Dalton *principle* states that whenever a utility transfer takes place which reduces the difference between the two, then that transfer should be considered socially beneficial [Mou88]. Indeed, as discussed by Endriss et al. in [EMST06], Pigou-Dalton transfers capture certain egalitarian principles though they not sufficient as acceptability criteria to guarantee negotiation outcomes with maximal egalitarian social welfare.

- *Equitable deals*: A deal $\delta = (P, Q)$ is called equitable iff it satisfies the following criterion:

$$min\ \{\ u_i(P) \mid i \in \mathcal{A}^\delta\ \}\ <\ min\ \{\ u_i(Q) \mid i \in \mathcal{A}^\delta\ \}$$

  Agents operating according to this criterion help any of their fellow agents that are worse off than they are themselves (as long as they can afford to do so without themselves ending up even worse). Indeed, the purpose of any exchange of resources is to improve

the welfare of the weakest agent involved in the respective deal. It is shown by Endriss et al. in [EMST06] that Pigou-Dalton transfers are equitable deals but not vice-versa.

It is necessary to choose appropriate rationality criteria, as well as interaction mechanisms (see Section 2.2), for negotiating agents in view of different notions of social welfare, for example, to achieve Pareto optimal outcomes in negotiation (by means of a sequence of deals), the agents should negotiate cooperatively rational deals. Indeed, in [EMST06], it is shown by Endriss et al. that *any* sequence of cooperatively rational deals will eventually result in a Pareto optimal allocation of resources.

The types of deals of interest in this thesis are cooperatively rational deals which are not accompanied by monetary side payments.

As discussed by Endriss et al. in [EMST06], a *class of deals* may be characterised by both *structural* constraints (number of agents and resources involved, etc.) as well as *rationality* constraints (relating to the changes in utility experienced by the agents involved) as above. In the work of Sandholm [San98], a number of structurally different types of deals (contracts) are distinguished, and shown to be sufficient and necessary for distributed task allocation where agents negotiate on a basis of *individual rationality*. The concept of individual rationality is invoked by Sandholm in [San98] on a per contract basis, i.e. a contract is individually rational to an agent if that agent is better off with the contract than without it. The classification of contract types in [San98] is as follows:

- *O-contract*: one task given by an agent i to an agent j (plus the contract price i pays to j for handling the task set).
- *C-contract*: a cluster (more than one) of tasks given by an agent i to an agent j (plus the contract price i pays to j for handling the task set).
- *S-contract*: swaps of tasks where agent i subcontracts a (single) task to agent j and vice-versa (plus the amount i pays to j and the amount j pays to i).
- *M-contract*: A multi-agent contract involving at least three agents, wherein each agent involved gives away a single resource to another agent (plus payment).

- *OCSM-contract*: combines/merges characteristics of the above contract types into one contract type - where the ideas of the above four contract types can be applied simultaneously.

Each of the first four contract types above is shown by Sandholm in [San98] to be necessary (and avoids some of the local optima that the other three do not) but is not sufficient in and of itself for reaching the global optimum via *individually rational* contracts. Building upon [San98], in [EMST06], Endriss et al. investigate the different kinds of negotiation outcomes that agents can achieve by using different classes of deals. It is shown by Endriss et al. in [EMST06], for negotiation frameworks with and without monetary side payments, deals of any structural complexity may be necessary in order to be able to guarantee an optimal allocation of a negotiation (with respect to the chosen notion of social welfare) if deals are required to conform to the rationality criterion in question. Accordingly:

> The allocation/negotiation mechanism we propose in this thesis realises the "OCSM-contract" type.

2.2 Dialogues in Multi-agent Systems

This section introduces the need and uses of dialogues in multi-agent system contexts where agents are likely to differ in their beliefs and goals, and are required to communicate. It is useful to distinguish firstly communication between autonomous agents and communication in an object-oriented programming context, as is done by Wooldridge in [Woo09]. Communication is treated in object-oriented programming as method invocation — an *object* $o1$ may invoke and thereby execute a "publicly available" *method* $m1$ of another *object* $o2$. However, in an agent-oriented context, an *agent* $a1$ may communicate (utter) to another *agent* $a2$ to execute an action α, but the decision to execute or not lies entirely with $a2$. It cannot be taken for granted that $a2$ will execute action α just because another agent wants it to. Performing the action α may not be in the best interests of $a2$, or $a2$ may still decide not to do α even if it is in its best interests (because, for example, it might violate certain norms of the agent society or obligations it finds itself in if it was to do α).

Walton and Krabbe [WK95] define a dialogue as an orderly sequence of utterances (moves) between two participants where each participant has a goal and the dialogue, as a whole, has a goal. We interpret the "goal" of a dialogue, as discussed by McBurney and Parsons in [MP09], to mean the reaching of a desired type of outcome. An utterance is evaluated as good to the extent that it contributes to the goal of the dialogue. An utterance is evaluated as bad to the extent that it blocks the goal of the dialogue. It is possible to distinguish several important, characteristic types of dialogue, where each has its own distinctive rules and goals (desired types of outcomes), its permitted types of move, and its conventions for managing the commitments incurred by the participants as a result of the moves they make. Table 2.1 (adapted from [WK95]) characterises the six main types of dialogue as distinguished by Walton and Krabbe [WK95] according to the primary or main goal of the dialogue, which the participants implicitly subscribe to, the aims of the participant to get the best out of that dialogue type for itself, the initial situation that gives rise to the dialogue and some side benefits that can be important motives for entering the dialogue. Also in [WK95] are four kinds of dialogue rules that can be used to characterise the different types of dialogue, as follows:

- The "locution rules" of a dialogue define the permissible locutions, like statements, questions, inferences, and so on.
- The "structural rules" define turn taking and the order in which moves can be made by each participant (i.e. which moves are permitted or required after each move).
- The "commitment rules" define which propositions go in or out of commitment stores in each type of move, where a commitment store is a type of information store used as a way of tracking the claims/assertions/propositions made by participants in dialogue.
- The "win-loss rules" define the participants' aims in the dialogue and determine the conditions under which a player wins or loses the game.

The kind of dialogues of interest to us in this thesis are such that allow agents to communicate in spite of their differing preferences over resources and to agree on *deals* (exchanges of resources) of the kind defined in Section 2.1.6.

Type	**Initial Situation**	**Main Goal**	**Participant's Aims**	**Side Benefits**
Persuasion Dialogue (Critical Discussion)	Conflicting Points of View	Resolution of Such Conflicts by Verbal Means	Persuade the Other(s)	Develop & Reveal Positions, Build Confidence, Influence Onlookers, Add to Prestige
Negotiation	Conflict of Interests & Need for Cooperation	Making a Deal[1]	Get the Best out of It for Oneself	Agreement, Build Up Confidence, Reveal Positions, Influence Onlookers Add to Prestige
Inquiry	General Ignorance	Growth of Knowledge & Agreement	Find a "Proof" or Destroy One	Add to Prestige, Gain Experience Raise Funds
Deliberation	Need for Action	Reach a Decision	Influence Outcome	Agreement, Develop & Reveal Positions, Add to Prestige, Vent Emotions
Information-Seeking Dialogue	Personal Ignorance	Spreading Knowledge & Revealing Positions	Gain, Pass on, Show, or Hide Personal Knowledge	Agreement, Develop position, Influence Onlookers, Add to Prestige, Vent Emotions
Eristics	Conflict & Antagonism	Reaching a (Provisional) Accommodation in a Relationship	Strike the Other Party & Win in the Eyes of Onlookers	Develop & Reveal Positions, Add to Prestige, Gain Experience, Amusement, Vent Emotions

Table 2.1: Types of Dialogue

Similar to [AS99, NMJ08], the goal of participating agents in the kinds of (negotiation) dialogues we will consider is to obtain their preferred (desired) resources and the goal of the dialogues as a whole is to facilitate exchanges of resources between agents. In order for agents to generate such dialogues (and represent a resource allocation mechanism/procedure), it is necessary to define a shared communication language and *protocol*, as well as identifying the *policies* that allow agents to act desirably best exploiting the protocol. A protocol in this thesis is taken to mean the possible actions that agents can take at different points of the interaction (the "locution" and "structual" rules as above), addressing issues in our case such as what types of *deals* (exchanges of resources) are possible, as well as what messages/locutions agents have to exchange and in what order to agree on such deals. A policy on the other hand is taken to mean the agent's strategy for determining what move to make at any given point of the interaction given the choices that the protocol allows.

In the following subsections we summarise existing work in the multi-agent field for agent communication languages, negotiation in general, logic-based negotiation, and then, in the subsequent section, the specific type of negotiation we consider in this thesis, namely argumentation-based negotiation.

2.2.1 Agent Communication Languages

To allow inter-operability between autonomous (heterogeneous) agents, a commonly understood agent communication language (ACL) is used. We present here background on two of the main languages developed in recent past: the Knowledge Query and Manipulation Language (KQML) and the Foundation for Intelligent Physical Agents (FIPA) ACL. The FIPA ACL in particular supersedes KQML, as we will see, and is the product of a standardisation body for the specification of a syntax and semantics for a 'standard' ACL. Both languages (KQML and FIPA ACL) are loosely based on speech act theory [Aus62, Sea69, CL90], which treats communication as action, and are defined in terms of a set of communicative acts as well as a formal semantics for the communicative acts. A communicative act occurs whenever one agent

[1] A "deal" here is meant in the general sense of the word and not as in Section 2.1.6.

sends a message to another. In particular, agents perform actions - communicative actions - in an attempt to influence other agents appropriately [Woo09].

KQML is an 'outer' message-based language for agent communication. It defines a common 'envelope' format for messages and is not concerned with the *content* part of messages [MLF96, PFPS+92]. A KQML message may be crudely thought of as an object (in the sense of object-oriented programming): each message has a *performative* (which may be thought of as the class of the message), and a number of *parameters* (attribute/value pairs, which may be thought of as instance variables) [Woo09]. Some of the main parameters used in KQML messages are summarized as follows:

- `:content` — content of the message or, equivalently, the object of the communicative action;
- `:reply-with` — whether the sender expects a reply to this message, and, if so, an identifier or expression which will be used in the reply by the agent responding to identify the original message;
- `:in-reply-to` — an expression that references an earlier action/message (i.e. the `:reply-with` parameter) to which this message is a reply;
- `:sender` — the identity of the sender of the message, i.e. the name of the agent of the communicative act.
- `:receiver` — the identity of the intended recipient of the message.

Different performatives require different sets of parameters. Examples of performatives are the following, for some sender S of the message, receiver R and content C:

- `achieve` — S wants R to make something true of their environment;
- `ask-one` — S wants to know one of R's answers to queston C;
- `tell` — S claims to R that C is in S's knowledge base.

A more complete list containing a total of 41 performatives can be found in the specification of the KQML ACL [FWW+93]. An example of a KQML message, adapted from [Woo09], is the following:

```
(ask-one
    :content (PRICE IBM ?price)
    :receiver stock-server
)
```

The intended interpretation of this message is that the sender is asking about the price of IBM stock.

Some criticisms of KQML regarding the performatives and lack of adequate semantics in particular lead to the development of a new, but rather closely related, agent communication language by the FIPA consortium [ffipa]. As discussed by Wooldridge in [Woo09], this ACL is superficially similar to KQML: it defines an 'outer' language for messages, it defines a number of performatives for defining the intended interpretation of messages, and it does not mandate any specific language for message content. In addition, the concrete syntax/structure for FIPA ACL messages closely resembles that of KQML — each message has a *performative* and a number of *parameters* (attribute/value pairs) very similar to KQML. Probably the most important difference between KQML and the FIPA ACL is in the collection of performatives it provides. Some of the performatives provided by the FIPA ACL, as summarised in its specification [FIP98], are as follows:

- `accept-proposal` — a general-purpose acceptance of a proposal that was previously submitted (typically through a `propose` act).
- `agree` — a general-purpose agreement to a previously submitted request to perform some action (typically through a `request` act).
- `cancel` — this performative (action) allows an agent to stop another agent from continuing to perform (or expecting to perform) an action which was previously requested.

- `cfp` — a general-purpose action to initiate a negotiation process by making a call for proposals to perform the given action.
- `confirm` — the sender informs the receiver that a given proposition is true, where the receiver is known to be uncertain about the proposition.
- `disconfirm` — the sender informs the receiver that a given proposition is false, where the receiver is known to believe, or believe it likely that, the proposition is true.
- `failure` — the action of telling another agent that an action was attempted but the attempt failed.
- `inform` — the sender informs the receiver that a given proposition is true.
- `not-understood` — the sender received a communicative act which it did not understand.
- `propose` — a general-purpose action to make a proposal or respond to an existing proposal during a negotiation process by proposing to perform a given action subject to certain conditions being true.
- `query-if` — the action of asking another agent whether or not a given proposition is (believed to be) true.
- `query-ref` — the action of asking another agent for the object referred to (or identified) by an expression (or definite descriptor).
- `refuse` — the action of refusing to perform a given action, and explaining the reason for the refusal.
- `reject-proposal` — a general-purpose rejection to a previously submitted proposal. The agent sending the rejection informs the receiver that it has no intention that the recipient performs the given action under the given preconditions.
- `request` — the sender requests the receiver to perform some action. The content of the message is a description of the action to be performed, in some language the receiver understands.

As well as an informal English description for each performative, the developers of the FIPA ACL provided a formal semantics to the language, defining *feasibility conditions* and *rational effects* of the communicative actions in terms of the intentional stance of the agent (i.e. mentalistic notions such as uncertain and certain beliefs, as well as desires and intentions). These logical definitions, further details of which can be found in the FIPA ACL specification [FIP98], generally characterise the belief state of the sending agent before performing the communicative act, and the expected possible belief states of the receiving agent after the act has been performed.

Singh argues in [Sin98] that there are considerable limitations of intentionality as a basis for defining the semantics of an ACL (i.e. using mental attitudes for standardising the semantics of the ACL). Notably, intentionality is concerned with agent internals and a communicative act is an external phenomenon. Intentionality can give guidance to developers but may be too strong a constraint for autonomous (heterogeneous) agents given that communication is inherently context-dependent and different applications may require variations on the agent behaviour and the performative semantics [Sin98]. In addition to this, internal states of agents cannot be verified and also the performatives and protocols specified as part of the FIPA ACL are not exhaustive [Sin98]. Recognising the above, we adopt an alternative semantic framework with an emphasis on pragmatics [MP09] and the protocols that agents agree to communicate under [PM99a, PM99b], as follows:

> In this thesis we give meaning to the communicative acts which we consider in terms of the protocols governing the interactions, where representation of the protocols as finite-state automata will suffice.

2.2.2 Negotiation

Negotiation arises out of a conflict of interests and need for interaction, and the desired outcome is to make a deal[1] [WK95, MP09]. Indeed, in a society of self-interested agents, agents cannot be assumed to share a common goal or a common set of preferences. One agent's interests may therefore conflict with those of others. Despite the potential for conflicts of interest,

the agents in a multiagent system will ultimately need to cooperate in order to achieve their goals [Woo09]. How agents can negotiate or compromise to reach agreement in situations where they are apparantly at loggerheads is an important area of research in artificial intelligence and computer science [RZ94, Kra01]. The kinds of negotiation and deals of interest in this thesis are those wherein participants bargain over the division of some (scarce) resources. We give an overview in this section of a couple of the protocols developed for negotiation over resources in a distributed setting. Namely, the Contract-Net [Smi80] and Task-Swap [GMR97] negotiation protocols. These protocols allow for an agent to negotiate simultaneously with multiple agents to decide which (single) agent to give or exchange resources with.

The Contract-Net Protocol [Smi80] is a protocol initially developed for task allocation over a distributed network of sensors, but is also perfectly suited to negotiation over resources in multi-agent distributed settings. It is inspired by a market-like model of negotiation. Ignoring much of the lower-level detail, the protocol consists of four interaction phases, involving two roles (initiator and bidders):

- *Announcement phase*: The initiator sends out a call for proposals of a task (resource) and then acts as the manager of the resource. An announcement can be addressed to all agents, to a subset of agents, or to a single agent.

- *Bidding phase*: Agents maintain a rank-ordered list of announcements that have been received and have not yet expired, and each agent can formulate a bid for each announcement received. The agent checks its list of announcements and selects one on which to submit a bid. The agents (bidders) send their proposals to the initiator.

- *Assignment phase*: The initiator maintains a rank-ordered list of bids that have been received for the task. When a bid is received, the initiator ranks the bid relative to others under consideration. If a bid is received that is determined to be satisfactory, then the resource is awarded immediately to the associated bidder. Otherwise, the initiator waits for further bids. If the expiration time is reached and the task has not yet been awarded, several actions are possible:

  - awarding the task to the most acceptable bidder(s);
  - transmitting another announcement (if no (acceptable) bids have been received);
  - ending without awarding the task or transmitting another announcement.

  Successful bidders are informed through an announced award message.

- *Confirmation phase*: The elected bidder confirms its intention to obtain the task, and the initiator rejects other bids.

Any agent can initiate an interaction, and the protocol is really a one-to-many protocol, possibly leading to the assignment of a single task (resource) to a single bidder.

The Contract-Net negotiation process is discussed under the assumption that an agent cannot submit a bid until it is idle (i.e. it completes processing of its current task and becomes available for processing another task). This strategy can lead to difficulty. For example, an initiator that issues an announcement may not receive any bids because there are no idle agents. Therefore, an initiator requires a way of determining what caused the lack of response. A possible technique for doing so is suggested by Smith in [Smi80]. Namely, for a bidding agent to specify why it cannot bid, e.g. it is busy, or ineligible, or it gives the task a low level ranking. The initiator (manager) agent can then proceed more appropriately. Indeed such a technique can be seen as a kind of argumentation-based negotiation (see Section 2.3). Other potential complications and possible extensions of the Contract-Net Protocol are also discussed in [Smi80].

The Task-Swap Negotiation Protocol developed by Golfarelli et al. [GMR97] uses a variant of the classical Contract-Net Protocol to allow negotiation over the exchange of bundles of resources and to cope with domains where task swap is the only possible type of contract (i.e. tasks are not sold and there are no monetary side payments as part of deals). The agents considered are heterogeneous, self-interested, have limited communication capabilities and as in the Contract-Net Protocol are not coordinated by any central supervisor. Indeed many realistic applications may benefit from adopting swap-based contracts instead of sale-based ones. In particular, as is discussed by Golfarelli et al. in [GMR97], autonomous robotic agents can decrease the costs for executing their missions by swapping the tasks they have been assigned.

The Task-Swap Negotiation Protocol is based on three main phases:

- *Announcement Phase*: The announcing agent formulates an announcement and broadcasts it to the other agents. An announcement consists of one or more tasks which the announcer is interested in exchanging.
- *Bidding Phase*: Each agent receiving the announcement formulates a bid and sends it to the announcer. The bid associated to an announcement consists of one or more tasks, owned by the bidder, which the bidder is interested in exchanging with the tasks included in the announcement. A swap is proposed only if it is individual rational (i.e. advantageous to the bidder). The bidder cannot bid other announcements nor can it start a new announcement until it receives either an award or an acknowledgement from the announcer.
- *Award Phase*: The announcer collects the bids for a fixed time. When this time expires the announcer determines, among all the bids received, the swap having the highest utility. If the utility is positive, then the swap is *individual rational* for both the announcer and the bidder. In such a case an award is broadcast to the winner and the swap is confirmed. Otherwise, no swap is confirmed. An acknowledgement is broadcast to all the non-winner bidders. The negotiation session is over.

Like the Contract-Net Protocol, the Task-Swap Negotiation Protocol is a one-to-many protocol possibly leading to the assignment of a single task (resource) to a single bidder.

The negotiation protocol we present in this thesis is an adaptation of the Contract-Net and and Task-Swap negotiation protocols, allowing not only for an agent to negotiate simultaneously with multiple agents but allowing also for agents to agree on exchanges of resources involving multiple (two or more) agents.

2.2.3 Logic-based Negotiation

Sadri et al. present in [STT01] a formal, logic-based approach to one-to-one agent negotiation, in the context of goal achievement in systems of agents with limited resource availability. Agents

have a *dichotomous* (0-1) cardinal utility function as in this thesis, such that an agent's utility value of a bundle is 1 if that bundle allows the agent to achieve its goal, 0 otherwise. More specifically, each agent is equipped with a knowledge base, which would contain beliefs such as the resources held by the agent, as well as the goal the agent is equipped with, an intention that might consist of the goal, a plan to achieve that goal, which may be to obtain a certain bundle of resources, and an associated set of available and missing resources that are needed to carry out the plan.

The purpose of negotiation in [STT01] is for agents to obtain missing resources, while retaining available ones that are necessary for the plan in its current intention. A *dialogue move* (or utterance) in [STT01], and somewhat similarly in this thesis, is an instance of a schema `tell(X, Y, Subject, T)`, where `X` is the utterer and `Y` is the receiver of the dialogue move, and `T` is the time when the dialogue move is uttered. `Subject` is the content of the dialogue move, expressed in some given content language. A *language for negotiation* in [STT01] is a set of dialogue moves and, as part of a given language, there are two subsets of dialogue moves: *initial moves* and *final moves*. An example of an initial move could be a request that is initiated by an agent for a resource that it requires. Each final move is either *successful* or *unsuccessful*. Examples of the types of dialogue moves that may make up a language are resource requests, acceptance of moves, refusal of moves and promise to exchange resources (see [STT01] for a possible concrete language for negotiation).

Conversation between agents in [STT01], and similarly in this thesis, is based on *dialogues*, where a single dialogue is a sequence of dialogue moves (utterances), according to a pre-defined acceptable sequence of utterances (the protocol), used to obtain a resource and such that agents alternate utterances in a dialogue. Intuitively, a dialogue should begin with an initial move, according to the given language for negotiation. Agent dialogues can be connected within sequences (of dialogues), all aimed at achieving an individual agent's goal. Sets of sequences of dialogues aim at allowing all agents in the system to achieve their goals, and can be used to solve the resource reallocation problem.

The proposed solution in [STT01] for allowing agents to achieve their goals (i.e. obtain needed

resources) is based on fully distributed agent dialogues as a way to request resources, propose resource exchanges, suggest alternative resources, and so on. The negotiation framework presented is a general one for representing policies governed by protocols and generating dialogues accordingly. A *negotiation policy* for an agent is specified as a set of *dialogue contraints*, which are if-then rules of the form $p(T) \wedge C \Rightarrow \hat{p}(T+1)$ where $p(T)$ and $\hat{p}(T+1)$ are shorthand for dialogue moves of the form `tell(X, Y, Subject, T)` as above and C is a conjunction of literals in the language of the knowledge base of the agent. If this is a dialogue constraint for an agent x, for example, then it is interpreted as follows: if at a certain time T in a dialogue some other agent y utters to x a dialogue move $p(T)$, then the corresponding instance of the dialogue constraint is triggered and, if the condition C is entailed by the knowledge base of x, then x will utter a dialogue move $\hat{p}(T+1)$, with y as receiver, at the next time $T+1$. An *agent program* (set of *dialogue constraints* associated with an agent) should ideally guarantee that the agent's utterances are conformant to the protocols governing its dialogues and that the agent's dialogues terminate. Other ideal properties desired of an agent program are discussed by Sadri et al. [STT01] and Endriss et al. [EMST03], e.g. the agent always produces *exactly one* reply to a (non final) move made by the other agent.

Our *communication rules* in this thesis for representing agent negotiation policies combine ideas from the dialogue constraints of Sadri et al. [STT01], which do not say how agents' beliefs are modified, and the*actions* of Rueda et al. [RGS02]. An action (ability) A in [RGS02] is an ordered triple $< P, X, C >$ denoted as $\{X_1, \ldots, X_n\} \stackrel{A}{\Leftarrow} \{P_1, \ldots, P_m\},\ \{not\ C_1, \ldots,\ not\ C_k\}$, where P is a set of m literals representing *preconditions* for A, C is a set of k *constraints* of the form *not* L (L is a literal) and X is a set of n literals representing *consequences* of executing A. An agent, according to [RGS02], can consider an action only if its beliefs tell it that the preconditions and constraints are satisfied. The consequences specify the revisions to be made to the agent's set of beliefs as a result of applying the action, where revision consists of removing any literal in the belief base of the agent that is complimentary of any literal in X and then adding X to the resulting set.

It is assumed in [STT01] that all dialogues between an agent i and any other agents are *atomic*, i.e., it is not possible for i to be engaged in two different dialogues at the same time. This is

an assumption we relax in this thesis, as follows:

> Conversation between agents in this thesis will be based on *dialogues* (sequences of utterances by agents) generated by the agents' negotiation policies (sets of *communication rules*) and governed by protocols represented as finite-state machines. The negotiation policies will allow for an agent to be engaged in two or more different dialogues at the same time.

Moreover, we consider one-to-many agent dialogues in this thesis as well as one-to-one dialogues as in [STT01].

2.3 Argumentation-Based Negotiation

Negotiation deals with the problem of agents coming to agreement on matters of common interest (exchanges of resources in this thesis). Argumentation on the other hand deals with the problem of exchanging *arguments* (which could be for example reasons or justifications supporting a certain stance) for deciding what to believe or do. Implicitly, it involves providing principled techniques for handling inconsistencies among the beliefs of multiple agents, which would come to be known as agents put forward arguments, and extracting rationally justifiable positions from the inconsistent pool of arguments [Woo09]. We give in this section a general introduction to argumentation and then a discussion on the use of argumentation for negotiation.

2.3.1 Introduction to Argumentation

Argumentation is a large topic of interest spanning over a number of research areas ranging from philosophy to artificial intelligence. As pointed out van Eemeren et al. in [vEGH02], it is important to note that there are statements whose acceptability can be established with no problem. Examples of these are factual statements whose truth can be verified. The acceptability of nonfactual statements can also sometimes be agreed on quickly, for instance, when they concern commonplace values or judgements (e.g. "Parents should take care of their children").

In many other instances it is very difficult to agree on the acceptability of a statement, particularly if it involves a complex matter or is strongly tied to particular values and norms (e.g. "Reading is (not) the best way to improve your language skills"). It is such statements that must be supported by further proof for the speaker's argumentation as a whole to be accepted as an adequate defence (or refutation) of the standpoint.

An *argument* can be seen as a claim with some justification, e.g. "This film should have won an Oscar because it was a good movie with an edge." As discussed in [BH08a], *argumentation* is a key way humans deal with conflicting information. It can be seen as a cognitive process involving identifying arguments and counterarguments (e.g. "What are the pros and cons for the safety of mobile phones for children?"); weighing, comparing, or evaluating arguments (e.g. "What sense can we make of the arguments concerning mobile phones for children?"); or drawing conclusions (e.g. "A parent answering the question 'Are mobile phones safe for my children?'"). Moreover, it may be monological involving one agent (e.g. a newspaper article by a journalist, a political speech by a politician etc) or dialogical involving more than one agent (e.g. lawyers arguing in a court, politicians debating about new legislation etc). As discussed in [BH08a], monological argumentatation is an internal process involving collating and analysing information, possibly producing a tangible output (e.g. an article or speech or decision). The emphasis of the monological view is on how to construct the arguments and how to draw conclusions from the assembled arguments. Dialogical argumentation on the other hand can be seen as incorporating monological argumentation but the emphasis is on interaction, i.e. the nature of the interactions between the agents and on the process of building up (constructing) the set of arguments (for and against a particular claim) until the agents collectively reach a conclusion.

A few examples of approaches which exist for representing arguments are the Toulmin Model [Tou58], Argument Schemes [vEGH02, Wal95] and the Carneades Argumentation Framework [GPW07]. These allow capturing justification for a claim by bringing together various interrelated components in specific ways. In evaluating the soundness of arguments, defective argumentation can occur due to contradictions in the argumentation as a whole, or individual arguments may be unacceptable or otherwise flawed. In assessing the soundness of argumenta-

tion all complex argumentation must be broken down into single arguments, each of which must be assessed. However, it is often advisable not to proceed to the assessment of the individual arguments before determining whether the argumentation as a whole is consistent. According to van Eemeren et al. [vEGH02], this holistic checking for consistency occurs at two levels:

- *Logical Inconsistency*: When statements are made that, because they contradict each other or have conflicting implications, cannot possibly both be true.

- *Pragmatic Inconsistency*: When argumentation contains two statements that, although not logically inconsistent, have consequences in the real world that are contradictory. As an example, the promise "I'll pick you up in the car" does not in a strict sense logically contradict the statement "I don't know how to drive", but in everyday conversation, it is unacceptable for such a promise to be followed by this statement.

To handle arguments systematically and "formalise" the argumentation process, it is necessary to capture and analyse the arguments and counterarguments relevant to a particular claim. Moreover, a framework for doing so should have in place principled criteria for suggesting whether the claim is "warranted" or "unwarranted" given the "constellation" of arguments and counterarguments [BH08a]. Numerous formal theories for modelling the argumentation process exist, e.g. abstract systems [Dun95], defeasible systems [GS04, DKT09] and coherence systems [BH08a].

The main approach we consider for modelling the argumentation process in this thesis is *Assumption-Based Argumentation* [DKT09] which can be seen as a *defeasible* logic-based "instantiation" of *Abstract Argumentation* [Dun95].

See Section 2.4 for a more detailed discussion on Abstract Argumentation and Assumption-Based Argumentation.

2.3.2 Argumentation for Negotiation

The basic idea of argumentation-based negotiation (ABN) is stated as allowing additional information to be exchanged as part of the negoatiation process, over and above proposals. For example, two agents negotiate over which should carry out a particular task or receive some resource, putting forward proposals and counter-proposals. Then the idea is, in addition, for an agent to put forward arguments as a means of justifying its negotiation stance and/or persuading the other to change its negotiation stance [JFL+01]. In doing so, the agents dialogue with one another in an argumentative sense, each seeking to make its proposal more attractive (acceptable) by providing additional meta-level information in the form of arguments for its position. It is argued that doing so can increase the efficiency of the negotiation process and increase the range of possible agreements [JFL+01, RRJ+03], a claim which has begun to be analysed formally and experimentally in recent works [RPSD07, PHD+07, KJRM09].

An example of a model developed to allow argumentation to fit into the general negotiation process is that of van Veenen and Prakken [vVP05]. Here a simple negotiation language for trading proposals as that of Wooldridge and Parsons [WP00] is augmented with a series of illocutionary moves which allow for the passing of arguments. In this model, as in other models such as [RSD03, KJRM09], the passing of an argument or a *challenge* for justification can be seen as marking a transition from the negotiation protocol to a separate argumentation protocol. Possible protocols for such argumentative dialogues have been suggested and analysed in [AMP00, APM00, Pra05]. On completion of the argumentation dialogue, the dialogue as a whole would make the reverse transition and pick up the negotiation dialogue once again. This general idea of one dialogue type being embedded within or following another is discussed by McBurney and Parsons in [MP02] where a logic-based formalism is presented for modelling such complex dialogue occurrences between intelligent and autonomous software agents.

As discussed by Rahwan et al. in [RRJ+03], there is a growing community of researchers that advocate the idea that agents can increase the likelihood and quality of an agreement by exchanging arguments which influence each others' states, and that sometimes this argument exchange is essential when various assumptions about agent rationality cannot be satisfied.

Section 2.2.2 discussed briefly how the Contract-Net protocol could benefit from allowing agents to exchange arguments. Namely, for bidding agents to specify why they cannot bid (when they cannot bid) and for the initiator (manager) agent to use this information and proceed more appropriately. The logic-based dialogue approach of Sadri et al. [STT01] (discussed in Section 2.2.3) also includes a form of argumentation using counter-proposals intended to change the *intention* (and perhaps the *plan*) of the other agent. A counter-proposal is an alternative proposal generated in response to a proposal, with the promise of a different resource deal. The framework could readily be extended to incorporate more argumentation moves, such as critique (comments on which parts of a proposal the agent like or dislikes), and moves to challenge and justify prior proposals and decisions. In using these argumentative moves the negotiator provides additional meta-level information and seeks to make her position or proposal more attractive (acceptable). As stated by Jennings et al. in [JFL+01], the nature and types of the arguments can vary enormously. Common categories, as occurring in [KSE98], include: threats (i.e. failure to accept the proposal means something negative will happen), rewards (i.e. acceptance of the proposal means something positive will happen), and appeals (i.e. this option should be preferred over alternatives for some reason).

In [JFL+01], Jennings et al. state two main areas of future work for ABN, as follows:

1. The definition of suitable argumentation protocols, that is, sets of rules that specify how agents generate and respond to arguments based upon what they know.

2. The transition between the underlying negotiation protocol and the argumentation protocol. When is the right time to make this transition, when is it right to start an argument?

We address the first challenge above in this thesis for the resource reallocation problem. We do not consider the second challenge above and instead, similar to [SJNP97, PSJ98], define a single protocol that combines both negotiation and argumentation. The work of Rahwan et al. [RRJ+03] aims at setting up a broader research agenda for ABN in multi-agent systems. It does so by splitting the elements of an ABN framework into two main parts, the interaction environment and the agents participating in that interaction, as follows:

- the elements that are external to the agents, namely the communication and domain languages, the negotiation protocol, and various information stores.
- the internal elements and functionalities necessary to enable an agent to conduct ABN. More precisely, the process of argument and proposal evaluation, argument and proposal generation, and argument selection.

We address both parts of ABN in this thesis in putting together a generative (fully worked out) ABN framework for the resource reallocation problem. In particular, firstly, we present a rich communication and domain language that enables agents to exchange meta-information (e.g. why a particular outcome is not suitable for the agent), as well as information that describes outcomes (potential agreements or potential deals). Secondly, we specify a protocol for our ABN dialogues, constraining the use of the language and governing the interaction of participants. We do so by specifying at each stage of a dialogue instance who is allowed to say what. Thirdly, we define the agents' internal specification (communication rules and associated procedures) to conform to the protocols. In addition to interpreting and evaluating incoming proposals regarding exchanges of resources and generating outgoing proposals and responses, as is minimally required in negotiation, our agents are capable of generating and evaluating accompanying arguments (and updating their mental states accordingly).

> We put together a generative (fully worked out) ABN framework in this thesis for the resource reallocation problem, defining the necessary internal and external elements required of agents and the interaction environment.

2.4 Computational Argumentation

The different forms of reasoning that an agent may perform (e.g. epistemic, practical, etc) are often defeasible as the information available to the agent may be conflicting in general (e.g. if this information comes from different sources) [Ton08] or may be presumptious in the sense defined by Walton in [Wal95] (i.e. tentative and subject to later retraction)[2]. Thus the

[2]Presumptions are ofen required, for example, if a decision or stance needs to be taken in the presence of incomplete information. A presumption, according to Walton [Wal95], is functionally opposed to burden of

approaches to representing argumentation in logic are often based on non-monotonic systems. With monotonic inference rules, once the premises of a rule are proved, the rule can be applied, the conclusion of the rule can be derived and will continue to hold no matter no matter how many new premises are added (even to the point where they form an inconsistent set) [AvH04]. However, in non-monotonic systems, a rule may not be applied even if all premises are known because contrary reasoning chains have to be considered [AvH04]. In general, such rules are called defeasible because they can be defeated by other rules. As new information comes in it can change the circumstances of a given situation. The Assumption-Based Argumentaion (ABA) approach explained in this section is such that a rule is defeasible if one (or more) of its premises is an "assumption". An assumption can be seen as a presumption in the sense defined by Walton in [Wal95]. We discuss first in this section the abstract view of argumentation and then the particular argumentation framework (ABA) which we use and build upon in this thesis to instantiate the decision-making of the agents and their ability to exchange arguments as part of the negotiation process. An overview of other possible argumentation frameworks can be found in [RS09].

2.4.1 Dung's Abstract Argumentation Framework

The work of Dung [Dun95] develops a highly abstract theory for argumentation whose central notion is the acceptability of arguments. This is done in two steps. Firstly, a formal, abstract but simple theory of argumentation is developed to capture the notion of acceptability of arguments in a general way. Secondly, the "correctness" (or "appropriateness") of this theory is demonstrated. The theory is based on a notion of argumentation framework defined as a pair of a set of arguments and a binary relation representing the attack-relationship between arguments. Here, an argument is an abstract entity whose role is solely determined by its relations to other arguments. No special attention is paid to the internal structure of the arguments.

Dung [Dun95] defines various notions of semantics (that define the acceptability of sets of

proof, in the sense that it removes or absolves one side from the burden and shifts the burden to the other side.

arguments) in terms of the attack relation between arguments; ranging from the credulous to the sceptical, and the relations between these semantics are given. Given two sets of arguments X and Y, we say that X attacks Y iff there exists $x \in X$ and $y \in Y$ such that x attacks y. As summarised in [Ton08], a set X of arguments then is

- conflict-free iff X does not attack itself;
- admissible iff X is conflict-free and X attacks every set of arguments Y such that Y attacks X;
- preferred iff X is maximally admissible;
- sceptically preferred iff X is the intersection of all preferred sets of arguments;
- complete iff X is admissible and X contains all arguments x such that X attacks all attacks against x;
- grounded iff X is minimally complete;
- ideal iff X is admissible and it is contained in every preferred set of arguments.

Deciding upon a semantics can be seen as a means of choosing/deciding the kinds of sets of arguments (positions) that are acceptable for an agent. *Conflict-freeness* can be thought of as the most basic requirement for a rational position: it basically says that an argument set should be internally consistent. An *admissibile* position is one that is internally consistent and which defends itself against all attackers. The *preferred* semantics can be seen as going a step further than admissibility and encompassing as much of the available evidence (arguments) as possible [Woo09]. More sceptical semantics still are the *sceptically preferred*, *complete*, *grounded* and *ideal* semantics, which unlike the previous three determine unique solutions and do not allow multiple (different) sets of arguments to be acceptable for a given argumentation framework. The last notion was not in the original [Dun95], but has been proposed recently by Dung et al. [DKT06] as an alternative, less sceptical semantics than the grounded semantics.

2.4.2 Assumption-Based Argumentation

The abstract view of argumentation in particular [Dun95] does not deal with the problem of actually finding arguments and attacks amongst them. This section introduces Assumption-Based Argumentation (ABA), an approach for representing argumentation in logic, giving the arguments themselves *structure*, giving a precise meaning to the notion of "attack" between arguments and presenting a notion of semantics for arguments in terms of ABA. Further details to that presented here can be found in [DKT06, DMT07, DKT09].

ABA Framework

Typically, as in Defeasible Logic Programming [GS04] and Deductive Argumentation [BH08a], arguments are built by connecting rules in the belief set, and attacks arise from conflicts among such arguments. In ABA, arguments are (implicitly meant to be) obtained by reasoning backwards with a given set of inference rules (the belief set), from conclusions to premises that are *assumptions*, and attacks are defined in terms of a notion of *contrary* of assumptions. The ABA approach for representing argumentation, as presented in this section, as well as approaches such as Defeasible Logic Programming [GS04] and Deductive Argumentation [BH08a], deal with cases where there is a single agreed upon knowledge-base. Thus, these approaches can be seen as intra-agent argumentation, whereby an agent engages in a dialectical argumentation process with itself to determine its "acceptable" beliefs (desires, intentions).

An agent's belief set and backward reasoning in ABA are defined in terms of a deductive system, which is defined as follows:

Definition 2.6 (Deductive System) *A deductive system is a pair* $(\mathcal{L}, \mathcal{R})$ *where*

- $\mathcal{L}$ *is a formal language consisting of countably many sentences, and*
- $\mathcal{R}$ *is a countable set of inference rules of the form*

$$\frac{x_1, \ldots, x_n}{x}$$

where $x \in \mathcal{L}$ is called the conclusion and $x_1, \ldots, x_n \in \mathcal{L}$ are called the premises of the inference rule, and $n \geq 0$.

A deduction of a conclusion by means of a deductive system $(\mathcal{L}, \mathcal{R})$ then is obtained by applying backwards the rules in $\mathcal{R}$, as follows:

Definition 2.7 (Deduction of a conclusion) *Given a deductive system $(\mathcal{L}, \mathcal{R})$ and a selection function f (any function that maps from sets of elements to elements), a (backward) deduction of a conclusion x based on (or supported by) a set of premises P, denoted $P \vdash x$, is a sequence of multi-sets $S_1, \ldots, S_m$, where $S_1 = x$, $S_m = P$, and for every $1 \leq i < m$, where σ is the sentence occurrence in S_i selected by f:*

1. *If σ is not in P then $S_{i+1} = S_i - \{\sigma\} \cup S$ for some inference rule of the form $\sigma \leftarrow S \in R$.*
2. *If σ is in P then $S_{i+1} = S_i$.*

Each S_i is referred to as a step in the deduction.

Deductions are the basis for the construction of arguments in ABA. To obtain an argument from a backward deduction the premises are restricted to ones that are *assumptions*. Indeed a set of assumptions and a contrary mapping, together with a deductive system as above, make up an ABA framework, as follows:

Definition 2.8 (ABA framework) *An ABA framework is a tuple $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, c \rangle$ where*

- *$(\mathcal{L}, \mathcal{R})$ is a* deductive system*, consisting of a language $\mathcal{L}$ and a set $\mathcal{R}$ of inference rules;*
- *$\mathcal{A} \subseteq \mathcal{L}$, $\mathcal{A} \neq \{\}$. $\mathcal{A}$ is referred to as the set of (candidate) assumptions;*
- *If $x \in A$, then there is no inference rule of the form $x \leftarrow x_1, \ldots, x_n \in \mathcal{R}$;*
- *c is a (total) mapping from $\mathcal{A}$ into $\mathcal{L}$, where $c(x)$ is referred to as the* contrary *of x.*

Intuitively, assumptions are sentences that can be assumed to hold but can be questioned and disputed (as opposed to axioms that are instead beyond dispute), and the contrary of an assumption stands for the reason why that assumption may be undermined and thus may need to be dropped. Indeed an inference rule containing one or more assumptions in the premises can be seen as a "defeasible rule" in the sense of [GS04], used for representing defeasible knowledge, i.e. weak or tentative information that may be used if nothing could be posted against it. On the other hand, an inference rule containing no assumptions in the premises can be seen as a "strict rule" in the sense of [GS04], used for representing non-defeasible information, i.e. strict (sound) knowledge.

We assume in this thesis that the inference rules in $\mathcal{R}$ have the syntax $l_0 \leftarrow l_1, \ldots l_n$ (for $n \geq 0$) where $l_i \in \mathcal{L}$. We will represent $l_0 \leftarrow$ simply as l_0, and refer to this as a *fact*. As in [DMT07] we restrict attention to *flat* ABA frameworks, such that if $l_0 \in \mathcal{A}$, then there exists no inference rule of the form $l_0 \leftarrow l_1, \ldots, l_n \in \mathcal{R}$, for any $n \geq 0$. Lastly, an *argument* in favour of a sentence $p \in \mathcal{L}$ is a backward deduction $A \vdash p$ such that $A \subseteq \mathcal{A}$

Attack-relationship between arguments

Unlike Dung's abstract argumentation framework [Dun95] wherein the attack-relationship between arguments is explicit, in ABA this relationship is determined by the internal structure of the arguments. In ABA, as defined above, arguments are deductions from conclusions based solely upon assumptions. A set of assumptions attacks another set of assumptions as follows:

Definition 2.9 (Attack in ABA) *A set of assumptions A* attacks *a set of assumptions B iff there exists an assumption x in B and, for some $y \in c(x)$, a deduction $A' \vdash y$ such that A' is a subset of A: if this is the case, it is said that A attacks B on x.*

This notion of attack between sets of assumptions implicitly gives a notion of attack between arguments supported by sets of assumptions: the attacking argument needs to have as conclusion the contrary of an assumption in the support of the attacked argument.

Argument Semantics

Standard argument semantics as per abstract argumentation (see Section 2.4.1) can be ascribed to ABA frameworks, as follows: A set of assumptions A is

- conflict-free iff it does not attack itself;
- admissible iff A is conflict-free and A attacks every set of assumptions B that attacks A;
- preferred iff it is maximally admissible;
- sceptically preferred iff A is the intersection of all preferred sets of assumptions;
- complete iff A is admissible and contains all assumptions x such that A attacks all attacks against $\{x\}$;
- grounded iff it is minimally complete;
- ideal iff A is admissible and it is contained in every preferred set of assumptions.

Our focus in this thesis is the admissibility semantics. We will say that *a claim is admissible* if it is the conclusion of an argument that is supported by a set of assumptions that can be extended to an admissible set of assumptions. Indeed computational mechanisms exist for computing sets of assumptions corresponding to the various semantics, which we discuss next.

Dialectic proof procedures

Within ABA, implicitly, a set of assumptions stands for the set of all arguments whose premises are contained in the given set of assumptions. Thus, the computation of "acceptable" sets of arguments amounts to computing "acceptable" sets of assumptions. The work of Dung et al. [DMT07] present a family of dialectical proof procedures for ABA that can compute "acceptable" sets of assumptions according to various semantics. The proof procedures find a set of assumptions, to defend a given belief (claim), by starting from an initial set of assumptions that supports an argument for the claim and adding defending assumptions incrementally to

counter-attack all attacks. They can be seen as generating a winning strategy for a proponent to win a dispute against an ideal opponent who attacks in every possible way the initial and defending arguments of the proponent. The proponent wins if it has a counter-attack against every attacking argument by the opponent.

The proof procedures in [DMT07] generate and find arguments by reasoning backwards from conclusions to assumptions. They use backward reasoning both to find an initial argument for a given belief and to find attacking and defending arguments for the contrary of an assumption. Each step in a backward argument is viewed as a partially completed, potential argument. Any assumption in such a potential argument can be attacked (by finding an argument for its contrary) before the argument is completed. We present next the AB-dispute derivations of Dung et al. [DMT07] which prove that a claim is admissible by building an admissible set of assumptions which support/defend the claim. The proponent's winning strategy is represented as a sequence of quadruples consisting of proponent and opponent nodes labelled by multi-sets of sentences, representing steps of potential arguments, as well as the set of defence assumptions (assumptions used by the proponent) and culprits (opponent's assumptions that have been attacked) generated so far, as follows:

Definition 2.10 (AB-dispute derivation) *Let* $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, c \rangle$ *be an ABA framework. Given a selection function, an* AB-dispute derivation of a defence set A *for a sentence* α *is a finite sequence of* quadruples

$$\langle \mathcal{P}_0, \mathcal{O}_0, A_0, C_0 \rangle, \ldots, \langle \mathcal{P}_i, \mathcal{O}_i, A_i, C_i \rangle, \ldots, \langle \mathcal{P}_n, \mathcal{O}_n, A_n, C_n \rangle$$

where $\mathcal{P}_0 = \{\alpha\}$, $A_0 = \mathcal{A} \cap \{\alpha\}$, $\mathcal{O}_0 = C_0 = \{\}$, $\mathcal{P}_n = \mathcal{O}_n = \{\}$, $A = A_n$ *and for every* $0 \leq i < n$, *only one* σ *in* $\mathcal{P}_i$ *or one* S *in* $\mathcal{O}_i$ *is selected, and:*

1. *If* $\sigma \in \mathcal{P}_i$ *is selected then*

   (i) if σ *is an assumption, then*

   - $\mathcal{P}_{i+1} = \mathcal{P}_i - \{\sigma\}$

- $A_{i+1} = A_i$
- $C_{i+1} = C_i$
- $\mathcal{O}_{i+1} = \mathcal{O}_i \cup \{\{c(\sigma)\}\}$

(ii) if σ is not an assumption, then there exists some inference rule $\sigma \leftarrow R \in \mathcal{R}$ such that $C_i \cap R = \{\}$ (filtering of potential defence arguments by culprits) and

- $\mathcal{P}_{i+1} = \mathcal{P}_i - \{\sigma\} \cup (R - A_i)$ *(filtering of defence assumptions by defences)*
- $A_{i+1} = A_i \cup (\mathcal{A} \cap R)$
- $C_{i+1} = C_i$
- $\mathcal{O}_{i+1} = \mathcal{O}_i$

2. If S is selected in $\mathcal{O}_i$ and σ is selected in S then

(i) if σ is an assumption, then

(a) either σ is ignored, i.e.

- $\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\} \cup \{S - \{\sigma\}\}$
- $\mathcal{P}_{i+1} = \mathcal{P}_i$
- $A_{i+1} = A_i$
- $C_{i+1} = C_i$

(b) or $\sigma \notin A_i$ (filtering of culprits by defence assumptions) and $\sigma \in C_i$ (filtering of culprits by culprits) and

- $\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\}$
- $\mathcal{P}_{i+1} = \mathcal{P}_i$
- $A_{i+1} = A_i$
- $C_{i+1} = C_i$

(c) or $\sigma \notin A_i$ (filtering of culprits by defence assumptions) and $\sigma \notin C_i$ (filtering of culprits by culprits) and

(c.1) if $c(\sigma)$ is not an assumption, then

- $\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\}$

- $\mathcal{P}_{i+1} = \mathcal{P}_i \cup \{c(\sigma)\}$
- $A_{i+1} = A_i$
- $C_{i+1} = C_i \cup \{\sigma\}$

(c.2) if $c(\sigma)$ is an assumption, then

- $\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\}$
- $\mathcal{P}_{i+1} = \mathcal{P}_i$ *(filtering of defence assumptions by defences)*
- $A_{i+1} = A_i \cup \{c(\sigma)\}$
- $C_{i+1} = C_i \cup \{\sigma\}$

(ii) if σ is not an assumption, then

- $\mathcal{P}_{i+1} = \mathcal{P}_i$
- $A_{i+1} = A_i$
- $C_{i+1} = C_i$
- $\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\} \cup \left\{S - \{\sigma\} \cup R \,\middle|\, \sigma \leftarrow R \in \mathcal{R}, \text{ and } R \cap C_i = \{\}\right\}$ *(filtering of culprits by culprits)*

Note that the dispute derivations as above incorporate filtering of and filtering by defence and culprit assumptions so that the final assumption-defence set constructed by the derivation does not attack itself, and so that the dispute derivations can be more efficient and can (finitely) compute infinite dispute trees. Further details can be found in [DMT07], including discussions of the soundness and completeness of the dispute derivations.

In closing, it is important to note that ABA, and to a large extent argumentation frameworks in general (e.g. [GS04, BH08a]), up to now have been considered in a single-agent setting. ABA, in particular, is such that an agent engages in a dispute (dialectic proof procedure) with itself (an imaginary opponent) to decide whether a claim is acceptable according to some acceptability criteria. Amongst our aims in this thesis are to present a generalised proof procedure for the admissibility semantics of ABA, which is still a dispute by an agent with itself but such that the outcome can be readily communicated to other agents. This is important for applications in multi-agent systems (an example of which is argumentation-based negotiation, see Section 2.3.2)

wherein agents may differ in the knowledge they have and may need to communicate their arguments between one another to convince each other of the acceptability or not of a given claim (or negotiation stance as the case may be).

2.5 Summary

We have presented in this chapter background knowledge on the key subject areas underpinning this thesis. The introduction of resources, allocations, agent preferences and social welfare in Section 2.1, as well the choices we make in this thesis regarding these concepts, will be useful in framing the resource reallocation problem of interest in this thesis (see Chapter 3). The introduction, also in Section 2.1, of distributed allocation procedures and deals will be useful in capturing the kinds of deals that are necessary and sufficient for solving our resource reallocation problem of interest in this thesis (see Chapter 4). The discussion in Section 2.2 of issues involved with agent communication and interaction for solving multi-agent problems such as the resource reallocation problem by means of agent dialogues (negotiation), as well as the discussion in Section 2.3 of issues surrounding argumentation-based negotiation, will be useful in presenting an argumentation-based negotiation protocol and policy for solving our resource reallocation problem of interest in this thesis (see Chapter 5). Lastly, the presentation of assumption-based argumentation in Section 2.4 will provide a basis for the general-purpose argumentation framework which we define for multi-agent settings (see Chapter 6) and which we use for grounding our argumentative negotiation policy (see Chapter 7).

Chapter 3

Agent Systems for the Resource Reallocation Problem

We define in this chapter the key components of our problem domain, amounting to the notion of agent system (Section 3.1), resource reallocation problem (Section 3.3), and individual and social welfare (Section 3.4). Informally, the problem domain is as follows: An agent is initially allocated a set of resources, possibly none, where resources are single-unit (distinguishable by name), indivisible (an agent either has a particular resource or it does not) and unshareable (no two agents have the same resource). Each agent has a goal which it desires to fulfil. A certain goal may be fulfilled by a choice of different resources. Also, a certain resource may fulfil a choice of different goals. We demonstrate in Section 3.2 an example application domain.

Throughout the thesis we will adopt the following notation: $\neg$ stands for (classical) negation; terms beginning with a capital letter are variables; terms beginning in small-case are constants; _ stands for an anonymous variable (as in Prolog).

3.1 Agent System

We consider agent systems where (i) each (resource reallocation) agent may have in its possession some (or no) resources; and (ii) each agent has a goal, which is fulfilled if the agent has a particular resource which the agent believes to fulfil its goal. In addition to beliefs about resources fulfilling goals, agents may hold beliefs about other agents existing in the system, holding resources, and having goals.

Definition 3.1 (Agent System) *An* agent system *is a triple* $(\mathcal{G}, \mathcal{R}, \mathcal{A})$ *such that*

- $\mathcal{G}$ *is a (finite) set of constants, representing* goals *of agents in the system*
- $\mathcal{R}$ *is a (finite) set of constants, representing* resources *(owned by agents) in the system*
- $\mathcal{A}$ *is a (finite) set of (resource reallocation)* agents*, where an agent is a quadruple* $\langle x,$ $Res(x)$, $G(x)$, $B(x)\rangle$ *such that*
  - x*, the* name*, is a constant, uniquely identifying the agent in* $\mathcal{A}$
  - $Res(x) \subseteq \mathcal{R}$ *is the set of* resources allocated to x*,*
  - $G(x) \in \mathcal{G}$ *is the* goal *of* x*,*
  - $B(x)$*, the* belief set *of* x*, is a consistent set of ground literals including (some) instances (for* $Y \neq x$*,* $\langle Y, _, _, _\rangle \in \mathcal{A}$*) of*
    - *(i)* `isAgent(`Y`)`*, representing the information that* Y *is another agent in the system;*
    - *(ii)* `has(`Y`,`R`)`*, for* $R \in \mathcal{R}$*, indicating* x*'s belief that agent* Y *is currently allocated resource* R*;*
    - *(iii)* $\neg$`has(`Y`,`R`)`*, for* $R \in \mathcal{R}$*, indicating* x*'s belief that agent* Y *is not currently allocated resource* R*;*
    - *(iv)* `goal(`Y`,`G`)`*, for* $G \in \mathcal{G}$*, indicating* x*'s belief that* G *is the goal of agent* Y*;*
    - *(v)* `fulfils(`R`,`G`)`*, for* $R \in \mathcal{R}$ *and* $G \in \mathcal{G}$*, representing that resource* R *fulfils goal* G*.*

Note that each agent has a single goal but several agents may have the same goal. Note also that each agent may have several resources. Given $AS = (\mathcal{G}, \mathcal{R}, \mathcal{A})$, let $names(AS) = \{x \mid \langle x, _ , _ , _ \rangle \in \mathcal{A}\}$.[1] We will assume that $names(AS) \cap \mathcal{G} = \{\}$, $names(AS) \cap \mathcal{R} = \{\}$ and $\mathcal{G} \cap \mathcal{R} = \{\}$ (namely, the constants used to refer to agents, goals and resources are distinct). We will also assume that resources held by agents do not overlap. Namely: $Res(x) \cap Res(y) = \emptyset$ for $x, y \in names(AS)$ where $x \neq y$, and that $\bigcup_{x \in names(AS)} Res(x) = \mathcal{R}$. An *allocation of resources* in an agent system then is a partitioning of $\mathcal{R}$ amongst the agents in $\mathcal{A}$, as follows:

Lemma 3.1 (allocation in an agent system) *Given an agent system $AS = (\mathcal{G}, \mathcal{R}, \mathcal{A})$, let P: $\mathcal{A} \rightarrow 2^{\mathcal{R}}$ such that $P(\langle x, _ , _ , _ \rangle) = Res(x)$. Then, P is an allocation of resources (in the sense of Definition 2.1).*

We will refer to the allocation P as Res, and say that *AS contains the allocation Res*.

The lemma above implies that, in our agent systems, resources are *indivisible* (i.e. agents cannot receive fractions of resources) and *non-sharable* (i.e. a resource cannot be allocated to two or more agents at the same time). Note also that, by using constants to distinguish resources, these are *single-unit* (i.e. there is only one copy of each resource in our agent systems).

In our agent systems it may be the case that two (or more) agents have the same goal. Each goal may be achieved by acquiring any of a number of resources, as indicated by the `fulfils` beliefs in the agents' belief sets. Note that we assume, for simplicity, that single resources suffice for the fulfilment of goals. Each resource fulfilling a goal may be seen as a *plan* for that goal.

Definition 3.2 (agent's beliefs about plans in an agent system) *An agent x's beliefs about plans in an agent system AS is defined as follows: $Pl_{AS}(x) = \{$ `fulfils(`R`,`G`)` $\mid$ `fulfils(`R`,`G`)` $\in B(x)\ \}$. We denote all beliefs about plans in an agent system as follows: $Pl_{AS} = \bigcup_{x \in names(AS)} Pl_{AS}(x)$.*

[1] By abuse of notation we will sometimes refer to an agent $\langle x, _ , _ , _ \rangle \in \mathcal{A}$ simply by its name, i.e. x.

With an abuse of terminology, from now on we will refer to beliefs about plans simply as plans. Note that it may be the case that there are no (beliefs about) plans for a goal in an agent system.

In general, in our agent systems, agents may have incorrect and only partial beliefs of the resources and goals of the other agents in the system. In this thesis we assume initial agent systems such that agents are aware of the existence of all their fellow agents in the agent system and have beliefs as to which resources fulfil which goals but have no beliefs (initially) about the resources held (or not held) by other agents, nor any beliefs about the goals of other agents, as follows:

Definition 3.3 (initial agent system) *For an initial agent system* $AS = (\mathcal{G}, \mathcal{R}, \mathcal{A})$*, all agents* $\langle x, Res(x), G(x), B(x)\rangle \in \mathcal{A}$ *are such that:*

- $\texttt{isAgent}(y) \in B(x)$ *for all* $y \in names(AS)$ *where* $y \neq x$*;*
- $Pl_{AS}(x) = Pl_{AS}(y)$ *for all agents* $y \in names(AS)$ *where* $y \neq x$*;*
- $\texttt{has}(_,_)$, $\neg\texttt{has}(_,_)$, $\texttt{goal}(_,_) \notin B(x)$.

We will refer to the allocation that the initial agent system AS contains as the initial allocation.

Agents would come to hold beliefs about the resources and goals held by other agents by means of dialogue (see Chapter 5).

To summarise:

In this thesis we assume that single resources suffice for the fulfilment of goals, that agents know which resources fulfil which goals, and that agents are aware of the existence of all their fellow agents in the agent system.

As a consequence of assuming that agents know which resources fulfil which goals, we have that $Pl_{AS} = Pl_{AS}(x)$ for any $x \in names(AS)$.

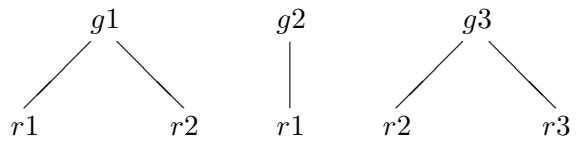


Figure 3.1: Plans Pl in Example 3.1.

3.2 Example Application Domain

Although simple, our agent systems are sufficiently powerful for representing real applications. We described in Chapter 1, Section 1.2 the kinds of problems that are of interest to us in this thesis. We illustrate in the following example how one such problem, namely the allocation and possible reallocation of medical appointments to booking agents, can be captured according to our representation of agent systems. Note that this would typically be a very large problem involving hundreds if not thousands of agents and although the example we present below consists of just three agents, this is for the sake of simplicity and clarity of illustration and not a restriction imposed on the framework which we present in this thesis. This example will be built upon and referred back to throughout the thesis to demonstrate the key concepts which we present.

Example 3.1 (application) *Consider an initial agent system consisting of three agents ($ag1$, $ag2$, $ag3$) representing patients; three resources ($r1$, $r2$, $r3$) representing medical appointments; three possible goals ($g1$, $g2$, $g3$), e.g. to see the doctor by the end of the week ($g1$), to see the doctor today ($g2$), and to have a morning appointment ($g3$). Consider also plans Pl as follows and as depicted in Figure 3.1:*

$Pl = \{$`fulfils(`$r1$`,`$g1$`)`, `fulfils(`$r2$`,`$g1$`)`, `fulfils(`$r1$`,`$g2$`)`, `fulfils(`$r1$`,`$g3$`)`, `fulfils(`$r3$`,`$g3$`)`$\}$

i.e. the first two appointments ($r1$ and $r2$) allow to see the doctor by the end of the week, the first appointment only ($r1$) allows to see the doctor today, and the first and third appointments ($r1$ and $r3$) are morning appointments. This can be represented as $AS = (\mathcal{G}, \mathcal{R}, \mathcal{A})$ with:

- $\mathcal{A} = \{\langle ag1, Res(ag1), G(ag1), B(ag1)\rangle,$

  $\langle ag2, Res(ag2), G(ag2), B(ag2)\rangle,$

$\langle ag3, Res(ag3),\ G(ag3), B(ag3)\rangle\}$

- $\mathcal{G} = \{g1, g2, g3\}$
- $\mathcal{R} = \{r1, r2, r3\}$

where

$ag1$	$ag2$	$ag3$
$Res(ag1) = \{r1\}$,	$Res(ag2) = \{r2\}$,	$Res(ag3) = \{r3\}$,
$G(ag1) = g1$,	$G(ag2) = g2$,	$G(ag3) = g3$,
$B(ag1) = \{$`isAgent(`$ag2$`),` `isAgent(`$ag3$`)}` $\cup\ Pl$	$B(ag2) = \{$`isAgent(`$ag1$`),` `isAgent(`$ag3$`)}` $\cup\ Pl$	$B(ag3) = \{$`isAgent(`$ag1$`),` `isAgent(`$ag2$`)}` $\cup\ Pl$

This system may represent the situation where patients ($ag1$, $ag2$, $ag3$) compete for available medical appointments ($r1$, $r2$, $r3$) with the same doctor, to achieve their goals $g1$ (for the first patient to see the doctor by the end of the week), $g2$ (for the second patient to see the doctor today) and $g3$ (for the third patient to have a morning appointment). The first two appointments ($r1$ and $r2$) suit the first patient. Only the first appointment ($r1$) suits the second patient. The first and third appointments ($r1$ and $r3$) suit the third patient. Note that agents have no beliefs about the appointments or goals of other agents, though they know which appointments fulfil which goals.

The problem of booking medical appointments is usually solved by a centralised system that does not take into account the patients' preferences, expressed in the example above simply as goals, and does not allow for assigned appointments to be later changed as new agents are considered, even if a change would satisfy the goals of all agents. Avaliable appointments are naturally indivisible, non-sharable and single-unit resources, as in our agent systems. Moreover, in the setting of booking medical appointments, goals are fulfilled by individual resources (appointments), and goals may be shared by several agents (patients), again as in our agent systems.

3.3 Solving the Resource Reallocation Problem

The *resource reallocation problem (r.r.p.)* amounts to identifying a reallocation of resources in an agent system wrt the initial allocation in the system, so that as many agents as possible can fulfil their goals.

Definition 3.4 (solving the r.r.p.) *Given an agent system* $AS = (\mathcal{G}, \mathcal{R}, \mathcal{A})$

- *the* r.r.p. for an agent $\langle x, Res(x), G(x), B(x)\rangle \in \mathcal{A}$ *is the problem of finding an agent system* $AS' = (\mathcal{A}', \mathcal{G}, \mathcal{R})$ *with* $\langle x, Res'(x), G(x), B'(x)\rangle \in \mathcal{A}'$ *such that* $r \in Res'(x)$ *and* `fulfils(`r`,`$G(x)$`)` $\in B'(x)$*;*

  we say that this AS' solves the r.r.p. for x *(wrt* AS*);*

  and we say that any such r solves the r.r.p. for x*;*

- *the* r.r.p. for the agent system AS *is the problem of finding* $AS' = (\mathcal{G}, \mathcal{R}, \mathcal{A}')$ *with* $names(AS') = names(AS)$ *solving the r.r.p. for as many agents in* $\mathcal{A}$ *as possible wrt* AS*, namely, given that, for any agent system* Z*,* $happy_{AS}(Z)$ *is a set of agent names such that* $happy_{AS}(Z) \subseteq names(AS)$ *and* Z *solves the r.r.p. for every* $x \in happy_{AS}(Z)$*,* AS' *is such that*

  - $happy_{AS}(AS) \subseteq happy_{AS}(AS')$
  - *there does not exist* $AS'' \neq AS'$ *such that* $happy_{AS}(AS'') \supset happy_{AS}(AS')$*;*

  we say that this AS' solves the r.r.p. for AS*;*

  we also say that the r.r.p. for AS is solved *if such* AS' *is found.*

Note that we take the term "solving the r.r.p." to mean "optimising the r.r.p.", i.e. finding an agent system and a possible reallocation of resources such that as many agents as possible have fulfilled their goals wrt the initial agent system. Also, we assume that the agents' goals ($G(x)$) are fixed and cannot change during the reallocation. Instead, the reallocation may require changing the agents' belief sets, e.g. so as to include information about other agents'

resources and goals. This will be discussed in greater detail in Chapter 5. Note also that our definition above disallows for the set of agents in a solution AS' to the r.r.p. for an agent system AS to contain agents other than the ones originally in AS (namely such that $names(AS) \neq names(AS')$). Our definition allows different plans (i.e. $Pl_{AS}(x) \neq Pl_{AS'}(x)$) in principle. However, in this thesis, our aim is to obtain AS' with the same plans ($Pl_{AS}(x)$) as in AS. In summary:

> Our aim, in this thesis, is to obtain an agent system AS' that solves the r.r.p. for an agent system AS with the same agents in AS' as in AS (namely such that $names(AS) = names(AS')$), with the same goals ($\mathcal{G}$) and resources ($\mathcal{R}$), and such that the agents' goals ($G(x)$) and plans ($Pl_{AS}(x)$) are the same in AS' as in AS. It is the resources and beliefs of agents that may undergo change.

We demonstrate below how the resource reallocation problem of Example 3.1 could be solved.

Example 3.2 (solving the r.r.p. for an agent system) *Given the agent system* $AS = (\mathcal{G}, \mathcal{R}, \mathcal{A})$ *of Example 3.1, the r.r.p. for* $ag1$ *is solved since it has goal* $g1$*, resource* $r1$ *and there is a plan such that* $r1$ *fulfils* $g1$*, and the r.r.p. for* $ag3$ *is solved since it has goal* $g3$*, resource* $r3$ *and there is a plan such that* $r3$ *fulfils* $g3$*. The r.r.p. for* $ag2$ *on the other hand is not solved since it has goal* $g2$*, resource* $r2$ *but no plan such that* $r2$ *fulfils* $g2$*. The r.r.p. for the agent system AS is solved by the agent system* $AS' = (\mathcal{G}, \mathcal{R}, \mathcal{A}')$ *with* $\mathcal{A}' = \{\langle ag1, Res'(ag1), G(ag1), B'(ag1)\rangle, \langle ag2, Res'(ag2), G(ag2), B'(ag2)\rangle, \langle ag3, Res(ag3), G(ag3), B'(ag3)\rangle\}$ *as follows, with resources that have changed hands between agents underlined:*

$ag1$	$ag2$	$ag3$
$Res'(ag1) = \{\underline{r2}\}$,	$Res'(ag2) = \{\underline{r1}\}$,	$Res(ag3) = \{r3\}$,
$G(ag1) = g1$,	$G(ag2) = g2$,	$G(ag3) = g3$,
$B'(ag1) = B(ag1)$	$B'(ag2) = B(ag2)$	$B'(ag3) = B(ag3)$

Indeed AS' *solves the r.r.p. for all agents in AS, and hence solves the r.r.p. for AS.*

The agent system AS' in the example above could be seen as arising from AS by allowing agents $ag1$ and $ag2$ to exchange resources $r1$ and $r2$. In our realistic example, this would amount to the two patients swapping medical appointments. Note that $ag3$ (i.e. its resources, goal and beleifs) is unchanged by the swap and hence still happy, and $ag1$ is happy before the swap and still happy after the swap. $ag2$ is happy after the swap since it has obtained a resource that solves its r.r.p. Generally:

Lemma 3.2 (obtaining a resource that solves the r.r.p. for an agent) *Let* $AS = (\mathcal{G}, \mathcal{R}, \mathcal{A})$ *be an agent system that contains allocation* Res *and such that* $\langle x, Res(x), G(x), B(x)\rangle \in \mathcal{A}$. *Let* $AS' = (\mathcal{G}, \mathcal{R}, \mathcal{A}')$ *be an agent system that contains allocation* Res' *and such that* $\langle x, Res'(x), G(x), B'(x)\rangle \in \mathcal{A}'$. *Assume* $Pl_{AS}(x) = Pl_{AS'}(x)$. *If* AS' *solves the r.r.p. for* x *and* AS *does not solve the r.r.p. for* x, *then there exists some* $r \in \mathcal{R}$ *such that* `fulfils(`r`,`$G(x)$`)` $\in Pl_{AS}(x)$, $r \notin Res(x)$ *and* $r \in Res'(x)$.

The lemma above follows trivially from Definition 3.4. The assumption in the lemma above makes sense for the kind of settings considered in this thesis, where plans are common knowledge and do not change.

Lastly, note that the agent system AS' found in Example 3.2 is such that all agents are happy and is trivially a solution of the resource reallocation problem for the initial agent system AS. However, in general, it may not be the case that such an "ideal" solution can be found where all agents are happy.

3.4 Individual and Social Welfare

We can restate the problem of solving the r.r.p. of an agent system in terms of the utilitarian social welfare of the system (see Chapter 2, Section 2.1.4). In turn, this requires a notion of individual welfare (utility) for agents, as follows:

Definition 3.5 (individual welfare wrt an agent system) *Given an agent system* $AS =$

$(\mathcal{G}, \mathcal{R}, \mathcal{A})$ and $\langle x,\ Res(x),\ G(x),\ B(x)\rangle \in \mathcal{A}$, let

$$u_{AS}(x) = \begin{cases} 1 & \textit{if } AS \textit{ solves the r.r.p. for } x \\ 0 & \textit{otherwise.} \end{cases}$$

$u_{AS}(x)$ is x's utility function *in AS and gives a measure of x's individual welfare wrt AS.*

The definition of individual welfare (utility) above is a specialisation of the class of utility functions described in Chapter 2, Section 2.1.3. Notably, the presentation of an agent's utility function in the definition above and here on after (i.e. $u_{AS}(x)$) is different from that of Section 2.1.3 where an agent's utility function was given as $u_i(P)$ (and $u_i(P(i))$) for some agent i and *allocation* P (*resource bundle* $P(i)$ respectively). The allocation Res (and resource bundle $Res(x)$ of an agent x) contained in an agent system AS is implicit in our notation. Note also that the utility an agent assigns to a set of resources depends only on those resources and not on the resources assigned to other agents or any other *externalities*. We can write $happy_{AS}(AS')$, given in Definition 3.4 to represent the set of agents that have solved their r.r.p. in an agent system AS' wrt to an initial agent system AS, in terms of the agents' individual welfare, as follows:

- Given agent systems AS and AS', $happy_{AS}(AS') = \{\ x \mid x \in names(AS),\ u_{AS'}(x) = 1\ \}$

We adopt a utilitarian view similar to that described in Chapter 2, Section 2.1.4 in defining the social welfare of an agent society, i.e. we use the sum of the utilities that the agents in that society ascribe to the sets of resources they hold.

Definition 3.6 (utilitarian social welfare wrt an agent system) *Given agent systems AS and AS', the* utilitarian social welfare (usw) of AS' wrt AS *is given by* $sw_{AS}(AS') = \Sigma_{x \in names(AS)} u_{AS'}(x)$.

Note that in the case where $names(AS) = names(AS')$, as is the case of interest in this thesis, $sw_{AS}(AS') = sw_{AS'}(AS')$. We say that an agent system AS' maximises the usw of an initial

system AS iff all agents in AS' are at least as happy as in AS and there is no other agent system AS'' with greater usw than AS' and such that all agents in AS'' are at least as happy as in AS:

Definition 3.7 (maximising usw of an agent system) *Given AS, AS' agent systems, AS'* maximises the usw of *AS if and only if*

1. $u_{AS'}(x) \geq u_{AS}(x)$ *for all* $x \in names(AS)$*; and*
2. *there exists no AS'' such that $u_{AS''}(x) \geq u_{AS}(x)$ for all $x \in names(AS)$ and $sw_{AS}(AS'') > sw_{AS}(AS')$.*

Note that the definition above differs from the definition of optimal (maximal) usw given in Chapter 2 (Section 2.1.4) where the maximal usw measure is defined independently of the initial allocation. This is illustrated in the following example:

Example 3.3 (maximising usw of an agent system versus maximal usw) *Let us consider an agent system AS consisting of three agents $ag1$, $ag2$ and $ag3$, two resources $r1$ and $r2$ and two possible goals $g1$ and $g2$, such that all agents know of one another, know of plans for $r1$ to fulfil $g1$ and $r2$ to fulfil $g2$ as depicted in Figure 3.2, agents have goals as follows:*

$G(ag1) = g1$	$G(ag2) = g1$	$G(ag3) = g2$

and AS contains allocation Res as follows:

$Res(ag1) = \{r1\}$	$Res(ag2) = \{r2\}$	$Res(ag3) = \emptyset$

Only one of the agents ($ag1$) has its r.r.p. solved in this setup. Also, there are two agent systems with maximal usw as given in Chapter 2 (Section 2.1.4): one, AS_1, containing allocation Res_1 as follows:



Figure 3.2: Plans in Example 3.3.

$Res_1(ag1) = \{r1\}$	$Res_1(ag2) = \emptyset$	$Res_1(ag3) = \{r2\}$

and the other, AS_2, containing allocation Res_2 as follows:

$Res_2(ag1) = \emptyset$	$Res_2(ag2) = \{r1\}$	$Res_2(ag3) = \{r2\}$

Indeed, $sw_{AS}(AS_i) > sw_{AS}(AS')$ for any agent system AS' such that $AS' \neq AS_i$ ($i = 1, 2$). However, there is only one agent system that maximises the usw of the initial agent system AS, i.e. the agent system AS_1 wherein $ag1$ has $r1$ and $ag3$ has $r2$. This is because $r1$ solves the r.r.p. of $ag1$ already and hence $ag1$ would not give it away (as required in AS_2) except if receiving another resource that solves its r.r.p.

Note that an agent system that maximises the usw of another contains an allocation that is a "Pareto optimal" allocation (as defined in Chapter 2, Section 2.1.4), and vice versa. We will prove this in Chapter 4 (Theorem 4.2) after introducing the kinds of *deals* which we consider in this thesis. We have the following result also:

Lemma 3.3 (solving the r.r.p. for an agent system is to maximise its usw) *Given agent systems AS and AS', AS' solves the r.r.p. for AS if and only if AS' maximises the usw of AS.*

The lemma above is an immediate consequence of Definitions 3.4 and 3.7.

Whenever there are sufficiently many resources available in an agent system to fulfil all goals of all agents in the system, a solution to the r.r.p. of the system exists that is a solution for each agent in the system (i.e. with usw of $|\mathcal{A}|$, the set of agents in the system). The necessary conditions for this are as follows:

Lemma 3.4 (necessary conditions for solving the r.r.p. of all agents) *Given an agent system $AS = (\mathcal{G}, \mathcal{R}, \mathcal{A})$, if an agent system AS' solves the r.r.p. for* all *agents wrt AS (such that $sw_{AS}(AS') = |\mathcal{A}|$), then:*

1. $|\mathcal{A}| \leq |\mathcal{R}|$; *and*
2. *for all $\langle x, Res(x), G(x), B(x)\rangle \in \mathcal{A}$, there exists a plan* `fulfils(`r`,`$G(x)$`)` $\in Pl_{AS}$ *such that $r \in \mathcal{R}$; and*
3. *there exists no agent $\langle x, Res(x), G(x), B(x)\rangle \in \mathcal{A}$ such that, for each* `fulfils(`r`,`$G(x)$`)` $\in Pl_{AS}$, *there exists an agent $\langle x'$, $Res(x')$, $G(x')$, $B(x')\rangle \in \mathcal{A}$ such that* `fulfils(`r`,`$G(x')$`)` $\in Pl_{AS}$ *and there exists no* `fulfils(`r'`,`$G(x')$`)` $\in Pl_{AS}$ *for $r' \neq r$.*

The first condition in the lemma above says that there are at least as many resources as agents in the agent system. The second condition amounts to saying that, for each agent, there is at least one resource in the system that can fulfil its goal. The third condition says that there are no two agents needing the same resource in a way that both agents are dependent on that resource for fulfilling their goal. Note that these are necessary, not sufficient, conditions for the existence of a solution to the r.r.p. for the agent sytem. This is illustrated by the following example:

Example 3.4 (necessary not sufficient conditions for solving the r.r.p. of all agents) *Let us consider an agent system AS consisting of three agents $ag1$, $ag2$ and $ag3$, three resources $r1$, $r2$ and $r3$ (owned by some agents) and one possible goal $g1$, such that all agents know of one another, know of plans for $r1$ to fulfil $g1$ and $r2$ to fulfil $g1$ as depicted in Figure 3.3, and all agents have goal $g1$. In this agent system:*

1. *there are at least as many resources as agents (first condition of Lemma 3.4);*
2. *for each agent, there is at least one resource in the system that can fulfil its goal (second condition of Lemma 3.4);*

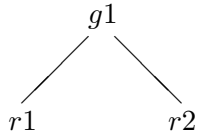


Figure 3.3: Plans in Example 3.4.

3. *there are no two agents needing the same resource in a way that both agents are dependent on that resource for fulfilling their goal (third condition of Lemma 3.4).*

The third condition holds in paricular since there is more than one way (i.e. resource by means of which) each agent can fulfil its goal. However, it is clear that any agent system (say AS'*) that maximises the usw of* AS *has* $sw_{AS}(AS') = 2$ *and it cannot be the case that* $sw_{AS}(AS') = |\mathcal{A}| = 3$*, since there are only two resources (*$r1$ *and* $r2$*) that are useful for solving the r.r.p. of the agents.*

3.5 Discussion

We have presented in this chapter a means of capturing the kinds of resource reallocation problems of interest to us in this thesis (as described in Chapter 1, Section 1.2). Given the goals and resources that make up a particular system, we show how agents in that system can be represented to understand the goals that they have, as well as the resources and also beliefs (about themselves and others) that they hold at the outset. We have demonstrated this in the case of a multi-agent medical booking system where the resources to be allocated are the appointments (bookings) and the goal of an agent would represent the date/time for when an appointment is required, or for the agent to be seen by a particular doctor, or other factors or even a combination of these factors. Although we have focused on small-sized systems in our examples involving just a few agents, this is only for purposes of illustration and our analysis is intended on the contrary to be applied to large scale systems involving hundreds of agents at the least. Moreover, the means of capturing agent systems which we present can be easily applied to other resource reallocation problems and is not limited to this particular application domain. For example, in the case of a grid computing system, the resources to be distributed (allocated)

would be various amounts of computational power held/controlled by various institutions and required for performing calculations (or some other computing intensive tasks), and the goal of each agent would represent the amount of computational power required for the task assigned to it, which could be fulfilled by a choice of some of the resources. Based on such a representation of an agent system (i.e. the goals, resources and agents involved), we have discussed in this chapter what it then means for agents to fulfil their goals and also what it means for the agent system to be solved, i.e. to make best use of the resources available between the agents given their individual goals and the resources and beliefs that they hold at the outset.

3.6 Conclusion

In this chapter we have set out the key components of our problem domain amounting to the notion of agent system, resource reallocation problem, and individual and social welfare. We have defined an agent system in terms of the agents, goals and resources comprising the system, and an agent in terms of its name, resources held, goal and beliefs. Indeed our formulation of an agent system paves the way to a distributed solution of the reallocation problem as is the focus in this thesis. An allocation is defined as a partitioning of the set of resources amongst the agents. We have discussed what it means for the resource reallocation problem to be solved for an agent as well as for an agent system. Namely, that an agent (as many agents as possible, respectively) have a resource that *fulfils* its goal. We have introduced a utility function for computing the individual welfare of an agent, and a utilitarian social welfare measure based on the utility function. Using this utility function and social welfare measure we have discussed what it means for an agent system to maximise the utilitarian social welfare of another initial agent system, and show that this amounts to solving (optmising) the resource reallocation problem for the initial system. Finally, we have presented an example application to demonstrate the concepts introduced in this chapter and to be referred back to in later chapters for demonstrating the various concepts which we cover in this thesis.

Chapter 4

Minimally Cooperative Rational Deals for the Resource Reallocation Problem

We define in this chapter the kinds of deals sufficient and necessary for solving the class of resource reallocation problems as defined in Chapter 3. We prove that any sequence of these deals, which we term "minimally cooperative rational" deals, leads to an optimal allocation. Moreover, we prove that whenever there exists a deal, of any kind, resulting in an optimal allocation, then there exists also a sequence of the deals we define which results in an optimal allocation. This chapter is structured as follows: In Section 4.1 we define deals that transition an agent system from one allocation to another, and we define a particular kind of deal which we term "cooperatively rational". In Section 4.2 we define *minimally cooperative rational* deals. In Section 4.3 we prove that any sequence of minimally cooperative rational deals leads to an allocation that maximises the utilitarian social welfare (usw) of an agent system as defined in Chapter 3. In Section 4.4 we break down and discuss characteristics of minimally cooperative rational deals. In Section 4.5 we prove completeness of minimally cooperative rational deals for finding an allocation that maximises the usw of an agent system. Lastly, in Sections 4.6 and 4.7, we discuss related work and conclude. In the definitions and lemmas that follow, we assume an agent system $AS = (\mathcal{G}, \mathcal{R}, \mathcal{A})$ (as in Definition 3.1) containing an allocation Res (see Lemma 3.1). We will assume also that all agent systems considered will have the same

agent names ($names(AS)$), goals ($\mathcal{G}$) and resources ($\mathcal{R}$) as in AS.

4.1 Deals and Cooperative Rationality

A *deal*, the set of agents *involved* in a deal, a *composition* of two deals and a *sequence* of three or more deals are defined, similarly as for the definitions of Endriss et al [EMST06] and Dunne and Chevaleyre [DC08] (see Chapter 2, Section 2.1.6, Definitions 2.2–2.5 respectively), as follows:

- A *deal* is a pair $\delta = (Res, Res')$ where Res and Res' are allocations contained in agent systems AS and AS' respectively and $Res \neq Res'$;
- the set of agents *involved* in a deal $\delta = (Res, Res')$ is given by $Agents^{\delta} = \{x \in names(AS) \mid Res(x) \neq Res'(x)\}$. Note that $Agents^{\delta}$ contains the names of the agents and not the agents themselves, and also that the symbol used here onwards differs slightly from Definition 2.3 where the symbol $\mathcal{A}^{\delta}$ was used;
- The *composition* of two deals $\delta_1 = (Res, Res')$ and $\delta_2 = (Res', Res'')$ is given by $\delta_1 \circ \delta_2 = (Res, Res'')$.
- A *sequence* of n deals $\delta_1 \circ \delta_2 \circ \ldots \circ \delta_n$ is given by $(\ldots((\delta_1 \circ \delta_2) \circ \delta_3) \circ \ldots) \circ \delta_n$.

A deal results in a new agent system, as follows:

Definition 4.1 (agent system resulting from a deal) *Given a deal $\delta = (Res, Res')$, let*

$AS' = (\mathcal{G}, \mathcal{R}, \mathcal{A}')$ *where*
$\mathcal{A}' = \{ \langle x, Res'(x), G(x), B'(x)\rangle \mid \langle x, Res(x), G(x), B(x)\rangle \in \mathcal{A} \}$

We say that the agent system AS' results *from δ. We also say that a deal $\delta = (Res, Res')$* takes us from *agent system AS to agent system AS', and that δ is* possible from *AS.*

Note that the agent system resulting from a deal $\delta = (Res, Res')$ contains allocation Res'. A deal can be seen as a set of *actions* comprising the deal (carried out by the agents involved), as follows:

Definition 4.2 (actions comprising a deal) *The set of* actions comprising a deal $\delta = (Res, Res')$ *is given by* $Actions^{\delta} = \{\texttt{give}(x,r,y) \mid r \in Res(x),\ r \in Res'(y),\ x \neq y\}$. *For each* $\texttt{give}(x,r,y) \in Actions^{\delta}$ *we say that* x gives away r *(to* y*) and* y *receives* r *(from* x*). If* $\texttt{give}(x,r,y) \in Actions^{\delta}$ *and there is no* $\texttt{give}(_,r,x) \in Actions^{\delta}$*, we say that* x simply gives away r *(receiving no resource in the deal).*

The actions comprising a deal capture the resources that change hands between agents in a deal, and the agents between which the resources change hands. If an agent increases utility following a deal, then it must have received, in that deal, a resource that solves its r.r.p., as follows:

Lemma 4.1 (necessary action in a deal) *If* $\delta = (Res, Res')$ *is a deal resulting in an agent system* AS' *and, for some agent* $x \in names(AS)$, $u_{AS}(x) = 0$ *and* $u_{AS'}(x) = 1$, *then* $\texttt{give}(y,r,x) \in Actions^{\delta}$ *for some agent* $y \in names(AS)$ $(y \neq x)$ *and resource* $r \in \mathcal{R}$ *that solves the r.r.p. of* x.

Proof. If $u_{AS}(x) = 0$ and $u_{AS'}(x) = 1$, then AS' solves the r.r.p. of x and AS does not. So, by Lemma 3.2, x must have a resource (say r) in Res', which it does not have in Res, and which solves its r.r.p., i.e. $r \notin Res(x)$ and $r \in Res'(x)$. This is only possible if $r \in Res(y)$ for some agent $y \in names(AS)$ $(y \neq x)$ and hence, by Definition 4.2, $\texttt{give}(x,r,y) \in Actions^{\delta}$. □

The types of deals we consider in this thesis are those wherein none of the agents involved end up worse off and at least one ends up better off. We define deals conforming to this criterion as a variant of the *cooperatively rational* deals given by Endriss et al. in [EMST06] (see Chapter 2, Section 2.1.6), adapted according to our notion of individual welfare (utility) as follows:

Definition 4.3 (cooperatively rational deal resulting in an agent system) *A deal* $\delta = (Res, Res')$ *resulting in an agent system* AS' *is called* cooperatively rational (CR) *iff* $u_{AS'}(x) \geq u_{AS}(x)$ *for all* $x \in names(AS)$ *and there exists* $y \in names(AS)$ *such that* $u_{AS'}(y) > u_{AS}(y)$.

CR deals require at least one agent to strictly increase in utility. We will see in Chapter 5 that, when we allow agents to propose deals for other agents to consider, the agent to increase utility in the deal will be the proposing agent. The social welfare of any agent system resulting from CR deals increases wrt the initial system, as shown below:

Lemma 4.2 (CR deal implies increase in social welfare) *If a deal* $\delta = (Res, Res')$ *resulting in agent system* AS' *is cooperatively rational, then* $sw_{AS}(AS') > sw_{AS}(AS)$.

The lemma above is an immediate consequence of Definitions 3.6 and 4.3. In general, as also discussed by Endriss et al. in [EMST06], it is not the case that there exists a CR deal $\delta = (Res, Res')$ giving AS' from AS whenever $sw_{AS}(AS') > sw_{AS}(AS)$. This is illustrated in the following example:

Example 4.1 (increase in social welfare does not always imply a CR deal) *Consider the agent systems* AS, AS_1 *and* AS_2 *of Example 3.3 containing allocations* Res, Res_1 *and* Res_2 *respectively. Then, the deal* $\delta_1 = (Res, Res_1)$ *resulting in* AS_1 *is a CR deal since* $u_{AS_1}(x) \geq u_{AS}(x)$ *for all* $x \in names(AS)$ *and* $u_{AS'}(ag3) > u_{AS}(ag3)$. *However, the deal* $\delta_2 = (Res, Res_2)$ *resulting in* AS_2 *is not a CR deal even though* $sw_{AS}(AS_2) = 2 > sw_{AS}(AS) = 1$. *This is because* $u_{AS_2}(ag1) < u_{AS}(ag1)$.

It is the case, however, that there exists a CR deal $\delta = (Res, Res')$ giving AS' from AS when all of the agents in AS have equal or better utility in AS' than in AS, as follows:

Lemma 4.3 (conditions for a CR deal) *Let* AS' *be an agent system that contains allocation* Res' *such that* $sw_{AS}(AS') > sw_{AS}(AS)$ *and* $u_{AS'}(x) \geq u_{AS}(x)$ *for all* $x \in names(AS)$. *Then* $\delta = (Res, Res')$ *is a CR deal.*

Proof. If $sw_{AS}(AS') > sw_{AS}(AS)$ and $u_{AS'}(x) \geq u_{AS}(x)$ for all $x \in names(AS)$, then, by the definition of usw (Definition 3.6), there exists an agent $y \in names(AS)$ such that $u_{AS'}(y) > u_{AS}(y)$. Hence, by the definition of CR deals (Definition 4.3), $\delta = (Res, Res')$ is a CR deal. □

In addition, we show that a sequence of CR deals is itself a CR deal, as follows:

Theorem 4.1 (sequence of CR deals is a CR deal) *If $\delta = \delta_1 \circ \ldots \circ \delta_n = (Res, Res_n)$ where each δ_i $(1 \leq i \leq n)$ is a CR deal, then δ is a CR deal.*

Proof. After every CR deal δ_i $(1 \leq i \leq n)$, some agents increase in utility and no agents decrease in utility. Then, the accumulation of all such deals is also a CR deal, since overall some agents have increased in utility and no agents have decreased in utility. □

Lastly, we prove (Theorem 4.2 below) that an agent system which maximises the usw of another (see Definition 3.7) contains a Pareto optimal allocation, defined by adapting to our problem domain the definition found in [HP02, CDE+06] (see Chapter 2, Section 2.1.4), as follows:

- An allocation *Res* contained in an agent system *AS* is *Pareto optimal* if there is no allocation *Res′* contained in an agent system *AS′* such that
  - $u_{AS'}(x) \geq u_{AS}(x)$ for all $x \in names(AS)$; and
  - $u_{AS'}(x) > u_{AS}(x)$ for some $x \in names(AS)$.

  We also say that such *AS* is Pareto optimal.

It is easy to see that in our problem domain, Pareto optimal agents systems and CR deals are linked, as follows:

Lemma 4.4 (agent system containing a Pareto optimal allocation) *An agent system AS contains a Pareto optimal allocation Res iff there exists no CR deal possible from AS.*

The lemma above is a consequence of Definition 4.3 and the adapted definition of Pareto optimality immediately above.

Theorem 4.2 (maximising usw amounts to Pareto optimal allocations) *Given a CR deal $\delta = (Res, Res')$ resulting in an agent system AS', then AS' maximises the usw of AS iff AS' is Pareto optimal.*

Proof. It is easy to see, by Definition 3.7, that if AS' maximises the usw of AS, then there are no CR deals possible from AS', and hence, by Lemma 4.4, AS' is Pareto optimal. It remains to show that, if AS' is Pareto optimal, then AS' maximises the usw of AS. Suppose, for the sake of contradiction, that AS' does not maximise the usw of AS. Then, since δ is a CR deal, the first condition of Definition 3.7 does indeed hold (i.e. $u_{AS'}(x) \geq u_{AS}(x)$ for all $x \in names(AS)$) and it must be the case that the second condition of Definition 3.7 does not hold, i.e. there is some agent system AS'' such that $u_{AS''}(x) \geq u_{AS}(x)$ for all $x \in names(AS)$ and $sw_{AS}(AS'') > sw_{AS}(AS')$. Let Res', Res'' be the allocations contained in AS', AS'' respectively. There are two cases to consider:

1. $happy_{AS}(AS') \subset happy_{AS}(AS'')$: trivially this implies that $u_{AS''}(x) \geq u_{AS'}(x)$ for all $x \in names(AS)$ and $u_{AS''}(y) > u_{AS'}(y)$ for some $y \in names(AS)$, and thus there is a CR deal $\delta' = (Res', Res'')$. But this cannot be, by Lemma 4.4, since Res' is Pareto optimal.

2. $happy_{AS}(AS') \not\subset happy_{AS}(AS'')$: together with "$u_{AS''}(x) \geq u_{AS}(x)$ for all $x \in names(AS)$ and $sw_{AS}(AS'') > sw_{AS}(AS')$" as above, this implies that there is some agent $x \in names(AS)$ whose r.r.p. is not solved in AS' but is solved in AS'', and thus some resource $r \in \mathcal{R}$ such that r solves the r.r.p. of x, $r \in Res''(x)$, $r \in Res'(y)$ ($y \in names(AS)$, $y \neq x$), and

   - r does not solve the r.r.p. of y; or
   - y has some other resource in AS' as well as r that solves its r.r.p.

   But then there is a CR deal δ' possible from AS' where $Actions^{\delta'} = \{\texttt{give(}y\texttt{,}r\texttt{,}x\texttt{)}\}$. But this cannot be, by Lemma 4.4, since Res' is Pareto optimal. □

The second case of the proof of Theorem 4.2 above captures a specific case of how a CR deal which does not maximise the usw of an initial agent system is not prohibitive of a successive

CR deal which does. This is demonstrated in the following example and it will be proven more generally in Section 4.3 (Theorem 4.3) that a CR deal which does not maximise the usw of an agent system is never prohibitive of a successive CR deal which does.

Example 4.2 (existence of CR deal when usw not maximised) *Suppose we have agent systems AS, AS' and AS'' such that:*

- $happy_{AS}(AS) = \emptyset$;
- $happy_{AS}(AS') = \{x_1\}$;
- $happy_{AS}(AS'') = \{x_2, x_3\}$.

Trivially, $happy_{AS}(AS') \not\subset happy_{AS}(AS'')$, $u_{AS''}(x) \geq u_{AS}(x)$ for all $x \in names(AS)$, and $sw_{AS}(AS'') > sw_{AS}(AS')$. Necessarily, also, x_1 has some resource r_1 in $Res'(x_1)$ which solves its r.r.p., and z (one of x_2 or x_3) has some resource r_2 ($r_2 \neq r_1$) in $Res''(z)$ which solves its r.r.p. Then, there is an agent y in AS' that has r_2 and either

- *$y = x_1$ (and x_1 does not "need" r_2 in AS' since its r.r.p. is solved by r_1); or*
- *$y \neq x_1$ (and r_2 does not solve the r.r.p. of y, since otherwise the r.r.p. of y would be solved in AS').*

Either way, the deal $\delta' = (Res', Res'')$ where $Actions^{\delta'} = \{give(y, r_2, z)\}$ is a CR deal.

Whilst CR deals are useful in that no agents end up worse off, a CR deal could contain useless exchanges of resources that do not affect the utilities of the agents involved. We introduce next a specific kind of CR deal for which this is not the case.

4.2 Minimally Cooperative Rational Deals

We define here a more refined kind of cooperatively rational (CR) deal that contains only exchanges of resources that are meaningful in that they contribute to agent(s) increasing utility

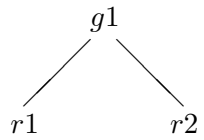


Figure 4.1: Plans in Example 4.3 (and as in Example 3.4).

in the deal.

Definition 4.4 (minimally cooperative rational deals) *A deal $\delta = (Res, Res')$ is called* minimally cooperative rational *(Min CR) iff δ is cooperatively rational and there is no CR deal $\delta' = (Res, Res'')$ such that $Actions^{\delta'} \subset Actions^{\delta}$.*

We illustrate next a Min CR deal.

Example 4.3 (Min CR deal) *Given the agent system AS as in Example 3.4 with plans as depicted in Figure 4.1 (and Figure 3.3), and agents' goals and resources as follows:*

$G(ag1) = g1$	$G(ag2) = g1$	$G(ag3) = g1$
$Res(ag1) = \{r1, r2, r3\}$	$Res(ag2) = \emptyset$	$Res(ag3) = \emptyset$

then, the deal $\delta = (Res, Res')$ where $Actions^{\delta} = \{$`give(`$ag1$`,`$r2$`,`$ag2$`)`, `give(`$ag1$`,`$r3$`,`$ag2$`)`$\}$ is a CR deal. This is because $ag2$ increases in utility and $ag1$ does not decrease in utility. However, δ is not a Min CR deal. This is because there exists a deal $\delta' = (Res, Res'')$ where $Actions^{\delta'}$ = $\{$`give(`$ag1$`,`$r2$`,`$ag2$`)`$\}$, $Actions^{\delta'} \subset Actions^{\delta}$ and δ' is itself a CR deal. This δ' is a Min CR deal.

In what follows, we say that a CR deal δ *contains* a CR deal δ' if $Actions^{\delta'} \subseteq Actions^{\delta}$. We show that every CR deal *contains* a Min CR deal, as follows:

Lemma 4.5 (CR deals contain Min CR deals) *If $\delta' = (Res, Res')$ is a CR deal, then there is a Min CR deal $\delta'' = (Res, Res'')$ such that δ' contains δ''.*

Proof. If δ' is itself a Min CR deal, then this is trivial. On the other hand, if δ' is not a Min CR deal, then, by Definition 4.4, there is a CR deal δ_1 such that $Actions^{\delta_1} \subset Actions^{\delta'}$. If δ_1 is a Min CR deal, we are done. On the other hand, if δ_1 is not a Min CR deal, then, repeatedly by applying Definition 4.4, there is a CR deal δ_{i+1} ($i \geq 1$) such that $Actions^{\delta_{i+1}} \subset Actions^{\delta_i}$, until δ_n is reached where δ_n is a Min CR deal. Indeed, δ_n exists because the actions comprising deals are finite sets. Since $Actions^{\delta_n} \subset \ldots \subset Actions^{\delta_1} \subset Actions^{\delta'}$, then, trivially, $Actions^{\delta_n} \subseteq Actions^{\delta'}$. □

We also have that a sequence of Min CR deals is itself a CR deal, as follows:

Corollary 4.1 (Sequence of Min CR deals is a CR deal) *If $\delta = \delta_1 \circ \ldots \circ \delta_n = (Res, Res_n)$ where each δ_i ($1 \leq i \leq n$) is a Min CR deal, then δ is a CR deal.*

Proof. This is an immediate consequence of Theorem 4.1, since all Min CR deals are cooperatively rational. □

This corollary will be used to prove results in Section 4.3. Note that it is not necessarily the case that δ, a sequence of a number of Min CR deals, is itself minimally cooperative rational. To see this suppose we have $\delta = \delta_1 \circ \delta_2$ where $Actions^{\delta_1} = \{\texttt{give}(x, r_1, y)\}$, $Actions^{\delta_2} = \{\texttt{give}(y, r_2, x)\}$ and both δ_1 and δ_2 are Min CR deals. Then, $r_1 \neq r_2$, $Actions^{\delta} = \{\texttt{give}(x, r_1, y), \texttt{give}(y, r_2, x)\}$ but δ is not a *minimally* cooperative rational deal since δ_1 is a CR deal and $Actions^{\delta_1} \subset Actions^{\delta}$.

We use the properties of CR and Min CR deals proven in this section and the previous section to prove next that Min CR deals are "sufficient" for maximising usw of agent systems.

4.3 Maximising Utilitarian Social Welfare

We prove in this section that Min CR deals are "sufficient" for maximising the utilitarian social welfare (usw) of agent systems. More so, we prove that *any* sequence of Min CR deals leads to

an allocation with maximal usw wrt an initial agent system. We establish firstly termination for the class of CR deals, as follows:

Lemma 4.6 (Termination) *There can be no infinite sequence of CR deals.*

Proof. Given that both the set of agents $\mathcal{A}$ as well as the set of resources $\mathcal{R}$ are required to be finite, there can only be a finite number of distinct allocations of resources. Furthermore, by Lemma 4.2, any CR deal will strictly increase usw. Hence, after a finite number of CR deals no further increase of usw will be possible and thus no further CR deals can take place. □

Using this termination result, we show that the class of CR deals is sufficient to guarantee an allocation that maximises the usw of any given agent system, as follows:

Theorem 4.3 (CR deals maximise usw of agent systems) *Any sequence of CR deals will eventually result in a resource allocation (agent system) that maximises the usw of AS.*

Proof. Given an initial agent system AS with allocation Res, then, by Lemma 4.6, any sequence of CR deals must terminate. Let the terminal agent system be AS' containing allocation Res'. Note that, by Theorem 4.1, $\delta = (Res, Res')$ resulting in AS' is a CR deal. By Theorem 4.2, it suffices to prove that AS' is Pareto optimal. For the sake of contradiction, assume AS' is not Pareto optimal. Then, there exists another agent system AS'' with allocation Res'' such that $u_{AS''}(x) \geq u_{AS'}(x)$ for all $x \in names(AS)$ and $u_{AS''}(y) > u_{AS'}(y)$ for some $y \in names(AS)$. But then, the deal $\delta' = (Res', Res'')$ is cooperatively rational, which contradicts our earlier assumption of Res' being a terminal allocation. Given that δ is a CR deal and AS' is Pareto optimal, then, by Theorem 4.2, AS' maximises the usw of AS. □

Theorem 4.3 above is an adaptation of Theorem 4 presented by Endriss et al. [EMST06]. Similarly as for Theorem 4 of [EMST06], the true power of Theorem 4.3 above is in the fine print: *any* sequence of CR deals will result in an allocation that maximises the usw of the initial system. That is, whatever deals are agreed on in the early stages of negotiation, the system will never get stuck in a local optimum and finding an allocation that maximises social

welfare remains an option throughout. We prove next that the result above also holds for Min CR deals, as follows:

Theorem 4.4 (Min CR deals maximise usw of agent systems) *Any sequence of Min CR deals will eventually result in a resource allocation (agent system) that maximises the usw of AS.*

Proof. Given an initial agent system AS containing allocation Res and the fact that every Min CR deal is a CR deal, then, by Lemma 4.6, any sequence of Min CR deals must terminate. Let the terminal agent system be AS' containing allocation Res'. Note that, by Corollary 4.1, $\delta = (Res, Res')$ resulting in AS' is a CR deal. By Theorem 4.2, it suffices to prove that AS' is Pareto optimal. For the sake of contradiction, assume AS' is not Pareto optimal. Then, there exists another agent system AS'' with allocation Res'' such that $u_{AS''}(x) \geq u_{AS'}(x)$ for all $x \in names(AS)$ and $u_{AS''}(y) > u_{AS'}(y)$ for some $y \in names(AS)$. But then, the deal $\delta' = (Res', Res'')$ is cooperatively rational, and, by Lemma 4.5, there is a Min CR deal possible from Res' (i.e. AS'). However, this contradicts our earlier assumption of Res' being a terminal allocation. Given that δ is a CR deal and AS' is Pareto optimal, then, by Theorem 4.2, AS' maximises the usw of AS. □

This last theorem above will be useful in Chapters 5 and 7 when defining a generative agent negotiation framework that is able to identify and propose Min CR deals, which would solve the r.r.p. of the proposing agent and lead towards an allocation that maximises the usw of the agent system. We focus next on breaking down and understanding the actions that make up Min CR deals.

4.4 Properties of Min CR deals

We characterise in the following three theorems the actions comprising any Min CR deal, and the agents and resources involved, depending on the number of actions comprising the deal — one, two or more. These theorems break down Min CR deals in such a way that they

can be captured by our generative agent negotiation framework to be defined in Chapters 5 and 7. Firstly, we consider Min CR deals comprising one action (of the kind illustrated in Example 4.3):

Theorem 4.5 (Min CR deal comprising one action) *If $\delta = (Res, Res')$ is a Min CR deal resulting in an agent system AS' such that $|Actions^{\delta}| = 1$, then $Actions^{\delta} = \{\texttt{give}(y,r,x)\}$ for some $x, y \in names(AS)$ $(x \neq y)$ and $r \in \mathcal{R}$, where:*

(1) $x \notin happy_{AS}(AS)$ and $x \in happy_{AS}(AS')$; and

(2) $y \in happy_{AS}(AS')$ iff $y \in happy_{AS}(AS)$.

Proof. Proving (1) and (2) amounts to proving $u_{AS}(x) = 0$, r solves the r.r.p. of x (and thus $u_{AS'}(x) = 1$), and one of the following conditions hold:

- $u_{AS}(y) = 0$ and $u_{AS'}(y) = 0$; or
- $u_{AS}(y) = 1$ and $u_{AS'}(y) = 1$.

By Definition of a CR deal, there is at least one agent that has its r.r.p. solved in AS' but not in AS, i.e. $u_{AS}(x) = 0$ and $u_{AS'}(x) = 1$ for some $x \in names(AS)$. This is only possible if x has a resource $r \in \mathcal{R}$ in AS' that solves its r.r.p., which it does not have in AS. Since $|Actions^{\delta}| = 1$, it must then be the case that $Actions^{\delta} = \{\texttt{give}(y,r,x)\}$ for some $y \in names(AS)$. Since utilities are dichotomous, then either $u_{AS}(y) = 0$ or $u_{AS}(y) = 1$. In the case $u_{AS}(y) = 0$: trivially $u_{AS'}(y) = 0$ since y receives no resource as part of δ. In the case $u_{AS}(y) = 1$: assume, for the sake of contradiction, that the r.r.p. of y is unsolved by y simply giving away r, i.e. $u_{AS'}(y) = 0$. But then $u_{AS'}(y) < u_{AS}(y)$, which cannot be, since δ is a CR deal. Hence, it must be the case that the r.r.p. of y is still solved even after simply giving away r, i.e. $u_{AS'}(y) = 1$. □

We characterise next a Min CR deal comprising two actions.

Theorem 4.6 (Min CR deal comprising two actions) *If $\delta = (Res, Res')$ is a Min CR deal resulting in an agent system AS' such that $|Actions^{\delta}| = 2$, then $Actions^{\delta} = \{$*`give(`y`,`r`,`x`)`*,* `give(`z_1`,`r_1`,`y`)`*$\}$ for some $x, y, z_1 \in names(AS)$ ($x \neq y$, $y \neq z_1$, x and z_1 possibly the same) and $r, r_1 \in \mathcal{R}$ ($r \neq r_1$), where:*

(1) $x \notin happy_{AS}(AS)$ and $x \in happy_{AS}(AS')$; and

(2) $y \in happy_{AS}(AS)$ and $y \in happy_{AS}(AS')$, and
the r.r.p. of y is unsolved by y simply giving away r; and

(3) if $z_1 \neq x$, then $z_1 \in happy_{AS}(AS')$ iff $z_1 \in happy_{AS}(AS)$.

Proof. Proving (1), (2) and (3) is equivalent to proving $u_{AS}(x) = 0$, r solves the r.r.p. of x (i.e. $u_{AS'}(x) = 1$), $u_{AS}(y) = 1$, the r.r.p. of y is unsolved by y simply giving away r, r_1 solves the r.r.p. of y (and thus $u_{AS'}(y) = 1$), and one of the following conditions holds:

- $u_{AS}(z_1) = 0$ and $u_{AS'}(z_1) = \begin{cases} 0 & \text{if } z_1 \neq x, \\ 1 & \text{if } z_1 = x; \end{cases}$ or
- $u_{AS}(z_1) = 1$ and $u_{AS'}(z_1) = 1$.

By Definition of a CR deal, there is at least one agent that has its r.r.p. solved in AS' but not in AS, i.e. $u_{AS}(x) = 0$ and $u_{AS'}(x) = 1$ for some $x \in names(AS)$. This is only possible if x has a resource $r \in \mathcal{R}$ in AS' that solves its r.r.p., which it does not have in AS, i.e. `give(`y`,`r`,`x`)` $\in Actions^{\delta}$ for some $y \in names(AS)$. Assume, for the sake of contradiction, that either

- $u_{AS}(y) = 0$; or
- $u_{AS}(y) = 1$ and the r.r.p. of y is still solved even after simply giving away r.

Then the deal comprising the set of actions {`give(`y`,`r`,`x`)`} would be cooperatively rational. But then, since `give(`y`,`r`,`x`)` $\in Actions^{\delta}$ and $|Actions^{\delta}| = 2$, {`give(`y`,`r`,`x`)`} $\subset Actions^{\delta}$ and δ cannot be *minimally* cooperative rational. Hence $u_{AS}(y) = 1$ and the r.r.p. of y is unsolved after

simply giving away r, which implies $\texttt{give}(z_1,r_1,y) \in Actions^{\delta}$ for some $z_1 \in names(AS)$ and resource $r_1 \in \mathcal{R}$ that solves the r.r.p. of y. We then have that $\{\texttt{give}(y,r,x), \texttt{give}(z_1,r_1,y)\} \subseteq Actions^{\delta}$. Together with $|Actions^{\delta}| = 2$, this implies $Actions^{\delta} = \{\texttt{give}(y,r,x),\texttt{give}(z_1,r_1,y)\}$. Since utilities are dichotomous, then either $u_{AS}(z_1) = 0$ or $u_{AS}(z_1) = 1$. Note that the latter case implies $z_1 \neq x$ since $u_{AS}(x) = 0$. In the case $u_{AS}(z_1) = 0$: either $z_1 = x$ and, trivially, $u_{AS'}(z_1) = 1$, or $z_1 \neq x$ and $u_{AS'}(z_1) = 0$ since z_1 receives no resource as part of δ. In the case $u_{AS}(z_1) = 1$: assume, for the sake of contradiction, that the r.r.p. of z_1 is unsolved by simply giving away r_1, and thus $u_{AS'}(z_1) = 0$. But then $u_{AS'}(z_1) < u_{AS}(z_1)$, which cannot be, since δ is a CR deal. Hence, it must be the case that the r.r.p. of z_1 is still solved even after simply giving away r_1, i.e. $u_{AS'}(z_1) = 1$. □

We illustrate Min CR deals comprising two actions in the following two examples:

Example 4.4 (Min CR deal comprising two actions, involving two agents) *Given the agent system AS as in Example 3.1 with plans as depicted in Figure 4.2 (and Figure 3.1), and agents' goals and resources as follows:*

$G(ag1) = g1$	$G(ag2) = g2$	$G(ag3) = g3$
$Res(ag1) = \{r1\}$	$Res(ag2) = \{r2\}$	$Res(ag3) = \{r3\}$

The r.r.p. of $ag1$ and $ag3$ is solved, but the r.r.p. of $ag2$ is not. Then, the deal $\delta = (Res, Res')$ where $Actions^{\delta} = \{\texttt{give}(ag2,r2,ag1), \texttt{give}(ag1,r1,ag2)\}$ results in the agent system AS' containing Res' as follows:

$Res'(ag1) = \{\underline{r2}\}$	$Res'(ag2) = \{\underline{r1}\}$	$Res'(ag3) = \{r3\}$

Here, δ is a Min CR deal comprising two actions. The underlined resources are the resources that have changed hands between agents. Note that $ag2$ increases in utility, $ag1$ does not decrease in utility, and there is no strict subset of $Actions^{\delta}$ for which this is possible. Indeed δ is the deal described in Example 3.2 resulting in a solution of the r.r.p. for all agents in AS (and hence AS itself).

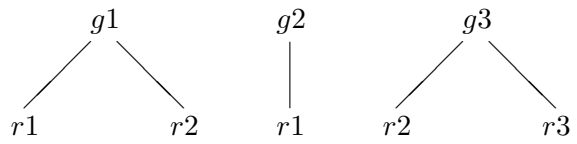


Figure 4.2: Plans in Examples 4.4–4.6 (and as in Example 3.1).

Example 4.5 (Min CR deal comprising two actions, involving three agents) *Given the agent system AS as in Example 3.1 and Example 4.4 with plans as depicted in Figure 4.2 (and Figure 3.1), and agents' goals as previously given but containing a resource allocation instead as follows:*

$Res(ag1) = \{r1\}$	$Res(ag2) = \emptyset$	$Res(ag3) = \{r2, r3\}$

The r.r.p. of $ag1$ and $ag3$ is solved, but the r.r.p. of $ag2$ is not. Then, the deal $\delta = (Res, Res')$ where $Actions^{\delta}$ = {`give`($ag1$,$r1$,$ag2$), `give`($ag3$,$r2$,$ag1$)} results in the agent system AS' containing Res' as follows:

$Res'(ag1) = \{\underline{r2}\}$	$Res'(ag2) = \{\underline{r1}\}$	$Res'(ag3) = \{r3\}$

Here, δ is a Min CR deal comprising two actions. The underlined resources are the resources that have changed hands between agents. Note that $ag2$ increases in utility, $ag1$ and $ag3$ do not decrease in utility, and there is no strict subset of $Actions^{\delta}$ for which this is possible.

We characterise next a Min CR deal comprising more than two actions.

Theorem 4.7 (Min CR deal comprising more than two actions) *If $\delta = (Res, Res')$ is a Min CR deal resulting in an agent system AS' such that $|Actions^{\delta}| = n + 2$ ($n \geq 1$), then $Actions^{\delta}$ = {`give`(y,r,x), `give`(z_1,r_1,y), ..., `give`(z_{n+1},r_{n+1},z_n)} for some $x, y, z_1, \ldots, z_{n+1} \in names(AS)$ ($x, y, z_1, \ldots, z_n$ all different, $z_n \neq z_{n+1}$ and one of $x, y, z_1, \ldots, z_{n-1}$ possibly the same as z_{n+1}) and $r, r_1, \ldots, r_{n+1} \in \mathcal{R}$ ($r, r_1, \ldots, r_{n+1}$ all different), where*

(1) $x \notin happy_{AS}(AS)$ and $x \in happy_{AS}(AS')$; and

(2.1) $y \in happy_{AS}(AS)$ *and* $y \in happy_{AS}(AS')$*, and*

the r.r.p. of y *is unsolved by* y *simply giving away* r*; and*

(2.2) for each z_i $(1 \leq i \leq n)$*:*

$z_i \in happy_{AS}(AS)$ *and* $z_i \in happy_{AS}(AS')$*, and*

the r.r.p. of z_i *is unsolved by* z_i *simply giving away* r_i*; and*

(3) if $z_{n+1} \neq x$*, then* $z_{n+1} \in happy_{AS}(AS')$ *iff* $z_{n+1} \in happy_{AS}(AS)$*.*

Proof. Proving (1), (2) and (3) is equivalent to proving the following:

- $u_{AS}(x) = 0$ and r solves the r.r.p. of x (such that $u_{AS'}(x) = 1$); and
- $u_{AS}(y) = 1$, the r.r.p. of y is unsolved by y simply giving away r, and r_1 solves the r.r.p. of y (and thus $u_{AS'}(y) = 1$); and
- for each z_i $(1 \leq i \leq n)$, $u_{AS}(z_i) = 1$, the r.r.p. of z_i is unsolved by z_i simply giving away r_i, and r_{i+1} solves the r.r.p. of z_i (and thus $u_{AS'}(z_i) = 1$); and
- one of the following conditions holds:
  - $u_{AS}(z_{n+1}) = 0$ and $u_{AS'}(z_{n+1}) = \begin{cases} 0 & \text{if } z_{n+1} \neq x, \\ 1 & \text{if } z_{n+1} = x; \end{cases}$ or
  - $u_{AS}(z_{n+1}) = 1$ and $u_{AS'}(z_{n+1}) = 1$.

By Definition of a CR deal, there is at least one agent that has its r.r.p. solved in AS' but not in AS, i.e. $u_{AS}(x) = 0$ and $u_{AS'}(x) = 1$ for some $x \in names(AS)$. This is only possible if x has a resource $r \in \mathcal{R}$ in AS' that solves its r.r.p., which it does not have in AS, i.e. `give(`y`,`r`,`x`)` $\in Actions^{\delta}$ for some $y \in names(AS)$. Assume, for the sake of contradiction, that either

- $u_{AS}(y) = 0$; or
- $u_{AS}(y) = 1$ and the r.r.p. of y is still solved even after simply giving away r.

Then the deal comprising the set of actions $\{\texttt{give}(y,r,x)\}$ would be cooperatively rational. But then, since $\texttt{give}(y,r,x) \in \mathit{Actions}^\delta$ and $|\mathit{Actions}^\delta| = n+2$ $(n \geq 1)$, $\{\texttt{give}(y,r,x)\} \subset \mathit{Actions}^\delta$ and δ cannot be *minimally* cooperative rational. Hence $u_{AS}(y) = 1$ and the r.r.p. of y is unsolved after simply giving away r, which implies $\texttt{give}(z_1,r_1,y) \in \mathit{Actions}^\delta$ for some $z_1 \in \mathit{names}(AS)$ and resource $r_1 \in \mathcal{R}$ that solves the r.r.p. of y. Assume, for the sake of contradiction, that either

- $u_{AS}(z_1) = 0$; or
- $u_{AS}(z_1) = 1$ and the r.r.p. of z_1 is still solved even after simply giving away r_1.

Then the deal compromising the set of actions $\{\texttt{give}(y,r,x), \texttt{give}(z_1,r_1,y)\}$ would be cooperatively rational. But then, since $\{\texttt{give}(y,r,x), \texttt{give}(z_1,r_1,y)\} \subseteq \mathit{Actions}^\delta$ and $|\mathit{Actions}^\delta| = n+2$ $(n \geq 1)$, $\{\texttt{give}(y,r,x), \texttt{give}(z_1,r_1,y)\} \subset \mathit{Actions}^\delta$ and δ cannot be *minimally* cooperative rational. Hence $u_{AS}(z_1) = 1$ and the r.r.p. of z_1 is unsolved after simply giving away r_1, which implies $\texttt{give}(z_2,r_2,z_1) \in \mathit{Actions}^\delta$ for some $z_2 \in \mathit{names}(AS)$ and resource $r_2 \in \mathcal{R}$ that solves the r.r.p. of z_2. Likewise, for all $1 < i \leq n$, we have that $u_{AS}(z_i) = 1$, $z_i \notin \{x, y, z_1, \ldots, z_{i-1}\}$ (since otherwise z_i will receive a resource that solves its r.r.p. in the deal at that ith point) and the r.r.p. of z_i is unsolved by z_i simply giving away r_i, which implies $\texttt{give}(z_{i+1},r_{i+1},z_i) \in \mathit{Actions}^\delta$ for some $r_{i+1} \in \mathcal{R}$ that solves the r.r.p. of z_i (and thus $u_{AS'}(z_i) = 1$). We then have that $\{\texttt{give}(y,r,x), \texttt{give}(z_1,r_1,y), \ldots, \texttt{give}(z_{n+1},r_{n+1},z_n)\} \subseteq \mathit{Actions}^\delta$. Together with $|\mathit{Actions}^\delta| = n+2$, this implies $\{\texttt{give}(y,r,x), \texttt{give}(z_1,r_1,y), \ldots, \texttt{give}(z_{n+1},r_{n+1},z_n)\} = \mathit{Actions}^\delta$. Since utilities are dichotomous, then either $u_{AS}(z_{n+1}) = 0$ or $u_{AS}(z_{n+1}) = 1$. Note that the former case ($u_{AS}(z_{n+1}) = 0$) implies $z_{n+1} \notin \{y, z_1, \ldots, z_{n-1}\}$ since $u_{AS}(z) = 1$ for each $z \in \{y, z_1, \ldots, z_{n-1}\}$, and the latter case ($u_{AS}(z_{n+1}) = 1$) implies $z_{n+1} \neq x$ since $u_{AS}(x) = 0$. (Trivially, $z_{n+1} \neq z_n$ since $\texttt{give}(z_{n+1},r_{n+1},z_n) \in \mathit{Actions}^\delta$.) In the case $u_{AS}(z_{n+1}) = 0$: either $z_{n+1} = x$ and, trivially, $u_{AS'}(z_{n+1}) = 1$, or $z_{n+1} \neq x$ and $u_{AS'}(z_{n+1}) = 0$ since z_{n+1} receives no resource as part of δ. In the case $u_{AS}(z_{n+1}) = 1$ and $z_{n+1} \in \{y, z_1, \ldots, z_{n-1}\}$: trivially $u_{AS'}(z_{n+1}) = 1$ since z_{n+1} receives a resource as part of δ which solves its r.r.p. In the case $u_{AS}(z_{n+1}) = 1$ and $z_{n+1} \notin \{y, z_1, \ldots, z_{n-1}\}$: assume, for the sake of contradiction, that the r.r.p. of z_{n+1} is unsolved by z_{n+1} simply giving away r_{n+1}, and

thus $u_{AS'}(z_{n+1}) = 0$. But then $u_{AS'}(z_{n+1}) < u_{AS}(z_{n+1})$, which cannot be, since δ is a CR deal. Hence, it must be the case that the r.r.p. of z_{n+1} is still solved even after simply giving away r_{n+1}, i.e. $u_{AS'}(z_{n+1}) = 1$. □

We illustrate a Min CR deal comprising more than two actions in the following example:

Example 4.6 (Min CR deal comprising more than two actions) *Given the agent system AS as in Example 3.1 and Example 4.4 with plans as depicted in Figure 4.2 (and Figure 3.1), and agents' goals as previously given but containing a resource allocation instead as follows:*

$Res(ag1) = \{r1\}$	$Res(ag2) = \{r3\}$	$Res(ag3) = \{r2\}$

The r.r.p. of $ag1$ *and* $ag3$ *is solved, but the r.r.p. of* $ag2$ *is not. Then, the deal* $\delta = (Res, Res')$ *where* $Actions^{\delta} = \{$`give(`$ag2$`,`$r3$`,`$ag3$`)`, `give(`$ag3$`,`$r2$`,`$ag1$`)`, `give(`$ag1$`,`$r1$`,`$ag2$`)`$\}$ *results in the agent system* AS' *containing* Res' *as follows:*

$Res'(ag1) = \{\underline{r2}\}$	$Res'(ag2) = \{\underline{r1}\}$	$Res'(ag3) = \{\underline{r3}\}$

Here, δ *is a Min CR deal comprising more than two actions. The underlined resources are the resources that have changed hands between agents. Note that* $ag2$ *increases in utility,* $ag1$ *and* $ag3$ *do not decrease in utility, and there is no strict subset of* $Actions^{\delta}$ *for which this is possible.*

Having characterised the actions comprising any Min CR deal and the agents involved, we can determine a number of properties of Min CR deals. Firstly, it is possible to determine the number of resources that agents involved in a Min CR deal receive, as follows:

Corollary 4.2 (Number of resources agents involved in a Min CR deal receive) *For any Min CR deal* $\delta = (Res, Res')$ *resulting in an agent system* AS'*, one agent involved in* δ *receives zero or one resources and all other agents involved in* δ *receive exactly one resource each.*

Proof. This is an immediate consequence of Corollaries A.1, A.2 and A.3, which can be found in Appendix A. □

Secondly, as well as the number of resources that agents involved in a Min CR deal receive, it is possible to determine the number of resources agents give away, as follows:

Corollary 4.3 (Number of resources agents involved in a Min CR deal give away) *For any Min CR deal $\delta = (Res, Res')$ resulting in an agent system AS', one agent $x \in Agents^{\delta}$ gives away zero or one resources, one agent $y \in Agents^{\delta}$ gives away one or two resources and all other agents $z \in Agents^{\delta}$ $(z \neq x, z \neq y)$ give away exactly one resource each.*

Proof. This is an immediate consequence of Corollaries A.4, A.5 and A.6, which can be found in Appendix A. □

Thirdly, for any Min CR deal, it is always the case that only one agent increases in utility and none of the agents decrease in utility, as follows:

Corollary 4.4 (Changes in utility of agents involved) *For any Min CR deal $\delta = (Res, Res')$ resulting in an agent system AS', only one agent increases in utility and none of the agents decrease in utility, i.e. $sw_{AS}(AS') = sw_{AS}(AS) + 1$.*

Proof. This is an immediate consequence of Corollaries A.7, A.8 and A.9, which can be found in Appendix A. □

We discuss next the completeness of Min CR deals for maximising the usw of an agent system.

4.5 Completeness of Min CR deals

In this section we show how any CR deal can be decomposed as a sequence of Min CR deals, such that any CR deal which maximises the utilitarian social welfare (usw) of an agent system can be broken down to a sequence of Min CR deals which also maximise the usw of the agent

system. We do so by first proving three subsidiary lemmas and then we prove our main theorem. Firstly, we can know that a certain resource is held by a certain agent following a deal if some conditions are met, as follows:

Lemma 4.7 (Resource Held) *If $\delta' = (Res, Res')$ and $\delta'' = (Res, Res'')$ are deals such that $Actions^{\delta''} \subset Actions^{\delta'}$, $\texttt{give}(x,r,y) \in Actions^{\delta'}$ and $\texttt{give}(x,r,y) \notin Actions^{\delta''}$, then $r \in Res''(x)$.*

Proof. First of all, since $\texttt{give}(x,r,y) \in Actions^{\delta'}$, it is clear that $r \in Res(x)$, $r \in Res'(y)$ and it cannot be that $\texttt{give}(x,r,z) \in Actions^{\delta'}$ for any $z \in names(AS)$ $(z \neq y)$. Then, since $Actions^{\delta''} \subset Actions^{\delta'}$ and $\texttt{give}(x,r,y) \notin Actions^{\delta''}$, it cannot be that $\texttt{give}(x,r,z) \in Actions^{\delta''}$ for any $z \in names(AS)$. Hence $r \in Res''(x)$. □

The lemma above is useful in knowing which agent has a particular resource following a particular deal. We show for each agent x that increases utility in a CR deal δ, there is a Min CR deal δ' where δ *contains* δ' (i.e. $Actions^{\delta'} \subseteq Actions^{\delta}$) and x increases utility in δ', as follows:

Lemma 4.8 (CR deal contains Min CR deal where agent increases utility) *If $\delta' = (Res, Res')$ is a CR deal resulting in an agent system AS' wherein x increases utility (i.e. $u_{AS}(x) = 0$, $u_{AS'}(x) = 1$), then $\texttt{give}(y,r,x) \in Actions^{\delta'}$ for some $y \in names(AS)$ and $r \in \mathcal{R}$ that solves the r.r.p. of x, and there is a Min CR deal $\delta'' = (Res, Res'')$ such that $Actions^{\delta''} \subseteq Actions^{\delta'}$ and $\texttt{give}(y,r,x) \in Actions^{\delta''}$.*

Proof. Given that x increases utility in δ', then, by Lemma 4.1, $\texttt{give}(z_1,r_1,x) \in Actions^{\delta'}$ for some $z_1 \in names(AS)$ and $r_1 \in \mathcal{R}$ that solves the r.r.p. of x. If the utility of z_1 does not decrease by simply giving away r_1, then $\{\texttt{give}(z_1,r_1,x)\} \subseteq Actions^{\delta'}$ is a set of actions comprising a Min CR deal and we are done. Otherwise, there must be some $\texttt{give}(z_2,r_2,z_1) \in Actions^{\delta'}$ ($z_2 \in names(AS)$, $r_2 \in \mathcal{R}$ solves the r.r.p. of z_1), since otherwise δ' would not be a CR deal. If the utility of z_2 does not decrease in the deal $\{\texttt{give}(z_1,r_1,x), \texttt{give}(z_2,r_2,z_1)\} \subseteq Actions^{\delta'}$, then it is a Min CR and we are done. Otherwise, $z_2 \notin \{x, z_1\}$, z_2 decreases

utility by simply giving away r_2, and there must be some $\texttt{give}(z_3,r_3,z_2) \in Actions^{\delta'}$ ($z_3 \in names(AS)$, $r_3 \in \mathcal{R}$ solves the r.r.p. of z_2). Continuing in this manner for each z_i whose utility would decrease by simply giving away r_i, a set of actions $\{\texttt{give}(z_1,r_1,x), \texttt{give}(z_2,r_2,z_1), \ldots, \texttt{give}(z_{n+1},r_{n+1},z_n)\} \subseteq Actions^{\delta'}$ comprising a Min CR deal can be found. □

As well as containing a single Min CR deal, as shown above, if two or more agents increase utility in a CR deal δ, we show next that δ also contains a composition of two Min CR deals δ_1 and δ_2, such that δ_1 and δ_2 comprise of sets of actions that do not overlap.

Lemma 4.9 (CR deals wherein two or more agents increase utility) *If $\delta = (Res, Res')$ and $\delta_1 = (Res, Res_1)$ are CR deals resulting in agent systems AS' and AS_1 respectively such that $Actions^{\delta_1} \subset Actions^{\delta}$ and an agent $x \in names(AS)$ increases utility in δ but does not increase utility in δ_1, then there exists a Min CR deal $\delta_2 = (Res_1, Res_2)$ where $Actions^{\delta_2} \subset Actions^{\delta}$, x increases utility in δ_2 and $Actions^{\delta_1} \cap Actions^{\delta_2} = \emptyset$.*

Proof. Since x increases utility in δ, then the r.r.p. of x is unsolved in AS and $\texttt{give}(z_1,r_1,x) \in Actions^{\delta}$ for some agent $z_1 \in names(AS)$ and resource $r_1 \in \mathcal{R}$ that solves the r.r.p. of x. Also, since x does not increase utility in δ_1, then $\texttt{give}(z_1,r_1,x) \notin Actions^{\delta_1}$ and, by Lemma 4.7, $r_1 \in Res_1(z_1)$. If z_1 can simply give away r_1 in AS_1 without decreasing in utility, then the deal $\delta_2 = (Res_1, Res_2)$ where $Actions^{\delta_2} = \{\texttt{give}(z_1,r_1,x)\}$ is a Min CR deal. Note that $Actions^{\delta_2} \subset Actions^{\delta}$, x increases utility in δ_2 and $Actions^{\delta_1} \cap Actions^{\delta_2} = \emptyset$. On the other hand, if simply giving away r_1 in AS_1 decreases z_1's utility, this implies that r_1 solves the r.r.p. of z_1, z_1 has no other resource in AS_1 that solves its r.r.p. and there is no $\texttt{give}(z,r,z_1) \in Actions^{\delta_1}$ for any agent $z \in names(AS)$ and resource $r \in \mathcal{R}$ that solves the r.r.p. of z_1. But, since δ is a CR deal and $\texttt{give}(z_1,r_1,x) \in Actions^{\delta}$, it must be the case that $\texttt{give}(z_2,r_2,z_1) \in Actions^{\delta}$ for some agent $z_2 \in names(AS)$ and resource $r_2 \in \mathcal{R}$ that solves the r.r.p. of z_1. We know $\texttt{give}(z_2,r_2,z_1) \notin Actions^{\delta_1}$ and hence, by Lemma 4.7, $r_2 \in Res_1(z_2)$. If z_2 does not decrease in utility in the deal $\delta_2 = (Res_1, Res_2)$ where $Actions^{\delta_2} = \{\texttt{give}(z_1,r_1,x), \texttt{give}(z_2,r_2,z_1)\}$, then it is a Min CR deal. Note that $Actions^{\delta_2} \subset Actions^{\delta}$, x increases utility in δ_2 and $Actions^{\delta_1} \cap Actions^{\delta_2} = \emptyset$. On the other hand, if z_2 decreases in utility in

the deal $\{\texttt{give}(z_1,r_1,x), \texttt{give}(z_2,r_2,z_1)\}$, then, similarly as for z_1 not being able to simply give away r_1, we can deduce $\texttt{give}(z_3,r_3,z_2) \in \mathit{Actions}^{\delta}$ for some agent $z_3 \in \mathit{names}(AS)$ and resource $r_3 \in \mathcal{R}$ that solves the r.r.p. of z_2. Continuing in this manner we can find a Min CR deal $\delta_2 = (\mathit{Res}_1, \mathit{Res}_2)$ such that $\mathit{Actions}^{\delta_2} = \{\texttt{give}(z_1,r_1,x), \texttt{give}(z_2,r_2,z_1), \ldots, \texttt{give}(z_{n+1},r_{n+1},z_n)\}$, $\mathit{Actions}^{\delta_2} \subset \mathit{Actions}^{\delta}$, x increases utility in δ_2 and $\mathit{Actions}^{\delta_1} \cap \mathit{Actions}^{\delta_2} = \emptyset$. □

Lastly we bring together the lemmas and corollaries above to prove that every CR deal δ can be decomposed into a sequence of Min CR deals δ' wherein the exact same agents increase utility in δ as they do in δ', and δ' contains no meaningless exchanges of resources, as follows:

Theorem 4.8 (CR deals decomposed as sequences of Min CR deals) *If $\delta = (\mathit{Res}, \mathit{Res}')$ is a CR deal resulting in an agent system AS' where $sw_{AS}(AS') = sw_{AS}(AS) + n$ ($n \geq 1$), then there is a CR deal $\delta_1 \circ \ldots \circ \delta_n = (\mathit{Res}, \mathit{Res}_n)$ resulting in an agent system AS_n where each δ_i ($1 \leq i \leq n$) is a Min CR deal and $happy_{AS}(AS_n) = happy_{AS}(AS')$.*

Proof. Let the n agents that increase in utility in δ be $happy_{AS}(AS') \setminus happy_{AS}(AS) = \{x_1, \ldots, x_n\}$. Since δ is a CR deal and x_1 increases utility in δ, then, by Lemma 4.8, there is a Min CR deal $\delta_1 = (\mathit{Res}, \mathit{Res}_1)$ such that $\mathit{Actions}^{\delta_1} \subseteq \mathit{Actions}^{\delta}$ and x_1 increases utility in δ_1. Note that, by Corollary 4.4, only x_1 increases utility in δ_1. If $n = 1$, we are done. If $n > 1$, we continue. Since δ and δ_1 are CR deals, and x_2 increases utility in δ but not in δ_1, then $\mathit{Actions}^{\delta_1} \subset \mathit{Actions}^{\delta}$ and, by Lemma 4.9, there is a Min CR deal $\delta_2 = (\mathit{Res}_1, \mathit{Res}_2)$ resulting in an agent system AS_2 where $\mathit{Actions}^{\delta_2} \subset \mathit{Actions}^{\delta}$ and x_2 increases utility in δ_2. If $n = 2$, we have $\delta_1 \circ \delta_2$ where each δ_i ($1 \leq i \leq 2$) is a Min CR deal, and hence, by Theorem 4.1, $\delta_1 \circ \delta_2$ is a CR deal. Also, as required, $happy_{AS}(AS_2) = happy_{AS}(AS')$. If $n > 2$, continuing successively for each i ($1 < i < n$) in this way, from each CR deal $\delta_1 \circ \ldots \circ \delta_i = (\mathit{Res}, \mathit{Res}_i)$, we can find a Min CR deal $\delta_{i+1} = (\mathit{Res}_i, \mathit{Res}_{i+1})$ where $\mathit{Actions}^{\delta_{i+1}} \subset \mathit{Actions}^{\delta}$ and (only) x_{i+1} increases utility in δ_{i+1}. Then, we end with a a CR deal $\delta_1 \circ \ldots \circ \delta_n = (\mathit{Res}, \mathit{Res}_n)$ resulting in an agent system AS_n where each δ_i ($1 \leq i \leq n$) is a Min CR deal and $happy_{AS}(AS_n) = happy_{AS}(AS')$. □

As a result of the theorem above we have that there exists a sequence of Min CR deals which result in agent system that maximises the usw of an initial agent system AS whenever there exists a CR deal which results in an agent system that maximises the usw of AS, as follows:

Corollary 4.5 (completeness of Min CR deals) *If there exists a CR deal $\delta = (Res, Res')$ resulting in an agent system AS' which maximises the usw of AS and such that $sw_{AS}(AS') = sw_{AS}(AS) + n$ $(n \geq 1)$, then there exists a sequence of Min CR deals $\delta_1 \circ \ldots \circ \delta_n = (Res, Res_n)$ resulting in an agent system AS_n which maximises the usw of AS.*

Proof. Since δ is a CR deal and $sw_{AS}(AS') = sw_{AS}(AS) + n$ $(n \geq 1)$, we know by Theorem 4.8 that there exists a sequence of Min CR deals $\delta_1 \circ \ldots \circ \delta_n = (Res, Res_n)$ resulting in an agent system AS_n such that $happy_{AS}(AS_n) = happy_{AS}(AS')$. Then, since AS' maximises the usw of AS, trivially by Definition 3.7, AS_n also maximises the usw of AS. □

4.6 Discussion

Given any resource reallocation problem in terms of the goals, resources and agents involved and represented according to the framework set out in Chapter 3, we have identified in this chapter the steps (deals) that would take the system from its initial allocation to an optimal allocation of resources. Though our discussion in this chapter is independent of how these deals are to be generated and also the nature of the solution (i.e. centralised or distributed), the deals lend themselves to a distributed solution, as is the focus in this thesis. Notably, since deals are to be found (agreed) by the participating agents themselves (which have only partial knowledge at their disposal) and not a central decision-making agent (which would have a more complete knowledge of the agents, their goals and resources), it is important that the deals are kept simple and involve as few agents and resources as possible. The deals which we identify meet this criterion since all actions as part of the deal are relevant and necessary for it to be agreeable to the agents involved, and hence the number of agents and resources involved in each deal is kept to a minimum. This is important for the large-scale (Grid computing, scheduling/timetabling

etc) applications as described in Chapter 1, Section 1.2 (and Chapter 3, Section 3.5) that our method is targeting where the total number of agents and resources comprising the system would be immense and it makes sense to limit the number of agents a particular agent needs to interact with (and also the number of resources that are subject to discussion) in order to agree on a suitable deal.

4.7 Conclusion

We have defined in this chapter deals that transition an agent system from one allocation (agent system) to another allocation (agent system). In particular, we have defined a specific kind of deal, namely the minimally cooperative rational deal, and have extended results of Endriss et al. [EMST06] to show that these deals are sufficient for solving the resource reallocation problem of Chapter 3. We have characterised the actions that comprise these deals in terms of the number of resources that agents involved receive and give away, and in terms of which agents change (increase) utility as a result of these deals. The work in this chapter of course does not address the important and complex problem of how agents would actually find and generate such deals. The problem is complicated further given that our aim in this thesis is for deals to be generated locally, in a distributed manner, by the agents involved and given that agents may have incomplete and/or incorrect information about the other agents. Moreover, as we have seen in this chapter, minimally cooperative rational deals may involve multiple agents and an agent may need to give away more than one resource in the deal, i.e. the negotiation process needs to capture 'OCSM contract types' (see Section 2.1.6) as identified by Toumas Sandholm [San98]. The definition of such a (generative) negotiation process will be the focus of the next chapter. There, we define a communication language and protocol that captures minimally cooperative rational deals and which would allow agents to negotiate towards allocations of resources that solve their own r.r.p. and, as a result, the r.r.p. of the entire agent system.

Chapter 5

Dialogues for Solving the Resource Reallocation Problem

We present in this chapter a communication language and protocol for argumentation-based negotiation, which allows agents that have only limited knowledge to negotiate deals (exchanges of resources) and share arguments (knowledge) in the process. We present also a means of defining agent policies/strategies that allow agents to generate moves conforming to the protocol. We discuss how such an agent policy would need to be defined to negotiate the kinds of deals described in Chapter 4 for solving the resource reallocation problem (r.r.p.) of interest in this thesis. A concretely defined agent policy is such that an agent can decide what move to make in any situation it finds itself in, as well as knowing what to do with what it is told in terms of updating its internal state (beliefs etc) and responding accordingly.

5.1 Introduction

The inter-agent interaction method of choice in this thesis is that of dialogues, namely orderly sequences of message exchanges between agents, where each agent has a goal and the dialogues, as a whole, also have a goal, notably that of solving the resource reallocation problem for all dialogue participants. Concretely, we focus on *negotiation* dialogues (see Table 2.1 in Chap-

ter 2), whereby agents *involved* in a dialogue try to arrive at a decision concerning possession of resources by agreeing a *deal*. We allow two types of dialogue that agents use intermittently to try and obtain a needed resource. The first, a *request*, is a dialogue between two agents (an initiator and a responder). The second, a *proposal*, is a dialogue between three or more agents (an initiator and a set of responders).

We adopt a "generative" approach to negotiation, similar to the approach of [BH07] for inquiry dialogues, by providing a constructive *policy* for the automatic generation of dialogues by autonomously reasoning agents. We describe in this chapter the policy which specifies how an agent initiates *requests* or *proposals*, directed at one or more agents respectively, depending on their beliefs about themselves and other agents. As discussed in Chapter 3, these beliefs may be partial, in that agents may have incomplete beliefs about the goals and resources held by other agents. The policy we define allows agents to provide "reasons" for their refusals to other agents, thus filling in gaps in their knowledge. The policy relies on argumentation (i) to fill the gaps in the agents' knowledge at request and proposal time and (ii) to compute "reasons" (in the form of arguments). When agents share from the onset knowledge as to which resources fulfil which goals, as assumed in Chapter 3, the policy used by agents allows agents to compute socially optimal allocations (with a maximal number of agents obtaining a resource that fulfils its goal) finitely and effectively. In particular, an agent makes use of reasons it has received previously to make more effective its search for a deal that solves its r.r.p.

For the purposes of this chapter, we will assume an argumentation framework where arguments will be of the form `<Conclusion, Support>`. We will not address in this chapter exactly how such arguments are generated (instantiated concretely) by the agents. This will be addressed in Chapter 7 for a particular general-purpose argumentation framework. Intuitively, the `Support` of arguments include some of the utterer's beliefs as well as information about the allocated and needed resources, which may be useful for the receiver to know.

This chapter is structured as follows: In Section 5.2 we define the two types of dialogue (i.e. the protocols) considered in this thesis. In Section 5.3 we present the communication rules for a general negotiation policy that conforms to the protocols. In Section 5.4 we give a definition

of the procedures found in the communication rules that would solve our resource reallocation problem. In Section 5.5 we provide examples of the kinds of dialogues agents would generate using the policy and procedures defined in this chapter. In Section 5.6 we discuss properties of these policies and procedures. Lastly, in Sections 5.7 and 5.8, we discuss related work and conclude.

5.2 Communication Language and Protocols for Dialogues

We assume in this chapter an agent system $AS = (\mathcal{G}, \mathcal{R}, \mathcal{A})$ (as in Definition 3.1) containing an allocation *Res* (see Definition 2.1). We define in this section a communication language and two protocols for dialogue which allow an agent to *request* an exchange of resources with another agent and to wait for a response (agreement or refusal). Or, alternatively, to *propose* an exchange of resources involving a number of agents and to wait for a response (acceptance or rejection) from each agent involved. In the case of all agents accepting the proposal, the initating agent confirms, sealing the exchange. In the case of receiving rejection from any agent, the initiating agent disconfirms the proposal with all agents that accepted, cancelling the exchange. Requests, proposals, refusals of requests and rejections of proposals may be accompanied by *reasons* which are *arguments*. The dialogue moves and protocols (instances) are presented generally in this section independent of the particular resource reallocation problem which we consider in this thesis. In our particular setting, as we will see in Section 5.4, a dialogue instance allows an agent to attempt to obtain a resource needed to achieve its goal.

Definition 5.1 (Dialogue Moves) *The* dialogue moves *consist of all instances of the following schemata (for $X, Y \in names(AS)$, $X \neq Y$, $Z \subseteq names(AS)$, $X \notin Z$):*

- `request(`X`,`Y`,Actions,Reason)`
- `propose(`X`,`Z`,Actions,Reason)`
- `agree(`X`,`Y`,Actions)`

- `refuse(`X`,`Y`,Actions,Reason)`
- `accept_proposal(`X`,`Y`,Actions)`
- `reject_proposal(`X`,`Y`,Actions,Reason)`
- `confirm(`X`,`Z`,Actions)`
- `disconfirm(`X`,`Z`,Actions)`

where

- X *is the* utterer *and* Y *or* Z *are the* receivers*;*
- `Actions`*, the set of actions (or exchanges of resources) in the dialogue move, is a non-empty set of atoms of the form* `give(`A`,`R`,`B`)`*, i.e. the agent named* A *is to give resource* R *to the agent named* B*, where* $A, B \in \{X, Y\}$ *or* $A, B \in \{X\} \cup Z$*;*
- `Reason` *is* `null` *or an argument of the form* `<Conclusion, Support>`*.*

Note that the dialogue moves take the form of negotiation moves regarding the exchange of resources. A set of actions in a dialogue move of the form {`give(`y`,`r`,`x`)`}, for example, is interpreted as a request for y to give r to x, with y receiving nothing in exchange, and {`give(`y`,`r`,`x`)`, `give(`x`,`r'`,`y`)`} is interpreted as a request for x and y to exchange resources r' and r.

> The kinds of actions in the dialogue moves which will we consider in this thesis are actions that comprise a minimally cooperative rational (Min CR) deal (see Theorems 4.5–4.7).

The dialogue moves we consider can be seen as FIPA ACL (or KQML) communicative acts in the sense that they consist of a *performative* and a number of *parameters* as follows: *utterer* of the dialogue move (X in Definition 5.1), *receiver(s)* (Y or Z in Definition 5.1) and *content* (`Actions` together with `Reason` as in Definition 5.1). Whilst each of the performatives in Definition 5.1 is a performative in the FIPA ACL also, our dialogue moves do not necessarily conform to the semantics of communicative acts (performatives) in FIPA ACL (see Section 2.2.1).

Dialogue instances are of two types: request and proposal. *Request dialogue instances* begin with a `request` dialogue move and *proposal dialogue instances* begin with a `propose` dialogue move, as follows:

Definition 5.2 (Request Dialogue Instances) *A* request dialogue *involving an initiating agent named x and a responding agent named y regarding* `Actions` *is a sequence of dialogue moves of one of the following forms:*

- `request(`x`,`y`,Actions,_ )`, `agree(`y`,`x`,Actions)` *(*successful *request dialogue instance)*
- `request(`x`,`y`,Actions,_ )`, `refuse(`y`,`x`,Actions,_ )` *(*unsuccessful *request dialogue instance)*

In both cases, `Actions` *is a non-empty set of atoms of the form* `give(`A`,`R`,`B`)` *($A, B \in \{x, y\}$, $A \neq B$, $R \in \mathcal{R}$), there is at least one* `give(`A`,_,_ )` *or* `give(_,_,`A`)` *in* `Actions` *for each $A \in \{x, y\}$, and there is at most one* `give(_,`R`,_ )` *in* `Actions` *for each $R \in \mathcal{R}$.*

Definition 5.3 (Proposal Dialogue Instance) *A* proposal dialogue instance *involving an initiating agent named x and a set of responding agents Z regarding* `Actions` *is a sequence of dialogue moves of one of the following forms:*

- `propose(`x`,`Z`,Actions,_ )`, `accept_proposal(`y_1`,`x`,Actions)`, ...,
  `accept_proposal(`y_n`,`x`,Actions)`, `confirm(`x`,`Z`,Actions)`,
  *(*successful proposal dialogue instance*)*
  for $Z = \{y_1, \ldots, y_n\}$
- `propose(`x`,`Z`,Actions,_ )`, `FailedSubSeq`,
  `disconfirm(`x`,`$\{y_{n+1}, \ldots, y_{n+m}\}$`,Actions)`
  *(*unsuccessful proposal dialogue instance*)*
  where `FailedSubSeq` *is a permutation of*

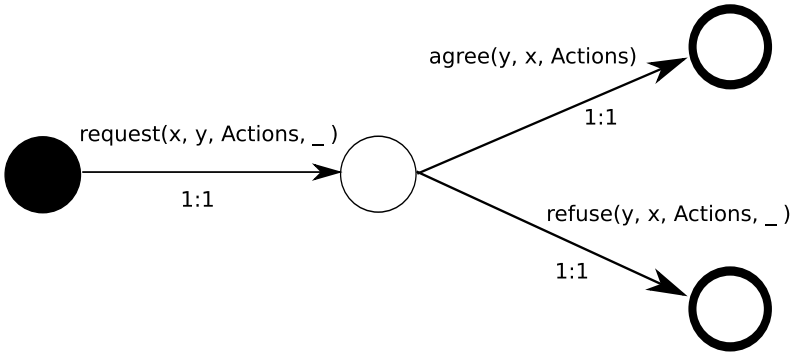


Figure 5.1: Request Dialogue Instance as Finite State Automaton

`reject_proposal`$(y_1,x,$`Actions`$,_)$, ..., `reject_proposal`$(y_n,x,$`Actions`$,_)$,

`accept_proposal`$(y_{n+1},x,$`Actions`$)$, ..., `accept_proposal`$(y_{n+m},x,$`Actions`$)$,

for $Z = \{y_1, \ldots, y_n, y_{n+1}, \ldots, y_{n+m}\}$ *with* $n > 0$ *and* $m > 0$

- `propose`$(x,Z,$`Actions`$)$, `reject_proposal`$(y_1,x,$`Actions`$,_)$, ...,

  `reject_proposal`$(y_n,x,$`Actions`$,_)$

  *(*unsuccessful proposal dialogue instance*)*

  for $Z = \{y_1, \ldots, y_n\}$.

In all cases, `Actions` *is a non-empty set of atoms of the form* `give`(A,R,B) *(*$A, B \in \{x\} \cup Z$*,* $A \neq B$*,* $R \in \mathcal{R}$*), and there is at least one* `give`$(A,_,_)$ *or* `give`$(_,_,A)$ *in* `Actions` *for each* $A \in \{x\} \cup Z$*, and there is at most one* `give`$(_,R,_)$ *in* `Actions` *for each* $R \in \mathcal{R}$.

Basically, the kinds of dialogue instances we consider conform to the protocols given in Figures 5.1 and 5.2 as finite state automata.

We assume in this thesis that agents initiate requests or proposals to *obtain* a certain resource, as follows:

Definition 5.4 (dialogue instance initiated by agent to obtain resource) *A request or proposal dialogue instance* d *regarding* `Actions` *is initiated by an agent* x *to obtain a resource* r *iff* x *is the initiating agent of* d *and* `give`$(_,r,x) \in$ `Actions`.

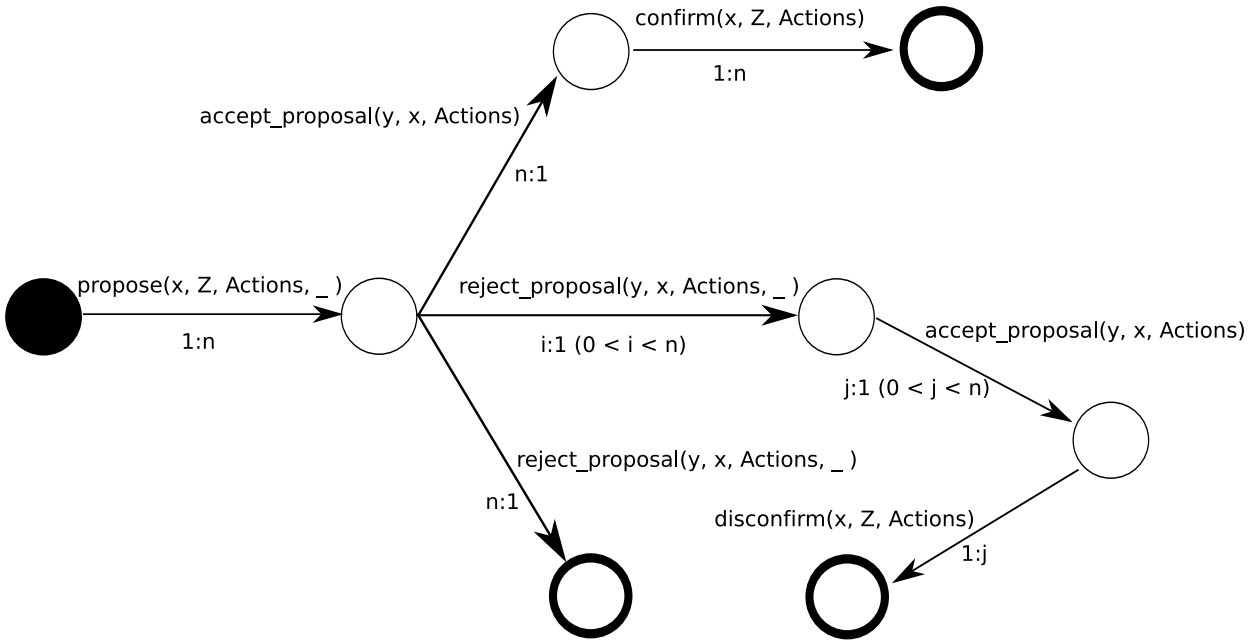


Figure 5.2: Proposal Dialogue Instance as Finite State Automaton

Given that our interest in this thesis are actions in requests and proposals that comprise a Min CR deal, it is necessary then that at least one agent is to *obtain* a resource that solves its resource reallocation problem. We will see in Section 5.4 that this will be the agent initiating the dialogue instance. We present next examples of (successful) request and proposal dialogue instances. Here and throughout the chapter, we associate timepoints to utterances in order to show how they are sequenced. In addition, we omit the argument (`Reason`) in writing the `request`, `propose`, `refuse` and `reject_proposal` dialogue moves when `Reason` = `null`.

Example 5.1 (request dialogue instance) *A request dialogue instance that captures the deal of Example 4.4 could proceed as follows, where the dialogue instance is initiated by* $ag2$ *to obtain* r_2*, and* t_1 *is a timepoint prior to* t_2*:*

t_1*:* `request(`$ag2$`,`$ag1$`,{give(`$ag2$`,`$r2$`,`$ag1$`), give(`$ag1$`,`$r1$`,`$ag2$`)})`

t_2*:* `agree(`$ag1$`,`$ag2$`,{give(`$ag2$`,`$r2$`,`$ag1$`), give(`$ag1$`,`$r1$`,`$ag2$`)})`

Example 5.2 (proposal dialogue instance) *A dialogue instance that captures the deal of Example 4.6 could proceed as follows, where the dialogue instance is initiated by* $ag2$ *to obtain* r_2*, and* t_1*,* t_2*,* t_3*,* t_4 *are successive timepoints:*

t_1: propose($ag2$,{$ag1$,$ag3$},{give($ag2$,$r3$,$ag3$),give($ag3$,$r2$,$ag1$),give($ag1$,$r1$,$ag2$)})

t_2: accept_proposal($ag1$,$ag2$,{give($ag2$,$r3$,$ag3$),give($ag3$,$r2$,$ag1$),give($ag1$,$r1$,$ag2$)})

t_3: accept_proposal($ag3$,$ag2$,{give($ag2$,$r3$,$ag3$),give($ag3$,$r2$,$ag1$),give($ag1$,$r1$,$ag2$)})

t_4: confirm($ag2$,{$ag1$,$ag3$},{give($ag2$,$r3$,$ag3$),give($ag3$,$r2$,$ag1$),give($ag1$,$r1$,$ag2$)})

Only a successful (request or proposal) dialogue instance results in an actual exchange of resources such that (some) agents have different sets of resources following the dialogue instance than they did before the dialogue instance, as follows:

Definition 5.5 (resources held by agents following successful dialogue instance) *Let Res be the* allocation *of resources contained in the agent system AS before a successful (request or proposal) dialogue instance d regarding* Actions*, where an allocation contained in an agent system is as in Lemma 3.1. $Res^{Actions}$ is the* assignment *of resources to agents following d, defined for each agent $x \in names(AS)$ as follows:*

$$Res^{Actions}(x) = Res(x) \cup \{\ r \mid \texttt{give}(_\ ,\ r,\ x) \in \texttt{Actions}\ \} - \{\ r \mid \texttt{give}(x,\ r,\ _\ ) \in \texttt{Actions}\ \}$$

We will assume in this thesis that every resource to be given away by an agent in a dialogue instance regarding an exchange of resources Actions is held by that agent, i.e. $r \in Res(x)$ for each give(x,r,_) $\in$ Actions. We show next that, under this assumption, the *assignment* of resources to agents following a successful dialogue instance is an *allocation* in the sense of Lemma 3.1, as follows:

Lemma 5.1 (successful dialogue instance results in new allocation of resources) *Let Res be the allocation of resources contained in AS before a successful (request or proposal) dialogue instance d regarding* Actions*. Then, the assignment of resources to agents following d, i.e. $Res^{Actions}$ as in Definition 5.5, is an allocation.*

Proof. Firstly, for each $r \in \mathcal{R}$ such that `give(_,`r`,_ )` $\notin$ `Actions`, we have, by assumption of $\bigcup_{x \in names(AS)} Res(x) = \mathcal{R}$, that $r \in Res(x)$ for some $x \in names(AS)$, and, by Definition 5.5, that $r \in Res^{Actions}(x)$ and $r \notin Res^{Actions}(y)$ (for all $y \neq x$). Secondly, by Definitions 5.2 and 5.3, there is at most one `give(_,`r`,_ )` $\in$ `Actions` for each $r \in \mathcal{R}$. Then, for each $r \in \mathcal{R}$ such that `give(`x`,`r`,`y`)` $\in$ `Actions` $(x, y \in names(AS))$, by assumption of Res being an allocation and every agent holding the resource it is to give away, we have that $r \in Res(x)$ and $r \notin Res(z)$ (for all $z \neq x$), and, by Definition 5.5, that $r \in Res^{Actions}(y)$ and $r \notin Res^{Actions}(z)$ (for all $z \neq y$). So, every resource is held by one and only one agent before the dialogue instance, and it is either retained by that agent or given to one other agent as a result of the dialogue instance. Therefore, $Res^{Actions}(x) \cap Res^{Actions}(y) = \emptyset$ for $x \neq y$ and $\bigcup_{x \in names(AS)} Res^{Actions}(x) = \mathcal{R}$, and hence $Res^{Actions}$ is an allocation. □

As a corollary of the previous lemma, a successful dialogue instance then results in a *deal* in the sense of Definition 2.2, as follows:

Corollary 5.1 (successful dialogue instance results in a deal) *Let Res be the allocation of resources contained in AS before a successful (request or proposal) dialogue instance d regarding* `Actions`*, and* $Res^{Actions}$ *the assignment of resources to agents following d as in Definition 5.5. Then,* $\delta = (Res, Res^{Actions})$ *is a deal.*

Proof. Trivial since we know by Lemma 5.1 that $Res^{Actions}$ is an allocation, and, since `Actions` is a non-empty set (i.e. resources change hands between agents), $Res \neq Res^{Actions}$. □

In lieu of the above lemma and corollary we will say that a successful dialogue instance results in a new allocation, or that *a successful dialogue instance results in a deal.* The agents involved in a deal (i.e. those that receive or give resources as part of the deal, see Definition 2.3) resulting from a successful dialogue instance are exactly those that are involved in the dialogue instance (i.e. the initiating and responding agents).

Lemma 5.2 (agents involved in a deal are those involved in the dialogue instance) *Let* `Actions` *be a set of atoms of the form* `give(`A`,`R`,`B`)` *as in Definition 5.2 or 5.3 for a*

successful request or proposal dialogue instance, respectively, involving a set of (initiating and responding) agents Z. Let $\delta = (Res, Res^{Actions})$ be the deal resulting from the dialogue instance. Then, $Agents^{\delta} = Z$.

Proof. By definition of agents involved in a deal (Definition 2.3) we have $x \in Agents^{\delta}$ if $x \in names(AS)$ and $Res(x) \neq Res^{Actions}(x)$. By Definition 5.5, $Res(x) \neq Res^{Actions}(x)$ for all $x \in names(AS)$ such that `give(`x`,_,_ )` $\in$ `Actions` or `give(_,_,`x`)` $\in$ `Actions`, and $Res(x) = Res^{Actions}(x)$ otherwise. Since, by Definitions 5.2 and 5.3, there is at least one `give(`x`,_,_ )` or `give(_,_,`x`)` in `Actions` for each $x \in Z$, then $Agents^{\delta} = Z$. □

In general, an agent may need to initiate a number of dialogue instances with a number of agents before a particular resource is obtained or before it gives up its intention of obtaining the resource.

Definition 5.6 (Dialogue Sequence) *A* dialogue sequence *wrt an agent x and a resource r is a sequence of (request or proposal)* dialogue instances *all initiated by x to obtain r such that a dialogue instance in the sequence is only initiated after the previous dialogue instance has completed and the only successful dialogue instance in the sequence, if any, is the last one in the sequence. A dialogue sequence* terminates successfully *if it is finite and the last dialogue instance is successful. A dialogue sequence* terminates unsuccessfully *if it is finite and the last dialogue instance is unsuccessful.*

In our problem domain, an agent attempts to obtain a resource that fulfils its goal (solves its r.r.p.) and can be seen as succeeding if any such dialogue sequence terminates successfully. By virtue of Definition 5.6, we assume in this thesis that an agent does not instantiate concurrent dialogue instances, such that an agent that has initiated a dialogue instance will not instantiate another dialogue instance until the first has ended. Each dialogue sequence can then be seen as a serial sequence of dialogue instances. Note however that an agent may be running multiple dialogue sequences concurrently, as follows:

Example 5.3 (concurrent dialogue sequences) *Let d_1, d_2, d_3 be unsuccessful dialogue instances initiated by an agent x to obtain a resource r_1. Let d_4 and d_5 be unsuccessful and*

successful dialogue instances respectively initiated by x to obtain a resource r_2. Assume the dialogue instances proceed in the following order: d_1, d_2, d_4, d_3, d_5. This merges an unsuccessful dialogue sequence d_1, d_2, d_3 wrt x and r_1, and a successful dialogue sequence d_4, d_5 wrt x and r_2, in a way that both dialogue sequences are intertwined (running concurrently).

We will allow an agent to ask the same resource from the same agent more than once within a dialogue sequence, though the particulars of the request or proposal would be expected to differ. Note that agents will take the role of initiator or responder in different dialogue instances, possibly occurring simultaneously, and different agents may be initiating dialogue instances simultaneously each in pursuit of a resource that fulfils their own individual goal. We illustrate this as follows:

Example 5.4 (concurrent dialogue instances) *Consider the following dialogue moves uttered at successive timepoints t_1, t_2, t_3, t_4:*

t_1: `request(`$ag1$`,`$ag2$`,{give(`$ag2$`,`$r2$`,`$ag1$`)})`

t_2: `request(`$ag3$`,`$ag1$`,{give(`$ag1$`,`$r1$`,`$ag3$`)})`

t_3: `agree(`$ag1$`,`$ag3$`,{give(`$ag1$`,`$r1$`,`$ag3$`)})`

t_4: `agree(`$ag2$`,`$ag1$`,{give(`$ag2$`,`$r2$`,`$ag1$`)})`

Then, the dialogue moves uttered at timepoints t_1 and t_4 make up a dialogue instance d_1 initiated by $ag1$ to obtain $r2$, and the dialogue moves uttered at timepoints t_2 and t_3 make up a dialogue instance d_2 initiated by $ag3$ to obtain $r1$. Notice in particular that d_2 is initiated after d_1 but completed before d_1, and that ag_1 is initiator in d_1 and responder in d_2.

We will discuss in the next section how agents can be defined to ensure dialogical correctness properties given that an agent may be involved in multiple dialogue instances simultanously. For example, it should never be the case that an agent offers to give away the same resource to two agents simultaneously.

5.3 Communication Rules

We present in this section the procedures and communication rules that make up a communication policy which generates dialogue instances between agents for the reallocation of resources according to the protocols of the previous section (Definitions 5.2 and 5.3). We present these generally in this section, to be given a specific definition in Section 5.4 for our particular resource reallocation problem of interest.

We begin by defining the *state* of an agent as follows:

Definition 5.7 (state of an agent) *The* state *of an agent is a triple* ($\langle x$,$Res(x)$,$G(x)$,$B(x)\rangle$, $M(x)$, $D(x)$) *such that:*

- $\langle x, Res(x), G(x), B(x)\rangle$ *is the name, resources, goal and beliefs of the agent as in Definition 3.1 (Chapter 3);*
- $M(x)$ *is the set of dialogue* moves *received in x's incoming "message queue", i.e. the dialogue moves with x as recipient which x has not yet "dealt" with;*
- $D(x)$ *is the* dialogical status *of the agent, i.e. a set of ground literals operating as flags.*

Flags will be used to "remember" when the agent has initiated a dialogue instance, accepted a proposal or received acceptance of a proposal.

We will see that, as a result of engaging in dialogue instances, the state of an agent undergoes changes.

The negotiation policy is defined as a set of *communication rules*. These rules have associated with them "preconditions", possibly the reception of an "incoming" dialogue move, and "consequences", which may be the utterance of an "outgoing" dialogue move and/or a number of "revisions" to the state of the (uttering) agent. Formally, a communication rule for an agent x is specified as a tuple

$$\langle In, P, Out, X\rangle$$

where

- In, the *incoming dialogue move*, is a dialogue move (with x as receiver) or is `null`;
- P, the *preconditions*, is a set (conjunction) of boolean "procedures";
- Out is the *outgoing dialogue move* (with x as utterer) or is `null`;
- X, the *revisions*, is a set of "procedures" that modify the agent's resources ($R(x)$), beliefs ($B(x)$) and dialogical status ($D(x)$).

It is intended that agent x applies a communication rule $\langle In, P, Out, X\rangle$ removing the dialogue move corresponding to In from $M(x)$ (if $In \neq$ `null`), generating (uttering) the corresponding Out (if $Out \neq$ `null`) and enforcing the revisions in X whenever

- $In \neq$ `null` and In matches a dialogue move in $M(x)$, or $In =$ `null`; and
- the preconditions P "hold" (evaluate to true).

The evaluation of P will typically lead to instantiation of variables in the communication rule. The procedures used in the communication rules will be split into two kinds:

- "general procedures", which can be realised in several ways according to an agent's owned resources, goal and beliefs in order for it to fulfil its purpose;
- "control procedures", guaranteeing that dialogues are appropriately sequenced and interleaved.

We describe informally in this section the role of the different "general procedures" and "control procedures", and present the communication rules accordingly.

The "general procedures" called within the preconditions of communication rules of an agent x are:

- `generateRequest(`Y`,Actions,Reason)`, that evaluates to true (returns successfully) and instantiates an exchange of resources `Actions` to request of an agent Y ($Y \neq x$) as well as an argument (`Reason`, possibly `null`) justifying the request, or otherwise evaluates to false (returns unsuccessfully);

- `generateProposal(`Z`,Actions,Reason)`, that evaluates to true and instantiates an exchange of resources `Actions` to *propose* to a set of agents Z ($x \notin Z$) as well as an argument (`Reason`, possibly `null`) justifying the proposal, or otherwise evaluates to false;

- `acceptableExchange(Actions,ReasonIn)`, that evaluates to true if the exchange of resources `Actions` with justification `ReasonIn` is acceptable for x,[1] and evaluates to false otherwise;

- ¬`acceptableExchange(Actions,ReasonIn,ReasonOut)`, that evaluates to true if the exchange of resources `Actions` with justification `ReasonIn` is *not* acceptable for x [1] and instantiates an argument (`ReasonOut`, possbily `null`) justifying the decision, and evaluates to false otherwise.

The "general procedures" called within the revisions of communication rules of agent x are:

- `exchangeResources(Actions)`, that revises x's owned resources and belief set as to which agents have which resources according to `Actions`;

- `reviseBeliefs(`Y`,Reason)`, that modifies x's belief set given an argument `Reason` uttered to it by some agent Y.

In Section 5.4 we will define more formally but at a high-level the "general procedures" that allow us to capture Min CR deals sufficient for solving our resource reallocation problem of interest. In Chapter 7 we will define these procedures concretely using a particular argumentation framework.

[1]The acceptability or not of an exchange of resources, and to what extent the justification provided is considered, will depend on the particular problem domain and definition given to this procedure. It could be required for example that the "utility" of this agent increase as a result of the exchange or, as we will see in Section 5.4, that the "utility" of this agent does not decrease.

The "control procedures" called within the preconditions and revisions of the communication rules are of the form `valid( _ )` and `do( _ )` respectively. Indeed procedures of the form `valid( _ )` check if certain dialogical properties hold for the agent, and procedures of the form `do( _ )` are used to make modifications of an agent's dialogical status. We assume that the "control procedures" govern the policy according to the dialogue instance protocols (see Definitions 5.2 and 5.3) and our assumption of dialogue sequences (see Definition 5.6), and maintain correctness so that, for example, the agent does not offer the same resource, or the only two resources it has that solve its r.r.p., to two or more agents simultaneously. We provide an example definition of these procedures in Appendix B, as well as describing these informally in the remainder of this section along with the communication rules (*CR1–CR15*) defining a general policy for a generic agent x.

Initiating Request Dialogues

(*CR1*)	
$In_1 =$	`null`
$P_1 =$	{`generateRequest(`Y`,Actions,Reason)`, `valid(initiateDialogue(Actions))`}
$Out_1 =$	`request(`x`,`Y`,Actions,Reason)`
$X_1 =$	{`do(initiateDialogue({`Y`},Actions))`}

As a result of performing *CR1*, the agent (x) utters a `request` dialogue move to the agent Y regarding `Actions` and revises its dialogical status according to `do(initiateDialogue({`Y`}, Actions))`. Note that no incoming dialogue move is required to execute *CR1*. This is because *CR1* is not executed in response to a previous dialogue move. The precondition `valid( initiateDialogue(Actions))` could be defined, for example, as in Definition B.1 (see Appendix B) to check that x is not currently awaiting a response of any kind from any agent for a resource that would solve its r.r.p., and that the request (for `Actions`) at this point in time would not mean x has offered the same resource to two agents simultaneously. The revision `do(initiateDialogue({`Y`},Actions))` would record in x's state that it has initiated a dialogue with Y regarding `Actions` (see Definition B.7 in Appendix B).

Initiating Proposal Dialogues

(*CR2*)	
$In_2 =$	`null`
$P_2 =$	{`generateProposal(`Z`,Actions,Reason)`, `valid(initiateDialogue(Actions))`}
$Out_2 =$	`propose(`x`,`Z`,Actions,Reason)`
$X_2 =$	{`do(initiateDialogue(`Z`,Actions))`}

As a result of performing *CR2*, the agent (x) utters a `propose` dialogue move to the set of agents Z regarding `Actions` and revises its dialogical status according to `do(initiateDialogue(`Z`,Actions))`. As for *CR1*, no incoming dialogue move is required to execute *CR2*. The precondition `valid(initiateDialogue(Actions))` and the revision `do(initiateDialogue(`Z`, Actions))` are as in *CR1*.

Responding to Requests

(*CR3*)	
$In_3 =$	`request(`Y`,`x`,Actions,Reason)`
$P_3 =$	{`acceptableExchange(Actions,Reason)`, `valid(accept(Actions))`}
$Out_3 =$	`agree(`x`,`Y`,Actions)`
$X_3 =$	{`reviseBeliefs(`Y`,Reason)`, `exchangeResources(Actions)`}

As a result of performing *CR3*, the agent (x) removes the received `request` dialogue move from $M(x)$, utters an `agree` dialogue move and revises its state according to `reviseBeliefs(`Y`, Reason)` and `exchangeResources(Actions)`. The precondition `valid(accept(Actions))` could be defined, for example, as in Definition B.2 (see Appendix B) to check that x has not presently offered to give away any of its resources which solve its r.r.p., nor any resources which it would be required to give away in `Actions`.

(*CR4*)	
$In_4 =$	`request(`Y`,`x`,Actions,ReasonIn)`
$P_4 =$	{¬`acceptableExchange(Actions,ReasonIn,ReasonOut)`}
$Out_4 =$	`refuse(`x`,`Y`,Actions,ReasonOut)`
$X_4 =$	{`reviseBeliefs(`Y`,ReasonIn)`}

The agent (x) refuses a request (*CR4*) which is not acceptable for it and provides an argument for its decision (`ReasonOut`), which could be useful for the requesting agent (Y) in revising its belief set and generating future requests/proposals or not. x also revises its belief set according to the argument it received in the `request` via `reviseBeliefs(`Y`,ReasonIn)`.

Responding to Proposals

(*CR5*)	
$In_5 =$	`propose(`Y`,`Z`,Actions,ReasonIn)`
$P_5 =$	{$x \in Z$, `acceptableExchange(Actions,ReasonIn)`, `valid(accept(Actions))`}
$Out_5 =$	`accept_proposal(`x`,`Y`,Actions)`
$X_5 =$	{`reviseBeliefs(`Y`,ReasonIn)`, `do(accept_proposal(`Y`,Actions))`}

In the case of uttering `accept_proposal` in response to a `propose` dialogue move (*CR5*), the agent (x) does not immediately modify its state according to `exchangeResources(Actions)` as it does in *CR3*; it awaits a `confirm` before doing so (see *CR13* below). The precondition `valid(accept(Actions))` is as in *CR3*. The revision `do(accept_proposal(`Y`,Actions))` would record in x's state that it has accepted the proposal of Y for `Actions` (see Definition B.8 in Appendix B).

(*CR6*)	
$In_6 =$	`propose(`Y`,`Z`,Actions,ReasonIn)`
$P_6 =$	{$x \in Z$, ¬`acceptableExchange(Actions,ReasonIn,ReasonOut)`}
$Out_6 =$	`reject_proposal(`x`,`Y`,Actions,ReasonOut)`
$X_6 =$	{`reviseBeliefs(`Y`,ReasonIn)`}

CR6 is the analogous of *CR4* for proposals.

Receiving Responses of Requests

(*CR7*)	
$In_7 =$	`agree(`Y`,`x`,Actions)`
$P_7 =$	{`valid(receiveResponse(`Y`,Actions))`}
$Out_7 =$	`null`
$X_7 =$	{`exchangeResources(Actions)`, `do(receiveResponse(`Y`,Actions,agree))`}

In the case of receiving `agree` (*CR7*), the agent (x) revises its state according to `exchangeResources(Actions)`. The precondition `valid(receiveResponse(`Y`,Actions))`, which occurs in *CR8–CR10* also, checks that the agent is indeed awaiting a response from Y regarding `Actions` (see Definition B.3 in Appendix B). The revision `do(receiveResponse(`Y`, Actions,Response))`, which occurs in *CR8–CR10* also, records in x's state that it has received a response from Y regarding `Actions` (see Definition B.9 in Appendix B).

(*CR8*)	
$In_8 =$	`refuse(`Y`,`x`,Actions,Reason)`
$P_8 =$	{`valid(receiveResponse(`Y`,Actions))`}
$Out_8 =$	`null`
$X_8 =$	{`reviseBeliefs(`Y`,Reason)`, `do(receiveResponse(`Y`,Actions,refuse))`}

In the case of receiving `refuse` (*CR8*), the agent (x) revises its state according to `reviseBeliefs(`Y`,Argument)`.

Receiving Responses of Proposals

(*CR9*)	
$In_9 =$	`accept_proposal(`Y`,`x`,Actions)`
$P_9 =$	{`valid(receiveResponse(`Y`,Actions))`}
$Out_9 =$	`null`
$X_9 =$	{`do(receiveResponse(`Y`,Actions,acceptProposal))`}

In the case of receiving `accept_proposal` (*CR9*), the agent (x) does not yet make any changes to its state according to `exchangeResources(Actions)` as it does in *CR7*. This is only done when all responding agents involved in the dialogue have accepted and x can `confirm` the exchange of resources (see *CR11* below).

(*CR10*)	
$In_{10} =$	`reject_proposal(`Y`,`x`,Actions,Reason)`
$P_{10} =$	{`valid(receiveResponse(`Y`,Actions))`}
$Out_{10} =$	`null`
$X_{10} =$	{`reviseBeliefs(`Y`,Reason)`, `do(receiveResponse(`Y`,Actions,rejectProposal))`}

CR10 is the analogous of *CR8* for proposals.

Ending Proposal Dialogues

(*CR11*)	
$In_{11} =$	`null`
$P_{11} =$	{`valid(confirm(`Z`,Actions))`}
$Out_{11} =$	`confirm(`x`,`Z`,Actions)`
$X_{11} =$	{`exchangeResources(Actions)`, `do(confirm(Actions))`}

The agent (x) confirms a proposal (*CR11* above), after receiving acceptance from all agents Z to whom the proposal was made. The precondition `valid(confirm(`Z`,Actions))` checks that all responding agents involved in the dialogue have accepted the proposal (see Definition B.4 in Appendix B). The revision `do(confirm(Actions))`, which also occurs in *CR12* and *13*, records in x's state that the proposal dialogue instance has been ended (see Definition B.10 in Appendix B).

(*CR12*)	
$In_{12} =$	`null`
$P_{12} =$	{`valid(disconfirm(`Z`,Actions))`}
$Out_{12} =$	`disconfirm(`x`,`Z`,Actions)`
$X_{12} =$	{`do(confirm(Actions))`}

The agent (x) disconfirms a proposal (*CR12* above) if the precondition `valid(disconfirm(`Z`, Actions))` holds. The precondition checks that all agents Z' to whom the proposal was made have responded (with acceptance or rejection) but only a strict subset $Z \subset Z'$ of agents responded with acceptance (see Definition B.5 in Appendix B). The revision `do(confirm(Actions))` is similar as in *CR11*.

(*CR13*)	
$In_{13} =$	`null`
$P_{13} =$	{`valid(disconfirm(`$\emptyset$`,Actions))`}
$Out_{13} =$	`null`
$X_{13} =$	{`do(confirm(Actions))`}

The agent knows that a proposal has ended unsuccessfully without having to utter `disconfirm` (*CR13*) in the case of receiving rejection from all the agents to whom the proposal was made.

Receiving Confirm and Disconfirm Responses

(*CR14*)	
$In_{14} =$	`confirm(`Y`,`Z`,Actions)`
$P_{14} =$	{$x \in Z$, `valid(receiveConfirm(`Y`,Actions))`}
$Out_{14} =$	`null`
$X_{14} =$	{`exchangeResources(Actions)`, `do(receiveConfirm(`Y`,Actions))`}

In the case of receiving `confirm` (*CR14*), the agent (x) revises its state according to `exchangeResources(Actions)`. The precondition `valid(receiveConfirm(`Y`,Actions))` checks that x is indeed awaiting a `confirm` or `disconfirm` utterance from Y (see Definition B.6 in Appendix B). The revision `do(receiveConfirm(`Y`,Actions))` would record in x's state that it is no longer awaiting a `confirm` or `disconfirm` utterance from Y (see Definition B.11 in Appendix B).

(*CR15*)	
$In_{15} =$	`disconfirm(`Y`,`Z`,Actions)`
$P_{15} =$	$\{x \in Z$, `valid(receiveConfirm(`Y`,Actions))`$\}$
$Out_{15} =$	`null`
$X_{15} =$	$\{$`do(receiveConfirm(`Y`,Actions))`$\}$

In the case of receiving `disconfirm` (*CR15*), the agent (x) only needs to update its dialogical status according to `do(receiveConfirm(`Y`,Actions))`, as given in *CR14*. The precondition `valid(receiveConfirm(`Y`,Actions))` is also similar as for *CR14*.

Lastly, dialogue instances and dialogue sequences generated by means of the communication rules above are as follows:

Definition 5.8 (generated dialogue instance) *We say that a dialogue instance is generated by (means of) communication rules* CR1–CR15 *if the agents utter dialogue moves making up a sequence as in Definition 5.2 or 5.3 by applying the communication rules* CR1–CR15*.*

Definition 5.9 (generated dialogue sequence) *We say that a dialogue sequence is generated by (means of) communication rules* CR1–CR15 *if it is a dialogue sequence as in Definition 5.6 made up of dialogue instances generated by the communication rules* CR1–CR15

5.4 Procedures

We give a definition of the "general procedures" occurring in the communication rules of the previous section for our particular resource reallocation problem, i.e. such that agents can generate dialogues between one another for the reallocation of resources in a way that their individual resource reallocation problem (r.r.p.) is solved and, consequently, the r.r.p. of the agent system as a whole is solved (where the r.r.p. for an individual agent and an agent system are as given in Chapter 3). The resulting policy for some agent x is such that x is willing to exchange a resource it has that solves its r.r.p. for another that solves its r.r.p.

or is willing to give away a resource it has but does not *need* (i.e. if as a result of the deal x's r.r.p. will still be solved or unsolved, because the resource(s) to be given away do not solve x's r.r.p. or x has another resource that solves its r.r.p.). The policy we present is such that reasons (arguments) are not presented with requests and proposals but only with refusals, and the procedures hereafter are presented accordingly. Namely, whenever the procedures `generateRequest(`Y`,Actions,Reason)` and `generateProposal(`Z`,Actions,Reason)` evaluate to true, the value of `Reason` instantiates to `null` and therefore we present the procedures instead as `generateRequest(`Y`,Actions)` and `generateProposal(`Z`,Actions)` respectively. Accordingly, we write `acceptableExchange(Actions)` and ¬`acceptableExchange(Actions, ReasonOut)` instead of `acceptableExchange(Actions,ReasonIn)` and ¬`acceptableExchange(Actions,ReasonIn,ReasonOut)` respectively. Whilst it would be useful and necessary in the general case to allow the requesting or proposing agent to include additional information along with its request or proposal, such as a plan which the responding agent might not be aware of, the need does not arise in our setting since we assume plans to be shared and known by all agents. As discussed in Chapter 5.1, the exact form arguments take and how arguments are generated (by the ¬`acceptableExchange(Actions,ReasonOut)` procedure) are left deliberately open/unspecified in this chapter to be grounded/instantiated according to the argumentation framework of choice in Chapter 7.

It is important that agents can operate (i.e. initiate and respond to requests and proposals) despite having only partial, incomplete and potentially incorrect (dated) information/beliefs about other agents. Thus we employ a notion of "legitimate assumption" in the sense of a *presumption* as discussed in Section 2.4. Moreover, it is important that an agent can make inferences from its state to aid its decision-making (reasoning), as follows:

Definition 5.10 (assumptions and inferences) *We denote that an agent x can legitimately assume $[L]$ or its negation $\neg[L]$ as $\vdash^x_{asm} [L]$ or $\vdash^x_{asm} \neg[L]$ respectively, and that the inference $[L]$ or its negation $\neg[L]$ follows from x's state as $\vdash^x [L]$ or $\vdash^x \neg[L]$ respectively, where $[L]$ and $\neg[L]$ are sentences in a meta-language $\mathcal{ML}$ (see Definition 5.11).*

Definition 5.11 (meta-language for assumptions and inferences) *The meta-language $\mathcal{ML}$ consists of all sentences of the form $[L]$ and $\neg[L]$ where $[L]$ takes one of the following forms (for $Y \in names(AS)$, $R \in \mathcal{R}$, $Rs \subseteq \mathcal{R}$):*

- *[Y has R];*
- *[the r.r.p. of Y is solved with R];*
- *[the r.r.p. of Y is solved];*
- *[the r.r.p. of Y would* still *be solved or unsolved by simply giving away R];*
- *[after giving away each resource in Rs, the r.r.p. of x would* no longer *be solved].*

An inference $\vdash^x [Y \text{ has } R]$ or $\vdash^x \neg[Y \text{ has } R]$, where $Y \neq x$, could be legitimate, for example, whenever $has(Y, R) \in B(x)$ or $has(Y, R) \notin B(x)$ respectively. Further examples of the conditions under which inferences could be legitimate for an agent can be found in Appendix C. Generally, exactly when an assumption or inference (by means of $\vdash^x_{asm}$ or $\vdash^x$ respectively) is legitimate depends on the underlying argumentation framework of choice. We do however expect certain consistency properties of $\vdash^x_{asm}$ and $\vdash^x$, as follows:

Postulate 5.1 (Consistency Properties of $\vdash^x_{asm}$ and $\vdash^x$) *For some sentence $[L] \in \mathcal{ML}$, the definition of $\vdash^x_{asm}$ and $\vdash^x$ should ensure the following consistency properties:*

- *It should not be possible that $\vdash^x [L]$ and $\vdash^x \neg[L]$;*
- *It should not be possible that $\vdash^x_{asm} [L]$ and $\vdash^x \neg[L]$.*

We have assumed thus far that the agent's beliefs are necessarily consistent (see Definition 3.1). We will show in Chapter 7 how an argumentation framework can be used to achieve the desired consistency properties of Postulate 5.1 even when the agent's beliefs are not necessarily consistent. We will show how the argumentation framework of Chapter 7 can be used to resolve

inconsistencies in the agent's beliefs, as well as for allowing an agent to make assumptions and inferences in the absence of knowledge.

The procedure `generateRequest(`Y`,Actions)` instantiates an exchange of resources `Actions` to request of an agent Y ($Y \neq x$) if the r.r.p. of x is not solved. By virtue of the exchange x would receive a resource that fulfils its goal (possibly also giving away a resource owned by x), as follows:

Definition 5.12 (`generateRequest(`Y`,Actions)`) *The procedure evaluates to true instantiating an intended recipient of the request Y and an exchange of resources* `Actions` *to request if (1) the r.r.p. of x is not solved and* `fulfils(`R`,`$G(x)$`)` $\in B(x)$ *for some $R \in \mathcal{R}$; (2) $Y \in names(AS)$ and $Y \neq x$; and (3)* `Actions` *takes one of the following two forms:*

(a) {`give(`Y`,`R`,`x`)`}*, if the following conditions hold:*

(i) `has(`Y`,`R`)` $\in B(x)$ *or*

$\vdash^{x}_{asm}$ [*Y has R*]*; and*

(ii) $\vdash^{x}$ [*the r.r.p. of Y would* still *be solved or unsolved by simply giving away R*] *or*

$\vdash^{x}_{asm}$ [*the r.r.p. of Y would* still *be solved or unsolved by simply giving away R*].

(b) {`give(`Y`,`R`,`x`), give(`x`,`R'`,`Y`)`}*, if the following conditions hold:*

(i) `has(`Y`,`R`)` $\in B(x)$*;*

(ii) $\vdash^{x}$ [*the r.r.p. of Y is solved*]*;*

(iii) $\vdash^{x} \neg$[*the r.r.p. of Y would* still *be solved or unsolved by simply giving away R*]*;*

(iv) $\vdash^{x}$ [*the r.r.p. of Y is solved with R'*]*;*

(v) $R' \in Res(x)$*.*

Otherwise, if the above conditions are not met, the procedure evaluates to false.

We note the following property about the `generateRequest` procedure:

Lemma 5.3 (requests generated are Min CR from the requestor's perspective) *A successful request dialogue instance initiated by x regarding* `Actions` *where* `Actions` *is an exchange of resources to request instantiated by the* `generateRequest(`Y`,Actions)` *procedure of Definition 5.12 is such that, from x's perspective (i.e. according to x's state),* `Actions` *is a set of actions comprising a Min CR deal in which x is to increase in utility.*

Proof. Firstly: x would increase utility as a result of `Actions` as its r.r.p. is not solved and it is to receive a resource that solves its r.r.p. as part of the resulting deal. Secondly: Y, the only other agent involved in the deal, would not decrease in utility according to x's state. If `Actions` is of the form {`give(`Y`,`R`,`x`)`} this is because x can legitimately infer or assume that the r.r.p. of Y would *still* be solved or unsolved by simply giving away R. If `Actions` is of the form {`give(`Y`,`R`,`x`), give(`x`,`R'`,`Y`)`} this is because x can infer that Y is to receive a resource R' that solves its r.r.p. Lastly: there is no exchange of resources `Actions'` $\subset$ `Actions` such that `Actions'` is cooperatively rational according to x's state. If `Actions` is of the form {`give(`Y`,`R`,`x`)`} this is trivial. If `Actions` is of the form {`give(`Y`,`R`,`x`), give(`x`,`R'`,`Y`)`} this is because x can infer that {`give(`Y`,`R`,`x`)`} would result in Y's r.r.p. becoming unsolved and that {`give(`x`,`R'`,`Y`)`} would not increase the utility of any of the agents involved. □

The procedure `generateProposal(`Z`,Actions)` instantiates an exchange of resources `Actions` to propose to a set of agents Z if the r.r.p. of x is not solved. By virtue of the exchange x would receive a resource that fulfils its goal (possibly also giving away a resource owned by x), as follows:

Definition 5.13 (`generateProposal(`Z`,Actions)`) *The procedure evaluates to true instantiating a set of intended recipients of the proposal Z and an exchange of resources* `Actions` *to propose if (1) the r.r.p. of x is not solved and* `fulfils(`R_2`,`$G(x)$`)` $\in B(x)$*; (2) $Z \subset names(AS)$ and $x \notin Z$; and (3)* `Actions` *is of the form* {`give(`Y_2`,`R_2`,`Y_1`)`, ..., `give(`Y_{n+1}`,` R_{n+1}`,`Y_n`)`} *for $n > 1$, $\{Y_1, \ldots, Y_{n+1}\} = \{x\} \cup Z$, $Y_1 = x$, $|Z| > 1$, $Y_2, \ldots, Y_n \in Z$, $Y_2, \ldots, Y_n$ all different, $Y_{n+1} \in \{x\} \cup Z$, $Y_n \neq Y_{n+1}$, $R_1, \ldots R_{n+1}$ all different, and satisfying the following conditions:*

(a) for all $Y_i \in \{Y_2, \ldots, Y_n\}$),

(i) $\texttt{has}(Y_i, R_i) \in B(x)$;

(ii) $\vdash^x$ [*the r.r.p. of* Y_i *is solved*];

(iii) $\vdash^x \neg$[*the r.r.p. of* Y_i *would* still *be solved or unsolved by simply giving away* R_i]; *and*

(iv) $\vdash^x$ [*the r.r.p. of* Y_i *is solved with* R_{i+1}];

(b) *(i)* $Y_{n+1} = x$ *and* $R_{n+1} \in Res(x)$; *or*

(ii) $Y_{n+1} \in \{Y_2, \ldots, Y_{n-1}\}$, *and either* $has(Y_{n+1}, R_{n+1}) \in B(x)$ *or* $\vdash^x_{asm}$ [Y_{n+1} *has* R_{n+1}]; *or*

(iii) $Y_{n+1} \notin \{x\} \cup \{Y_2, \ldots, Y_{n-1}\}$, *and either* $has(Y_{n+1}, R_{n+1}) \in B(x)$ *or* $\vdash^x_{asm}$ [Y_{n+1} *has* R_{n+1}], *and either*

- $\vdash^x$ [*the r.r.p. of* Y_{n+1} *would* still *be solved or unsolved by simply giving away* R_{n+1}], *or*
- $\vdash^x_{asm}$ [*the r.r.p. of* Y_{n+1} *would* still *be solved or unsolved by simply giving away* R_{n+1}].

Otherwise, if the above conditions are not met, the procedure evaluates to false.

We note the following property about the `generateProposal` procedure:

Lemma 5.4 (proposals generated are Min CR from the proposer's perspective) *A successful proposal dialogue instance initiated by* x *regarding* `Actions` *where* `Actions` *is an exchange of resources to propose instantiated by the* `generateProposal(`Z`,Actions)` *procedure of Definition 5.13 is such that, from* x*'s perspective (i.e. according to* x*'s state),* `Actions` *is a set of actions comprising a Min CR deal in which* x *is to increase in utility.*

Proof. Firstly: x would increase utility as a result of `Actions` as its r.r.p. is not solved and it is to receive a resource that solves its r.r.p. as part of the resulting deal. Secondly: none of the agents involved in the deal ($\{x\} \cup Z$) would decrease in utility according to x's state. This

is because for each `give(`Y`,`R`,`Y'`)` $\in$ `Actions` we have $Y \in \{x\} \cup Z$ and x can legitimately infer or assume that R does not solve the r.r.p. of Y or that Y has some other resource that solves its r.r.p. or that Y is to receive some other resource that solves its r.r.p. as part of the deal. Lastly: there is no exchange of resources `Actions'` $\subset$ `Actions` such that `Actions'` is cooperatively rational according to x's state. This is because every atom `give(`Y`,`R`,`Y'`)` $\in$ `Actions` is necessary for `Actions` to be cooperatively rational according to x's state, as follows: leaving out `give(`Y`,`R`,`x`)` $\in$ `Actions` means no agent increases utility in the resulting deal according to x and leaving out any `give(`Y`,`R`,`Y'`)` $\in$ `Actions` ($Y' \in Z$) means Y' decreases utility in the resulting deal according to x. □

The procedure `acceptableExchange(Actions)` determines whether the exchange of resources in `Actions` is acceptable for x. We assume that agents are self-interested but generous, and so we define this procedure so that x agrees to give away the resources required of it in `Actions` (i.e. $\{r \mid$ `give(`x`,`r`,_ )` $\in$ `Actions`$\}$) that it has if either its r.r.p. is *still* solved or unsolved by doing so or it receives a resource in the exchange that fulfils its goal, as follows:

Definition 5.14 (`acceptableExchange(Actions)`) *The procedure evaluates to true if given* $Rs = \{ R \mid$ `give(`x`,`R`,_)` $\in$ `Actions` $\}$*: (1) for each* $R \in Rs$, $R \in Res(x)$*; and (2) any one of the following conditions holds:*

(a) for all $R \in Rs$, `fulfils(`R`,`$G(x)$`)` $\notin B(x)$*, or*

(b) there is some $R \notin Rs$ *such that* $R \in Res(x)$ *and* `fulfils(`R`,`$G(x)$`)` $\in B(x)$*, or*

(c) `give(_,`R'`,`x`)` $\in$ `Actions` *and* `fulfils(`R'`,`$G(x)$`)` $\in B(x)$.

Otherwise, if the above conditions are not met, the procedure evaluates to false.

We note the following property about the `acceptableExchange` procedure:

Lemma 5.5 (agents responding with acceptance do not decrease in utility) *The agent x responding to a request or proposal regarding* `Actions` *with acceptance according to*

the `acceptableExchange(Actions)` *procedure of Definition 5.14 results in a deal in which* x *does not decrease in utility.*

Proof. A deal comprising a set of actions `Actions` is such that x only gives away resources that do not solve its r.r.p. (Condition (2a) of Definition 5.14), or x has some other resource which it is not giving away as part of the deal and which solves its r.r.p. (Condition (2b) of Definition 5.14), or x is receiving a resource that solves its r.r.p. as part of the deal (Condition (2c) of Definition 5.14). Each of these conditions implies that x is not to decrease utility in the deal. □

The procedure ¬`acceptableExchange(Actions,ReasonOut)` determines whether the exchange of resources in `Actions` is *not* acceptable for x and instantiates an argument (`ReasonOut`) justifying the decision, which may be because x does not have the resource R requested of it or x's r.r.p. will no longer be solved if it agrees to `Actions`, as follows:

Definition 5.15 (¬`acceptableExchange(Actions,ReasonOut)`**)** *The procedure evaluates to true instantiating an argument* `ReasonOut` *where* `ReasonOut` *takes one of the following two forms:*

(a) `<`¬`has(`x`,`R`), Support>`*: if* `give(`x`,`R`,_ )` ∈ `Actions`*, and* $R \notin Res(x)$*; or*

(b) `<`[*after giving away each resource in* Rs*, the r.r.p. of* x *would* no longer *be solved*]`, Support>`*: if* $Rs = \{ R \mid$ `give(`x`,`R`,_)` ∈ `Actions` $\}$ *and (i)* $\vdash^x$ [*after giving away each resource in* Rs*, the r.r.p. of* x *would* no longer *be solved*] *and (ii) there is no* `give(_,`R'`,`x`)` ∈ `Actions` *such that* `fulfils(`R'`,`$G(x)$`)` ∈ $B(x)$.

Otherwise, if neither of the above conditions are met, the procedure evaluates to false.

We ignore in this chapter the exact information contained in the `Support` of the argument and how it is represented. Intuitively it would contain some of the agent's beliefs as well as information about its allocted and needed resources. For example, in case (a) of Definition 5.15

above, the `Support` would be expected to represent the information $R \notin Res(x)$. We show in Chapter 7 how the argument `ReasonOut` in Definition 5.15 can be generated and instantiated concretely for a given general-purpose argumentation framework.

The procedure `exchangeResources(Actions)` revises x's owned resources and belief set as to which agents have which resources according to `Actions`, as follows:

Definition 5.16 (`exchangeResources(Actions)`) *The procedure revises x's owned resources $Res(x)$ and belief set $B(x)$ to produce a new set of owned resources $Res'(x)$ and beliefs $B'(x)$ as follows:*

$$Res'(x) = Res(x) \cup \{\, r \mid \texttt{give}(_, r, x) \in \texttt{Actions} \,\} - \{\, r \mid \texttt{give}(x, r, _) \in \texttt{Actions} \,\}$$

$$B'(x) = B(x) \cup \{\, \texttt{has}(y, r) \mid \texttt{give}(_, r, y) \in \texttt{Actions},\ y \neq x \,\}$$

$$- \{\, \neg\texttt{has}(y, r) \mid \texttt{give}(_, r, y) \in \texttt{Actions},\ y \neq x \,\}$$

$$- \{\, \texttt{has}(y, r) \mid \texttt{give}(y, r, _) \in \texttt{Actions},\ y \neq x \,\}$$

Note that the new set of owned resources computed is equivalent to the *assignment of resources to an agent following a successful dialogue instance* as in Definition 5.5.

The procedure `reviseBeliefs(`Y`,Reason)` modifies x's belief base given an argument `Reason` uttered to it by some agent Y. We define this procedure and the other procedures above concretely in Chapter 7 by means of a general-purpose argumentation framework. We show also how such an argumentation framework can be used to resolve contradictory beliefs (inconsistencies in the agent's belief set) so that explicit removing (cleaning) of beliefs to enforce consistency is not necessary.

5.5 Examples of Generated Dialogue Instances

We present in this section two examples demonstrating how agents could proceed with communication using the communication rules and procedures defined in this chapter. Note that we do not assume any explicit turn-taking between the agents during communication, such that agents are freely (simultaneously) initiating and responding to dialogues (see Example 5.4). In order to ensure agents do not deadlock (in the sense that each agent is waiting for a response from another and none can proceed), the communication rules pertaining to initiating dialogue (*CR1–CR2*) have least priority, such that in a situation where the In and P conditions hold for one of the communication rules *CR3–CR15*, then a rule in *CR3–CR15* will be chosen for application and not a rule in *CR1–CR2*. This ensures that an agent does not initiate dialogue until/unless it has processed all incoming dialogue moves and could be achieved, for example, by means of adding a condition requiring $M(x) \neq \emptyset$ in the `valid(initiateDialogue(Actions))` procedure. Thus also, by means of this prioritisation, agents initiate dialogues with up-to-date beliefs about the other agents, having processed all incoming dialogue moves.

We show firstly how the request dialogue instance of Example 5.1 could be generated following an unsuccessful request dialogue instance by means of the communication rules and procedures defined in Sections 5.3 and 5.4 respectively.

Example 5.5 (Unsuccessful and Successful Request Dialogue Instances) *Given the agent system* $AS = (\mathcal{A}, \mathcal{G}, \mathcal{R})$ *of Example 3.1 with* $\mathcal{A} = \{\langle ag1, Res(ag1), G(ag1), B(ag1)\rangle, \langle ag2, Res(ag2), G(ag2), B(ag2)\rangle, \langle ag3, Res(ag3), G(ag3), B(ag3)\rangle\}$, $\mathcal{G} = \{g1, g2, g3\}$, $\mathcal{R} = \{r1, r2, r3\}$, *plans* Pl *as depicted in Figure 5.3, and such that*

$Res(ag1) = \{r1\}$,	$Res(ag2) = \{r2\}$,	$Res(ag3) = \{r3\}$,
$G(ag1) = g1$,	$G(ag2) = g2$,	$G(ag3) = g3$,
$B(ag1) = \{$`isAgent(`$ag2$`),` `isAgent(`$ag3$`)}` $\cup\ Pl$	$B(ag2) = \{$`isAgent(`$ag1$`),` `isAgent(`$ag3$`)}` $\cup\ Pl$	$B(ag3) = \{$`isAgent(`$ag2$`),` `isAgent(`$ag2$`)}` $\cup\ Pl$

we show how, over successive time points t_0 to t_5, $ag2$ could go about obtaining a resource ($r1$) that fulfils its goal ($g2$) thus solving its own r.r.p. and, consequently, the r.r.p. of the agent system as a whole (the solution as identified in Example 3.2):

- t_0: *Since $ag2$ has no beliefs about the goals and/or resources of both $ag1$ and $ag3$, $ag2$ can* assume *about either $ag1$ or $ag3$ that it has and does not need $r1$. Applying* CR1 *for the instantiation of $Y = ag1$, $ag2$ utters*

  `request(`$ag2$`, `$ag1$`, {give(`$ag1$`,`$r1$`,`$ag2$`)})`

- t_1: *Since $r1$ solves the r.r.p. of $ag1$ and giving away $r1$ would mean the r.r.p. of $ag1$ is no longer solved, $ag2$'s request is not acceptable for $ag1$. Applying* CR4 *$ag1$ utters*

  `refuse(`$ag1$`, `$ag2$`, {give(`$ag1$`,`$r1$`,`$ag2$`)},`
  `<[`*after giving away each resource in* $\{r1\}$*, the r.r.p. of* $ag1$ *would* no longer *be solved*`],`
  `Support>`

  We ignore here the exact form `Support` *takes and how exactly $ag2$ would revise its beliefs according to the argument provided (to be defined concretely in Chapter 7). Assume* `Support` *contains the information that $ag1$ has $r1$ but its goal is $g1$ (and $r1$ fulfils the goal $g1$).*

- t_2: *$ag2$ receives $ag1$'s refusal by* CR8. *Assume $ag2$ revises $B(ag2)$ to include* `has(`$ag1$`,`$r1$`)` *and* `goal(`$ag1$`,`$g1$`)`.

- t_3: *Knowing now that $r2$ solves $ag1$'s r.r.p., $ag2$ applies* CR1 *again for $Y = ag1$ and makes a refined request such that the r.r.p. of both agents would be solved*

  `request(`$ag2$`, `$ag1$`, {give(`$ag1$`,`$r1$`,`$ag2$`),give(`$ag2$`,`$r2$`,`$ag1$`)})`

- t_4: *Now, $ag1$ can agree to $ag2$'s request because $ag1$'s r.r.p. would still be solved after the exchange. Applying* CR3 *$ag1$ utters*

  `agree(`$ag1$`, `$ag2$`, {give(`$ag1$`,`$r1$`,`$ag2$`),give(`$ag2$`,`$r2$`,`$ag1$`)})`

  and revises its state according to

  `exchangeResources({give(`$ag1$`,`$r1$`,`$ag2$`), give(`$ag2$`,`$r2$`,`$ag1$`)})`

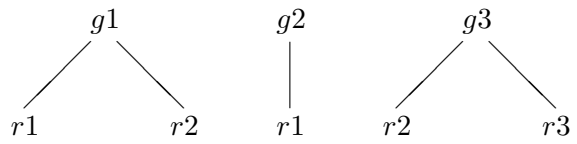


Figure 5.3: Plans Pl in Example 5.5 (and as in Example 3.1).

such that $Res(ag1) = \{r2\}$.

- t_5*: ag2 receives ag1's acceptance by* CR7 *and revises its state according to*

  `exchangeResources({give(`$ag1$`,`$r1$`,`$ag2$`),give(`$ag2$`,`$r2$`,`$ag1$`)})`

  such that $Res(ag2) = \{r1\}$.

  At this point, the r.r.p. of both agents is solved and neither agent initiates further dialogue.

We show next how the proposal dialogue instance of Example 5.2 could be generated by means of the communication rules and procedures defined in Sections 5.3 and 5.4 respectively.

Example 5.6 (Successful Proposal Dialogue Instance) *Given the agent system of Example 3.1 (and of the previous example) but such that the resource allocation is as follows:*

$Res(ag1) = \{r1\}$	$Res(ag2) = \{r3\}$	$Res(ag3) = \{r2\}$

i.e. $ag2$ *and* $ag3$ *have resources* $r3$ *and* $r2$ *respectively, and not* $r2$ *and* $r3$ *respectively as in the previous example. Assume also that* $ag2$ *has beliefs as to the owned resources and goals of agents* $ag1$ *and* $ag3$ *(i.e.* `has(`$ag1$`,`$r1$`)`, `goal(`$ag1$`,`$g1$`)`, `has(`$ag3$`,`$r3$`)`, `goal(`$ag3$`,`$g3$`)` $\in B(ag2)$*).*[2] *We show how, over successive time points* t_0 *to* t_6*,* $ag2$ *could go about obtaining a resource (*$r1$*) that fulfils its goal (*$g2$*) thus solving its own r.r.p. and, consequently, the r.r.p. of the agent system as a whole:*

- t_0*: ag2 is able to generate a proposal which it believes would leave the r.r.p. of all agents solved. Applying* CR2 *it utters*

[2]This could be, for example, a result of arguments (`Reason`) received in previous unsuccessful request dialogues initiated by $ag2$ with $ag1$ and $ag3$.

---

propose($ag2$, {$ag1$,$ag3$},
{give($ag2$,$r3$,$ag3$),give($ag3$,$r2$,$ag1$),give($ag1$,$r1$,$ag2$)})

- t_1: *$ag2$'s proposal is acceptable for $ag1$. Applying* CR5 *it utters*

---

accept_proposal($ag1$, $ag2$,
{give($ag2$,$r3$,$ag3$),give($ag3$,$r2$,$ag1$),give($ag1$,$r1$,$ag2$)})

- t_1: *$ag2$'s proposal is acceptable for $ag3$ also. Applying* CR5 *it utters*

---

accept_proposal($ag3$, $ag2$,
{give($ag2$,$r3$,$ag3$),give($ag3$,$r2$,$ag1$),give($ag1$,$r1$,$ag2$)})

- t_2: *Applying* CR9, *$ag2$ receives the* accept_proposal *utterance of $ag1$.*

- t_3: *Applying* CR9, *$ag2$ receives the* accept_proposal *utterance of $ag3$.*

- t_4: *Applying* CR11, *$ag2$ utters*

---

confirm($ag2$, {$ag1$,$ag3$},
{give($ag2$,$r3$,$ag3$),give($ag3$,$r2$,$ag1$),give($ag1$,$r1$,$ag2$)})

---

and revises its state according to

exchangeResources({give($ag2$,$r3$,$ag3$),give($ag3$,$r2$,$ag1$),give($ag1$,$r1$,$ag2$)})

- t_5–t_6: *Applying* CR14 *(separately), $ag1$ and $ag3$ receive the* confirm *utterance of $ag2$ and revise their state according to*

exchangeResources({give($ag2$,$r3$,$ag3$),give($ag3$,$r2$,$ag1$),give($ag1$,$r1$,$ag2$)})

At this point, the r.r.p. of all agents is solved.

The examples above demonstrate the kinds of dialogue instances (sequences) that agents would generate according to the negotiation protocols and policy defined in this chapter. It remains to define concretely the procedures found in the communication rules and the arguments that are generated and exchanged accordingly. This will be the focus of Chapter 7 for a general-purpose argumentation framework. We discuss next some properties of the negotiation policy that are independent of the argumentation framework used to ground the procedures.

5.6 Properties of Generated Dialogue Instances

We present in this section some properties of the communication policy based on the communication rules and procedure definitions as given in Sections 5.3 and 5.4. In the lemmas and theorems that follow we assume an agent system $AS = (\mathcal{A}, \mathcal{G}, \mathcal{R})$ consisting of agents using the communication rules *CR1–CR15* according to the definitions of the procedures as given in Definitions 5.12–5.16. We will say that an agent system results from a generated dialogue instance as follows:

Definition 5.17 (agent system resulting from a generated dialogue instance) *An agent system $AS' = (\mathcal{A}', \mathcal{G}, \mathcal{R})$ results from a dialogue instance generated by the communication rules* CR1–CR15 *if each agent in $\mathcal{A}$ is revised (producing $\mathcal{A}'$) according to the revisions of the communication rules applied in generating the dialogue instance.*

Note that the definitions of the control procedures occurring in the communication rules (see Appendix B) are such that an agent's resources do not change once it has initiated a request or accepted a proposal until it receives a response (agree/refuse or confirm/disconfirm respectively) and that dialogue instance ends, i.e. the agent is not engaged in two dialogue instances simultaneously in which its resources could change. We assume also that the `reviseBeliefs` procedure is defined such that an agent x adds beliefs regarding an agent Y and a resource R or a set of resources Rs such that $\vdash^x$ [the r.r.p. of Y is solved] and $\vdash^x \neg$[the r.r.p. of Y would *still* be solved or unsolved by simply giving away R] hold true only if x has been told by Y itself as such by means of an argument uttered by Y as part of a dialogue move (dialogue instance).

Firstly, dialogue instances generated are such that no agents decrease in utility, as follows:

Lemma 5.6 (agents' utilities do not decrease in a dialogue instance) *An agent system AS' resulting from a dialogue instance generated by the communication rules* CR1–CR15 *according to the procedures of Definitions 5.12–5.16 is such that for all agents $x \in names(AS)$, $u_{AS}(x) \leq u_{AS'}(x)$.*

Proof. For all agents x not involved in the dialogue instance (i.e. neither initiator nor responder in the dialogue instance), none of the communication rules will have been applied by x in generating the dialogue instance. Therefore x's resources are unchanged and $u_{AS}(x) \leq u_{AS'}(x)$. Also, if $u_{AS}(x) = 0$, then, trivially, $u_{AS}(x) \leq u_{AS'}(x)$. We show that this holds for an agent x involved in the dialogue instance and such that $u_{AS}(x) = 1$. Firstly, note that x's owned resources only change by means of the `exchangeResources` procedure which is called by x in ending a successful dialogue instance, i.e. communications rules *CR3* or *CR7* if x is the initiator or responder (respectively) of a successful request dialogue instance, and *CR11* or *CR14* if x is the initiator or responder (respectively) of a successful proposal dialogue instance. Hence these are the only cases we need to consider. In the case of x initiating a request or proposal dialogue instance, we know by Lemmas 5.3 and 5.4 that the resulting exchange of resources will not decrease x's utility. In the case of x responding to a request or proposal with acceptance, we know by Lemma 5.5 that the resulting exchange will not decrease x's utility. □

As a consequence of the result above we have that an agent's utility will never decrease as a result of any dialogue instance generated and, in particular, an agent that has one or more resources that solves its r.r.p., at any point thereafter will always have in its possession at least one resource that solves its r.r.p..

Secondly, an agent will never obtain another resource that solves its r.r.p. once it has one, whether actively as an initiator of a dialogue instance or accidently as a responder of a dialogue instance, except in exchange for a resource r it has if r is the only resource it has that solves its r.r.p., as follows:

Lemma 5.7 (happy agent does not obtain more resources that solve its r.r.p.) *Let* $Rs = \{\ R \mid$ `fulfils(`R`,`$G(x)$`)` $\in B(x),\ R \in Res(x)\ \}$ *for an agent* $x \in names(AS)$ *such that* $|Rs| \geq 1$*, i.e.* x *has at least one resource that solves its r.r.p. A successful dialogue instance is generated regarding* `Actions` *where* `give(_,`r`,`x`)` $\in$ `Actions` *for some* r *that solves the r.r.p. of* x *only if* $|Rs| = 1$ *and* `give(`x`,`r'`,_ )` $\in$ `Actions` *for some* $r' \in \mathcal{R}$ *such that* $r' \in Rs$*.*

Proof. Firstly, x cannot initiate a request or proposal dialogue instance according to communications rules *CR1* and *CR2* (or, rather, the definitions of the procedures `generateRequest` (Definition 5.12) and `generateProposal` (Definition 5.13)) since $|Rs| \geq 1$, i.e. the r.r.p. of x is solved. The only case to consider then is when x takes the role of responder in a dialogue instance. Suppose y ($y \neq x$) is the agent initiating the dialogue instance. By Definitions 5.12 and 5.13, y will only initiate a dialogue instance such that `give(_,`r`,`x`)` $\in$ `Actions` for some resource r that solves the r.r.p. of x if `give(`x`,`r''`,_ )` $\in$ `Actions` for some $r'' \in \mathcal{R}$ and $\vdash^y$ $\neg$[the r.r.p. of x would *still* be solved or unsolved by simply giving away r'']. Indeed y would only make such an inference if told by x by means of an argument in a previous (unsuccessful) dialogue instance, and x would only make such an utterance in case $|Rs| = 1$ and $r'' = r'$. $\square$

The above lemma is useful for proving that successful request and proposal dialogue dialogue instances generated by agents result in Min CR deals, as in the following two theorems:

Theorem 5.1 (successful request dialogue instances generated are Min CR) *A successful request dialogue instance regarding* `Actions` *generated by an initiating agent x and a responding agent y is such that* `Actions` *is a set of actions comprising a Min CR deal $\delta = (Res, Res')$ where Res is the allocation before the dialogue instance and Res' is the allocation after the dialogue instance.*

Proof. First of all, we know by Lemma 5.3 that an exchange of resources `Actions` to request generated by x according to `generateRequest(`Y`,Actions)` ($Y = y$) would increase x's utility. We know also by definition of `generateRequest` (Definition 5.12) that

- `Actions` is a non-empty set of atoms of the form `give(`A`,`R`,`B`)` as in Definition 5.2, i.e. $A, B \in \{x, y\}$, $A \neq B$, $R \in \mathcal{R}$, there is at least one `give(`A`,_,_ )` or `give(_,_,`A`)` in `Actions` for each $A \in \{x, y\}$, and there is at most one `give(_,`R`,_ )` in `Actions` for each $R \in \mathcal{R}$; and

- $r \in Res(x)$ for each `give(`x`,`r`,_ )` $\in$ `Actions`.

Secondly, we know by Lemma 5.5 that if `acceptableExchange(Actions)` evaluates to true for an agent, then that agent (y in this case) does not decrease in utility as a result of `Actions`. We know also by definition of `acceptableExchange(Actions)` (Definition 5.14) that $r \in Res(y)$ for each `give(`y`,`r`,_ )` $\in$ `Actions`. Thus every resource to be given away by an agent (i.e. x and y) in `Actions` is held by that agent. Lastly, x and y generate $Res'(x)$ and $Res'(y)$ according to `exchangeResources(Actions)` (Definition 5.16) which is equivalent to Definition 5.5. Moreover, $Res'(z) = Res(z)$ for all agents $z \in names(AS)$ such that $z \notin \{x, y\}$. Thus, by Corollary 5.1, $\delta = (Res, Res')$ is a deal. It remains to show that δ is a Min CR deal. Since, by Lemma 5.2, the only agents involved in the deal are x and y, and x increases in utility and y does not decrease in utility, then δ is a CR deal. If |`Actions`| = 1, then δ is trivially a Min CR deal. If |`Actions`| > 1, then, according to Definition 5.12, `Actions` is of the form {`give(`y`,`r`,`x`),` `give(`x`,`r'`,`y`)`} (for some $r, r' \in \mathcal{R}$), $\vdash^x$ [the r.r.p. of y is solved] and $\vdash^x \neg$[the r.r.p. of y would *still* be solved or unsolved by simply giving away r]. For x to make such inferences, it must have been told as such by y in a previous dialogue instance. Since, by Lemma 5.7, y would never obtain a second resource that solves its r.r.p., then the set of actions {`give(`y`,`r`,`x`)`} $\subset$ `Actions` does not comprise a CR deal. Also, the set of actions {`give(`x`,`r'`,`y`)`} $\subset$ `Actions` does not comprise a CR deal since no agents would increase utility. Therefore, `Actions` is a set of actions comprising a Min CR deal. □

Theorem 5.2 (successful proposal dialogue instances generated are Min CR) *A successful proposal dialogue instance regarding* `Actions` *generated by an initiating agent x and set of responding agents Z is such that* `Actions` *is a set of actions comprising a Min CR deal $\delta = (Res, Res')$ where Res is the allocation before the dialogue instance and Res' is the allocation after the dialogue instance.*

Proof. First of all, we know by Lemma 5.4 that an exchange of resources `Actions` to propose generated by x according to `generateProposal(`Z`,Actions)` would increase x's utility. We know also by definition of `generateProposal(`Z`,Actions)` (Definition 5.13) that

- `Actions` is a non-empty set of atoms of the form `give(`A`,`R`,`B`)` as in Definition 5.3, i.e. $A, B \in \{x\} \cup Z$, $A \neq B$, $R \in \mathcal{R}$, there is at least one `give(`A`,_,_ )` or `give(_,_,`A`)` in

`Actions` for each $A \in \{x\} \cup Z$, and there is at most one `give(_,`R`,_ )` in `Actions` for each $R \in \mathcal{R}$; and

- $r \in Res(x)$ for each `give(`x`,`r`,_ )` $\in$ `Actions`.

Secondly, we know by Lemma 5.5 that if `acceptableExchange(Actions)` evaluates to true for an agent, then that agent (each of $y \in Z$ in this case) does not decrease in utility as a result of `Actions`. We know also by definition of `acceptableExchange(Actions)` (Definition 5.14) that $r \in Res(y)$ for each `give(`y`,`r`,_ )` $\in$ `Actions` where $y \in Z$. Thus every resource to be given away by an agent (i.e. $y \in \{x\} \cup Z$) in `Actions` is held by that agent. Lastly, each agent $y \in \{x\} \cup Z$ generates $Res'(y)$ according to `exchangeResources(Actions)` (Definition 5.16) which is equivalent to Definition 5.5. Moreover, $Res'(y) = Res(y)$ for all agents $y \in names(AS)$ such that $y \notin \{x\} \cup Z$. Thus, by Corollary 5.1, $\delta = (Res, Res')$ is a deal. It remains to show that δ is a Min CR deal. Since, by Lemma 5.2, the only agents involved in the deal are $x \cup Z$, and x increases in utility and none of the agents in Z decrease in utility, then δ is a CR deal. According to Definition 5.13, `Actions` is of the form {`give(`Y_2`,`R_2`,`Y_1`), ..., give(`Y_{n+1}`,`R_{n+1}`,`Y_n`)`} ($n > 1$, $Y_1 = x$), $\vdash^x$ [the r.r.p. of Y_i is solved] (for all $1 < i \leq n$) and $\vdash^x \neg$[the r.r.p. of Y_i would *still* be solved or unsolved by simply giving away R_i] (for all $1 < i \leq n$). For x to make such inferences, it must have been told as such by each $Y_2, \ldots, Y_n$ in previous dialogue instances, and, by Lemma 5.7, none of the agents $Y_2, \ldots, Y_n$ could have obtained a second resource that solves its r.r.p. since then. Thus the set of actions {`give(`Y_2`,`R_2`,`Y_1`), ..., give(`Y_i`,`R_i`,`Y_{i-1}`)`} $\subset$ `Actions` (for all $2 < i \leq n$) cannot be cooperatively rational. Also, the r.r.p. of all agents in Z is solved and believed by x to be solved, and, by Lemma 5.6, will never be unsolved. Thus, any (strict) subset of `Actions` that does not contain {`give(`Y_2`,`R_2`,`x`)`} cannot be cooperatively rational since no agents would increase utility in the resulting deal. Therefore, `Actions` is a set of actions comprising a Min CR deal. □

The above two theorems are sufficient to prove that all successful dialogue instances generated by agents using the communication rules and procedures defined in this chapter result in Min CR deals, as follows:

Corollary 5.2 (successful dialogue instances generated are Min CR) *A successful dialogue instance regarding* `Actions` *generated by an initiating agent x and set of responding agents Z is such that* `Actions` *is a set of actions comprising a Min CR deal $\delta = (Res, Res')$ where Res is the allocation before the dialogue instance and Res' is the allocation after the dialogue instance.*

Proof. This is an immediate consequence of Theorems 5.1 and 5.2 since dialogue instances generated by agents are either request or proposal dialogue instances. □

Lastly, by means of the result above we have the following result regarding dialogue sequences generated by means of the communication rules and procedures:

Corollary 5.3 (successful dialogue sequences generated end with Min CR deals) *A successful dialogue sequence generated wrt an agent x and a resource r is such that the last dialogue instance generated in the sequence results in a Min CR deal.*

The result above is an immediate consequence of Definitions 5.6 and 5.9, and Corollary 5.2.

5.7 Discussion

In multi-agent resource reallocation problems of the kind we consider in this thesis where successive allocations of resources are to be arrived at (given an initial starting allocation of resources) by the participating agents themselves with minimal disclosure of information, it is vital that agents can communicate and agree on the exchanges of resources required, as well as exchange information between one another as necessary. We have presented in this chapter a mechanism for doing this exactly. Given representation of a resource reallocation problem in terms of the agents, goals and resources involved according to the framework of Chapter 3, we identify in this chapter how agents can interact and arrive at decisions (deals of the kind described in Chapter 4) leading incrementally to an arrangement (allocation) where resources are best utilised in the agent system, taking into consideration the initial setup. We have

illustrated at various points in this chapter the kinds of dialogue that are possible between agents and the communication exchanged between agents in arriving at exchanges of resources involving two or three agents. Note that our illustrations consist of two or three agents only for clarity of illustration and agent systems and deals (exchanges of resources) in our framework are not restricted to consist of just two or three agents. On the contrary we envision that the agents and resulting dialogues will take place in (Grid computing, scheduling/timetabling, disaster response etc) systems consisting of large numbers of agents as described in Chapter 1, Section 1.2 (and also Chapter 3, Section 3.5). In addition, note that the agent decision-making and communication framework we present in this chapter is truly distributed, in accordance with the kinds of problems we aim to solve, allowing agents to simultaneously engage in dialogue instances as initiators and responders, and not relying on turn-taking nor a central controlling agent to facilitate the communication between agents.

5.8 Conclusion

The results of Endriss et al. [EMST06] and indeed Chapter 4 contain sufficiency results for agents to reach "optimal" allocations of resources. We have expanded upon this in this chapter to address the important and complex problem of actually finding the necessary deals (exchanges of resources). We have presented in this chapter a distributed negotiation process (policy) in which agents can identify and generate deals that are *minimally* cooperative rational whenever such deals exist. The negotiation policy which we present allows agents to intermittently generate and request/propose exchanges of resources according to the different contract types identified by Sandholm [San98] as each being necessary to avoid some of the local optima that the others allow, but not sufficient in and of itself for reaching the global optimum: 'O-contracts' (a single resource is given by one agent to another), 'S-contracts' (single resources are exchanged between two agents), 'M-contracts' (three or more agents each give away a single resource) and 'OCSM-contracts' (a combination of the different contract types). The need for 'C-contracts' (more than one resource given by one agent to another) does not arise in our setting since single resources fulfil single goals. In extending our framework to allow

for multiple resources to fulfil goals, this additional contract type will need to be catered for.

We address a number of the challenges highlighted by Rahwan et al. [RRJ+03] in presenting our generative Argumentation-Based Negotiation (ABN) framework. Firstly, we present a rich communication and domain language that enables agent's to exchange meta-information, as well as information that describes outcomes. Namely, agent's are able to express reasons (arguments) regarding their beliefs and goals, as well as requests/proposals and responses regarding certain exchanges of resources. Secondly, we specify a protocol for each of our two types of ABN dialogue (requests and proposals), constraining the use of the language and governing the interaction of participants. We do so by specifying at each stage of a dialogue instance who is allowed to say what. Our protocols are based solely on the utterances made by agents rather than to the agents' internal mental attitudes, hence constituting a public syntax and pragmatics that can be enforced by an external regulator. Thirdly, we define the agents' internal specification (communication rules and associated procedures) to conform to the protocols. In particular, agents are defined such that dialogues terminate, and terminate successfully whenever successful termination is possible. Moreover, in addition to interpreting and evaluating incoming requests/proposals regarding exchanges of resources and generating outgoing requests/proposals and responses, as is minimally required in negotiation, our agents are capable of generating and evaluating accompanying arguments (and updating their mental states accordingly).

Note that the procedures in the communication rules and what exactly it means for an agent to make assumptions, as well as the exact form arguments take and how these can be generated, has been left deliberately open in the negotiation policy defined in this chapter. We will show in Chapter 7 how the negotiation policy can be grounded by means of a general-purpose argumentation framework defined in the next chapter, and also what role such an argumentation framework can take to resolve an agent's contradictory beliefs so that explicit removing (cleaning) of beliefs (assumed in this chapter) is not necessary.

Chapter 6

Assumption-Based Argumentation for Communicating Agents

In this chapter we extend the Assumption-Based Argumentation (ABA) framework of Dung et al. [DKT09] (see Section 2.4.2) for multi-agent systems, such that agents can build acceptable arguments according to a particular notion of acceptabilty, but which are also communicable and can be understood by other agents in the agent system. We term our new framework "Agent ABA" (AABA). We will use AABA in the next chapter to ground the argumentative negotiation policy presented in the previous chapter.

6.1 Introduction

ABA is a general-purpose framework for argumentation, where arguments are defined as *backward deductions* (using sets of *rules* in an underlying logic) supported by sets of *assumptions*, and the notion of *attack* amongst arguments is reduced to that of *contrary* of assumptions. Existing computational models (dispute derivations), such as that of Dung et al. [DMT07], allow to determine the "acceptability" of claims under the semantics of credulous, admissible extensions as well as under two sceptical semantics (of grounded and ideal extension). The dispute derivations find a set of assumptions, to defend a given claim, by starting from an initial

set of assumptions that supports an argument for the claim and adding defending assumptions incrementally to counter-attack all attacks.

It is important to note that the dispute derivations of Dung et al. [DMT07] take place within the mind of a single agent, between fictional proponent and opponent. Also, the resulting arguments (sets of assumptions) are not built to be communicated to other agents, but rather for an agent to determine within itself whether a claim is acceptable. In this chapter, we focus on generalising the computational model for admissibility, called *AB-dispute derivations* [DMT07] (see Section 2.4.2). The computational model we present is still a dispute derivation within the mind of a single agent but such that the resulting arguments can be readily communicated to other agents, similarly to enthymemes as presented by Hunter [Hun07]. Hunter argues in [Hun07] that arguments presented by real-world agents normally only explicitly represent some of the premises for entailing their claim and/or they do not explicitly state their claim. It is argued that this is because there is some common knowledge that can be assumed by a proponent of an argument and the recipient of it. This allows the proponent of an argument to encode it by ignoring the common knowledge, and it allows a recipient of an argument to decode it by drawing on the common knowledge.

The chapter is structured as follows: Section 6.2 presents our generalised ABA framework and introduces an example to be referred to throughout the chapter. Sections 6.3 and 6.4 describe our generalised dispute derivation procedure used to build communicable arguments for a claim. Section 6.4.1 demonstrates how exchanging communicable arguments can be useful in dialogue between agents. Section 6.5 gives theoretical results. Section 6.6 presents a discussion of the use of AABA for multi-agent systems as well as related work. Lastly, Section 6.7 concludes. Note that we adopt a generalisation of ABA frameworks in this thesis, as originally presented by Gaertner and Toni [GT08], whereby assumptions allow *multiple contraries* (i.e. c is a (total) mapping from $\mathcal{A}$ into $\wp(\mathcal{L}) - \{\emptyset\}$, where any $y \in c(x)$ is referred to as *a* contrary of x).

6.2 Agent ABA Framework

Let $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, c \rangle$ be an ABA framework as presented by Dung et al. (see Chapter 2, Section 2.4.2) for some agent x in a given *agent system*. In the spirit of Hunter [Hun07], with the view of allowing agents to exchange arguments and assuming that all agents in an agent system use the same common knowledge, we partition $\mathcal{R}$ for x into $\mathcal{R}_c$ and $\mathcal{R}_x^p$ (i.e. $\mathcal{R}_c \cap \mathcal{R}_x^p = \emptyset$, $\mathcal{R}_c \cup \mathcal{R}_x^p = \mathcal{R}$) and propose an *agent assumption-based argumentation* (AABA) framework $\langle \mathcal{L}, \mathcal{R}_c, \mathcal{R}_x^p, \mathcal{A}, c \rangle$, where $\mathcal{R}_c$ is meant to contain knowledge common to all the agents and $\mathcal{R}_x^p$ is meant to contain personal beliefs of x not necessarily shared by other agents.

Definition 6.1 (AABA framework) *An* AABA framework $\langle \mathcal{L}, \mathcal{R}_c, \mathcal{R}_x^p, \mathcal{A}, c \rangle$ *for an agent* x *is such that* $\mathcal{R}_c \cap \mathcal{R}_x^p = \emptyset$ *and* $\langle \mathcal{L}, \mathcal{R}_c \cup \mathcal{R}_x^p, \mathcal{A}, c \rangle$ *is an* ABA framework.

In what follows, we use $\mathcal{AF}$ to refer to an ABA framework $\langle \mathcal{L}, \mathcal{R}_c \cup \mathcal{R}_{Ag}^p, \mathcal{A}, c \rangle$ for some generic agent (named) Ag and $\mathcal{AF}_{Ag}$ to refer to an AABA framework $\langle \mathcal{L}, \mathcal{R}_c, \mathcal{R}_{Ag}^p, \mathcal{A}, c \rangle$ for Ag. We restrict $\mathcal{R}_{Ag}^p$ to containing *facts* only (i.e. inference rules of the form $l_0 \leftarrow$, see Section 2.4.2 in Chapter 2) and refer to these as *personal facts*. As we will see in Chapter 7, this is sufficient for realising dialogues as given in Chapter 5. Note that $\mathcal{A} \cap \mathcal{R}_{Ag}^p = \emptyset$, since we work with *flat* frameworks (see Section 2.4.2 in Chapter 2).

The generalised procedures for building *arguments*, which we present in Sections 6.3 and 6.4, are such that the necessary personal facts of an agent are accumulated, as well as the necessary assumptions. In standard ABA instead, only the assumptions are accumulated.

Note that we assume $\mathcal{L}$ (as well as $\mathcal{A}$ and c) is common to all agents in an agent system as communication (or, rather, *arguments*) exchanged between the agents in the system will consist of sentences in $\mathcal{L}$.

6.2.1 Example — Exchanging appointments between agents

We introduce here an example to be used throughout the remainder of this chapter to demonstrate our AABA framework. The example represents situations where agents may need to exchange appointments, e.g. on behalf of patients, and providing arguments would be useful for coming to agreement on whether to exchange an appointment or not. This setting has been touched upon in examples in previous chapters — Examples 3.1 and 5.5 in particular.

Consider two AABA frameworks $\mathcal{AF}_x = \langle \mathcal{L}, \mathcal{R}_c, \mathcal{R}_x^p, \mathcal{A}, c \rangle$ and $\mathcal{AF}_y = \langle \mathcal{L}, \mathcal{R}_c, \mathcal{R}_y^p, \mathcal{A}, c \rangle$ for agents x and y respectively.

The language $\mathcal{L} \setminus \mathcal{A}$ consists of all ground instances of the following schemata:

- $sa(Ag1, Ag2, App1, App2)$, i.e. agents $Ag1$ and $Ag2$ should *swap appointments* $App1$ and $App2$;
- $r(Ag, G)$, i.e. agent Ag *requires* G to be fulfilled;
- $fr(App, G)$, i.e. appointment App *fulfils requirement* G;
- $l(Ag, App)$, i.e. agent Ag *likes* appointment App;
- $h(Ag, App)$, i.e. agent Ag *has* appointment App;
- $\neg cs(Ag, App1, App2)$, i.e. agent Ag *cannot swap* appointment $App1$ for $App2$.

The set of assumptions $\mathcal{A}$ consists of all ground instances of the following schemata:

- $\neg l(Ag, App)$, i.e. agent Ag *does not like* appointment App;
- $cs(Ag, App1, App2)$, i.e. agent Ag *can swap* appointment $App1$ for $App2$.

The contraries are as follows:

- $c(\neg l(Ag, App)) = \{l(Ag, App)\}$;

- $c(cs(Ag, App1, App2)) = \{\neg cs(Ag, App1, App2)\}$.

$\mathcal{R}_c$ consists of inference rules *R1–R3* below. There, Ag, $Ag1$ and $Ag2$ can be any agents.

R1: $sa(Ag1, Ag2, App1, App2) \leftarrow h(Ag1, App1),\ l(Ag1, App2),$

$$\neg l(Ag1, App1),\ cs(Ag2, App2, App1)$$

Namely, $Ag1$ and $Ag2$ should swap $App1$ and $App2$ if, it is believed, $Ag1$ has $App1$ and $Ag1$ likes $App2$, and, it is assumed, $Ag1$ dislikes $App1$ and $Ag2$ can swap $App2$ for $App1$.

R2: $l(Ag, App) \leftarrow r(Ag, G),\ fr(App, G)$

Namely, Ag likes App if, it is believed, Ag requires G to be fulfilled and App fulfils G.

R3: $\neg cs(Ag, App1, App2) \leftarrow h(Ag, App1),\ l(Ag, App1),\ \neg l(Ag, App2)$

Namely, Ag cannot swap $App1$ for $App2$ if, it is believed, Ag has and likes $App1$, and, it is assumed, Ag dislikes $App2$.

We could include additional inference rules, for example, for the case that an agent cannot swap an appointment it does not have or to allow agents to have multiple requirements, but the above three inference rules will be sufficient to demonstrate the AABA framework as is the focus of the chapter.

The personal facts of agents x and y are as follows:

$\mathcal{R}^p_x = \{h(x, app1),\ h(y, app2),\ r(x, seeDrAli),\ fr(app1, fridayApp),\ fr(app2, seeDrAli)\}$

$\mathcal{R}^p_y = \{h(y, app2),\ r(y, fridayApp),\ fr(app2, fridayApp),\ fr(app2, seeDrAli)\}$

In general, agents may have incorrect, out of date, or inconsistent beliefs. However, we assume here, for the sake of simplicity of demonstration, that agents have correct beliefs about the appointments allocated to themselves and other agents. Our focus in this example are agents that have incomplete information. Indeed, e.g. y has no knowledge of x's appointments and goals. The assumptions in this example allow an agent to reason under the uncertainty due to incomplete information. In particular, an agent can make assumptions about an agent (itself

or another) not liking a certain appointment or being able to swap appointments until/unless it is told or knows otherwise.

6.3 Fact-supported Arguments

We generalise for our AABA framework the notion of an argument that has a support consisting of assumptions only as in conventional ABA (see Chapter 2, Section 2.4.2). We will define arguments as deductions for sentences with a support consisting of assumptions *as well as* personal facts. We define first finite (deduction) trees adapted from Dung et al. [DKT09] and then a specific kind of such trees that is of interest in this chapter. The symbol τ represents an extra-logical symbol not in $\mathcal{L}$.

Definition 6.2 (finite deduction tree) *Given an AABA framework $\mathcal{AF}_{Ag}$, a* finite (deduction) tree $\mathcal{T}$ *for $p \in \mathcal{L}$ (the* conclusion *or* claim*) is as follows:*

- *the root of $\mathcal{T}$ is labelled by p and denoted $root(\mathcal{T})$*
- *for every node N of $\mathcal{T}$*
  - *if N is a leaf then N is labelled by a sentence in $\mathcal{L}$ or by τ;*
  - *if N is not a leaf and l_N is the label of N, then there is an inference rule $l_N \leftarrow b_1, \ldots, b_m$ ($m \geq 0$) in $\mathcal{R}_c$ and*

    either $m = 0$ and the child of N is labelled by τ

    or $m > 0$ and N has m children, labelled by $b_1, \ldots, b_m$ (respectively)
- *the set of sentences, not including τ, labelling the leaves of $\mathcal{T}$ is denoted $leaves(\mathcal{T})$.*

Nodes in a tree as above are connected by the inference rules, with sentences matching the conclusion of an inference rule connected as parent nodes to sentences matching the premises of the inference rule as children nodes. Finite deduction trees as above differ from a deduction tree in Definition 2 of [DKT09] in that, firstly, only the inference rules in the *common* knowledge are

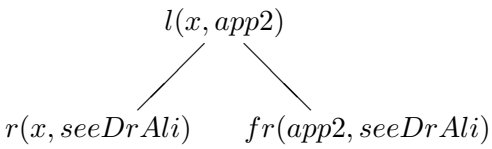


Figure 6.1: A f-argument for $l(x, app2)$ with support $\{r(x, seeDrAli),\ fr(app2, seeDrAli)\}$ wrt $\mathcal{AF}_x$ as defined in Section 6.2.1.

used in the definition above to produce new branches and, secondly, the leaves in the definition above may be non-assumptions as well as assumptions. This is different from [DKT09] where, firstly, all rules are used to produce new branches and, secondly, only assumptions are leaves. We define next *fact-supported arguments* (abbreviated '*f-arguments*') for a claim p with support S consisting only of assumptions and personal facts as finite trees with p at the root and S at the leaves.

Definition 6.3 (f-argument) *Given an AABA framework $\mathcal{AF}_{Ag}$, a* f-argument *for $p \in \mathcal{L}$ (the* conclusion *or* claim*) with* support *$S \subseteq \mathcal{A} \cup \mathcal{R}^p_{Ag}$ is a finite (deduction) tree $\mathcal{T}$ where $root(\mathcal{T}) = p$ and $leaves(\mathcal{T}) = S$.*

We illustrate in Figure 6.1 a finite tree that is a f-argument. F-arguments for a claim can be computed by means of backward deductions defined as follows:

Definition 6.4 (backward deduction) *Given an AABA framework $\mathcal{AF}_{Ag}$ and a selection function $f : 2^{\mathcal{L}} \to \mathcal{L}$,*[1] *a* backward deduction *of $S \subseteq \mathcal{A} \cup \mathcal{R}^p_{Ag}$ for a claim $p \in \mathcal{L}$ according to $\mathcal{AF}_{Ag}$ (wrt f) is a finite sequence of tuples*

$$\langle C_0, S_0\rangle,\ \ldots\ ,\ \langle C_i, S_i\rangle,\ \ldots\ ,\ \langle C_n, S_n\rangle$$

where $n \geq 1$, $C_0 = \{p\}$, $S_0 = C_n = \emptyset$, $S_n = S$, and, for every $0 \leq i < n$, if $f(C_i) = \sigma$, then

1. *If $\sigma \in \mathcal{R}^p_{Ag}$ or $\sigma \in \mathcal{A}$, then*

   - $C_{i+1} = C_i - \{\sigma\}$

[1] A selection function f takes as input a set of sentences C_i and returns a single element σ from C_i.

- $S_{i+1} = S_i \cup \{\sigma\}$

2. *Otherwise, choose* $\sigma \leftarrow b_1, \ldots, b_m \in \mathcal{R}_c$ *and*

- $C_{i+1} = (C_i - \{\sigma\}) \cup \{b_1, \ldots, b_m\}$
- $S_{i+1} = S_i$

We write $S \vdash^f_{\mathcal{AF}_{Ag}} p$ *if there exists a backward deduction of* S *for* p *according to* $\mathcal{AF}_{Ag}$ *(wrt* f*).*[2]

Backward deductions, as defined above, are a generalisation of SLD resolution (and also *deductions of conclusions based on sets of premises* as in Defition 2.2 of [DKT06]), which is the basis of proof procedures for logic programming. As in SLD, if there is a backward deduction using one selection function for picking sentences in C_i, then there is a backward deduction using any other selection function. Thus, different selection functions are simply different but equivalent ways of generating the same argument for a claim. A backward deduction as above differs from an *argument* ("a deduction whose premises are all assumptions") as in [DKT06] in that arguments in [DKT06] are only supported by assumptions whereas our backward deductions are supported by non-assumptions (personal facts of the agent) as well as assumptions.

The following example illustrates the notion of backward deduction.

Example 6.1 (backward deduction) *Given the AABA framework* $\mathcal{AF}_x$ *from Section 6.2.1, a backward deduction*

$$\{r(x, seeDrAli),\ fr(app2, seeDrAli)\} \vdash_{\mathcal{AF}_x} l(x, app2)$$

built by x *affirming that it likes* $app2$ *is obtained as shown in Table 6.1 and described here:*

- *At step* $i = 0$*,* $R2 \in \mathcal{R}_c$ *is chosen in applying Case 2 of Definition 6.4.*

[2]We write $\vdash_{\mathcal{AF}_{Ag}}$ instead of $\vdash^f_{\mathcal{AF}_{Ag}}$ where the particular selection function is unimportant in the given context.

i	C_i	S_i	Apply...
0	$\{\underline{l(x, app2)}\}$	$\emptyset$	Case 2
1	$\{\underline{r(x, seeDrAli)},\ fr(app2, seeDrAli)\}$	$\emptyset$	Case 1
2	$\{\underline{fr(app2, seeDrAli)}\}$	$\{r(x, seeDrAli)\}$	Case 1
3	$\emptyset$	$\{r(x, seeDrAli),\ fr(app2, seeDrAli)\}$	

Table 6.1: Backward deduction for $l(x, app2)$ according to $\mathcal{AF}_x$ where $\mathcal{AF}_x$ is as defined in Section 6.2.1. The underlined atoms are picked by the selection function.

- *At step $i = 1$, Case 1 is applied since $r(x, seeDrAli) \in \mathcal{R}_x^p$.*
- *At step $i = 2$, Case 1 is applied since $fr(app2, seeDrAli) \in \mathcal{R}_x^p$.*

In Definition 6.4, we use the word "choose" to identify backtrackable points in the search for a backward deduction. The need for this is illustrated by the following example:

Example 6.2 (backtracking in the search for a backward deduction) *Consider an AABA framework $\mathcal{AF}_x$ for some agent x with*

$$\mathcal{L} = \{p, q, r\},\ \mathcal{A} = \emptyset,\ \mathcal{R}_c = \{p \leftarrow q\} \cup \{p \leftarrow r\},\ \mathcal{R}_x^p = \{r\}$$

If in the search for a backward deduction for p, $p \leftarrow q \in \mathcal{R}_c$ is chosen when applying case 2 of Definition 6.4, then, since $q \notin \mathcal{R}_x^p$, $q \notin \mathcal{A}$ and there is no inference rule $q \leftarrow b_1, \ldots, b_m \in \mathcal{R}_c$, the search for a backward deduction needs to backtrack and choose $p \leftarrow r \in \mathcal{R}_c$, leading eventually to $\{r\} \vdash_{\mathcal{AF}_x} p$.

We end this section by proving the relationship between f-arguments and backward deductions.

Lemma 6.1 *If, for some selection function f, there is a backward deduction $S \vdash^f_{\mathcal{AF}_{Ag}} p$, then there is a f-argument for p with support S wrt $\mathcal{AF}_{Ag}$.*

Lemma 6.2 *If there is a f-argument for p with support S wrt $\mathcal{AF}_{Ag}$, then there is a backward deduction $S \vdash^f_{\mathcal{AF}_{Ag}} p$ for any selection function f.*

The proofs of Lemmas 6.1 and 6.2 can be found in Appendix D.

6.4 Computation of Admissible Arguments

The AB-dispute derivations (which we abbreviate '*dd*s') of Dung et al. [DMT07] (see Definition 2.10) can be used to construct an admissible set of assumptions supporting a claim. We generalise this for AABA frameworks and present a new computational model, which we call *fact-inclusive AB-dispute derivations* (abbreviated '*fdd*s'), that constructs an admissible defence for a claim as well as personal facts used during the construction. These derivations, similar to *dd*s, can be seen as a game between two fictional players - a proponent and an opponent. The rules of the game are roughly as follows: the proponent puts forward a f-argument for a claim; the opponent disputes the proponent's argument by presenting f-arguments attacking assumptions of the proponent's argument; the proponent in turn defends its argument by counter-attacking the opponent's attacks with other f-arguments, which the opponent in turn attacks, and so on, until the proponent has defended itself against all attacks.

As for *dd*s, in conducting this game, the proponent cannot attack any of its own assumptions, nor does it need to counter-attack any assumption it has attacked previously, nor does it need to defend any assumption it has already defended. Differently from *dd*s, in *fdd*s, f-arguments are fully constructed before being attacked. Thus only actual f-arguments are manipulated (rather than "potential", see [GT08]). Also, in *fdd*s unlike *dd*s, the proponent keeps track of personal facts used in its arguments. Together, the assumptions and personal facts used by the proponent make up its defence for a claim.

We define a *fdd* as a sequence of tuples of the form $\langle \mathcal{P}_i, \mathcal{O}_i, \mathcal{D}_i, \mathcal{C}_i \rangle$. Such a tuple represents the state of the derivation at the ith step and is to be read as follows: $\mathcal{P}_i$ is a set of sentences, the claims of the proponent to be expanded and defended; $\mathcal{O}_i$ are the sets consisting of personal facts and assumptions that support attacks of the opponent on assumptions of the proponent, and each such set is to be attacked by the proponent; $\mathcal{D}_i$ (*Defences*) is the set of personal facts and assumptions encountered so far in the derivation sequence that support the claims of the proponent; $\mathcal{C}_i$ (*Culprits*) is the set of assumptions of the opponent that have already been attacked by the proponent.

Definition 6.5 (fact-inclusive AB-dispute derivation) *Given an AABA framework $\mathcal{AF}_{Ag}$, and selection functions $f : 2^{\mathcal{L}} \to \mathcal{L}$ and $g : 2^{2^{\mathcal{L}}} \to 2^{\mathcal{L}}$,*[3] *a* fdd *of a defence set $\mathcal{D} \subseteq \mathcal{A} \cup \mathcal{R}^p_{Ag}$ for a set of sentences $\mathcal{P} \subseteq \mathcal{L}$ according to $\mathcal{AF}_{Ag}$ (wrt f and g) is a finite sequence of tuples*

$$\langle \mathcal{P}_0, \mathcal{O}_0, \mathcal{D}_0, \mathcal{C}_0 \rangle, \ \ldots, \ \langle \mathcal{P}_i, \mathcal{O}_i, \mathcal{D}_i, \mathcal{C}_i \rangle, \ \ldots, \ \langle \mathcal{P}_n, \mathcal{O}_n, \mathcal{D}_n, \mathcal{C}_n \rangle$$

where $\mathcal{P}_0 = \mathcal{P}$, $\mathcal{D}_0 = \mathcal{P} \cap (\mathcal{A} \cup \mathcal{R}^p_{Ag})$, $\mathcal{O}_0 = \mathcal{C}_0 = \emptyset$, $\mathcal{P}_n = \mathcal{O}_n = \emptyset$, $\mathcal{D}_n = \mathcal{D}$, and, for every $0 \leq i < n$, only one $f(\mathcal{P}_i) = \sigma$ or one $g(\mathcal{O}_i) = \Sigma$ is picked [4] *and*

1. *If $f(\mathcal{P}_i) = \sigma$ is picked, then*

   (a) *if $\sigma \in \mathcal{R}^p_{Ag}$, then*

   - $\mathcal{P}_{i+1} = \mathcal{P}_i - \{\sigma\}$
   - $\mathcal{O}_{i+1} = \mathcal{O}_i$
   - $\mathcal{D}_{i+1} = \mathcal{D}_i$
   - $\mathcal{C}_{i+1} = \mathcal{C}_i$

   (b) *if $\sigma \in \mathcal{A}$, then*

   - $\mathcal{P}_{i+1} = \mathcal{P}_i - \{\sigma\}$
   - $\mathcal{O}_{i+1} = \mathcal{O}_i \cup \{S \mid \sigma' \in c(\sigma),\ S \vdash^f_{\mathcal{AF}_{Ag}} \sigma',\ S \cap \mathcal{C}_i = \emptyset\}$ *(filtering of culprits by culprits)*
   - $\mathcal{D}_{i+1} = \mathcal{D}_i$
   - $\mathcal{C}_{i+1} = \mathcal{C}_i$

   (c) *if $\sigma \notin \mathcal{A} \cup \mathcal{R}^p_{Ag}$, choose a backward deduction $S \vdash^f_{\mathcal{AF}_{Ag}} \sigma$ such that $S \cap \mathcal{C}_i = \emptyset$ (filtering of defences by culprits), then*

   - $\mathcal{P}_{i+1} = (\mathcal{P}_i - \{\sigma\}) \cup ((S \cap \mathcal{A}) - \mathcal{D}_i)$ *(filtering of defences by defences)*
   - $\mathcal{O}_{i+1} = \mathcal{O}_i$
   - $\mathcal{D}_{i+1} = \mathcal{D}_i \cup S$

[3] f takes as input a set of sentences $\mathcal{P}_i$ and returns a single element $\sigma \in \mathcal{P}_i$, while g takes as input a set of sets of sentences $\mathcal{O}_i$ and returns a single element $\Sigma \in \mathcal{O}_i$.

[4] Depending on the player choice, as discussed later.

- $\mathcal{C}_{i+1} = \mathcal{C}_i$

2. *If* $g(\mathcal{O}_i) = \Sigma$ *is picked, then*

   (a) *if* $\Sigma \cap \mathcal{C}_i \neq \emptyset$ *(filtering of culprits by culprits), then*

   - $\mathcal{P}_{i+1} = \mathcal{P}_i$
   - $\mathcal{O}_{i+1} = \mathcal{O}_i - \{\Sigma\}$
   - $\mathcal{D}_{i+1} = \mathcal{D}_i$
   - $\mathcal{C}_{i+1} = \mathcal{C}_i$

   (b) *if* $\Sigma \cap \mathcal{C}_i = \emptyset$*, choose* $\sigma \in \Sigma$ *such that* $\sigma \in \mathcal{A}$ *and* $\sigma \notin \mathcal{D}_i$ *(filtering of culprits by defences), and choose* $\sigma' \in c(\sigma)$*, then*

   - $\mathcal{P}_{i+1} = \mathcal{P}_i \cup \{\sigma'\}$
   - $\mathcal{O}_{i+1} = \mathcal{O}_i - \{\Sigma\}$
   - $\mathcal{D}_{i+1} = \mathcal{D}_i \cup (\{\sigma'\} \cap (\mathcal{A} \cup \mathcal{R}^p_{Ag}))$
   - $\mathcal{C}_{i+1} = \mathcal{C}_i \cup \{\sigma\}$

We write $\mathcal{D} \models^{f,g}_{\mathcal{AF}_{Ag}} \mathcal{P}$ *if there exists a* fdd *of* $\mathcal{D}$ *for* $\mathcal{P}$ *according to* $\mathcal{AF}_{Ag}$ *(wrt* f *and* g*). We write* $\mathcal{D} \models^{f,g}_{\mathcal{AF}_{Ag}} p$ *if* $\mathcal{P} = \{p\}$*.* [5]

Similarly to Definition 6.4, we use the word "choose" in Definition 6.5 above to identify backtrackable points in the search for a *fdd*.

Intuitively, two choices (at least) have to be made at each step in a derivation. First, a (fictional) *player* must be chosen: either the proponent ($\mathcal{P}_i$) or the opponent ($\mathcal{O}_i$). If the proponent is chosen, then a sentence σ in $\mathcal{P}_i$ is picked and one of three cases (1a, 1b or 1c) apply. If the opponent is chosen, then a set Σ in $\mathcal{O}_i$ is picked and one of two cases (2a or 2b) apply.

In case 1a, the proponent plays and σ is a personal fact: this is simply removed from the proponent's set. Indeed, σ will have been added to the defence set previously.

[5] We write $\models_{\mathcal{AF}_{Ag}}$ instead of $\models^{f,g}_{\mathcal{AF}_{Ag}}$ where the particular selection functions are unimportant in the given context.

In case 1b, the proponent plays and σ is an assumption: this is removed from the proponent's set and all (suppports of) f-arguments that attack σ and contain no culprit assumptions are added to the opponent's set.

In case 1c, the proponent plays and σ is neither a personal fact nor an assumption: σ is removed from the proponent's set and, choosing a support S for σ that contains no culprit assumptions, replaced by the assumptions of S after the defence assumptions have been filtered.

In case 2a, the opponent plays and Σ contains a culprit assumption: Σ is simply removed from the opponent's set, as it has already been counter-attacked.

In case 2b, the opponent plays and Σ contains no culprit assumptions: Σ is removed from the opponent's set. A non-defence (trivially non-culprit) assumption σ in Σ is chosen and added to the set of culprits. Also, a contrary σ' of σ is chosen, added to the proponent's set and, if σ' is a personal fact or assumption, also added to the defence set.

We demonstrate in Examples 6.3 and 6.4 below a successful and a failed (respectively) *fdd*. Here and in the remainder of the chapter, "*fdd* of $\mathcal{D}$ for p" will stand for "*fdd* of $\mathcal{D}$ for $\{p\}$".

Example 6.3 (successful fdd) *Given the AABA framework $\mathcal{AF}_x$ of Section 6.2.1, a* fdd $S \models_{\mathcal{AF}_x} sa(x, y, app1, app2)$ *is obtained as shown in Table 6.2 and described as follows:*

- *At step $i = 0$, $sa(x, y, app1, app2) \in \mathcal{P}_0$ is picked and $S \vdash_{\mathcal{AF}_x} sa(x, y, app1, app2)$ chosen in applying Case 1c, where $S = \{h(x, app1), r(x, seeDrAli), fr(app2, seeDrAli), \neg l(x, app1), cs(y, app2, app1)\}$.*

- *At step $i = 1$, $\neg l(x, app1) \in \mathcal{P}_1$ is picked and Case 1b applied. Note that $c(\neg l(x, app1)) = \{l(x, app1)\}$ and $l(x, app1)$ has no support in $\mathcal{AF}_x$, hence $\mathcal{O}_2 = \mathcal{O}_1 = \emptyset$.*

- *At step $i = 2$, $cs(y, app2, app1) \in \mathcal{P}_2$ is picked and Case 1b applied. Note that $c(cs(y, app2, app1)) = \{\neg cs(y, app2, app1)\}$ and $\neg cs(y, app2, app1)$ has no support in $\mathcal{AF}_x$, hence $\mathcal{O}_3 = \mathcal{O}_2 = \emptyset$.*

i	$\mathcal{P}_i$	$\mathcal{O}_i$	$\mathcal{D}_i$	$\mathcal{C}_i$	Apply...
0	$\{\underline{sa(x, y, app1, app2)}\}$	$\emptyset$	$\emptyset$	$\emptyset$	Case 1c
1	$\{\underline{\neg l(x, app1)}, cs(y, app2, app1)\}$	$\emptyset$	S	$\emptyset$	Case 1b
2	$\{\underline{cs(y, app2, app1)}\}$	$\emptyset$	S	$\emptyset$	Case 1b
3	$\emptyset$	$\emptyset$	S	$\emptyset$	

Table 6.2: Fdd for $sa(x, y, app1, app2)$ according to $\mathcal{AF}_x$ where $\mathcal{AF}_x$ is as defined in Section 6.2.1. $S = \{h(x, app1), r(x, seeDrAli), fr(app2, seeDrAli), \neg l(x, app1), cs(y, app2, app1)\}$. The underlined atoms are picked by f.

Note that S includes facts to be communicated, which would not be accumulated by the dd *procedure.*

Example 6.4 (failed fdd attempt) *Given the AABA framework $\mathcal{AF}_y$ of Section 6.2.1 but such that additionally $h(x, app1), r(x, seeDrAli) \in \mathcal{R}^p_y$, a* fdd *for $sa(x, y, app1, app2)$ according to $\mathcal{AF}_y$ cannot be found, as shown in Table 6.3 and described as follows:*

- *At step $i = 0$, $sa(x, y, app1, app2) \in \mathcal{P}_0$ is picked and $S \vdash_{\mathcal{AF}_y} sa(x, y, app1, app2)$ chosen in applying Case 1c of Definition 6.5 (where S is as in Example 6.3).*

- *At step $i = 1$, $cs(y, app2, app1) \in \mathcal{P}_1$ is picked and Case 1b applied. Note that $c(cs(y, app2, app1)) = \{\neg cs(y, app2, app1)\}$ and S' is the only support for $\neg cs(y, app2, app1)$ in $\mathcal{AF}_y$, hence $\mathcal{O}_2 = \mathcal{O}_1 \cup \{S'\}$ (where $S' = \{h(y, app2), r(y, fridayApp), fr(app2, fridayApp), \neg l(y, app1)\}$).*

- *At step $i = 2$, $S' \in \mathcal{O}_2$ is picked and in applying Case 2b, (the only assumption) $\neg l(y, app1) \in S'$ and (the only contrary) $l(y, app1) \in c(\neg l(y, app1))$ chosen.*

- *At step $i = 3$, picking $l(y, app1) \in \mathcal{P}_3$, the* fdd *fails since none of the cases 1a to 1c are applicable for $l(y, app1)$ according to $\mathcal{AF}_y$. In particular, $l(y, app1) \notin \mathcal{R}^p_y$ nor is there a support for $l(y, app1)$ in $\mathcal{AF}_y$. Further, there are no points in the derivation sequence where backtracking can be successfully applied.*

i	$\mathcal{P}_i$	$\mathcal{O}_i$	$\mathcal{D}_i$	$\mathcal{C}_i$	Apply...
0	$\{\underline{sa(x, y, app1, app2)}\}$	$\{\}$	$\{\}$	$\{\}$	Case 1c
1	$\{\neg l(x, app1), \underline{cs(y, app2, app1)}\}$	$\{\}$	S	$\{\}$	Case 1b
2	$\{\neg l(x, app1)\}$	$\{\underline{S'}\}$	S	$\{\}$	Case 2b
3	$\{\neg l(x, app1), \underline{l(y, app1)}\}$	$\{\}$	S	$\{\neg l(y, app1)\}$	

Table 6.3: Attempt to build a fdd for $sa(x, y, app1, app2)$ according to $\mathcal{AF}_y$ where $\mathcal{AF}_y$ is as defined in Section 6.2.1. S is as defined in Table 6.2. $S' = \{h(y, app2), r(y, fridayApp), fr(app2, fridayApp), \neg l(y, app1)\}$. The underlined elements are picked by f and g.

6.4.1 Example — Arguments in Dialogue

*Fdd*s, giving $\mathcal{D} \models_{\mathcal{AF}_{Ag}} p$, can be seen as a means of generating "communicable arguments" for claims p to be exchanged between communicating agents. Note that typically these are not f-arguments as they could include in $\mathcal{D}$ additional facts and assumptions needed to defend the claim against attacks (see step **t=3** in the example that follows below where the support of the f-argument $S \models_{\mathcal{AF}_x} sa(x, y, app1, app2)$ is augmented with S'' to produce the fdd $S \cup S'' \models_{\mathcal{AF}_x} sa(x, y, app1, app2)$). We refer to these "arguments" as *fdd-arguments*.

We demonstrate in this section how exchanging fdd-arguments between agents can be useful in dialogue. Note that this section is solely for illustration and not intended to define a dialogue game for argumentation. Whilst frameworks have been presented for argumentation-based dialogue games, e.g. [Pra05], our work here in this chapter is orthogonal and presents instead a means of representing and generating arguments that could be fed into such frameworks.

Given the AABA frameworks $\mathcal{AF}_x$ and $\mathcal{AF}_y$ of Section 6.2.1 and ignoring details of the communication protocol and policy, we demonstrate how, by exchanging arguments, agents x and y could come to an agreement that $sa(x, y, app1, app2)$ is an acceptable action. Informally, the dialogue between the two agents proceeds as follows:

x: We should swap $app1$ and $app2$ because ... (*reasons*) ...

... and I assume you can swap $app2$ for $app1$.

y: I cannot swap $app2$ for $app1$ because ... (*reasons*) ...

... and I assume $app1$ is no good for me.

x: Ah, but $app1$ is good for you because $\ldots(reasons)\ldots$,

hence we should swap $app1$ and $app2$.

y: Ok. I agree to the swap.

We show in Table 6.4 and explain here the fdd-arguments uttered by the agents and the beliefs each agent adds to its personal facts as a result of the dialogue. The dialogue takes place over time points 1–4. Time 0 is prior to the dialogue.

t=0: There is a *fdd* $S \models_{\mathcal{AF}_x} sa(x, y, app1, app2)$ (as derived in Table 6.2). However, there is no *fdd* for $sa(x, y, app1, app2)$ according to $\mathcal{AF}_y$.

t=1: x communicates its fdd-argument $S \models_{\mathcal{AF}_x} sa(x, y, app1, app2)$ to y and y revises $\mathcal{R}_y^p$ to include $h(x, app1)$ and $r(x, seeDrAli)$, given in S. Then, y knows what x has and requires.

t=2: y can now build $S \vdash_{\mathcal{AF}_y} sa(x, y, app1, app2)$. However, a *fdd* for $sa(x, y, app1, app2)$ according to $\mathcal{AF}_y$ still does not exist, as shown in Table 6.3. The successfully attacking argument in the failed *fdd* sequence is $S' \models_{\mathcal{AF}_y} \neg cs(y, app2, app1)$. Then, y communicates this fdd-argument to x and x revises $\mathcal{R}_x^p$ to include $r(y, fridayApp)$ and $fr(app2, fridayApp)$, given in S'.

t=3: x can now build a f-argument $S' \vdash_{\mathcal{AF}_x} \neg cs(y, app2, app1)$. However, this is not an acceptable argument in $\mathcal{AF}_x$ since the (only) assumption $\neg l(y, app1)$ in S' has a contrary $l(y, app1)$ for which there is an acceptable *fdd* $S'' \models_{\mathcal{AF}_x} l(y, app1)$. By means of this defending argument, a *fdd* $S \cup S'' \models_{\mathcal{AF}_x} sa(x, y, app1, app2)$ exists. Then x communicates this fdd-argument to y and y revises $\mathcal{R}_y^p$ to include $fr(app1, fridayApp)$, given in S''.

t=4: y is now able to build a f-argument $S'' \vdash_{\mathcal{AF}_y} l(y, app1)$ as well as an acceptable *fdd* for $sa(x, y, app1, app2)$ according to $\mathcal{AF}_y$. Hence the agents end in agreement.

We present next properties of the AABA framework.

t	x utters...	y utters...
1	$S \models_{\mathcal{AF}_x} sa(x, y, app1, app2)$	
2		$S' \models_{\mathcal{AF}_y} \neg cs(y, app2, app1)$
3	$S''' \models_{\mathcal{AF}_x} sa(x, y, app1, app2)$	
4		$S''' \models_{\mathcal{AF}_x} sa(x, y, app1, app2)$

t	$\mathcal{R}_x^p$ plus...	$\mathcal{R}_y^p$ plus...
1		$h(x, app1)$, $r(x, seeDrAli)$
2	$r(y, friApp)$,$fr(app2, friApp)$	
3		$fr(app1, fridayApp)$
4		

Table 6.4: Arguments exchanged between x and y regarding the acceptability of $sa(x, y, app1, app2)$, and respective changes to each agent's personal facts. S and S' are as defined in Tables 6.2 and 6.3 (respectively). $S'' = \{r(y, fridayApp),\ fr(app1, fridayApp)\}$. $S''' = S \cup S''$.

6.5 Theoretical Results for AABA

In this section, we state the following main results for any ABA framework $\mathcal{AF}$ and corresponding AABA framework $\mathcal{AF}_{Ag}$:

- *fdd*s correspond to (a variant of) the *dd*s of Dung et al. [DMT07] (see Theorems 6.1 and 6.2, and Corollary 6.1);
- *fdd*s are correct in that the assumptions in the defence of the input claims they compute is indeed admissible (see Corollary 6.2);
- *fdd*s are complete for p-acyclic ABA frameworks (see Corollary 6.3).

The variant of *dd*s we use is as defined by Gaertner [Gae09]. The cases 2ic.1 and 2ic.2 of the *dd*s of Dung et al. [DMT07] (see Definition 2.10) are combined into one single case 2ic which states:

$$\mathcal{P}_{i+1} = \mathcal{P}_i \cup \{c(\sigma)\};\ \mathcal{O}_{i+1} = \mathcal{O}_i - S;\ \mathcal{D}_{i+1} = \mathcal{D}_i \cup (\mathcal{A} \cap \{c(\sigma)\});\ \mathcal{C}_{i+1} = \mathcal{C}_i \cup \{\sigma\}.$$

The variant is proven by Gaertner in [Gae09] to be correct, and to be equivalent to the original version under the restriction that, when the contrary of an assumption is an assumption, its original contrary is the original assumption. This restriction also held (implicitly) in [DMT07]. We prove our results for ABA frameworks where the contrary of every assumption is a singleton

set. This can be done without loss of generality since there is a one-to-one correspondence between such ABA frameworks and ABA frameworks with multiple contraries [GT08]. When $c(\sigma)$ is a singleton set $\{\sigma'\}$, we denote $c(\sigma)$ simply as σ', i.e. $c(\sigma) = \sigma'$.

Theorem 6.1 (fdd implies dd) *If, using some selection functions f and g, $\mathcal{D} \models^{f,g}_{\mathcal{AF}_{Ag}} p$, then, using some selection function f', there is a* dd *of $\mathcal{D} \cap \mathcal{A}$ for p.*

Theorem 6.2 (dd implies fdd) *If, using some selection function f, there is a* dd *of a defence set $A \subseteq \mathcal{A}$ for a claim p, then, for some selection function g, $A \cup F \models^{f,g}_{\mathcal{AF}_{Ag}} p$ for some $F \subseteq \mathcal{R}^p_{Ag}$.*

The proofs of Theorems 6.1 and 6.2 can be found in Appendix D.

Corollary 6.1 (fdd iff dd) *If $\mathcal{R}^p_{Ag} = \emptyset$, then there is a* fdd *of a defence set $\mathcal{D}$ for p according to $\mathcal{AF}_{Ag}$ iff there is a* dd *of $\mathcal{D}$ for p.*

The proof of Corollary 6.1 follows straightforwardly from Theorems 6.1 and 6.2 above.

Corollary 6.2 (soundness of fdd) *For every* fdd *of a defence set $\mathcal{D} = A \cup F$ ($A \subseteq \mathcal{A}$, $F \subseteq \mathcal{R}^p_{Ag}$) for $p \in \mathcal{L}$, the set of assumptions A is admissible and there exist some $A' \subseteq A$ and $F' \subseteq F$ such that $A' \cup F' \vdash_{\mathcal{AF}_{Ag}} p$.*

The proof of Corollary 6.2 follows straightforwardly from Theorem 6.1 above and the soundness theorem for *dd*s (i.e. Theorem 4.3 of [DMT07], as formulated in [Gae09]).

We define a *positively acyclic* (or *p-acyclic* for short) ABA framework as in [DMT07]. By $\mathcal{AF}^+$ we denote the ABA framework obtained by deleting all assumptions appearing in the premises of the inference rules of $\mathcal{AF}$. The *dependency graph* of $\mathcal{AF}^+$ is a directed graph where:

- the nodes are the atoms occurring in $\mathcal{AF}^+$;
- a (directed) arc from a node p to a node q is in the graph if and only if there exists an inference rule $p \leftarrow B$ in $\mathcal{AF}^+$ such that q occurs in B.

Definition 6.6 (p-acyclic frameworks) *An ABA framework $\mathcal{AF}$ is* p-acyclic *if the dependency graph of $\mathcal{AF}^+$ is acyclic. An AABA framework $\langle \mathcal{L}, \mathcal{R}_c, \mathcal{R}^p_{Ag}, \mathcal{A}, c\rangle$ is* p-acyclic *iff $\langle \mathcal{L}, \mathcal{R}_c \cup \mathcal{R}^p_{Ag}, \mathcal{A}, c\rangle$ is* p-acyclic.

Definition 6.7 (Partial Deduction) *A* partial deduction *for a claim p, given a selection function f, is a sequence $\langle C_0, S_0\rangle$, ..., $\langle C_j, S_j\rangle$ where $j \geq 0$, $C_0 = \{p\}$, $S_0 = \emptyset$ and each $\langle C_{i+1}, S_{i+1}\rangle$ $(0 \leq i < j)$ is constructed from $\langle C_i, S_i\rangle$ by picking $f(C_i) = \sigma$ and applying case 1 or 2 of Definition 6.4.*

Theorem 6.3 (no infinite partial deductions for p-acyclic frameworks) *Given a p-acyclic framework, there exists no infinite partial deduction.*

The proof of Theorem 6.3 is trivial since there is a well-ordering of all sentences in the language of p-acyclic frameworks, i.e. whenever a sentence belongs to the premise of an inference rule, then the sentence is lower in ordering than the conclusion of the inference rule.

Theorem 6.3 guarantees that the search for a f-argument always terminates (either successfully, finding such an argument, or not). In the case of p-acyclic frameworks with a finite underlying language $\mathcal{L}$ the *fdd*s are complete, in the following sense:

Corollary 6.3 (completeness of fdds) *Let $\mathcal{AF}_{Ag}$ be a p-acyclic AABA framework such that $\mathcal{L}$ is finite. Then, for each sentence $l \in \mathcal{L}$, if l is an admissible claim then*

- *there exists a* fdd *for l according to $\mathcal{AF}_{Ag}$;*
- *for each admissible set $A \subseteq \mathcal{A}$, if, for some $A' \subseteq A$ and $F \subseteq \mathcal{R}^p_{Ag}$, $A' \cup F \vdash_{\mathcal{AF}_{Ag}} l$, then there is a* fdd $\mathcal{D} \models_{\mathcal{AF}_{Ag}} l$ *such that $A' \cup F \subseteq \mathcal{D}$.*

The proof of Corollary 6.3 follows straightforwardly from Theorem 6.2 above and the completeness theorem for *dd*s for p-acyclic ABA frameworks (i.e. Theorem 4.4 of [DMT07]).

6.6 Discussion

We have proposed an extension to the ABA framework of Dung et al. [DKT09] for multi-agent system settings where agents have some common knowledge but may also have some personal beliefs not necessarily shared by all agents. Moreover, our extension of the ABA framework ties in to the kinds of problems of interest to us in this thesis where agents need to interact — constructing and sharing arguments between one another to come to agreement over actions to be undertaken. In exchanging arguments between one another, agents would need to include personal beliefs that the recipient agent may be unaware of and which could result in the recipient agent revising its belief base and thus in turn revising its stance or position in an interaction. In the context of a resource reallocation problem, as is the focus in this thesis, these arguments might express, for example, why an agent is requesting a particular resource (e.g. "I need it for my goal which is ...") or why an agent cannot give away a particular resource (e.g. "The resource is not currently free as it is allocated to ..."). Such arguments, which would include only some (and not necessarily all) of the premises for entailing the claim, can be seen as enthymemes [Hun07]. Notably, beliefs that are assumed by the utterer to be common between it and the recipient(s) are omitted in contruction of an argument.

The benefits of allowing agents to exchange arguments can be found in many different types of dialogues, e.g. information-seeking [DMWB06], inquiry [STT01, BH08b], negotiation [RRJ+03, APM00, HT08, KJRM09], deliberation [MHP02] as well as of course persuasion [AMP00, Pra05]. Thus a computational model that can generate communicable, acceptable arguments as part of a larger dialogue is clearly needed. The computational model which we have set out in this chapter is useful for this purpose exactly and for the purposes of the kinds of distributed (resource allocation) applications which we consider in this thesis (see Chapter 1, Section 1.2). Notably, to fit in and serve as the agent's decision-making mechanism as part of its negotiation (dialogue) policy for deciding upon deals (exchange of resources) to propose to other agents and for evaluating deals proposed by other agents, and to generate communicable arguments accordingly. Beside the framework which we aim to set up in this thesis, the only other framework we are aware of which makes use of enthymemes in dialogue is that of Black

and Hunter [BH08b]. However, there enthymemes are characterised in inquiry dialogues rather than negotation dialogues, as is the focus in this thesis.

6.7 Conclusion

We have proposed a revised notion of ABA argument that accumulates an agent's personal beliefs as well as its assumptions. Accordingly, we have proposed a computational model for generating admissible arguments, called fact-inclusive AB-dispute derivations (*fdd*s), which accumulates an agent's personal facts as well as its assumptions. We have illustrated how such outcomes of *fdd*s could be usefully exchanged between agents in dialogue. In the next chapter we will apply our framework and the exchange of arguments in the context of concrete inter-agent communication, where consistency in the agents' beliefs is not explicitly enforced and agents may have inconsistent (internally or across agents) as well as incomplete beliefs.

Chapter 7

AABA Negotiation Policy for Socially Optimal Resource Allocation

We complete in this chapter the generative agent policy of Chapter 5 which supports distributed multi-agent resource reallocation amongst goal-driven agents. The policy specifies how agents initiate requests or proposals, directed at one or more agents respectively, depending on their beliefs about themselves and other agents. These beliefs may be partial, in that agents may have incomplete knowledge about the goals and resources held by other agents. Agents also need to take into consideration the possibility of resources changing hands between agents and thus their beliefs becoming incorrect over time. Furthermore, the policy as completed in this chapter does not require an explicit removing (cleaning) of the agent's beliefs to enforce consistency in the agent's beliefs as assumed in Chapters 3 and 5 for when the agent engages in dialogue instances. The policy relies on Agent Assumption-Based Argumentation (AABA) as defined in Chapter 6 (i) to fill the gaps in the agents' knowledge at request and proposal time, (ii) to resolve inconsistencies arising in the beliefs of agents over time, and (iii) to compute "reasons" (in the form of arguments) to be communicated to other agents. When agents share from the onset knowledge as to which resources fulfil which goals, the policy allows to compute socially optimal allocations (with a maximal number of agents obtaining a resource that fulfils its goal) finitely and effectively.

7.1 Introduction

Argumentation-based negotiation is a particular class of negotiation allowing agents to provide arguments and justifications as part of the negotiation process, thus increasing the likelihood and/or speed of agreements being reached (see Section 2.3, Chapter 2). We focus in this chapter on substantiating this claim, by showing how AABA can be used to realise the policy of Chapter 5.

We consider the two types of dialogue introduced in Chapter 5 that agents use to try obtain a needed resource. The first, a request, is a dialogue between two agents (an initiator and a responder). The second, a proposal, is a dialogue between three or more agents (an initiator and a set of responders). Agents agree to release resources they have but do not need or agree to exchange a resource for another of at least equal value, and refuse otherwise. We allow agents to provide "reasons" (arguments) with their refusals to other agents, thus filling in gaps in their knowledge.

Whilst there has been discussion on protocols for argumentation-based negotiation (e.g. [vVP05, RSD03]), little work exists on the agents' decision making models to support argumentative negotiation, i.e. how the agents would be built and the strategic reasoning that is to go on in the agents' minds to determine the best course of action at any given time. We adopt a fully "generative" approach to negotiation, by providing constructive policies for the automatic generation of dialogue instances by autonomously reasoning agents.

In order to be fully generative, our negotiation policy presented in Chapter 5 is complemented here by AABA for defining and evaluating the conditions in the procedures of the communication rules constituting the policy. The evaluation includes the computation of arguments. Moreover, AABA allows agents to make assumptions to complete their partial/incomplete beliefs when it is admissible to do so and to deal with inconsistencies arising over time from conflicting information, as well as computing arguments to be provided with dialogue moves.

The chapter is structured as follows: In Section 7.2 we provide the core representation in AABA of the (resource reallocation) agents in agent systems as given in Chapter 3, and generating

dialogue instances as given in Chapter 5. In Section 7.3 we discuss properties of the agents grounded in AABA. In Section 7.4 we give a concrete instantiation of the procedures for the policies of Chapter 5 by means of AABA, and discuss the correlation between the two sets of definitions. In Section 7.5 we present an example of the kinds of dialogue instances our AABA agents would generate. Lastly, in Sections 7.6 and 7.7, we discuss related work and conclude.

7.2 Resource Reallocation Agents as AABA frameworks

Resource reallocation agents in our agent systems (see Definition 5.7) can be represented as AABA frameworks combining an agent's beliefs, owned resources, goal and dialogical status, as well as introducing "reasoning rules" and "assumptions" needed to support/generate dialogue instances. The elements of the AABA framework pertaining to an agent's dialogical status, as well as the "control procedures" outlined in Section 5.3, Chapter 5, can be defined in a number of ways to guarantee its dialogical properties and, straightforwardedly, in a manner that is independent of AABA. Thus we focus instead in this chapter on the elements of the AABA framework needed to define the "general procedures" which are based on an agent's beliefs, goal and resources. We do, however, provide an example definition of an agent's dialogical status and the "control procedures" by means of AABA in Appendix E.

We refer to agent $\langle x, _ , _ , _ \rangle$ as x, as throughout the thesis. An agent x can be represented in AABA as a tuple $\mathcal{AF}_x = \langle \mathcal{L}, \mathcal{R}_c, \mathcal{R}_x^p, \mathcal{A}, c \rangle$ where

- $(\mathcal{L}, \mathcal{R}_c \cup \mathcal{R}_x^p)$ is a *deductive system*, consisting of a language $\mathcal{L}$ and a set $\mathcal{R}_c \cup \mathcal{R}_x^p$ of inference rules ($\mathcal{R}_c \cap \mathcal{R}_x^p = \emptyset$) such that $\mathcal{R}_c$ contains knowledge *common* to all agents in an agent system and $\mathcal{R}_x^p$ contains knowledge personal (or *private*) to x, not necessarily shared by all agents;
- $\mathcal{A} \subseteq \mathcal{L}$ is the set of (candidate) *assumptions*;
- c is a (total) mapping from $\mathcal{A}$ into $\wp(\mathcal{L}) - \{\emptyset\}$, where any $y \in c(x)$ is referred to as a *contrary* of x.

A computational mechanism, called *fact-inclusive dispute derivations* (*fdds*), has been defined in Chapter 6 for computing admissible supports (defences) for conclusions and this will be used in this chapter in defining the procedures occurring in the communication rules of the policies. We use the following terminology in this chapter: given some $L \subseteq \mathcal{L}$,

- $\models_{\mathcal{AF}_x} L$ stands for "there exists a fdd of some defence set $X \subseteq \mathcal{R}^p_x \cup \mathcal{A}$ for L according to $\mathcal{AF}_x$";
- $X \models_{\mathcal{AF}_x} L$ stands for "there exists a fdd of defence set X for L according to $\mathcal{AF}_x$";
- $\not\models_{\mathcal{AF}_x} L$ stands for "it is not the case that $\models_{\mathcal{AF}_x} L$".

Note that we write $\models_{\mathcal{AF}_x} L$ if $\models_{\mathcal{AF}_x} \{L\}$ as in Chapter 6, and likewise $X \models_{\mathcal{AF}_x} L$ and $\not\models_{\mathcal{AF}_x} L$ for $X \models_{\mathcal{AF}_x} \{L\}$ and $\not\models_{\mathcal{AF}_x} \{L\}$ respectively.

Aside from the personal facts, as in Chapter 6, the other elements of the AABA framework are intended to be shared between the agents in the agent system, such that agents can, to some extent, reconstruct arguments exchanged between one another. Indeed the personal facts of an agent is a belief set from which the agent's belief base is derived.

We define now the elements that make up the agent framework for our negotiation domain, beginning with the non-assumptions that make up the language, as follows:

Definition 7.1 (Non-assumptions in $\mathcal{L}$) *The non-assumptions $\mathcal{L} \setminus \mathcal{A}$ are (instances of) the following schemata, for $X, Y \in names(AS)$, $R \in \mathcal{R}$, $Rs \in 2^{\mathcal{R}}$, $G \in \mathcal{G}$, $T \in \mathbb{N}$, $L \in \mathcal{L}$:*

- *$thisAgent(X)$, standing for "the agent's own name is X";*
- *$isAgent(X)$, standing for "X is an agent other than itself in the agent system";*
- *$(\neg)\ has(X, R)$, standing for "X has (does not have) resource R";*
- *$goal(X, G)$, standing for "X's goal is G";*
- *$fulfils(R, G)$, standing for "resource R fulfils goal G";*

- *(¬) useful(R, X), standing for "R is (not) useful for X in that it (does not) fulfils X's goal";*

- *needs(X, Rs), standing for "X's r.r.p. is solved and would become unsolved if it simply gave away the set of resources Rs, or is unsolved and would become solved if it obtained Rs";*

- *needsToRetain(X, Rs), standing for "X's r.r.p. is solved and would become unsolved if it simply gave away the set of resources Rs";*

- *¬needs(X, Rs), standing for "X's r.r.p. is solved and would still be solved even if it simply gave away the set of resources Rs";*

- *gaveAwayAfter(X, R, T), standing for "X has given away R at some point in time after time T";*

- *done$(give(X, R, Y), T)$, standing for "X has given R to Y at time T";*

- *uttered(X, L, T), standing for "X has uttered L at time T".*

Note that *thisAgent*(X), *isAgent*(X), $\neg has(X, R)$, $has(X, R)$, $goal(X, G)$ and $fulfils(R, G)$ as above correspond to literals in $B(x)$. However, as we will see in defining $\mathcal{R}_c$ below (Definition 7.4), $\neg has(X, R)$ and $has(X, R)$ are not meant in $\mathcal{AF}_x$ as primitive facts as they are in the agent's belief set $B(x)$. Instead, $\neg has(X, R)$ and $has(X, R)$ are meant in $\mathcal{AF}_x$ as inferences, as are $useful(R, X)$, $\neg useful(R, X)$, $needs(X, Rs)$, $needsToRetain(X, Rs)$ and $\neg needs(X, Rs)$. Note also the asymmetry in the interpretation of $needs(X, Rs)$ and $\neg needs(X, Rs)$ above. This will be explained when these sentences are used to define $\mathcal{R}_c$ below.

In the absence of beliefs about other agents and in the absence of contrary beliefs, by virtue of their representation as AABA frameworks, agents are allowed to make assumptions about the resources and goals that other agents have. The assumptions that make up the language are as follows:

Definition 7.2 (Assumptions and Contraries) *The assumptions $\mathcal{A}$ (and their associated contraries) in the language $\mathcal{L}$ are all instances of the following schemata, for $X \in names(AS)$, $R \in \mathcal{R}$, $Rs \in 2^{\mathcal{R}}$, $T \in \mathbb{Z}$:*

- *$asm(has(X,R))$, standing for "assume that agent X has resource R",*

  with $c(asm(has(X,R))) = \{\neg has(X,R)\} \cup \{has(Y,R) \mid Y \neq X\}$

- *$asm(\neg has(X,R))$, standing for "assume that X does not have R",*

  with $c(asm(\neg has(X,R))) = \{has(X,R)\}$

- *$asm(\neg useful(R,X))$, standing for "assume that R does not solve the r.r.p. of X",*

  with $c(asm(\neg useful(R,X))) = \{useful(R,X)\}$

- *$asm(needs(X,Rs))$, standing for "assume that X's r.r.p. is solved and would become unsolved if it simply gave away the set of resources Rs, or is unsolved and would become solved if it obtained Rs",*

  with $c(asm(needs(X,Rs))) = \{\neg needs(X,Rs)\}$

- *$asm(\neg needs(X,Rs))$, standing for "assume that X's r.r.p. is solved and would still be solved even if it simply gave away the set of resources Rs, or is unsolved and would still be unsolved even if it obtained Rs",*

  with $c(asm(\neg needs(X,Rs))) = \{needs(X,Rs)\}$

- *$asm(\neg gaveAwayAfter(X,R,T))$, standing for "assume that X has not given away resource R after time T",*

  with $c(asm(\neg gaveAwayAfter(X,R,T))) = \{gaveAwayAfter(X,R,T)\}$

An agent x is initiated with facts in $\mathcal{R}_x^p$ representing which resources it itself has and its goal, as well as beliefs analogous to its beliefs in Definition 3.3 about the identity of the other agents in the system and available plans. Beliefs as to the resources and goals of other agents would be accumulated and modified (within $\mathcal{R}_x^p$) as a result of dialogues. Formally, an agent's initial personal facts are as follows:

Definition 7.3 (Initial Personal Facts) *The set of personal facts in $\mathcal{R}^p_x$ for an agent $x \in names(AS)$ are initially as follows*

$$\{isAgent(Y) \mid Y \in names(AS), Y \neq x\} \cup Pl_{AS}(x) \cup$$
$$\{done(give(x, R, x), 0) \mid R \in Res(x)\} \cup \{goal(x, g) \mid G(x) = g\}$$

The initial facts represent x's beliefs (first two subsets), allocated resources (third subset) and goal (fourth subset). Note that in our AABA representation $has(X, R)$ and $\neg has(X, R)$ are not facts like they are beliefs in the belief set $B(x)$. We will see in Section 7.3 how holding facts regarding the exchange of resources between agents rather than which agents have (or do not have) which resources directly allows us to deal with the possibilities of agents having and communicating to one another conflicting claims as to which agents have (or do not have) which resources, without requiring an explicit removing of facts from $\mathcal{R}^p_x$ to maintain consistency.

Example 7.1 (Initial Personal Facts) *Given plans Pl as in Example 3.1 as follows:*

$$Pl = \{fulfils(r1, g1),\ fulfils(r2, g1),\ fulfils(r1, g2),\ fulfils(r1, g3),\ fulfils(r3, g3\}$$

and agent $ag1$ specified in Example 3.1 as follows:

$Res(ag1) = \{r1\}$	$G(ag1) = g1$	$B(ag1) = \{isAgent(ag2), isAgent(ag3)\} \cup Pl$

Then, initially, $\mathcal{R}^p_{ag1} = \{isAgent(ag2), isAgent(ag3), done(give(ag1, r1, ag1), 0), goal(ag1, g1)\}$ $\cup\ Pl$.

As a consequence of dialogues, agents modify their personal facts (and thus their representation as an AABA framework) by adding *facts* about goals held by other agents, resources changing hands between agents and assumptions uttered by other agents. We define now the general purpose inference rules. These will be used in the next section along with the AABA fdd mechanism for determining when claims and assumptions are "admissible" for an agent, for

computing fdd-arguments to be communicated accordingly, and thereby concretely defining the procedures that make up the negotiation policy.

Definition 7.4 (General-purpose Inference rules in $\mathcal{R}_c$) *The set of general-purpose inference rules $\mathcal{R}_c$ for an agent $x \in names(AS)$ are all the instances of the following schemata, for $X, Y \in names(AS)$, $R \in \mathcal{R}$, $Rs \in 2^{\mathcal{R}}$, $G \in \mathcal{G}$, $T, T' \in \mathbb{N}$:*

$$gaveAwayAfter(X, R, T) \leftarrow done(give(_, R, Y), T'),\ X \neq Y,\ T' > T$$
$$has(X, R) \leftarrow done(give(_, R, X), T),\ asm(\neg gaveAwayAfter(X, R, T))$$
$$\neg has(X, R) \leftarrow has(Y, R),\ X \neq Y$$
$$\neg has(X, R) \leftarrow uttered(X, asm(\neg has(X, R)), _),\ asm(\neg has(X, R))$$
$$\neg has(x, R) \leftarrow asm(\neg has(x, R))$$
$$useful(R, X) \leftarrow goal(X, G),\ fulfils(R, G)$$
$$\neg useful(R, X) \leftarrow asm(\neg useful(R, X))$$
$$\neg needs(X, Rs) \leftarrow has(X, R),\ R \notin Rs,\ useful(R, X)$$
$$needs(X, Rs) \leftarrow R \in Rs,\ useful(R, X),\ asm(needs(X, Rs))$$
$$needsToRetain(X, Rs) \leftarrow R \in Rs,\ has(X, R),\ useful(R, X),\ asm(needs(X, Rs))$$

We describe in the remainder of this section the intuitive reading of the above inference rules as well as demonstrating some of the benefits of an AABA representation of agents. We discuss in Section 7.3 instead more formally properties of the inference rules and our AABA agents defined as in this section.

The first general-purpose inference rule of Definition 7.4 allows an agent x to determine whether an agent X (which may or may not be itself) has given away a resource R after a time T (i.e. $gaveAwayAfter(X, R, T)$). This holds true if x has a belief (personal fact) that some other agent Y has been given R at a time T' which is later than T. The first general-purpose inference rule is used together with the second general-purpose inference rule above to determine whether an agent X can be believed to have a resource R (i.e. $has(X, R)$). Indeed a backward deduction exists for $has(X, R)$ according to x's personal facts if x has a belief that X has been given R at a time T. However, $has(X, R)$ is only considered an "admissible" claim for x if the assumption

$asm(\neg gaveAwayAfter(X, R, T))$ is admisssible for x, i.e. x does not have a contrary belief that X has given R away at a time after T.

We demonstrate in the following three examples the conditions under which $has(X, R)$ is and is not admissible, as well as demonstrating the use of assumptions for completing an agent's information, the use of fdds for generating communicable arguments and how contradictory assertions by agents as to which agents have which resources can be resolved by means of the AABA framework without requiring facts to be removed from an agent's personal facts.

Example 7.2 ($has(X, R)$ and assumptions) *Suppose an agent $ag1$ has no personal facts about a resource $r1$ changing hands between agents (i.e. $done(give(_, r1, _), _)) \notin \mathcal{R}^p_{ag1}$), then $\nvdash_{\mathcal{AF}_{ag1}} has(Ag, r1)$ for all agents $Ag \in names(AS)$. It is also the case that $\nvdash_{\mathcal{AF}_{ag1}} \neg has(Ag, r1)$ for all agents $Ag \in names(AS)$ such that $Ag \neq ag1$ and $uttered(Ag, asm(\neg has(Ag, r1)), _) \notin \mathcal{R}^p_{ag1}$. Then, though $ag1$ has no facts as to which agent has $r1$, it can make the assumption $\models_{\mathcal{AF}_{ag1}} asm(has(Ag, r1))$ for all agents $Ag \in names(AS)$ such that $Ag \neq ag1$ and $uttered(Ag, asm(\neg has(Ag, r1)), _) \notin \mathcal{R}^p_{ag1}$. We will see in Definitions 7.5 and 7.6 how such assumptions are useful for an agent to initiate requests and proposals in the absence of information.*

Example 7.3 ($has(X, R)$ and argument generation) *Suppose a resource $r2$ has changed hands from an agent $ag2$ to an agent $ag3$ at time 1, $ag2$ records this is in its personal facts (i.e. $done(give(ag2, r2, ag3), 1) \in \mathcal{R}^p_{ag2}$) and $ag2$ is not aware of $r2$ changing hands between agents at a later time point. Then,*

$$\{done(give(ag2, r2, ag3), 1),\ asm(\neg gaveAwayAfter(ag3, r2, 1))\} \models_{\mathcal{AF}_{ag2}} has(ag3, r2).$$

Indeed, the defence *and* claim *of such a fdd make up a communicable argument (as understood in Chapter 5)*

$$(has(ag3, r2),\ \{done(give(ag2, r2, ag3), 1),\ asm(\neg gaveAwayAfter(ag3, r2, 1))\})$$

which could be conveyed from $ag2$ to another agent to attempt to convince it that $ag3$ has $r2$.

Example 7.4 ($has(X, R)$ and inconsistency resolution) *Suppose an agent $ag1$ has been told by some agent that $ag3$ has $r2$ based on the fact $done(give(ag2, r2, ag3), 1)$ but told also contradictively by another agent that $ag4$ has $r2$ based on the fact $done(give(ag3, r2, ag4), 2)$. Suppose also that $ag1$'s personal facts $\mathcal{R}^p_{ag1}$ include $done(give(ag2, r2, ag3), 1)$, $done(give(ag3, r2, ag4), 2)$ and no other facts about $r2$ changing hands between agents. Then, the assumption $asm(\neg gaveAwayAfter(ag3, r2, 1))$ is not admissible for $ag1$ as it is for $ag2$ in Example 7.3, there is no backward deduction for $gaveAwayAfter(ag4, r2, 2)$ according to $\mathcal{AF}_{ag1}$, and so,*

$$\{done(give(ag3, r2, ag4), 2),\ asm(\neg gaveAwayAfter(ag4, r2, 2))\} \models_{\mathcal{AF}_{ag1}} has(ag4, r2).$$

The third, fourth and fifth general-purpose inference rules of Definition 7.4 allow an agent x to (respectively) exploit beliefs on the single-unity of resources, determine when an agent X that has previously said it does not have a resource R still does not have it, and impose a form of *closed world assumption* over the resources it does not itself have. Together with the first two rules these rules are used to determine whether an agent X (which may or may not be itself) can be believed to not have a resource R (i.e. $\neg has(X, R)$). This is demonstrated as follows:

Example 7.5 (fdds for $\neg has(X, R)$) *Given $\mathcal{AF}_{ag1}$ as in Example 7.4 such that $done(give(ag2, r2, ag3), 1)$, $done(give(ag3, r2, ag4), 2) \in \mathcal{R}^p_{ag1}$ and $ag1$ has no other facts about $r2$ changing hands between agents. Then,*

$$\{done(give(ag3, r2, ag4), 2),\ asm(\neg gaveAwayAfter(ag4, r2, 2))\} \models_{\mathcal{AF}_{ag1}} \neg has(Ag, r2)$$

for all $Ag \in names(AS)$ such that $Ag \neq ag4$. Moreover, the following fdd exists also for $\neg has(Ag, r2)$ according to $\mathcal{AF}_{ag1}$:

$$\{asm(\neg has(ag1, r2))\} \models_{\mathcal{AF}_{ag1}} \neg has(ag1, r2).$$

The sixth, seventh, eighth and ninth general-purpose inference rules in Definition 7.4 allow an agent x to determine when a resource R is believed to solve the r.r.p. of an agent X irrespective of other resources X may have (i.e. $useful(R, X)$), impose a form of *closed world assumption* over resources not being useful for agents (i.e. $\neg useful(R, X)$), determine when the r.r.p. of

an agent X is believed to be solved irrespective of whether it does or does not have some set of resources Rs (i.e. $\neg needs(X, Rs)$), determine when the r.r.p. of an agent X is believed to be unsolved but would be solved if given a set of resources Rs, or solved and would be unsolved if X gave away a set of resources Rs (i.e. $needs(X, Rs)$). We demonstrate in the following three examples the conditions under which $\neg needs(X, Rs)$ and $needs(X, R)$ are and are not admissible:

Example 7.6 (fdd for $\neg needs(X, Rs)$) *Suppose an agent $ag4$ has a resource $r2$ and $\mathcal{D}_1 \models_{\mathcal{AF}_{ag4}} has(ag4, r2)$ where $\mathcal{D}_1 = \{done(give(ag3, r2, ag4), 2),\ asm(\neg gaveAwayAfter(ag4, r2, 2))\}$. Suppose also that $\mathcal{D}_2 \models_{\mathcal{AF}_{ag4}} useful(r2, ag4)$ where $\mathcal{D}_2 = \{goal(ag4, g1),\ fulfils(r2, g1)\}$. Then, the r.r.p. of $ag4$ is solved by $r2$ and the following fdd exists for all sets of resources Rs such that $r2 \notin Rs$:*

$$\mathcal{D}_1 \cup \mathcal{D}_2 \models_{\mathcal{AF}_{ag4}} \neg needs(ag4, Rs)$$

Trivially, $\not\models_{\mathcal{AF}_{ag4}} asm(needs(ag4, Rs))$ and hence $\not\models_{\mathcal{AF}_{ag4}} needs(ag4, Rs)$ for all sets of resources Rs such that $r2 \notin Rs$.

Example 7.7 (fdd for $needs(X, Rs)$ where agent's r.r.p. is solved) *Given $\mathcal{AF}_{ag4}$ as in Example 7.6 for an agent $ag4$ whose r.r.p. is solved and such that for all resources $R \neq r2$: $\not\models_{\mathcal{AF}_{ag4}} \{has(ag4, R), useful(R, ag4)\}$, i.e. $ag4$ has no* useful *resource other than $r2$. Then,*

$$\mathcal{D}_2 \cup \{asm(needs(ag4, \{r2\}))\} \models_{\mathcal{AF}_{ag4}} needs(ag4, \{r2\})$$

where $\mathcal{D}_2$ is a defence for $useful(r2, ag4)$ as in Example 7.6.

Suppose instead that $\mathcal{D}_3 \models_{\mathcal{AF}_{ag4}} has(ag4, r3)$ and $\mathcal{D}_4 \models_{\mathcal{AF}_{ag4}} useful(r3, ag4)$ for some resource $r3 \neq r2$ and defence sets $\mathcal{D}_3$ and $\mathcal{D}_4$, then the following fdd exists for all sets of resources Rs such that $r3 \notin Rs$ (including $Rs = \{r2\}$):

$$\mathcal{D}_3 \cup \mathcal{D}_4 \models_{\mathcal{AF}_{ag4}} \neg needs(ag4, Rs).$$

The fdd of Example 7.6 also still holds for all sets of resources Rs such that $r2 \notin Rs$ (including $Rs = \{r3\}$). Moreover, now, $\not\models_{\mathcal{AF}_{ag4}} needs(ag4, \{r2\})$ since $\models_{\mathcal{AF}_{ag4}} \neg needs(ag4, \{r2\})$. In-

tuitively, this is because an agent only needs *one resource that solves its r.r.p., so any set of resources not containing either of* $r2$ *or* $r3$ *is not "needed" by* $ag4$.

Example 7.8 (fdd for $needs(X, Rs)$ where agent's r.r.p. is not solved) *Suppose an agent* $ag1$ *is such that* $\not\models_{\mathcal{AF}_{ag1}} \{has(ag1, R), useful(R, ag1)\}$ *for all resources* $R \in \mathcal{R}$, *i.e.* $ag1$ *has no resource that solves it r.r.p. Then,* $\models_{\mathcal{AF}_{ag1}} needs(ag1, Rs)$ *for all sets of resources* Rs *such that* $R \in Rs$ *and* $\models_{\mathcal{AF}_{ag1}} useful(R, ag1)$.

The last general-purpose inference rule ($needsToRetain(X, Rs)$) in Definition 7.4 is exactly the same as the inference for $needs(X, Rs)$ except that it is more specific in that it is for the case where X is believed to have a resource $R \in Rs$, which if X gave away together with the other resources in Rs, X's r.r.p. would no longer be solved. Naturally, the defence of an fdd for $needsToRetain(X, Rs)$ would include that X has such a resource R. This will be useful, as we will see in Section 7.4, for the agent when refusing a request or proposal and providing argument as to which particular resource, if given away, would cause its r.r.p. to become unsolved.

We end this section with an example of the kinds of claims that could be admissible for an agent at start-up.

Example 7.9 (admissible claims at start-up) *Given* $\mathcal{AF}_{ag1} = \langle \mathcal{L}, \mathcal{R}_c, \mathcal{R}^p_{ag1}, \mathcal{A}, c \rangle$ *where* $\mathcal{L}$, $\mathcal{R}_c$, $\mathcal{A}$ *and* c *are as defined in this section, and* $\mathcal{R}^p_{ag1}$ *is as in Example 7.1, some of the claims admissible for* $ag1$ *initially are:* $has(ag1, r1)$, $\neg has(ag2, r1)$, $\neg has(ag1, r2)$, $useful(r1, ag1)$, $needs(ag1, r1)$, $\neg needs(ag1, r2)$.

Note that a choice of two assumptions which conflict in their pragmatic meaning could be admissible for an agent, though of course an assumption and its *contrary* claims can never both be admissible. For example, in the example above, $ag1$ could (admissibly) assume either $asm(has(ag2, r2))$ or $asm(\neg has(\ ag2, r2))$. This is because $ag1$ has no information regarding whether or not $ag2$ has $r2$, and thus no contrary beliefs for these assumptions. Likewise, $ag1$ could (admissibly) assume either $asm(needs(ag2, r2))$ or $asm(\neg needs(ag2,\ r2))$. Indeed it is important for agents to be able to make assumptions in the absence of beliefs to proceed with obtaining resources that fulfil their goals, as we will see in Section 7.4.

7.3 Consistency Properties of an AABA agent

We prove in this section some consistency properties of an AABA agent x defined as in Section 7.2. We assume $\mathcal{R}_x^p$ is such that if $done(give(Y, R, Z), T)$, $done(give(Y', R, Z'), T) \in \mathcal{R}_x^p$ then $Y = Y'$ and $Z = Z'$ (i.e. resources are not given away or received by two different agents at the same point in time and agents have no facts as such). We assume also that facts $done(give(Y, R, X), T)$ are correct (i.e. such a fact originates and is communicated only if Y has given R to X at time T).

Structuring the first two inference rules of Definition 7.4 for determining when an agent has a resource as they are avoids the possibility of holding conflicting beliefs in this regard. Notably, an agent x will never believe a resource to be held by two agents simultaneously, as in the lemma below. In addition, the need to manually remove facts from $\mathcal{R}_x^p$ as to which agent has a given resource for the sake of maintaining consistency of the agent's belief set (personal facts) is eliminated.

Lemma 7.1 (consistency of *has*) *If $\models_{\mathcal{AF}_x} has(X, R)$ for some agent $X \in names(AS)$ and resource $R \in \mathcal{R}$, then $\not\models_{\mathcal{AF}_x} has(Y, R)$ for all agents $Y \in names(AS)$ such that $Y \neq X$.*

Proof. Assume, for the sake of contradiction, that

- $\{done(give(_, R, X), T), asm(\neg gaveAwayAfter(X, R, T))\} \models_{\mathcal{AF}_x} has(X, R)$; and
- $\{done(give(_, R, Y), T'), asm(\neg gaveAwayAfter(Y, R, T'))\} \models_{\mathcal{AF}_x} has(Y, R)$

for some $T, T' \in \mathbb{N}$ and $Y \in names(AS)$ such that $Y \neq X$. Note that $T \neq T'$ since, by assumption, a resource R cannot be received by two agents at the same point in time and facts about resources changing hands are correct. Then, either

- $T' > T$ and hence $\{done(give(_, R, Y), T')\} \models_{\mathcal{AF}_x} gaveAwayAfter(X, R, T)$. But then $asm(\neg gaveAwayAfter(X, R, T))$ cannot be an admissible assumption defending the claim $has(X, R)$; or

- $T > T'$ and hence $\{done(give(_, R, X), T)\} \models_{\mathcal{AF}_x} gaveAwayAfter(Y, R, T')$. But then $asm(\neg gaveAwayAfter(Y, R,\ T'))$ cannot be an admissible assumption defending the claim $has(Y, R)$.

Contradiction. □

Likewise for $has(X, R)$, structuring the inference rules for determining when an agent does not have some resource according to Definition 7.4 avoids the need to manually remove facts as to which agents do not have a a given resource for the sake of maintaining consistency of the agent's belief set (personal facts). Also, the possibility of deriving conflicting claims regarding an agent both having and not having a certain resource is avoided, as follows:

Lemma 7.2 (consistency of *has* and $\neg$*has*) *It can never be the case that both* $\models_{\mathcal{AF}_x} has(X, R)$ *and* $\models_{\mathcal{AF}_x} \neg has(X, R)$ *for any agent* $X \in names(AS)$ *and resource* $R \in \mathcal{R}$.

Proof. Assume, for the sake of contradiction, that $\models_{\mathcal{AF}_x} has(X, R)$ and $\models_{\mathcal{AF}_x} \neg has(X, R)$ for some $X \in names(AS)$ and $R \in \mathcal{R}$. Then, for it to be the case that $\models_{\mathcal{AF}_x} \neg has(X, R)$, either $\models_{\mathcal{AF}_x} has(Y, R)$ for some $Y \neq X$ (by the third inference rule of Definition 7.4) or $\models_{\mathcal{AF}_x} asm(\neg has(X, R))$ (by the fourth and fifth inference rules of Definition 7.4). We look at each case in turn. Firstly, if $\models_{\mathcal{AF}_x} has(Y, R)$ for some $Y \neq X$, then, by Lemma 7.1, $\not\models_{\mathcal{AF}_x} has(X, R)$. Contradiction. Secondly, if $\models_{\mathcal{AF}_x} asm(\neg has(X, R))$, then it cannot be the case that the contrary $has(X, R)$ is admissible, i.e. $\not\models_{\mathcal{AF}_x} has(X, R)$. Contradiction. □

Similarly as for Lemma 7.2 whereby an agent will never claim an agent to both have and not have a particular resource, an agent will never claim an agent to both *need* and *not need* a particular set of resources, as follows:

Lemma 7.3 (consistency of *needs* and $\neg$*needs*) *It can never be the case that both* $\models_{\mathcal{AF}_x} needs(X, Rs)$ *and* $\models_{\mathcal{AF}_x} \neg needs(X,\ Rs)$.

Proof. Assume, for the sake of contradiction, that both $\mathcal{D}_1 \models_{\mathcal{AF}_x} needs(X, Rs)$ and $\mathcal{D}_2 \models_{\mathcal{AF}_x} \neg needs(X, Rs)$ where $\mathcal{D}_1$ and $\mathcal{D}_2$ are admissible defence sets. Then, according to Definition 7.4,

$\mathcal{D}_1$ consists of a single (admissible) assumption $asm(needs(X, Rs))$. However, it cannot then be that the contrary $\neg needs(X, Rs)$ is an admissible claim, i.e. $\not\models_{\mathcal{AF}_x} \neg needs(X, Rs)$. Contradiction. □

We define $\vdash^x_{asm}$ and $\vdash^x$ (see Definition 5.10) now in terms of $\models_{\mathcal{AF}_x}$ for the sentences of the meta-language $\mathcal{ML}$ (see Definition 5.11) occurring in Definitions 5.12–5.15, as follows:

- $\vdash^x$ [Y has R] iff $\models_{\mathcal{AF}_x} has(Y, R)$;
- $\vdash^x \neg$[Y has R] iff $\models_{\mathcal{AF}_x} \neg has(Y, R)$;
- $\vdash^x_{asm}$ [Y has R] iff $\models_{\mathcal{AF}_x} asm(has(Y, R))$;
- $\vdash^x_{asm} \neg$[Y has R] iff $\models_{\mathcal{AF}_x} asm(\neg has(Y, R))$;
- $\vdash^x$ [the r.r.p. of Y is solved with R] iff $\models_{\mathcal{AF}_x} useful(R, Y)$;
- $\vdash^x \neg$[the r.r.p. of Y is solved with R] iff $\models_{\mathcal{AF}_x} \neg useful(R, Y)$;
- $\vdash^x_{asm} \neg$[the r.r.p. of Y is solved with R] iff $\models_{\mathcal{AF}_x} asm(\neg useful(R, Y))$;
- $\vdash^x$ [the r.r.p. of Y is solved] iff $\models_{\mathcal{AF}_x} \{useful(R, Y), has(Y, R)\}$ for some resource R;
- $\vdash^x$ [the r.r.p. of Y would *still* be solved or unsolved by simply giving away R] iff $\models_{\mathcal{AF}_x} \neg needs(Y, \{R\})$;
- $\vdash^x \neg$[the r.r.p. of Y would *still* be solved or unsolved by simply giving away R] iff $\models_{\mathcal{AF}_x} needs(Y, \{R\})$;
- $\vdash^x_{asm}$ [the r.r.p. of Y would *still* be solved or unsolved by simply giving away R] iff $\models_{\mathcal{AF}_x} asm(\neg needs(Y, \{R\}))$;
- $\vdash^x_{asm} \neg$[the r.r.p. of Y would *still* be solved or unsolved by simply giving away R] iff $\models_{\mathcal{AF}_x} asm(needs(Y, \{R\}))$;
- Given some set of resources Rs: $\vdash^x$ [after giving away each resource in Rs, the r.r.p. of Y would *no longer* be solved] iff $\models_{\mathcal{AF}_x} needsToRetain(Y, Rs)$.

Further, we donote the following conditions occurring in Definitions 5.12–5.15 in terms of $\mathcal{AF}_x$:

- the r.r.p. of x is not solved and `fulfils(`R`,`$G(x)$`)` $\in B(x)$ iff $\models_{\mathcal{AF}_x} \{\neg has(x,R), needs(x,\{R\})\}$;
- $Y \in names(AS)$ and $Y \neq x$ iff $\models_{\mathcal{AF}_x} isAgent(Y)$;
- It is the case that each of $\vdash^x$ [Y has R], $\vdash^x$ [the r.r.p. of Y is solved], $\vdash^x \neg$[the r.r.p. of Y would *still* be solved or unsolved by simply giving away R] and $\vdash^x$ [the r.r.p. of Y is solved with R'] holds iff $\models_{\mathcal{AF}_x} \{has(Y,R), needs(Y,\{R\}), useful(R',Y)\}$;
- Given some set of resources Rs: (for all $R \in Rs$, `fulfils(`R`,`$G(x)$`)` $\notin B(x)$, or there is some $R \notin Rs$ such that $R \in Res(x)$ and `fulfils(`R`,`$G(x)$`)` $\in B(x)$) iff ($\models_{\mathcal{AF}_x} asm(\neg needs(x,Rs))$).

The definition of $\mathcal{AF}_x$ (Definitions 7.1–7.4) as well as the consistency properties demonstrated in this section (Lemmas 7.1–7.3) make the mapping of $\vdash^x_{asm}$ and $\vdash^x$ to $\models_{\mathcal{AF}_x}$ as above legitimate according to the requirements of Postulate 5.1. Notably, for some sentence $[L] \in \mathcal{ML}$, it is not possible that $\vdash^x [L]$ and $\vdash^x \neg[L]$, nor is it possible that $\vdash^x_{asm} [L]$ and $\vdash^x \neg[L]$.

7.4 Procedures defined using AABA

We re-define now the procedures of Section 5.4 for an agent x in terms of the AABA framework $\mathcal{AF}_x$ defined above.

Firstly, the agent x requests for another agent Y to simply give away a resource R which x needs or to exchange R for another resource R' which x has according to the `generateRequest` procedure, which occurs in the *preconditions* of communication rule *CR1*, as follows:

Definition 7.5 (`generateRequest(`Y`,Actions)`) *The procedure evaluates to true instantiating an intended recipient of the request* Y *and an exchange of resources* `Actions` *to request if*

(1) $\models_{\mathcal{AF}_x} \neg has(x, R)$, $needs(x, \{R\})$*; (2)* $\models_{\mathcal{AF}_x} isAgent(Y)$*; and (3)* `Actions` *takes one of the following two forms:*

(a) {`give(`Y`,`R`,`x`)`}*, if the following conditions hold:*

(i) $\models_{\mathcal{AF}_x} has(Y, R)$ *or* $\models_{\mathcal{AF}_x} asm(has(Y, R))$*; and*

(ii) $\models_{\mathcal{AF}_x} \neg needs(Y, \{R\})$ *or* $\models_{\mathcal{AF}_x} asm(\neg needs(Y, \{R\}))$.

(b) {`give(`Y`,`R`,`x`), give(`x`,`R'`,`Y`)`}*, if the following conditions hold:*

(i) $\models_{\mathcal{AF}_x} has(Y, R)$, $needs(Y, \{R\})$, $useful(R', Y)$*; and*

(ii) $\models_{\mathcal{AF}_x} has(x, R')$.

Otherwise, if the above conditions are not met, the procedure evaluates to false.

We show next that the definition of `generateRequest` given above in terms of AABA is synonymous to the non-AABA definition of `generateRequest` given in Chapter 5, and thus Lemma 5.3 holds for the AABA definition also.

Lemma 7.4 (correspondence of `generateRequest(`Y`,Actions)` procedures) *The* `generateRequest(`Y`,Actions)` *procedure according to Definition 7.5 evaluates to true iff the* `generateRequest(`Y`, Actions)` *procedure of Definition 5.12 evaluates to true.*

Proof. Under both the non-AABA and AABA definitions of `generateRequest`, `Actions` can take the form {`give(`Y`,`R`,`x`)`} or {`give(`Y`,`R`,`x`),give(`x`,`R'`,`Y`)`}. We show that whenever the conditions for the AABA definition hold, they hold also for the corresponding conditions of the non-AABA definition, and vice versa: Firstly, Conditions 1 and 2 of Definition 7.5 hold iff the corresponding Conditions 1 and 2 (respectively) of Definition 5.12 hold. These conditions are common to both forms of `Actions`. Secondly, Conditions 3ai and 3aii of Definition 7.5 hold iff the corresponding Conditions 3ai and 3aii of Definition 5.12 hold. Thus, when all necessary conditions of Definition 7.5 hold for the case where `Actions` takes the form {`give(`Y`,`R`,`x`)`}, all necessary conditions of Definition 5.12 hold also, and vice versa. Thirdly, Conditions 3bi

and 3bii of Definition 7.5 hold iff the corresponding Conditions 3bi and 3bii (respectively) of Definition 5.12 hold. Thus, when all necessary conditions of Definition 7.5 hold for the case where `Actions` takes the form {`give(`Y`,`R`,`x`), give(`x`,`R'`,`Y`)`}, all necessary conditions of Definition 5.12 hold also, and vice versa. □

The `generateProposal` procedure, which occurs in the *preconditions* of communication rule *CR2*, on the other hand is used for proposing exchanges of resources involving two or more agents besides x, as follows:

Definition 7.6 (`generateProposal(Z,Actions)`) *The procedure evaluates to true instantiating a set of intended recipients of the proposal Z and an exchange of resources* `Actions` *to propose if (1)* $\models_{\mathcal{AF}_x} \neg has(x, R_2), needs(x, \{R_2\})$*; and (2) for each* $Y \in Z$, $\models_{\mathcal{AF}_x} isAgent(Y)$*; and (3)* `Actions` *takes the form* {`give(`Y_2`,`R_2`,`Y_1`)`, ..., `give(`Y_{n+1}`,`R_{n+1}`,`Y_n`)`} *for* $n > 1$, $\{Y_1, \ldots, Y_{n+1}\} = \{x\} \cup Z$, $Y_1 = x$, $|Z| > 1$, $Y_2, \ldots, Y_n \in Z$, $Y_2, \ldots, Y_n$ *all different,* $Y_{n+1} \in \{x\} \cup Z$, $Y_n \neq Y_{n+1}$, $R_1, \ldots R_{n+1}$ *all different, and satisfying the following conditions:*

(a) for all $Y_i \in \{Y_2, \ldots, Y_n\}$, $\models_{\mathcal{AF}_x} has(Y_i, R_i), needs(Y_i, \{R_i\}), useful(R_{i+1}, Y_i)$*;*

(b) (i) $Y_{n+1} = x$ *and* $\models_{\mathcal{AF}_x} has(x, R_{n+1})$*; or*

(ii) $Y_{n+1} \in \{Y_2, \ldots, Y_{n-1}\}$*, and either* $\models_{\mathcal{AF}_x} has(Y_{n+1}, R_{n+1})$ *or* $\models_{\mathcal{AF}_x} asm(has(Y_{n+1}, R_{n+1}))$*; or*

(iii) $Y_{n+1} \notin \{x\} \cup \{Y_2, \ldots, Y_{n-1}\}$*, and*

- $\models_{\mathcal{AF}_x} has(Y_{n+1}, R_{n+1})$ *or* $\models_{\mathcal{AF}_x} asm(has(Y_{n+1}, R_{n+1}))$*, and*
- $\models_{\mathcal{AF}_x} \neg needs(Y_{n+1}, \{R_{n+1}\})$ *or* $\models_{\mathcal{AF}_x} asm(\neg needs(Y_{n+1}, \{R_{n+1}\}))$.

Otherwise, if the above conditions are not met, the procedure evaluates to false.

We show next that the definition of `generateProposal` given above in terms of AABA is synonymous to the non-AABA definition of `generateProposal` given in Chapter 5, and thus Lemma 5.4 holds for the AABA definition also.

Lemma 7.5 (correspondence of `generateProposal(`Z`,Actions)` procedures) *The* `generateProposal(`Z`,Actions)` *procedure according to Definition 7.6 evalutes to true iff the* `generateProposal(`Z`, Actions)` *procedure of Definition 5.13 evalutes to true.*

Proof. Under both the non-AABA and AABA definitions of `generateRequest`, `Actions` takes the form $\{$`give(`Y_2`,`R_2`,`Y_1`)`, ..., `give(`Y_{n+1}`,`R_{n+1}`,`Y_n`)`$\}$ for $n > 1$, $\{Y_1, \ldots, Y_{n+1}\} = \{x\} \cup Z$, $Y_1 = x$, $x \notin Z$, $|Z| > 1$, $Y_2, \ldots, Y_n \in Z$, $Y_2, \ldots, Y_n$ all different, $Y_{n+1} \in \{x\} \cup Z$, $Y_n \neq Y_{n+1}$, $R_1, \ldots R_{n+1}$ all different. It is trivial to see that each of the Conditions 1, 2, 3a and 3bi–3biii of Definition 7.6 holds iff the corresponding Condition 1, 2, 3a and 3bi–3biii (respectively) of Definition 5.13 holds. □

The `acceptableExchange` procedure, which occurs in the *preconditions* of communication rules *CR3* and *CR5*, is used when the agent takes the role of responder in a dialogue instance for deciding when an exchange of resources is acceptable for it, as follows:

Definition 7.7 (`acceptableExchange(Actions)`) *The procedure evaluates to true if* $Rs = \{$ $R \mid$ `give(`x`, `R`,_)` $\in$ `Actions` $\}$ *and: (1) for each* $R \in Rs$, $\models_{\mathcal{AF}_x} has(x, R)$*; and (2) any one of the following conditions holds:*

(a) $\models_{\mathcal{AF}_x} asm(\neg needs(x, Rs))$*, or*

(b) `give(_,`R'`,`x`)` $\in$ `Actions` *and* $\models_{\mathcal{AF}_x} useful(R', x)$*.*

Otherwise, if the above conditions are not met, the procedure evaluates to false.

We show next that the definition of `acceptableExchange` given above in terms of AABA is synonymous to the non-AABA definition of `acceptableExchange` given in Chapter 5, and thus Lemma 5.5 holds for the AABA definition also.

Lemma 7.6 (correspondence of `acceptableExchange(Actions)` procedures) *The* `acceptableExchange(Actions)` *procedure according to Definition 7.7 evalutes to true iff the* `acceptableExchange(Actions)` *procedure of Definition 5.14 evalutes to true.*

Proof. It is trivial to see that Condition 1 of Definition 7.7 holds iff the corresponding Condition 1 of Definition 5.14 holds. Moreover, Condition 2a of Definition 7.7 holds iff Condition 2a or 2b of Definition 5.14 holds, and Condition 2b of Definition 7.7 holds iff Condition 2c of Definition 5.14 holds. □

The ¬`acceptableExchange` procedure, which occurs in the *preconditions* of communication rules *CR4* and *CR6*, is used when the agent takes the role of responder in a dialogue instance for deciding when an exchange of resources is not acceptable for it. The procedure generates an argument according to the reason for which the exchange of resources is not acceptable. This may be because the agent does not have a resource which it would be required to give away in the exchange, as follows:

Definition 7.8 (¬`acceptableExchange(Actions,Reason)`**)** *The procedure evaluates to true instantiating an argument* `Reason` *where* `Reason` *takes one of the following two forms:*

(a) `<`$\neg has(x, R)$`, Support>`*: if* `give(`x`,`R`,_ )` $\in$ `Actions` *and* `Support` $\models_{\mathcal{AF}_x} \neg has(x, R)$*; or*

(b) `<`$needsToRetain(x, Rs)$`, Support>`*: if* $Rs = \{ R \mid$ `give(`x`,`R`,_)` $\in$ `Actions` $\}$*, and (i)* `Support` $\models_{\mathcal{AF}_x} needsToRetain(x, Rs)$*, and (ii) there is no* `give(_,`R'`,`x`)` $\in$ `Actions` *such that* $\models_{\mathcal{AF}_x} useful(R', x)$.

Otherwise, if neither of the above conditions are met, the procedure evaluates to false.

Note that the argument generated as in Condition (a) above would include in the support who the agent believed to have the resource according to the associated general-purpose inference rule in $\mathcal{R}_c$, if it had such a belief. The other reason why the exchange of resources may not be acceptable for the agent (Condition (b)) is because it *needs to retain* the set of resources which it is required to give away in the exchange.

We show next that the definition of ¬`acceptableExchange` given above in terms of AABA is synonymous to the non-AABA definition of ¬`acceptableExchange` given in Chapter 5.

Lemma 7.7 (correspondence of ¬acceptableExchange(Actions,Reason) procedures) *The* ¬acceptableExchange(Actions,Reason) *procedure according to Definition 7.8 evaluates to true iff the* ¬acceptableExchange(Actions,Reason) *procedure of Definition 5.15 evaluates to true.*

Proof. It is trivial to see that Condition (a) or (b) of Definition 7.8 holds iff the corresponding Condition (a) or (b) (respectively) of Definition 5.15 holds. □

The exchangeResources procedure, which occurs in the *revisions* of communication rules *CR3*, *CR7*, *CR11* and *CR14*, records in the agent's belief base an exchanges of resources that it is involved in, as initiator or responder of a dialogue instance, as follows:

Definition 7.9 (exchangeResources(Actions)) *For each* give(Y,R,Z) ∈ Actions*, x adds $done(give(Y, R, Z), T)$ to $\mathcal{R}_x^p$, where T is the current time.*

We show next that the definition of exchangeResources given above in terms of AABA revises the agent's state as to which agents have which resources synonymously to the non-AABA definition of exchangeResources given in Chapter 5.

Lemma 7.8 (correspondence of exchangeResources(Actions) procedures) *The beliefs of agent x regarding the owners of resources following execution of the* exchangeResources(Actions) *procedure of Definition 7.9 corresponds to the beliefs of x following execution of the* exchangeResources(Actions) *procedure of Definition 5.16.*

Proof. According to Definition 7.9, for each give(Y,R,Z) ∈ Actions, x adds $done(give(Y, R, Z), T)$ to $\mathcal{R}_x^p$, where T is the current time, and thus $\models_{\mathcal{AF}_x} has(Z, R)$. Likewise, according to Definition 5.14:

- for each give(Y,R,x) ∈ Actions, the revised resources of x (i.e. $Res'(x)$) are such that $R \in Res'(x)$; and

- for each `give(`Y`,`R`,`Z`)` $\in$ `Actions` ($Z \neq x$), the revised beliefs of x (i.e. $B'(x)$) are such that $has(Z, R) \in B'(x)$.

Thus, for each `give(`Y`,`R`,`Z`)` $\in$ `Actions`, $\vdash^x$ [Z has R]. □

The `reviseBeliefs` procedure, which occurs in the *revisions* of communication rules *CR3–CR6*, *CR8* and *CR10*, modifies the agent's set of private facts according to an argument provided to it by another agent, as follows:

Definition 7.10 (`reviseBeliefs(`Y`,<Conclusion,Support>)`) *For each* $A \in$ `Support` *such that* $A \in \mathcal{A}$ *(i.e. A is an* assumption*), x adds uttered(Y, A, T) to $\mathcal{R}_x^p$, where T is the current time, and, for each* $F \in$ `Support` *such that* $F \notin \mathcal{A}$ *(i.e. F is a* fact*), x adds F to $\mathcal{R}_x^p$.*

Note that such revisions make sense since agents are assumed to be trustworthy and only share facts (in arguments) that they themselves hold to be true (admissible).

We have shown in Lemmas 7.4–7.8 the correspondence between Definitions 7.5–7.9 and Definitions 5.12–5.16. As a result, it is clear that all successful dialogue instances generated by the agents grounded in AABA do indeed result in minimally cooperatively rational deals, as follows:

Corollary 7.1 (successful dialogue instances generated are Min CR) *A successful dialogue instance generated by an initiating (AABA) agent x and set of responding (AABA) agents Z all defined according to Definitions 7.1–7.4 and all using the communication rules* CR1–CR15 *according to the definitions of the procedures given in Definitions 7.5–7.9 results in a minimally cooperatively rational deal $\delta = (Res, Res')$ where Res is the allocation before the dialogue instance and Res' is the allocation after the dialogue instance.*

The result above is an immediate consequence of Corollary 5.2 and Lemmas 7.4–7.8.

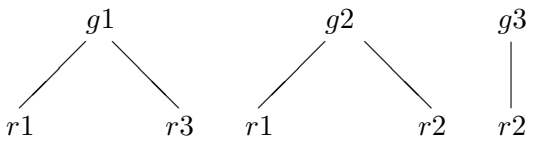


Figure 7.1: Plans Pl in Example 7.10.

7.5 Example of Dialogue Instances Generated by the AABA Agents

We present in this section an example of the different kinds of dialogue instances our AABA agents would generate in pursuit of resources that solve their r.r.p.

Example 7.10 (dialogue instances generated by AABA agents) *Consider a system of three agents, $ag1$, $ag2$, $ag3$, and a set of plans Pl (depicted in Figure 7.1) which all agents are aware of, as follows:*

$Res(ag1) = \{r1\}$, $G(ag1) = g1$

$Res(ag2) = \{r2\}$, $G(ag2) = g2$

$Res(ag3) = \{r3\}$, $G(ag3) = g3$

$Pl = \{fulfils(r1, g1),\ fulfils(r3, g1),\ fulfils(r1, g2),\ fulfils(r2, g2),\ fulfils(r2, g3)\}$

and with AABA representations and procedures as defined in Sections 7.2 and 7.4. Let $\mathcal{AF}_{ag1}$, $\mathcal{AF}_{ag2}$ and $\mathcal{AF}_{ag3}$ denote the AABA representations of agents $ag1$, $ag2$ and $ag3$ respectively.

The r.r.p. of $ag1$ and $ag2$ is solved but the r.r.p. of $ag3$ is not. $ag3$, starting with no knowledge of the other agents, would proceed to try and solve its r.r.p. as follows:

***t=0**: It is the case that $\models_{\mathcal{AF}_{ag3}} needs(ag3, \{r2\})$. Also, in the absence of contrary beliefs, the assumptions $asm(has(ag1, r2))$, $asm(\neg needs(ag1, \{r2\}))$, $asm(has(ag2, r2))$ and $asm(\neg needs(ag2, \{r2\}))$ are all admissible for $ag3$. Based on this, the procedure* `generateRequest(`Y`,Actions)` *evaluates to true for $Y = ag1$ and* `Actions` $= \{$`give(`$ag1$`,`$r2$`,`

$ag3$)}, *as well as* $Y = ag2$ *and* Actions = {give($ag2$,$r2$,$ag3$)}. *So, by* CR1, $ag3$ *can initiate request with* $ag1$ *or* $ag2$. *Assume it selects* $ag1$, *then it utters*

request($ag3$, $ag1$, {give($ag1$,$r2$,$ag3$)})

t=1*:* $ag1$ *receives* $ag3$*'s request by* CR2 *since the procedure* ¬acceptableExchange({give($ag1$, $r2$,$ag3$)},Reason) *evaluates to true for* Reason = <$\neg has(ag1, r2)$,{$asm(\neg has(ag1, r2))$}>. *In so doing,* $ag1$ *utters*

refuse($ag1$, $ag3$, {give($ag1$,$r2$,$ag3$)},<$\neg has(ag1, r2)$,{$asm(\neg has(ag1, r2))$}>)

t=2*:* $ag3$ *receives* $ag1$*'s refusal by* CR8 *and executes the* reviseBeliefs *procedure. In so doing,* $ag3$ *adds* $uttered(ag1, asm(\neg has(ag1, r2)), 2)$ *to* $\mathcal{R}^p_{ag3}$.

t=3*: Now,* $\not\models_{\mathcal{AF}_{ag3}} asm(has(ag1, r2))$ *and so* generateRequest(Y,Actions) *evaluates to true for* $Y = ag2$ *and* Actions = {give($ag2$,$r2$, $ag3$)} *only. So, by* CR1, $ag3$ *utters*

request($ag3$, $ag2$, {give($ag2$,$r2$,$ag3$)})

t=4*:* $ag2$ *receives* $ag3$*'s request by* CR2 *since the procedure* ¬acceptableExchange({give($ag2$, $r2$,$ag3$)},Reason) *evaluates to true for* Reason = <$needsToRetain(ag2, \{r2\})$,Support> *where* Support = $\{goal(ag2, g2)$, $fulfils(r2, g2)$, $done(give(ag2, r2, ag2), 0)$, $asm($ $\neg gaveAwayAfter(ag2, r2, 0))$, $asm(needs(ag2, \{r2\}))\}$. *In so doing,* $ag2$ *utters*

refuse($ag2$, $ag3$, {give($ag2$,$r2$,$ag3$)}, <$needsToRetain(ag2, \{r2\})$,Support>)

t=5*:* $ag3$ *receives* $ag2$*'s refusal by* CR8 *and executes the* reviseBeliefs *procedure according to the providing argument. In so doing,* $ag3$ *adds* F *to* $\mathcal{R}^p_{ag3}$ *for each* fact F *in* Support, *and* $uttered(ag2, A, 5)$ *for each* assumption A *in* Support.

t=6*: Now,* $\models_{\mathcal{AF}_{ag3}} needs(ag2, \{r2\})$. *However, given that* $\models_{\mathcal{AF}_{ag3}} useful(r1, ag2)$ *also, and that* $asm(has(ag1, r1))$ *and* $asm(\neg needs(ag1, \{r1\}))$ *are admissible assumptions, the procedure* generateProposal(Z,Actions) *evaluates to true for* $Z = \{ag1, ag2\}$ *and* Actions = {give($ag2$,$r2$,$ag3$),give($ag1$,$r1$,$ag2$)}. *Hence* $ag3$ *utters*

propose($ag3$, {$ag1$,$ag2$}, {give($ag2$,$r2$,$ag3$),give($ag1$,$r1$,$ag2$)})

t=7*: $ag2$ receives $ag3$'s proposal by* CR5 *and responds with* `accept_proposal` *since the procedure* `acceptableExchange({give(`$ag2$`,`$r2$`,`$ag3$`),give(`$ag1$`,`$r1$`,`$ag2$`)})` *evaluates to true.*

Separately, $ag1$ receives $ag3$'s proposal by CR6 *but responds with* `reject_proposal` *since the procedure* `¬acceptableEchange({give(`$ag2$`,`$r2$`,`$ag3$`),give(`$ag1$`,`$r1$`,`$ag2$`)},<`$needsToRetain(ag1, \{r1\})$`, Support>)` *evaluates to true, where* `Support` $= \{goal(ag1, g1)$, $fulfils(r1, g1)$, $done(give(ag1, r1, ag1), 0)$, $asm(\neg gaveAwayAfter(ag1, r1, 0))$, $asm(needs(ag1, \{r1\}))\}$.

t=8*: $ag3$ receives $ag2$'s* `accept_proposal` *utterance by* CR9.

t=9*: $ag3$ receives $ag1$'s* `reject_proposal` *utterance by* CR10 *and executes the* `reviseBeliefs` *procedure according to the provided argument.*

t=10*: Applying* CR12, *$ag3$ utters*

`disconfirm(`$ag3$`, {`$ag1$`}, {give(`$ag2$`,`$r2$`,`$ag3$`),give(`$ag1$`,`$r1$`,`$ag2$`)})`

t=11*: Now, $\not\models_{\mathcal{AF}_{ag3}} asm(\neg needs(ag1, \{r1\}))$. In addition, $\models_{\mathcal{AF}_{ag3}} needs(ag1, \{r1\})$ so the previous proposal cannot be generated. Instead, now that $\models_{\mathcal{AF}_{ag3}} useful(r3, ag1)$, as well as $\models_{\mathcal{AF}_{ag3}} has(ag3, r3)$ as before, the procedure* `generateProposal(`Z`,Actions)` *evaluates to true for $Z = \{ag1, ag2\}$ and* `Actions` $= \{$`give(`$ag2$`,`$r2$`,`$ag3$`),give(`$ag1$`,`$r1$`,`$ag2$`),give(`$ag3$`,`$r3$`,`$ag1$`)`$\}$. *Hence $ag3$ utters*

`propose(`$ag3$`, {`$ag1$`,`$ag2$`}, {give(`$ag2$`,`$r2$`,`$ag3$`),give(`$ag1$`,`$r1$`,`$ag2$`),give(`$ag3$`,`$r3$`,`$ag1$`)})`

t=12*: $ag3$'s proposal is acceptable for both $ag1$ and $ag2$ since the procedure* `acceptableExchange` *evalutes to true for each, and hence both $ag1$ and $ag2$ utter* `accept_proposal`.

t=13*: Applying* CR9, *$ag3$ receives the* `accept_proposal` *utterance of $ag1$.*

t=14*: Applying* CR9, *$ag3$ receives the* `accept_proposal` *utterance of $ag2$.*

t=15*: Applying* CR11, *$ag3$ utters* `confirm`.

t=16*: Applying* CR14 *(separately), $ag1$ and $ag2$ receive the* `confirm` *utterance of $ag3$. As a result, $ag3$ applying* CR11 *and agents $ag1$ and $ag2$ applying* CR13, *each agent revises its*

state according to the procedure `exchangeResources({give(`$ag2$`,`$r2$`,`$ag3$`),give(`$ag1$`,`$r1$`,`$ag2$`), give(`$ag3$`,`$r3$`,`$ag1$`)})` *and the r.r.p. of all agents is solved.*

7.6 Discussion

We have shown in this chapter how a general-purpose argumentation framework can be used to underpin an agent's dialogue policy, guiding its decision-making and thereby also its uterrances and supporting arguments. We have done this in this thesis for a particular negotiation problem (and such that agent utterances conform to the protocol set out in Chapter 5) but the principle, which we have demonstrated in this chapter, of using a general-purpose argumentation framework for concretely defining an agent's communication rules (policy) could be applied to other negotiation problems/protocols or, even, other types of dialogue besides negotiation (i.e. information-seeking, persuasion etc). In the context of resource reallocation as we consider in this thesis, we have presented in this chapter a fully worked out, generative policy for solving a (Grid computing, scheduling/timetabling etc) problem represented as an agent system in the format of Chapter 3. Our agents progress from an initial to an optimal allocation of resources by means of request-agree (if two agents involved; see Definition 5.2) and propose-accept-confirm (if three or more agents involved; see Definition 5.3) instances of dialogue which result in exchanges of resources. As part of the (dialogues) generated, our agents also disclose additional information as relevant and required in assisting recipient agents to find resources sufficient for fulfilling their goals or to become aware that a particular goal or plan is currently out of reach (not posssible). This additional disclosure of information in the negotiation method set out in this thesis takes the form of arguments justifying agent utterances as to why a particular resource cannot be given away (e.g. "I do not have it ...", "it has been given to another agent ...", "I need it ..."). The need does not rise in this thesis but in future work, in considering more general resource reallocation problems, it could be necessary at times, according to the situation, for agents to include arguments (justifications) with their requests/proposals also, to convince recipient agents of the acceptability of a request/proposal.

We compare in the remainder of this section the negotiation process/policy defined (completed) in this chapter and the scope of our work with similar recent work in the argumentation-based negotiation research community. Indeed there are a number of interest- (argument-) based multi-agent negotiation frameworks that are related to our framework. However, most are restricted to two agents (e.g. [RPSD07, RSD03, PHD+07, HDM10]), a restriction we do not impose, and most are protocols for the allowed types of negotiation (e.g. [RPSD07, vVP05, RSD03]) rather than (fully) generative models that discuss the agents' decision making models, as we have presented in this thesis. In addition, much in the framework of Rahwan et al. [RPSD07] rests on the setting of positive interaction between agents' shared or common goals such that an agent benefits from a common goal achieved by the other agent. The setting of Pasquier et al. [PHD+07], on the other hand, rests on there being no overlap in the agents' goals (and needed resources). Both of these differ from our setting wherein agents may have common goals which interact negatively such that agents 'compete' to fulfil their individual goals. Whilst the work of Hadidi et al. [HDM10] is more elaborate than others in that it presents a strategy for argumentative negotiation as well as a protocol, it differs from our focus in this thesis in that it is based on an abstract argumentation framework and not a concrete argumentation framework, which gives structure to the arguments such that arguments can be constructed (generated), as is the focus in this thesis.

Sierra et al. [SJNP97] present a negotiation language and protocol that allows for the exchange of complex proposals which can include compelling arguments for why a proposal should be adopted. Whilst [SJNP97] does not concentrate on the way in which arguments are built and analysed, the work is extended by Parsons et al. [PSJ98] by indicating how argumentation can be used to construct proposals, create critiques, provide explanations and meta-information. However, even in [PSJ98], further expansion is required for agents to be able to generate and rate arguments, and for any kind of implementation to be produced. In particular, the acceptability classes used in [PSJ98] to rank arguments are not sufficient to resolve inconsistencies that may arise within and between agents. A more fine-grained mechanism is required. We use an existing argumentation framework (AABA) for this purpose, that is able to build and determine the acceptability of arguments, even as the knowledge bases of agents change over time (as a result

of the dialogues). The AABA framework also allows agents to make assumptions, enabling agents to make decisions even despite incomplete information.

The ABN framework of Karunatillake et al. [KJRM09] is supplemented with a number of concrete negotiation strategies which allow agents to exchange arguments as part of the negotiation process. The example scenario/context considered allows for multiple (more than two) agents. However, in [KJRM09], the authors are mainly concerned with the systematic impact of ABN in an agent society, identifying how and when argumentation can be useful for agent societies. Hence the authors abstract away from the defeat-status computation of arguments. Instead, a simple validation heuristic that simulates a defeasible model is used for evaluating justifications, comparing arguments and, ultimately, deciding utterances. We use an argumentation framework to evaluate preconditions for utterances and to generate justifications/arguments for utterances. The argumentation framework we use allows an agent to make assumptions in the presence of incomplete information to resolve inconsistencies and avoid incorrect decisions. These assumptions are communicated and can be inspected by other agents. Also, in [KJRM09], deals involving more than two agents are not possible, as is required in our resource allocation setting in order to reach optimal allocations.

Multi-agent negotiation processes are also presented by Andersson and Sandholm [AS99] and Nongaillard et al. [NMJ08]. These negotiation processes considered in [AS99, NMJ08], however, are such that all agents engage in a single "contract type" only ('OCSM-contracts' are not considered, as well as 'M-contracts' in the case of [NMJ08]). The outcomes (solution quality in terms of social welfare) and the number of contracts tried/performed before a (local) optimum is reached are compared for societies of agents based on each of the different "contract types". The negotiation processes of [AS99, NMJ08] are exhaustive searches through the search space of all possible deals. It would be interesting to compare the efficiency of these approaches in reaching an optimum with our approach which allows agents to exchange beliefs/arguments to guide their search for goal-fulfilling resources.

7.7 Conclusion

We have instantiated in this chapter the agent policy described in Chapter 5 for solving the resource reallocation problem by means of agents and procedures grounded in AABA. AABA has allowed us (1) to realise the reasoning of agents, allowing each agent to deal with inconsistency and incompleteness in its beliefs, and in deciding what actions to undertake; and (2) to allow agents to exchange reasons (arguments) as part of their utterances (dialogue moves) which are understandable to one another and which can be used to revise their beliefs. In particular, we have defined the language which agents use internally for reasoning as well as externally for exchanging "arguments" between one another. The language includes "assumptions" which agents can use for reasoning and building arguments in the absence of contrary beliefs. Finally, we have defined the reasoning (inference) rules and discussed how these definitions together ensure certain consistency properties that are required of the agent reasonism mechanism. These fundamental components which make up the AABA framework for our negotiation agents are then used to ground the procedures of the negotiation policy, and we have shown that desired properties of the procedures persist.

Chapter 8

Conclusion

In this chapter we conclude the research described in this thesis, summarising the achievements of this thesis in Section 8.1, discussing possible directions for future work in Section 8.2 and ending with some final remarks in Section 8.3.

8.1 Summary of Thesis Achievements

In Chapter 2 we presented background on the resource reallocation problem, social welfare, agent communication (negotiation and argumentation-based negotiation dialogues in particular) and argumentation (assumption-based argumentation in particular). Following this, in Chapter 3, we set out the key components of our problem domain amounting to the notion of agent system, resource reallocation problem, and individual and social welfare.

In Chapter 4 we defined a class of deals which we termed "minimally cooperatively rational deals". We showed the sufficiency and completeness of these deals for solving our resource reallocation problem. We showed also that it is possible to characterise the actions that make up these deals in terms of the number of resources that agents involved receive and give away, and in terms of which agents change (increase) utility as a result of these deals.

In Chapter 5 we defined a communication language and protocol for negotiation that captured

and allowed for minimally cooperatively rational deals, such that more than two agents can be involved in a deal and the agents involved can give away more than one resource. The communication language and protocol defined allowed for agents to share arguments during negotiation. We defined also in that chapter a set of communication rules and associated procedures based on the communication language and protocol which made up a negotiation policy for generating dialogue moves (instances) that solve the resource reallocation problem of interest in this thesis.

In Chapter 6 we extended the Assumption-Based Argumentation (ABA) framework for multi-agent systems where agents have some common knowledge but may also have personal knowledge not necessarily shared by all agents. The framework defined was such that agents can build acceptable (admissible) arguments which are communicable and can be understood by other agents in the agent system. We termed our new framework Agent ABA (AABA).

In Chapter 7 we instantiated the negotiation policy of Chapter 5 by means of AABA. In particular, AABA realised the reasoning of agents, allowing each agent to deal with inconsistency and incompleteness in its beliefs, and in deciding what actions to undertake. Moreoever, AABA allowed a concrete means of representing and generating the arguments exchanged between agents as part of their dialogue moves during negotiation.

8.2 Future Work

We have made a number of simplifying assumptions. The negotiation policy is designed so that agents are truthful and only provide arguments that they hold as true. Moreover, the use of argumentation is limited to internal decision-making and reasoning, and for the exchange of arguments that are justifications for actions chosen: in general, it would be useful to allow agents to argue with one another, for example for persuasion or in case agents are found to be untruthful. It will also be interesting to investigate the case where agents do not share knowledge at the outset as to which resources fulfil which goals, and the role of agent dialogues and argumentation in resolving this incompleteness and inconsistency in beliefs for achieving

optimal allocations. Agents are assumed to have single goals: it would be interesting to extend this to allow agents to have multiple goals, and, possibly even, conflicting goals and preferences over goals. Also, we have assumed that single resources suffice for the achievement of goals: the next step on from this would be to consider the case where multiple resources may be required to fulfil a goal, and to investigate the class of deals that are required to solve the resource reallocation problem in such cases, as well as the definitions of the procedures required to capture the necessary deals. Lastly, resources are assumed to be single unit, indivisible and unsharable: this is often not the case in applications. Future work will look at dropping the above assumptions.

A further direction for future work will be to investigate the possibility and also the suitability of using other general-purpose argumentation frameworks for instantiating an agent's argumentation-based negotiation policy as opposed to Agent Assumption-Based Argumentation which we have used in this thesis. In addition to the above, we hope to define the operational semantics for the dialogue instances that our agens engage in, as has been done recently for some negotiation and argumentation dialogue protocols [Kar06, MVEPA03, MP05], and thus viewing the participating agents and their shared dialogues as part of a large abstract or virtual computer whose overall state may be altered by the utterance of valid locutions or by internal decision processes of the participants [MP09]. Lastly, we aim to translate our fully worked our generative argumentation-based negotiation policy into a fully-fledged implementation which can capture the different kinds of problems that we have set out to solve (as described in Chapter 1, Section 1.2). A possible approach, using existing agent programming tools, is to use Jade (Java Agent DEvelopment Framework) [BCG07] and SICStus Prolog [Proa], interfacing the two by means of PrologBeans [Prob]. Jade would be used for its naming, messaging and yellow pages services, as well as its capability to support the scheduling of cooperative 'behaviours' (where a 'behaviour' would represent a 'communcation rule' in our case). Prolog, on the other hand, would be used for the AABA representation of the agents (language, inference rules, assumptions, contrary relationships, backward deductions, dispute derivations). The implementation will be used additionally to test and validate findings in this thesis as well as to experiment and evaluate agent systems of varying sizes and complexity.

8.3 Final Remarks

We set out in this thesis to present a fully worked out, generative model for solving a particular class of resource reallocation problem (described in Section 3) by means of the multi-agent paradigm, but such that our model and its constituent parts would not be restricted to this class of problems and is instead a first step which can be built upon and applied more generally in future work. We set out to present our model in such a manner that the computation of optimal allocations of resources is arrived at by the participating agents themselves and, also, such that the disclosure of information (in the form of communicable arguments) between agents is kept minimal and relevant. The application domain of interest has been systems where agents could benefit, both individually and in a societal sense, from sharing (distributing) resources that would otherwise go un-utilised and thus making the best possible use of resources collectively despite their individualistic nature. Moreover, we have been interested in this thesis in application domains where it would be difficult or impossible to arrange a central, all-knowing, decision-making agent, citing a number of examples of real-life problems that are of interest to us in Chapter 1, Section 1.2.

Over the course of the thesis we have made individual contributions to three different subject areas — resource allocation (Chapter 4), argumentation-based negotiation (Chapter 5) and assumption-based argumentation (Chapter 6) — bridging the separate contributions together in Chapter 7 to present our generative, argumentation-based negotiation method for solving (optimising) resource allocation problems captured as agent systems in the format of Chapter 3. Although we have focused on defining a negotiation policy for a particular class of resource allocation problem, the tools we have set out can be applied more generally. Namely, the procedures used by an agent in evaluating a communication rule could be defined differently (by means of Agent Assumption-Based Argumentation as in this thesis or other general-purpose argumentation frameworks) or, even, the agent could have additional or different communication rules to those set out in this thesis, allowing it to engage in dialogue protocols different to those which we consider.

In closing, acknowledging the assumptions and limitations of the work as stated in this section

above and the previous section, we remind the reader of what we have achieved in relation to the state of the art in the research community, which has been to produce a representation format for resource reallocation problems in terms of multi-agent systems and an argumentation-based negotiation framework built (grounded) by means of a general-purpose argumentation framework. We use our general-purpose argumentation framework to underpin the agents' decision-making and for the automatic generation of dialogues by agents, allowing disclosure of relevant (useful) information, capturing deals involving multiple agents (and multiple resources), and leading to optimal allocations of resources for the problem domain focused on in this thesis.

Appendix A

Additional properties of the CR and Min CR deals of Chapter 4

We present in this appendix properties of cooperatively rational (CR) and minimally cooperatively rational (Min CR) deals additional to those given in Chapter 4.

A.1 Number of resources agents receive in Min CR deals

Below are the three corollaries that combine to prove Corollary 4.2 of Chapter 4.

Corollary A.1 *For any Min CR deal $\delta = (Res, Res')$ resulting in an agent system AS' such that $|Actions^{\delta}| = 1$, there are two agents involved (i.e. $|Agents^{\delta}| = 2$) — one agent $x \in Agents^{\delta}$ receives exactly one resource and the other agent $y \in Agents^{\delta}$ $(y \neq x)$ receives no resources.*

Proof. The proof of this is trivial since $Actions^{\delta} = \{\texttt{give}(y,r,x)\}$ for some $x, y \in names(AS)$ $(x \neq y)$, i.e. x receives exactly one resource r and y receives no resources. □

Corollary A.2 *For any Min CR deal $\delta = (Res, Res')$ resulting in an agent system AS' such that $|Actions^{\delta}| = 2$, one agent $z_1 \in Agents^{\delta}$ receives zero or one resources and all other agents $z \in Agents^{\delta}$ $(z \neq z_1)$ receive exactly one resource each.*

Proof. By Theorem 4.6 we know that $Actions^{\delta} = \{\texttt{give}(y,r,x), \texttt{give}(z_1,r_1,y)\}$ for some $x, y, z_1 \in names(AS)$ where $x \neq y$, $y \neq z_1$ and x and z_1 are possibly the same. Thus x and y receive exactly one resource each. The only other agent involved in the deal is z_1. In the case that $z_1 \neq x$, then z_1 receives no resources. Otherwise, if $z_1 = x$, then z_1 (i.e. x) receives exactly one resource. □

Corollary A.3 *For any Min CR deal $\delta = (Res, Res')$ resulting in an agent system AS' such that $|Actions^{\delta}| = n + 2$ ($n \geq 1$), one agent $z_{n+1} \in Agents^{\delta}$ receives zero or one resources and all other agents $z \in Agents^{\delta}$ ($z \neq z_{n+1}$) receive exactly one resource each.*

Proof. By Theorem 4.7 we know that $Actions^{\delta} = \{\texttt{give}(y,r,x), \texttt{give}(z_1,r_1,y), \ldots, \texttt{give}(z_{n+1},r_{n+1},z_n)\}$ for some $x, y, z_1, \ldots, z_{n+1} \in names(AS)$ where $x, y, z_1, \ldots, z_n$ are all different, $z_n \neq z_{n+1}$ and one of $x, y, z_1, \ldots, z_{n-1}$ is possibly the same as z_{n+1}. Thus $x, y, z_1, \ldots, z_n$ receive exactly one resource each. The only other agent involved in the deal is z_{n+1}. In the case that $z_{n+1} \notin \{x, y, z_1, \ldots, z_{n-1}\}$, then z_{n+1} receives no resources. Otherwise, if $z_{n+1} \in \{x, y, z_1, \ldots, z_{n-1}\}$, then z_{n+1} receives exactly one resource. □

A.2 Number of resources agents give away in Min CR deals

Below are the three corollaries that combine to prove Corollary 4.3 of Chapter 4.

Corollary A.4 *For any Min CR deal $\delta = (Res, Res')$ resulting in an agent system AS' such that $|Actions^{\delta}| = 1$, there are two agents involved (i.e. $|Agents^{\delta}| = 2$) — one agent $y \in Agents^{\delta}$ gives away exactly one resource and the other agent $x \in Agents^{\delta}$ ($x \neq y$) gives away no resources.*

Proof. Trivial since $Actions^{\delta} = \{\texttt{give}(y,r,x)\}$ for some $x, y \in names(AS)$ ($x \neq y$), i.e. y gives away exactly one resource r and x gives away no resources. □

Corollary A.5 *For any Min CR deal $\delta = (Res, Res')$ resulting in an agent system AS' such that $|Actions^\delta| = 2$, one agent $x \in Agents^\delta$ gives away zero or one resources and all other agents $z \in Agents^\delta$ $(z \neq x)$ give away exactly one resource each.*

Proof. By Theorem 4.6 we know that $Actions^\delta = \{\texttt{give}(y,r,x), \texttt{give}(z_1,r_1,y)\}$ for some $x, y, z_1 \in names(AS)$ where $x \neq y$, $y \neq z_1$ and x and z_1 are possibly the same. Thus y and z_1 give away exactly one resource each. The only other agent involved in the deal is x. In the case that $x \neq z_1$, then x gives away no resources. Otherwise, if $x = z_1$, then x (i.e. z_1) gives away exactly one resource. □

Corollary A.6 *For any Min CR deal $\delta = (Res, Res')$ resulting in an agent system AS' such that $|Actions^\delta| = n + 2$ $(n \geq 1)$, one agent $x \in Agents^\delta$ gives away zero or one resources, one agent $z_{n+1} \in Agents^\delta$ gives away one or two resources and all other agents $z \in Agents^\delta$ $(z \neq x$, $z \neq z_{n+1})$ give away exactly one resource each.*

Proof. By Theorem 4.7 we know that $Actions^\delta = \{\texttt{give}(y,r,x), \texttt{give}(z_1,r_1,y), \ldots, \texttt{give}(z_{n+1},r_{n+1},z_n)\}$ for some $x, y, z_1, \ldots, z_{n+1} \in names(AS)$, i.e. $Agents^\delta = \{x, y, z_1, \ldots, z_{n+1}\}$, where $x, y, z_1, \ldots, z_n$ are all different, $z_n \neq z_{n+1}$ and one of $x, y, z_1, \ldots, z_{n-1}$ is possibly the same as z_{n+1}. Thus, if $x \neq z_{n+1}$, then x gives away no resources. If $x = z_{n+1}$, then x gives away exactly one resource. If $z_{n+1} \in \{y, z_1, \ldots, z_{n-1}\}$, then z_{n+1} gives away exactly two resources. Lastly, since $y, z_1, \ldots, z_n$ are all distinct, each agent $z \in Agents^\delta$ $(z \neq x$, $z \neq z_{n+1})$ gives away exactly one resource. □

A.3 Changes in utility of agents in Min CR deals

Below are the three corollaries that combine to prove Corollary 4.4 of Chapter 4.

Corollary A.7 *For any Min CR deal $\delta = (Res, Res')$ resulting in an agent system AS' such that $|Actions^\delta| = 1$, only one agent increases in utility and none of the agents decrease in utility, i.e. $sw_{AS}(AS') = sw_{AS}(AS) + 1$.*

Proof. The fact that none of the agents decrease in utility is trivial since δ is a CR deal. It remains to show that only one agent increases in utility. Since $|Actions^{\delta}| = 1$, then $Actions^{\delta}$ = {`give(`y`,`r`,`x`)`} for some $x, y \in names(AS)$ and $r \in \mathcal{R}$. We know, by Theorem 4.5, that $x \notin happy_{AS}(AS)$ and $x \in happy_{AS}(AS')$. It suffices to show that y does not change (increase or decrease) in utility. This is the case since, by Theorem 4.5, $y \in happy_{AS}(AS')$ iff $y \in happy_{AS}(AS)$. □

Corollary A.8 *For any Min CR deal $\delta = (Res, Res')$ resulting in an agent system AS' such that $|Actions^{\delta}| = 2$, only one agent increases in utility and none of the agents decrease in utility, i.e. $sw_{AS}(AS') = sw_{AS}(AS) + 1$.*

Proof. The fact that none of the agents decrease in utility is trivial since δ is a CR deal. It remains to show that only one agent increases in utility. Since $|Actions^{\delta}| = 2$, then $Actions^{\delta}$ = {`give(`y`,`r`,`x`)`, `give(`z_1`,`r_1`,`y`)`} for some $x, y, z_1 \in names(AS)$ and $r, r_1 \in \mathcal{R}$. We know, by Theorem 4.6, that $x \notin happy_{AS}(AS)$, $x \in happy_{AS}(AS')$, $y \in happy_{AS}(AS)$ and $y \in happy_{AS}(AS')$. It suffices to show that z_1 does not change (increase or decrease) in utility except in the case where $z_1 = x$ (and x increases in utility). Indeed this is the case since, by Theorem 4.6, when $z_1 \neq x$ we have $z_1 \in happy_{AS}(AS')$ iff $z_1 \in happy_{AS}(AS)$. □

Corollary A.9 *For any Min CR deal $\delta = (Res, Res')$ resulting in an agent system AS' such that $|Actions^{\delta}| = n + 2$ ($n \geq 1$), only one agent increases in utility and none of the agents decrease in utility, i.e. $sw_{AS}(AS') = sw_{AS}(AS) + 1$.*

Proof. The fact that none of the agents decrease in utility is trivial since δ is a CR deal. It remains to show that only one agent increases in utility. Since $|Actions^{\delta}| = n + 2$ ($n \geq 1$), then, by Theorem 4.7, $Actions^{\delta}$ = {`give(`y`,`r`,`x`)`, `give(`z_1`,`r_1`,`y`)`, ..., `give(`z_{n+1}`,`r_{n+1}`,`z_n`)`} for some $x, y, z_1, \ldots, z_{n+1} \in names(AS)$ ($x, y, z_1, \ldots, z_n$ all different, $z_n \neq z_{n+1}$ and one of $x, y, z_1, \ldots, z_{n-1}$ possibly the same as z_{n+1}), where $x \notin happy_{AS}(AS)$, $x \in happy_{AS}(AS')$, $\{y, z_1, \ldots, z_n\} \subset happy_{AS}(AS)$ and $\{y, z_1, \ldots, z_n\} \subset happy_{AS}(AS')$. It suffices to show that z_{n+1} does not change (increase or decrease) in utility except in the case where $z_{n+1} = x$ (and

x increases in utility). Indeed this is the case since, by Theorem 4.7, when $z_{n+1} \neq x$ we have $z_{n+1} \in happy_{AS}(AS')$ iff $z_{n+1} \in happy_{AS}(AS)$. □

Appendix B

Example Definition of the Control Procedures of Chapter 5

We give in this section an example definition of the elements needed to ensure that dialogue instances, as defined in Chapter 5, are appropriately sequenced and interleaved.

Let $D(x)$, the dialogical status of the agent, be a consistent set of ground literals of the following form (for $X, Y \in names(AS)$, $Z \in 2^{names(AS)}$, $R \in \mathcal{R}$, `Response` $\in$ {`acceptProposal`, `rejectProposal`}):

- `initiatedDialogue(`X`,`Z`,Actions)`, representing that X has initiated a request or proposal dialogue instance directed at a set of agents Z regarding `Actions` and has not yet received responses from all agents Z;
- `acceptedProposal(`X`,`Y`,Actions)`, representing that X has accepted a proposal sent to it from Y regarding `Actions` and is awaiting confirmation;
- `receivedResponse(`X`,`Y`,Actions,Response)`: representing that X has received response (`Response`) from Y following a proposal dialogue move it sent regarding `Actions`.

Furthermore, we assume that the agent is able to make the following inferences:

- $\vdash^x$ [R does not solve the r.r.p. of x] iff `fulfils(`R`,`$G(x)$`)` $\notin Pl_{AS}(x)$.

- $\vdash^x$ [x is not awaiting a response from any agent] iff `initiatedDialogue(`x`,_,Actions)` $\notin D(x)$, and for all `acceptedProposal(`x`,_,Actions)` $\in D(x)$ such that `give(_,`R`,`x`)` $\in$ `Actions`, it is the case that $\vdash^x$ [R does not solve the r.r.p. of x].

- $\vdash^x$ [x has offered R to some agent] iff either of the following two conditions hold:

  - `initiatedDialogue(`x`,_,Actions)` $\in D(x)$ and `give(`x`,`R`,_ )` $\in$ `Actions`; or

  - `acceptedProposal(`x`,_,Actions)` $\in D(x)$ and `give(`x`,`R`,_ )` $\in$ `Actions`.

- $\vdash^x$ [x has not presently offered R to any agent] iff $\nvdash^x$ [x has offered R to some agent].

The first inference above represents that a resource R does not solve the r.r.p. of x if x does not know of a plan such that R fulfils its goal $G(x)$. The second inference above represents that x is not awaiting a response from any agent for a resource that could solve its r.r.p. if it does not "remember" initiating a (request or proposal) dialogue instance or accepting a proposal in which it is to receive a resource that solves its r.r.p. The third inference above represents that x has offered to give away a resource R if it "remembers" initiating a dialogue instance or accepting a proposal as such. The fourth inference above holds whenever the third does not.

We now provide definitions of the "control procedures" occurring in the communication rules of our negotiation policy for an agent x (see Chapter 5).

The `valid(initiateDialogue(Actions))` procedure, which occurs in the *preconditions* of communication rules *CR1* and *CR2*, is defined below to return successfully if the agent is deemed to be in a state where it is not currently *awaiting a response* from any agent, and it has not currently *offered* away any resources which it would also be offering away in `Actions`:

Definition B.1 (`valid(initiateDialogue(Actions))`) *The procedure evaluates to true if the following conditions hold:*

- $\vdash^x$ *[x is not awaiting a response from any agent]; and*

- *for all* `give(`x`,`R`,_ ) ∈ Actions`*, it is the case that* $\vdash^x$ *[x has not presently offered R to any agent].*

The procedure evaluates to false otherwise.

The `valid(accept(Actions))` procedure, which occurs in the *preconditions* of communication rules *CR3* and *CR5*, is defined below to return successfully if the agent has not presently offered to give away any of its resources which solve its r.r.p., nor any resources which it would be required to give away in `Actions`:

Definition B.2 (`valid(accept(Actions))`) *The procedure evaluates to true if the following conditions hold:*

- *for all resources R for which the condition* $\vdash^x$ *[x has offered R to some agent] holds, it is the case that* $\vdash^x$ *[R does not solve the r.r.p. of x]; and*
- *for all* `give(`x`,`R`,_ ) ∈ Actions`*, it is the case that* $\vdash^x$ *[x has not presently offered R to any agent].*

The procedure evaluates to false otherwise.

The reason for the first condition in the definition above being independent of `Actions` is to avoid the situation arising whereby the agent has offered to give away a resource which solves its r.r.p. as part of another dialogue instance, and then it offers to give another one away here also (as part of `Actions`), possibly offering to give away the only two resources it has that solve its r.r.p.

The `valid(receiveResponse(`Y`,Actions))` procedure, which occurs in the *preconditions* of communication rules *CR7–CR10*, is defined to check that the agent is indeed awaiting a response from Y regarding `Actions`, as follows:

Definition B.3 (`valid(receiveResponse(`Y`,Actions)))` *The procedure evaluates to true if the condition* `initiatedDialogue(`x`,`Z`,Actions)` $\in D(x)$ *holds, and* $Y \in Z$*. The procedure evaluates to false otherwise.*

Note that, according to Definition B.1, the agent only initiates one dialogue instance at a time. Therefore, it has at most one `initiatedDialogue(`x`,_,_ )` literal in $D(x)$ at any given time and, in receiving a response from an agent Y according to the procedure above, confusion would not arise as to which proposal it is a response to.

The `valid(confirm(`Z`,Actions))` procedure, which occurs in the *preconditions* of communication rule *CR11*, returns successfully if the agent has initiated a proposal directed at a set of (responding) agents Z and all agents in Z have accepted the proposal, as follows:

Definition B.4 (`valid(confirm(`Z`,Actions)))` *The procedure evaluates to true if* `initiatedDialogue(`x`,`Z`,Actions)` $\in D(x)$*, and for all* $Y \in Z$*, it is the case that* `receivedResponse(`x`,`Y`,Actions,acceptProposal)` $\in D(x)$*. The procedure evaluates to false otherwise.*

The `valid(disconfirm(`Z`,Actions))` procedure, which occurs in the *preconditions* of communication rules *CR12* and *CR13*, returns successfully if the agent has initiated a proposal directed at a set of (responding) agents Z', all agents Z' have responded but only a subset $Z \subset Z'$ have accepted the proposal, as follows:

Definition B.5 (`valid(disconfirm(`Z`,Actions)))` *The procedure evaluates to true if the following conditions hold:*

- `initiatedDialogue(`x`,`Z'`,Actions)` $\in D(x)$*; and*
- *for all* $Y \in Z'$*,* `receivedResponse(`x`,`Y`,Actions, _ )`*; and*
- $Z = \{\ Y \mid$ `receivedResponse(`x`,`Y`,Actions,acceptProposal)` $\in D(x)\ \} \subset Z'$.

The procedure evaluates to false otherwise.

The `valid(receiveConfirm(`Y`,Actions))` procedure, which occurs in the *preconditions* of communication rules *CR14* and *CR15*, returns successfully if the agent can receive a confirm or disconfirm dialogue move following an earlier `accept_proposal` dialogue move it itself made regarding `Actions`, as follows:

Definition B.6 (`valid(receiveConfirm(`Y`,Actions))`) *The procedure evaluates to true if the condition* `acceptedProposal(`x`,`Y`,Actions)` $\in D(x)$ *holds, and evaluates to false otherwise.*

The `do(initiateDialogue(`Z`,Actions))` procedure, which occurs in the *revisions* of communication rules *CR1* and *CR2*, updates the agent's state so that it knows it has initiated a dialogue instance with a set of responding agents Z regarding `Actions`, as follows:

Definition B.7 (`do(initiateDialogue(`Z`,Actions))`) x *adds* `initiatedDialogue(`x`,`Z`, Actions)` *to* $D(x)$ *when the procedure is called.*

The `do(accept_proposal(`Y`,Actions))` procedure, which occurs in the *revisions* of communication rule *CR5*, updates the agent's state so that it knows it has accepted a proposal from Y regarding `Actions`, as follows:

Definition B.8 (`do(accept_proposal(`Y`,Actions))`) x *adds* `acceptedProposal(`x`,`Y`, Actions)` *to* $D(x)$ *when the procedure is called.*

The `do(receiveResponse(`Y`,Actions,`*Response*`))` procedure, which occurs in the *revisions* of communication rules *CR7–CR10*, updates the agent's state so that it knows it has received a response from Y regarding `Actions` following an earlier request or proposal it sent, as follows:

Definition B.9 (`do(receiveResponse(`Y`,Actions,Response))`) x *updates its state according to the value of* `Response` *when the procedure is called, as follows:*

- *if* `Response` $\in$ {`agree`, `refuse`}, *then* `initiatedDialogue(`x`,{`Y`},Actions)` *is removed from* $D(x)$*;*
- *otherwise, if* `Response` $\in$ {`acceptProposal`, `rejectProposal`}, *then* `receivedResponse(`x`,` Y`,Actions,Response)` *is added to* $D(x)$.

The `do(confirm(Actions))` procedure, which occurs in the *revisions* of communication rules *CR11–CR13*, updates the agent's state so that it knows it has ended a proposal dialogue instance which it initiated, as follows:

Definition B.10 (`do(confirm(Actions))` x *removes* `initiatedDialogue(`x`,`Z`,Actions)` *from* $D(x)$ *when the procedure is called, and for all* $Y \in Z$*, removes* `receivedResponse(`x`,`Y`, Actions,_ )` *from* $D(x)$.

The `do(receiveConfirm(`Y`,Actions))` procedure, which occurs in the *revisions* of communication rules *CR14* and *CR15*, updates the agent's state so that it knows it has received a confirm or disconfirm dialogue move from Y regarding `Actions`, and thus the proposal dialogue instance has ended, as follows:

Definition B.11 (`do(receiveConfirm(`Y`,Actions))` x *removes* `acceptedProposal(`x`,`Y`, Actions)` *from* $D(x)$ *when the procedure is called.*

Note that in completing a dialogue instance and executing one of the procedures of Definitions B.9–B.11 the "flags" in $D(x)$ alluding to the agent's involvement in the dialogue instance are removed, as are the dialogue moves received in $M(x)$ as they are "processed" according to the communication rules. As such, no "record" of the dialogue instance is kept in the agent's state and the agent could well initiate the same initial dialogue move at some point in the future. We allow for this and whether or not the same initial dialogue move is made at some point in the future depends on the agent's beliefs and the "general procedures" (see Section 5.4).

Appendix C

Conditions for Inferences of Chapter 5

We present in this appendix conditions under which the sentences in $\mathcal{ML}$ (see Definition 5.11) could be inferred from an agent x's state, as follows:

- $\vdash^x$ [Y has R] if either
  - $Y = x$ and $R \in Res(x)$; or
  - $Y \neq x$ and $\texttt{has}(Y,R) \in B(x)$.
- $\vdash^x \neg$[Y has R] if either
  - $Y = x$ and $R \notin Res(x)$; or
  - $Y \neq x$ and $\texttt{has}(Y,R) \notin B(x)$.
- $\vdash^x$ [the r.r.p. of Y is solved with R] if either
  - $Y = x$ and $\texttt{fulfils}(R,G(x)) \in B(x)$; or
  - $Y \neq x$ and $\texttt{goal}(Y,G)$, $\texttt{fulfils}(R,G) \in B(x)$.
- $\vdash^x \neg$[the r.r.p. of Y is solved with R] if either
  - $Y = x$ and $\texttt{fulfils}(R,G(x)) \notin B(x)$; or
  - $Y \neq x$ and it is not the case that $\texttt{goal}(Y,G)$, $\texttt{fulfils}(R,G) \in B(x)$ for any $G \in \mathcal{G}$.

- $\vdash^x$ [the r.r.p. of Y is solved] if either
  - $Y = x$, $\texttt{fulfils(}R\texttt{,}G(x)\texttt{)} \in B(x)$ and $R \in Res(x)$ for some $R \in \mathcal{R}$; or
  - $Y \neq x$ and $\texttt{goal(}Y\texttt{,}G\texttt{)}$, $\texttt{fulfils(}R\texttt{,}G\texttt{)}$, $\texttt{has(}Y\texttt{,}R\texttt{)} \in B(x)$ for some $R \in \mathcal{R}$.
- $\vdash^x \neg$[the r.r.p. of Y is solved] if either
  - $Y = x$, and it is not the case that $\texttt{fulfils(}R\texttt{,}G(x)\texttt{)} \in B(x)$ and $R \in Res(x)$ for any $R \in \mathcal{R}$; or
  - $Y \neq x$, and it is not the case that $\texttt{goal(}Y\texttt{,}G\texttt{)}$, $\texttt{fulfils(}R\texttt{,}G\texttt{)}$, $\texttt{has(}Y\texttt{,}R\texttt{)} \in B(x)$ for any $R \in \mathcal{R}$.
- $\vdash^x$ [the r.r.p. of Y would *still* be solved or unsolved by simply giving away R] if either
  - $\vdash^x \neg$[the r.r.p. of Y is solved with R]; or
  - for some $R' \in \mathcal{R}$ where $R' \neq R$, it is the case that $\vdash^x$ [Y has R'] and $\vdash^x$ [the r.r.p. of Y is solved with R'].
- $\vdash^x \neg$[the r.r.p. of Y would *still* be solved or unsolved by simply giving away R] if
  - $\vdash^x$ [Y has R];
  - $\vdash^x$ [the r.r.p. of Y is solved with R]; and
  - there is no $R' \in \mathcal{R}$ such that $R' \neq R$, $\vdash^x$ [Y has R'] and $\vdash^x$ [the r.r.p. of Y is solved with R'].
- $\vdash^x$ [after giving away each resource in Rs, the r.r.p. of x would *no longer* be solved] if
  - for some $R \in Rs$, $\vdash^x$ [Y has R], $\vdash^x$ [the r.r.p. of Y is solved with R]; and
  - there is no $R' \in \mathcal{R}$ such that $R' \notin Rs$, $\vdash^x$ [Y has R'] and $\vdash^x$ [the r.r.p. of Y is solved with R'].
- $\vdash^x \neg$[after giving away each resource in Rs, the r.r.p. of Y would *no longer* be solved] if either
  - for each $R \in Rs$, $\vdash^x \neg$[the r.r.p. of Y is solved with R]; or

– there is some $R' \in \mathcal{R}$ such that $R' \notin Rs$, $\vdash^x$ [Y has R'] and $\vdash^x$ [the r.r.p. of Y is solved with R'].

We re-formulate the meta-language $\mathcal{ML}$ and inferences above for a general-purpose argumentation framework in Chapter 7, taking into consideration in particular "assumptions" (see Definition 5.10) for agents to make inferences in the absence of knowledge (i.e. given the incompleteness of their beliefs).

Appendix D

Proofs of Lemmas and Theorems pertaining to AABA (in Chapter 6)

We present in this appendix the proofs omitted in Chapter 6 pertaining to the backward deductions (Lemmas 6.1 and 6.2) and fact-inclusive AB-dispute derivations (Theorems 6.1 and 6.2) of AABA.

D.1 Proofs pertaining to backward deductions of AABA

D.1.1 Proof of Lemma 6.1

Let $< C_0, S_0 >, \ldots, < C_i, S_i >, \ldots, < C_n, S_n >$ be a backward deduction $S \vdash^{f}_{\mathcal{AF}_{Agent}} p$ where $C_0 = \{p\}$, $S_0 = C_n = \{\}$ and $S_n = S$. We show, inductively, that for each such backward deduction there exists a sequence of trees $\mathcal{T}_0, \ldots, \mathcal{T}_i, \ldots, \mathcal{T}_n$ and at each (ith) step of the backward deduction, $root(\mathcal{T}_i) = p$ and $leaves(\mathcal{T}_i) = C_i \cup S_i$.

For a given backward deduction, construct a sequence of trees as follows:

- for the initial step, $\mathcal{T}_0$ is a tree consisting of a single node (a leaf) labelled by p. Hence $root(\mathcal{T}_0) = p$ and $leaves(\mathcal{T}_0) = \{p\} = C_0 \cup S_0$.

- assume that $< C_0, S_0 >, \ldots, < C_i, S_i >$ $(i < n)$ is a sub-sequence of the backward deduction with corresponding sequence of trees $\mathcal{T}_0, \ldots, \mathcal{T}_i$ constructed such that $root(\mathcal{T}_i) = p$ and $leaves(\mathcal{T}_i) = C_i \cup S_i$. Then,

  - if $f(C_i) = \sigma$ and $\sigma \in \mathcal{R}^p \cup \mathcal{A}$, then by applying case 1 of Definition 6.4, $C_{i+1} = C_i - \{\sigma\}$, $S_{i+1} = S_i \cup \{\sigma\}$ and hence $C_{i+1} \cup S_{i+1} = C_i \cup S_i$. Construct $\mathcal{T}_{i+1} = \mathcal{T}_i$ and, trivially, $leaves(\mathcal{T}_{i+1}) = C_{i+1} \cup S_{i+1}$.

  - if $f(C_i) = \sigma$ and $\sigma \leftarrow b_1, \ldots b_m \in \mathcal{R}_c$ $(m = 0)$ is chosen in applying case 2 of Definition 6.4, then $C_{i+1} = C_i - \{\sigma\}$, $S_{i+1} = S_i$ and hence $C_{i+1} \cup S_{i+1} = (C_i \cup S_i) - \{\sigma\}$. Construct $\mathcal{T}_{i+1}$ from $\mathcal{T}_i$ by adding a child labelled by τ to each leaf node in $\mathcal{T}_i$ that is labelled by σ. Trivially, $leaves(\mathcal{T}_{i+1}) = leaves(\mathcal{T}_i) - \{\sigma\}$. Since $leaves(\mathcal{T}_i) - \{\sigma\} = (C_i \cup S_i) - \{\sigma\}$ by hypothesis, $leaves(\mathcal{T}_{i+1}) = C_{i+1} \cup S_{i+1}$.

  - if $f(C_i) = \sigma$ and $\sigma \leftarrow b_1, \ldots b_m \in \mathcal{R}_c$ $(m > 0)$ is chosen in applying case 2 of Definition 6.4, then $C_{i+1} = C_i - \{\sigma\} \cup \{b_1, \ldots, b_m\}$, $S_{i+1} = S_i$ and hence $C_{i+1} \cup S_{i+1} = (C_i \cup S_i) - \{\sigma\} \cup \{b_1, \ldots, b_m\}$. Construct $\mathcal{T}_{i+1}$ from $\mathcal{T}_i$ by adding m children, labelled by $b_1, \ldots b_m$ (respectively), to each leaf node in $\mathcal{T}_i$ that is labelled by σ. Trivially, $leaves(\mathcal{T}_{i+1}) = leaves(\mathcal{T}_i) - \{\sigma\} \cup \{b_1, \ldots, b_m\}$. Since $leaves(\mathcal{T}_i) - \{\sigma\} \cup \{b_1, \ldots, b_m\} = (C_i \cup S_i) - \{\sigma\} \cup \{b_1, \ldots, b_m\}$ by hypothesis, $leaves(\mathcal{T}_{i+1}) = C_{i+1} \cup S_{i+1}$.

- Then, by construction, $root(\mathcal{T}_n) = p$ and $leaves(\mathcal{T}_n) = C_n \cup S_n$. Since $C_n = \{\}$ and $S_n = S$, we have $leaves(\mathcal{T}_n) = S$, as required. □

D.1.2 Proof of Lemma 6.2

Let $\mathcal{T}$ be a tree corresponding to the f-argument for p supported by S (i.e. $root(\mathcal{T}) = p$ and $leaves(\mathcal{T}) = S$). We contruct a finite sequence of trees $\mathcal{T}_0, \ldots, \mathcal{T}_i, \ldots, \mathcal{T}_n$ $(0 \leq i \leq n)$ where $root(\mathcal{T}_i) = p$ (for $0 \leq i \leq n$), $leaves(\mathcal{T}_i)/(\mathcal{R}^p \cup \mathcal{A}) \neq \emptyset$ (for $0 \leq i < n$), $leaves(\mathcal{T}_0) = \{p\}$ and $\mathcal{T}_n = \mathcal{T}$. We show, inductively, that for each $\mathcal{T}_i$ $(0 \leq i \leq n)$ there exists a *partial deduction* $< C_0, S_0 >, \ldots, < C_{j_i}, S_{j_i} >$ (see Definition 6.7) such that $C_{j_i} = leaves(\mathcal{T}_i)/(\mathcal{R}^p \cup \mathcal{A})$ and $S_{j_i} = leaves(\mathcal{T}_i) \cap (\mathcal{R}^p \cup \mathcal{A})$. We construct the sequence of trees as follows:

- for the initial step, if $p \notin (\mathcal{R}^p \cup \mathcal{A})$, then, trivially, $C_0 = leaves(\mathcal{T}_0)/(\mathcal{R}^p \cup \mathcal{A})$ and $S_0 = leaves(\mathcal{T}_0) \cap (\mathcal{R}^p \cup \mathcal{A})$. Otherwise, if $p \in \mathcal{R}^p \cup \mathcal{A}$, then, by applying case 1 of Definition 6.4, $C_1 = C_0 - \{p\} = \{\} = leaves(\mathcal{T}_0)/(\mathcal{R}^p \cup \mathcal{A})$ and $S_1 = S_0 \cup \{p\} = \{p\} = leaves(\mathcal{T}_0) \cap (\mathcal{R}^p \cup \mathcal{A})$.

- assume given $\mathcal{T}_i$ $(0 \leq i < n)$ such that there exists a partial deduction $< C_0, S_0 >, \ldots, < C_{j_i}, S_{j_i} >$ and $C_{j_i} = leaves(\mathcal{T}_i)/(\mathcal{R}^p \cup \mathcal{A})$ and $S_{j_i} = leaves(\mathcal{T}_i) \cap (\mathcal{R}^p \cup \mathcal{A})$. Then,

  - Let $f(C_{j_i}) = \sigma$ ($\sigma \notin \mathcal{R}^p \cap \mathcal{A}$ by hypothesis). If $\mathcal{T}_{i+1}$ is constructed from $\mathcal{T}_i$ by choosing inference rule $\sigma \leftarrow b_1, \ldots, b_m \in \mathcal{R}_c$ $(m = 0)$ and adding a single child labelled by τ to each leaf node in $\mathcal{T}_i$ labelled by σ, then $leaves(\mathcal{T}_{i+1}) = leaves(\mathcal{T}_i) - \{\sigma\}$. Choose $\sigma \leftarrow b_1, \ldots, b_m \in \mathcal{R}_c$ in applying case 2 of Definition 6.4. Then, $C_{j_i+1} = C_{j_i} - \{\sigma\}$ and $S_{j_i+1} = S_{j_i}$. Since $C_{j_i} = leaves(\mathcal{T}_i)/(\mathcal{R}^p \cup \mathcal{A})$ by hypothesis, trivially, we have

    $$C_{j_i} - \{\sigma\} = leaves(\mathcal{T}_i)/(\mathcal{R}^p \cup \mathcal{A}) - \{\sigma\}$$

    which, since $\sigma \notin \mathcal{R}^p \cup \mathcal{A}$, is equivalent to

    $$C_{j_i} - \{\sigma\} = (leaves(\mathcal{T}_i) - \{\sigma\})/(\mathcal{R}^p \cup \mathcal{A})$$

    which is equivalent to

    $$C_{j_i+1} = leaves(\mathcal{T}_{i+1})/(\mathcal{R}^p \cup \mathcal{A}).$$

    Also, since $\sigma \notin \mathcal{R}^p \cup \mathcal{A}$ and $S_{j_i} = leaves(\mathcal{T}_i) \cap (\mathcal{R}^p \cup \mathcal{A})$ by hypothesis, we have $S_{j_i} = (leaves(\mathcal{T}_i) - \{\sigma\}) \cap (\mathcal{R}^p \cup \mathcal{A})$, which is equivalent to $S_{j_i+1} = leaves(\mathcal{T}_{i+1}) \cap (\mathcal{R}^p \cup \mathcal{A})$. Thus for $\mathcal{T}_{i+1}$ there exists a partial deduction $< C_0, S_0 >, \ldots, < C_{j_i}, S_{j_i} >, < C_{j_i+1}, S_{j_i+1} >$ such that $C_{j_i+1} = leaves(\mathcal{T}_{i+1})/(\mathcal{R}^p \cup \mathcal{A})$ and $S_{j_i+1} = leaves(\mathcal{T}_{i+1}) \cap (\mathcal{R}^p \cup \mathcal{A})$.

  - Let $f(C_{j_i}) = \sigma$ ($\sigma \notin \mathcal{R}^p \cap \mathcal{A}$ by hypothesis). If $\mathcal{T}_{i+1}$ is constructed from $\mathcal{T}_i$ by choosing inference rule $\sigma \leftarrow b_1, \ldots, b_m \in \mathcal{R}_c$ $(m > 0)$ and adding m children labelled by $b_1, \ldots, b_m$ (respectively) to each leaf node in $\mathcal{T}_i$ labelled by σ, then $leaves(\mathcal{T}_{i+1}) = leaves(\mathcal{T}_i) - \{\sigma\} \cup \{b_1, \ldots, b_m\}$. Construct $< C_{j_i+1}, S_{j_i+1} >$ by choosing $\sigma \leftarrow b_1, \ldots, b_m \in \mathcal{R}_c$ and applying case 2 of Definition 6.4. Subsequently, construct sequence $< C_{j_i+1}, S_{j_i+1} >, \ldots, < C_k, S_k >$ by applying case 1 of Definition 6.4 for each $b_h \in \{b_1, \ldots, b_m\} \cap (\mathcal{R}^p \cup \mathcal{A})$. Then,

* $C_k = C_{j_i} - \{\sigma\} \cup (\{b_1, \ldots, b_m\}/(\mathcal{R}^p \cup \mathcal{A}))$
* $S_k = S_{j_i} \cup (\{b_1, \ldots, b_m\} \cap (\mathcal{R}^p \cup \mathcal{A}))$

Since $C_{j_i} = leaves(\mathcal{T}_i)/(\mathcal{R}^p \cup \mathcal{A})$ by hypothesis, trivially, we have

$$C_{j_i} - \{\sigma\} \cup (\{b_1, \ldots, b_m\}/(\mathcal{R}^p \cup \mathcal{A})) = leaves(\mathcal{T}_i)/(\mathcal{R}^p \cup \mathcal{A}) - \{\sigma\} \cup (\{b_1, \ldots, b_m\}/(\mathcal{R}^p \cup \mathcal{A}))$$

which, since $\sigma \notin (\mathcal{R}^p \cup \mathcal{A})$, is equivalent to

$$C_{j_i} - \{\sigma\} \cup (\{b_1, \ldots, b_m\}/(\mathcal{R}^p \cup \mathcal{A})) = (leaves(\mathcal{T}_i) - \{\sigma\} \cup (\{b_1, \ldots, b_m\}/(\mathcal{R}^p \cup \mathcal{A})))/(\mathcal{R}^p \cup \mathcal{A})$$

which is equivalent to

$$C_k = leaves(\mathcal{T}_{i+1})/(\mathcal{R}^p \cup \mathcal{A}).$$

Also, since $S_{j_i} = leaves(\mathcal{T}_i) \cap (\mathcal{R}^p \cup \mathcal{A})$ by hypothesis, trivially, we have

$$S_{j_i} \cup (\{b_1, \ldots, b_m\} \cap (\mathcal{R}^p \cup \mathcal{A})) = (leaves(\mathcal{T}_i) \cap (\mathcal{R}^p \cup \mathcal{A})) \cup (\{b_1, \ldots, b_m\} \cap (\mathcal{R}^p \cup \mathcal{A})) = (leaves(\mathcal{T}_i) \cup \{b_1, \ldots, b_m\}) \cap (\mathcal{R}^p \cup \mathcal{A})$$

which, since $\sigma \notin (\mathcal{R}^p \cup \mathcal{A})$, is equivalent to

$$S_{j_i} \cup (\{b_1, \ldots, b_m\} \cap (\mathcal{R}^p \cup \mathcal{A})) = (leaves(\mathcal{T}_i) - \{\sigma\} \cup \{b_1, \ldots, b_m\}) \cap (\mathcal{R}^p \cup \mathcal{A})$$

which is equivalent to

$$S_k = leaves(\mathcal{T}_{i+1}) \cap (\mathcal{R}^p \cup \mathcal{A}).$$

Thus for $\mathcal{T}_{i+1}$ there exists a partial deduction $< C_0, S_0 >, \ldots, < C_{j_i}, S_{j_i} >, \ldots, < C_k, S_k >$ such that $C_k = leaves(\mathcal{T}_{i+1})/(\mathcal{R}^p \cup \mathcal{A})$ and $S_k = leaves(\mathcal{T}_{i+1}) \cap (\mathcal{R}^p \cup \mathcal{A})$.

- Then, by construction, for $\mathcal{T}_n$, there exists a partial deduction $< C_0, S_0 >, \ldots, < C_{j_n}, S_{j_n} >$ such that $C_{j_n} = leaves(\mathcal{T}_n)/(\mathcal{R}^p \cup \mathcal{A})$ and $S_{j_n} = leaves(\mathcal{T}_n) \cap (\mathcal{R}^p \cup \mathcal{A})$. Since $\mathcal{T}_n = \mathcal{T}$, $leaves(\mathcal{T}) = S$ and $S \subseteq \mathcal{R}^p \cup \mathcal{A}$, then $C_{j_n} = \{\}$, $S_{j_n} = S$ and hence $< C_0, S_0 >, \ldots, < C_{j_n}, S_{j_n} >$ is a backward deduction for p supported by S. □

D.2 Proofs pertaining to fact-inclusive AB-dispute derivations of AABA

Given a tuple $\langle \mathcal{P}_i, \mathcal{O}_i, \mathcal{D}_i, \mathcal{C}_i \rangle$ of a fact-inclusive AB-dispute derivation (*fdd*) (see Definition 6.5) and a tuple $\langle \mathcal{P}'_j, \mathcal{O}'_j, A'_j, \mathcal{C}'_j \rangle$ of a AB-dispute derivation (*dd*) (see Definition 2.10)[1], we write

$$\langle \mathcal{P}_i, \mathcal{O}_i, \mathcal{D}_i, \mathcal{C}_i \rangle \approx \langle \mathcal{P}'_j, \mathcal{O}'_j, A'_j, \mathcal{C}'_j \rangle$$

in the proofs of Theorems 6.1 and 6.2 below if $\mathcal{P}_i = \mathcal{P}'_j$, $\mathcal{O}_i = \mathcal{O}'_j$, $\mathcal{D}_i \cap \mathcal{A} = A'_j$ and $\mathcal{C}_i = \mathcal{C}'_j$.

For the proofs of Theorems 6.1 and 6.2 below, we need to use the fact that the existence of a *dd* is independent of the selection function used during the derivation as noted in [DKT06]. If there exists a derivation wrt some choices, then there exists a derivation with respect to any other choices.

D.2.1 Proof of Theorem 6.1

Let $\langle \mathcal{P}_0, \mathcal{O}_0, \mathcal{D}_0, \mathcal{C}_0 \rangle$, ..., $\langle \mathcal{P}_i, \mathcal{O}_i, \mathcal{D}_i, \mathcal{C}_i \rangle$, ..., $\langle \mathcal{P}_n, \mathcal{O}_n, \mathcal{D}_n, \mathcal{C}_n \rangle$ be a *fdd* $\mathcal{D} \models^{f,g}_{\mathcal{AF}_{Agent}} \mathcal{P}$, i.e. $\mathcal{P}_0 = \mathcal{P}$, $\mathcal{D}_n = \mathcal{D}$. We show that for each such derivation there exists a *dd* $\langle \mathcal{P}'_0, \mathcal{O}'_0, A'_0, \mathcal{C}'_0 \rangle$, ..., $\langle \mathcal{P}'_j, \mathcal{O}'_j, A'_j, \mathcal{C}'_j \rangle$, ..., $\langle \mathcal{P}'_m, \mathcal{O}'_m, A'_m, \mathcal{C}'_m \rangle$ $(m \leq n)$ where $\mathcal{P}'_0 = \mathcal{P}$ and $A'_m = \mathcal{D} \cap \mathcal{A}$.

For a given *fdd*, construct the *dd* as follows:

- For the initial step, trivially $\mathcal{P}_0 = \mathcal{P}'_0$, $\mathcal{O}_0 = \mathcal{O}'_0 = \{\}$, $\mathcal{D}_0 \cap \mathcal{A} = A'_0 = \mathcal{P} \cap \mathcal{A}$ and $\mathcal{C}_0 = \mathcal{C}'_0 = \{\}$. Hence $\langle \mathcal{P}_0, \mathcal{O}_0, \mathcal{D}_0, \mathcal{C}_0 \rangle \approx \langle \mathcal{P}'_0, \mathcal{O}'_0, A'_0, \mathcal{C}'_0 \rangle$.

- Assume that $\langle \mathcal{P}'_0, \mathcal{O}'_0, A'_0, \mathcal{C}'_0 \rangle$, ..., $\langle \mathcal{P}'_j, \mathcal{O}'_j, A'_j, \mathcal{C}'_j \rangle$ $(j < m)$ is a sub-sequence of a *dd* corresponding to the sub-sequence $\langle \mathcal{P}_0, \mathcal{O}_0, \mathcal{D}_0, \mathcal{C}_0 \rangle$, ..., $\langle \mathcal{P}_i, \mathcal{O}_i, \mathcal{D}_i, \mathcal{C}_i \rangle$ $(i < n)$ of a *fdd* such that $\langle \mathcal{P}_i, \mathcal{O}_i, \mathcal{D}_i, \mathcal{C}_i \rangle \approx \langle \mathcal{P}'_j, \mathcal{O}'_j, A'_j, \mathcal{C}'_j \rangle$. Then,

[1] We refer to the variant of the AB-dispute derivation procedure as discussed in Section 6.5 where the cases 2ic.1 and 2ic.2 are combined into one single case 2ic.

1. If the player choice is the proponent, $f(\mathcal{P}_i) = \sigma$ and $\sigma \in \mathcal{R}^p$, then by applying case 1a of Definition 6.5, $\mathcal{P}_{i+1} = \mathcal{P}_i - \{\sigma\}$, $\mathcal{O}_{i+1} = \mathcal{O}_i$, $\mathcal{D}_{i+1} = \mathcal{D}_i$ and $\mathcal{C}_{i+1} = \mathcal{C}_i$. Select $f'(\mathcal{P}'_j) = \sigma$ and apply case 1ii of the *dd*s, then $\mathcal{P}'_{j+1} = \mathcal{P}'_j - \{\sigma\}$ (since σ is a fact), $\mathcal{O}'_{j+1} = \mathcal{O}'_j$, $\mathcal{A}'_{j+1} = \mathcal{A}'_j$ (since σ is a fact) and $\mathcal{C}'_{j+1} = \mathcal{C}'_j$. The only change in both derivation sequences is the removal of σ from the proponent's set. Thus, trivially by hypothesis, $\langle \mathcal{P}_{i+1}, \mathcal{O}_{i+1}, \mathcal{D}_{i+1}, \mathcal{C}_{i+1} \rangle \approx \langle \mathcal{P}'_{j+1}, \mathcal{O}'_{j+1}, A'_{j+1}, \mathcal{C}'_{j+1} \rangle$.

2. If the player choice is the proponent, $f(\mathcal{P}_i) = \sigma$ and $\sigma \in \mathcal{A}$, then by applying case 1b of Definition 6.5, $\mathcal{P}_{i+1} = \mathcal{P}_i - \{\sigma\}$, $\mathcal{O}_{i+1} = \mathcal{O}_i \cup \{S \mid S \vdash^f_{\mathcal{AF}_{Agent}} c(\sigma),\ S \cap \mathcal{C}_i = \phi\ \}$, $\mathcal{D}_{i+1} = \mathcal{D}_i$ and $\mathcal{C}_{i+1} = \mathcal{C}_i$. Select $f'(\mathcal{P}'_j) = \sigma$ and construct sequence $\langle \mathcal{P}'_j, \mathcal{O}'_j, A'_j, \mathcal{C}'_j \rangle$, $\langle \mathcal{P}'_{j+1}, \mathcal{O}'_{j+1}, A'_{j+1}, \mathcal{C}'_{j+1} \rangle, \ldots, \langle \mathcal{P}'_k, \mathcal{O}'_k, A'_k, \mathcal{C}'_k \rangle$ $(j+1 \leq k)$ by applying case 1i of the *dd* procedure once, followed by applying case 2ii of the *dd* procedure repeatedly (initially selecting $\{c(\sigma)\} \in \mathcal{O}'_{j+1}$) unfolding the different f-arguments for $c(\sigma)$ that contain no culprit assumptions. Then, $\mathcal{P}'_k = \mathcal{P}'_j - \{\sigma\}$, $\mathcal{O}'_k = \mathcal{O}'_j \cup \{\ S \mid S \vdash^{f'}_{\mathcal{AF}_{Agent}} c(\sigma),\ S \cap \mathcal{C}'_j = \phi\ \}$, $\mathcal{A}'_k = \mathcal{A}'_j$ and $\mathcal{C}'_k = \mathcal{C}'_j$. Since $\mathcal{O}_i = \mathcal{O}'_j$ and $\mathcal{C}_i = \mathcal{C}'_j$ by hypothesis, and since the set of all f-arguments for $c(\sigma)$ is irrespective of the selection function used, trivially, we have

$$\mathcal{O}_i \cup \{S \mid S \vdash^f_{\mathcal{AF}_{Agent}} c(\sigma),\ S \cap \mathcal{C}_i = \phi\} = \mathcal{O}'_j \cup \{S \mid S \vdash^{f'}_{\mathcal{AF}_{Agent}} c(\sigma),\ S \cap \mathcal{C}'_j = \phi\}$$

which is equivalent to $\mathcal{O}_{i+1} = \mathcal{O}'_k$. The sets of culprits and defences are unchanged in both derivations, and the only change in the proponent's set in both derivations is the removal of σ. Thus, trivially by hypothesis, $\langle \mathcal{P}_{i+1}, \mathcal{O}_{i+1}, \mathcal{D}_{i+1}, \mathcal{C}_{i+1} \rangle \approx \langle \mathcal{P}'_k, \mathcal{O}'_k, A'_k, \mathcal{C}'_k \rangle$.

3. If the player choice is the proponent, $f(\mathcal{P}_i) = \sigma$, $\sigma \notin \mathcal{A} \cup \mathcal{R}^p$ and $S \vdash^f_{\mathcal{AF}_{Agent}} \sigma$ is chosen in applying case 1c of Definition 6.5, then, $\mathcal{P}_{i+1} = (\mathcal{P}_i - \{\sigma\}) \cup ((S \cap \mathcal{A}) - \mathcal{D}_i)$, $\mathcal{O}_{i+1} = \mathcal{O}_i$, $\mathcal{D}_{i+1} = \mathcal{D}_i \cup S$ and $\mathcal{C}_{i+1} = \mathcal{C}_i$. Construct sequence $\langle \mathcal{P}'_j, \mathcal{O}'_j, A'_j, \mathcal{C}'_j \rangle, \ldots,$ $\langle \mathcal{P}'_k, \mathcal{O}'_k, A'_k, \mathcal{C}'_k \rangle$ $(j < k)$ by applying case 1ii of the *dd* procedure repeatedly (initially selecting $f'(\mathcal{P}_j) = \sigma$) unfolding S from σ (with filtering of defence assumptions by defences), followed by applying case 1ii repeatedly removing each personal fact in S (leaving $S \cap \mathcal{A}$). Then, $\mathcal{P}'_k = (\mathcal{P}'_j - \{\sigma\}) \cup ((S \cap \mathcal{A}) - A'_j)$, $\mathcal{O}'_k = \mathcal{O}'_j$, $A'_k = A'_j \cup (\mathcal{A} \cap S)$

and $\mathcal{C}'_k = \mathcal{C}'_j$. Since $\mathcal{P}_i = \mathcal{P}'_j$ and $\mathcal{D}_i \cap \mathcal{A} = A'_j$ by hypothesis, trivially, $\mathcal{P}_{i+1} = \mathcal{P}'_k$. Since $\mathcal{D}_i \cap \mathcal{A} = A'_j$ by hypothesis, trivially, $\mathcal{D}_{i+1} \cap \mathcal{A} = A'_k$. The sets of culprits and defences are unchanged in both derivations. Thus $\langle \mathcal{P}_{i+1}, \mathcal{O}_{i+1}, \mathcal{D}_{i+1}, \mathcal{C}_{i+1} \rangle \approx \langle \mathcal{P}'_k, \mathcal{O}'_k, A'_k, \mathcal{C}'_k \rangle$.

4. If the player choice is the opponent, $g(\mathcal{O}_i) = \Sigma$ and there is a $\sigma \in \Sigma$ such that $\sigma \in \mathcal{C}_i$ (i.e. $\Sigma \cap \mathcal{C}_i \neq \phi$), then, applying case 2a of Definition 6.5, $\mathcal{P}_{i+1} = \mathcal{P}_i$, $\mathcal{O}_{i+1} = \mathcal{O}_i - \{\Sigma\}$, $\mathcal{D}_{i+1} = \mathcal{D}_i$ and $\mathcal{C}_{i+1} = \mathcal{C}_i$. Select $\Sigma \in \mathcal{O}'_j$ and $f'(\Sigma) = \sigma$, and applying case 2ib of the *dd* procedure, $\mathcal{P}'_{j+1} = \mathcal{P}_j$, $\mathcal{O}'_{j+1} = \mathcal{O}'_j - \{\Sigma\}$, $A'_{j+1} = A_j$ and $\mathcal{C}'_{j+1} = \mathcal{C}_j$. The only change in both derivation sequences is the removal of Σ from the opponent's set. Thus, trivially, $\langle \mathcal{P}_{i+1}, \mathcal{O}_{i+1}, \mathcal{D}_{i+1}, \mathcal{C}_{i+1} \rangle \approx \langle \mathcal{P}'_{j+1}, \mathcal{O}'_{j+1}, A'_{j+1}, \mathcal{C}'_{j+1} \rangle$.

5. If the player choice is the opponent, $g(\mathcal{O}_i) = \Sigma$ and $\sigma \in \Sigma$ is chosen in applying case 2b of Definition 6.5 (i.e. $\sigma \in \mathcal{A}$, $\sigma \notin \mathcal{C}_i$, $\sigma \notin \mathcal{D}_i$), then $\mathcal{P}_{i+1} = \mathcal{P}_i \cup \{c(\sigma)\}$, $\mathcal{O}_{i+1} = \mathcal{O}_i - \{\Sigma\}$, $\mathcal{D}_{i+1} = \mathcal{D}_i \cup (\{c(\sigma)\} \cap (\mathcal{A} \cup \mathcal{R}^p))$ and $\mathcal{C}_{i+1} = \mathcal{C}_i \cup \{\sigma\}$. Select $\Sigma \in \mathcal{O}'_j$ and $f'(\Sigma) = \sigma$, and applying case 2ic of the *dd* procedure, $\mathcal{P}'_{j+1} = \mathcal{P}_j \cup \{c(\sigma)\}$, $\mathcal{O}'_{j+1} = \mathcal{O}'_j - \{\Sigma\}$, $A'_{j+1} = A_j \cup (\{c(\sigma)\} \cap \mathcal{A})$ and $\mathcal{C}'_{j+1} = \mathcal{C}_j$. Since $\mathcal{P}_i = \mathcal{P}'_j$ by hypothesis, trivially, $\mathcal{P}_{i+1} = \mathcal{P}'_{j+1}$. Since $\mathcal{O}_i = \mathcal{O}'_j$ by hypothesis, trivially, $\mathcal{O}_{i+1} = \mathcal{O}'_{j+1}$. Since $\mathcal{D}_i \cap \mathcal{A} = A'_j$ by hypothesis, trivially, $\mathcal{D}_{i+1} \cap \mathcal{A} = A'_{j+1}$. Since $\mathcal{C}_i = \mathcal{C}'_j$ by hypothesis, trivially, $\mathcal{C}_{i+1} = \mathcal{C}'_{j+1}$. Thus $\langle \mathcal{P}_{i+1}, \mathcal{O}_{i+1}, \mathcal{D}_{i+1}, \mathcal{C}_{i+1} \rangle \approx \langle \mathcal{P}'_{j+1}, \mathcal{O}'_{j+1}, A'_{j+1}, \mathcal{C}'_{j+1} \rangle$.

By construction, $\langle \mathcal{P}_n, \mathcal{O}_n, \mathcal{D}_n, \mathcal{C}_n \rangle \approx \langle \mathcal{P}'_m, \mathcal{O}'_m, A'_m, \mathcal{C}'_m \rangle$ and hence $\mathcal{D}_n \cap \mathcal{A} = A'_m$. Thus a defence set derived by the *fdd* procedure (minus the *facts*) is also derivable by the *dd* procedure. □

D.2.2 Proof of Theorem 6.2

Let $\langle \mathcal{P}'_0, \mathcal{O}'_0, A'_0, \mathcal{C}'_0 \rangle, \ldots, \langle \mathcal{P}'_j, \mathcal{O}'_j, A'_j, \mathcal{C}'_j \rangle, \ldots, \langle \mathcal{P}'_m, \mathcal{O}'_m, A'_m, \mathcal{C}'_m \rangle$ be a *dd*. We show that for each such derivation there exists a *fdd* $\langle \mathcal{P}_0, \mathcal{O}_0, \mathcal{D}_0, \mathcal{C}_0 \rangle, \ldots, \langle \mathcal{P}_i, \mathcal{O}_i, \mathcal{D}_i, \mathcal{C}_i \rangle, \ldots, \langle \mathcal{P}_n, \mathcal{O}_n, \mathcal{D}_n, \mathcal{C}_n \rangle$ ($\mathcal{D}_n \models^{f,g}_{\mathcal{AF}_{Agent}} \mathcal{P}_0$) ($m \leq n$) such that the defence set $A'_m = (\mathcal{D}_n \cap \mathcal{A})$.

We assume that the player choice and selection function is such that if there is a Σ in the opponent's set that contains a $\sigma \notin \mathcal{A} \cap \mathcal{R}^p$, then the player choice will be the opponent and

Σ as well as σ will be selected. This ensures in the proof that the opponent's f-arguments are fully contructed in the *dd*s before being dealt with (by one of the cases 2ia–2ic) or before the proponent plays again.

For a given *dd*, construct the *fdd* as follows:

- For the initial step, trivially $\mathcal{P}_0 = \mathcal{P}'_0$, $\mathcal{O}_0 = \mathcal{O}'_0 = \{\}$, $\mathcal{D}_0 \cap \mathcal{A} = A'_0 = \mathcal{P} \cap \mathcal{A}$ and $\mathcal{C}_0 = \mathcal{C}'_0 = \{\}$. Hence $\langle \mathcal{P}_0, \mathcal{O}_0, \mathcal{D}_0, \mathcal{C}_0 \rangle \approx \langle \mathcal{P}'_0, \mathcal{O}'_0, A'_0, \mathcal{C}'_0 \rangle$.

- Assume that $\langle \mathcal{P}'_0, \mathcal{O}'_0, A'_0, \mathcal{C}'_0 \rangle, \ldots, \langle \mathcal{P}'_j, \mathcal{O}'_j, A'_j, \mathcal{C}'_j \rangle$ $(j < m)$ is a sub-sequence of a *dd* corresponding to the sub-sequence $\langle \mathcal{P}_0, \mathcal{O}_0, \mathcal{D}_0, \mathcal{C}_0 \rangle, \ldots, \langle \mathcal{P}_i, \mathcal{O}_i, \mathcal{D}_i, \mathcal{C}_i \rangle$ $(i < n)$ of a *fdd* such that $\langle \mathcal{P}_i, \mathcal{O}_i, \mathcal{D}_i, \mathcal{C}_i \rangle \approx \langle \mathcal{P}'_i, \mathcal{O}'_i, A'_i, \mathcal{C}'_i \rangle$. Then,

  1. If the player choice is the proponent, $f(\mathcal{P}'_j) = \sigma$ and $\sigma \in \mathcal{A}$, then, by applying case 1i of the *dd*s, $\mathcal{P}'_{j+1} = \mathcal{P}'_j - \{\sigma\}$, $\mathcal{O}'_{j+1} = \mathcal{O}'_j \cup \{\{c(\sigma)\}\}$, $A'_{j+1} = A'_j$, $\mathcal{C}'_{j+1} = \mathcal{C}'_j$. By assumption of the player choice and selection function, the sequence $\langle \mathcal{P}'_{j+1}, \mathcal{O}'_{j+1}, A'_{j+1}, \mathcal{C}'_{j+1} \rangle$, $\ldots$, $\langle \mathcal{P}'_k, \mathcal{O}'_k, A'_k, \mathcal{C}'_k \rangle$ $(j+1 \leq k)$ is constructed, applying case 2ii of the *dd* repeatedly unfolding all f-arguments for $c(\sigma)$ that contain no culprit assumptions (filtering of culprits by culprits). Then, $\mathcal{P}'_k = \mathcal{P}'_{j+1}$, $\mathcal{O}'_k = \mathcal{O}'_j \cup \{\ S \mid S \vdash^f_{\mathcal{AF}_{Agent}} c(\sigma),\ S \cap \mathcal{C}'_j = \phi\ \}$ $A'_k = A'_{j+1}$, $\mathcal{C}'_k = \mathcal{C}'_{j+1}$. Pick $f(\mathcal{P}_i) = \sigma$ and apply case 1b of Definition 6.5, then $\mathcal{P}_{i+1} = \mathcal{P}_i - \{\sigma\}$, $\mathcal{O}_{i+1} = \mathcal{O}_i \cup \{\ S \mid S \vdash^f_{\mathcal{AF}_{Agent}} c(\sigma),\ S \cap \mathcal{C}_i = \phi\ \}$, $\mathcal{D}_{i+1} = \mathcal{D}_i$, $\mathcal{C}_{i+1} = \mathcal{C}_i$. Since $\mathcal{O}_i = \mathcal{O}'_j$ and $\mathcal{C}_i = \mathcal{C}'_j$ by hypothesis, trivially, $\mathcal{O}_{i+1} = \mathcal{O}'_k$. The sets of culprits and defences are unchanged in both derivations, and the only change in the proponent's set in both derivations is the removal of σ. Thus, by hypothesis, $\langle \mathcal{P}_{i+1}, \mathcal{O}_{i+1}, \mathcal{D}_{i+1}, \mathcal{C}_{i+1} \rangle \approx \langle \mathcal{P}'_k, \mathcal{O}'_k, A'_k, \mathcal{C}'_k \rangle$.

  2. If the player choice is the proponent, $f(\mathcal{P}'_j) = \sigma$ and $\sigma \notin \mathcal{A}$, then there are two possibilities to consider.

     - $\sigma \in \mathcal{R}^p$. Apply case 1ii of the *dd*s, then $\mathcal{P}'_{j+1} = \mathcal{P}'_j - \{\sigma\}$, $\mathcal{O}'_{j+1} = \mathcal{O}'_j$, $A'_{j+1} = A'_j$, $\mathcal{C}'_{j+1} = \mathcal{C}'_j$. Pick $f(\mathcal{P}_i) = \sigma$ and apply case 1a of Definition 6.5, then $\mathcal{P}_{i+1} = \mathcal{P}_i - \{\sigma\}$, $\mathcal{O}_{i+1} = \mathcal{O}_i$, $\mathcal{D}_{i+1} = \mathcal{D}_i$, $\mathcal{C}_{i+1} = \mathcal{C}_i$. The only change in both derivation

sequences is the removal of σ from the proponent's set. Thus, by hypothesis, $\langle \mathcal{P}_{i+1}, \mathcal{O}_{i+1}, \mathcal{D}_{i+1}, \mathcal{C}_{i+1} \rangle \approx \langle \mathcal{P}'_{j+1}, \mathcal{O}'_{j+1}, A'_{j+1}, \mathcal{C}'_{j+1} \rangle$.

– $\sigma \notin \mathcal{R}^p$. Construct sequence $\langle \mathcal{P}'_j, \mathcal{O}'_j, A'_j, \mathcal{C}'_j \rangle$, ..., $\langle \mathcal{P}'_k, \mathcal{O}'_k, A'_k, \mathcal{C}'_k \rangle$ $(j < k)$ by applying case 1ii of the *dd*s repeatedly, unfolding a f-argument for σ with support S (such that $\mathcal{C}'_j \cap S \neq \phi$) while filtering defence assumptions by defences, followed by applying 1ii for each personal fact in S (removing it from the proponent's set). Then, $\mathcal{P}'_k = \mathcal{P}'_j - \{\sigma\} \cup ((S \cap \mathcal{A}) - A'_j)$, $\mathcal{O}'_k = \mathcal{O}'_j$, $A'_k = A'_j \cup (\mathcal{A} \cap S)$, $\mathcal{C}'_k = \mathcal{C}'_j$. Pick $f(\mathcal{P}_i) = \sigma$ and choose $S \vdash^f_{\mathcal{AF}_{Agent}} \sigma$ in applying case 1c of Definition 6.5, then $\mathcal{P}_{i+1} = \mathcal{P}_i - \{\sigma\} \cup ((S \cap \mathcal{A}) - \mathcal{D}_i)$, $\mathcal{O}_{i+1} = \mathcal{O}_i$, $\mathcal{D}_{i+1} = \mathcal{D}_i \cup S$, $\mathcal{C}_{i+1} = \mathcal{C}_i$. Since $\mathcal{P}_i = \mathcal{P}'_j$ and $\mathcal{D}_i \cap \mathcal{A} = A'_j$ by hypothesis, trivially, $\mathcal{P}_{i+1} = \mathcal{P}'_k$. Since $\mathcal{D}_i \cap \mathcal{A} = A'_j$ by hypothesis, trivially, $\mathcal{D}_{i+1} \cap \mathcal{A} = A'_k$. The opponent's set and the set of culprits are unchanged in both derivations. Thus, by hypothesis, $\langle \mathcal{P}_{i+1}, \mathcal{O}_{i+1}, \mathcal{D}_{i+1}, \mathcal{C}_{i+1} \rangle \approx \langle \mathcal{P}'_k, \mathcal{O}'_k, A'_k, \mathcal{C}'_k \rangle$.

3. If the player choice is the opponent, $\Sigma \in \mathcal{O}'_j$ is selected, $f(\Sigma) = \sigma$ and a sequence $\langle \mathcal{P}'_j, \mathcal{O}'_j, A'_j, \mathcal{C}'_j \rangle$, ..., $\langle \mathcal{P}'_k, \mathcal{O}'_k, A'_k, \mathcal{C}'_k \rangle$ is constructed by applying case 2ia of the *dd*s repeatedly, ignoring assumptions in Σ until $\langle \mathcal{P}'_{k+1}, \mathcal{O}'_{k+1}, A'_{k+1}, \mathcal{C}'_{k+1} \rangle$ is constructed by selecting $\sigma \in \Sigma$ and applying case 2ib (or 2ic) of the *dd*s. Then $\langle \mathcal{P}'_{k+1}, \mathcal{O}'_{k+1}, A'_{k+1}, \mathcal{C}'_{k+1} \rangle$ is exactly as it would be if constructed directly from $\langle \mathcal{P}'_j, \mathcal{O}'_j, A'_j, \mathcal{C}'_j \rangle$ by selecting $\sigma \in \Sigma$ and applying case 2ib (or 2ic respectively) of the *dd*s. Then, as shown in points 4 and 5 below, an $\langle \mathcal{P}_{i+1}, \mathcal{O}_{i+1}, \mathcal{D}_{i+1}, \mathcal{C}_{i+1} \rangle$ can be constructed from $\langle \mathcal{P}_i, \mathcal{O}_i, \mathcal{D}_i, \mathcal{C}_i \rangle$ by selecting $\Sigma \in \mathcal{O}_i$, $\sigma \in \Sigma$ and applying case 2a (or 2b respectively) of Definition 6.5 such that $\langle \mathcal{P}_{i+1}, \mathcal{O}_{i+1}, \mathcal{D}_{i+1}, \mathcal{C}_{i+1} \rangle \approx \langle \mathcal{P}'_{k+1}, \mathcal{O}'_{k+1}, A'_{k+1}, \mathcal{C}'_{k+1} \rangle$.

4. If the player choice is the opponent, $\Sigma \in \mathcal{O}'_j$ is selected, $f(\Sigma) = \sigma$, $\sigma \in \mathcal{A}$, $\sigma \notin A_i$ and $\sigma \in \mathcal{C}'$, then, by applying case 2ib of the *dd*s, $\mathcal{P}'_{j+1} = \mathcal{P}'_j$, $\mathcal{O}'_{j+1} = \mathcal{O}'_j - \{\Sigma\}$, $A'_{j+1} = A'_j$, $\mathcal{C}'_{j+1} = \mathcal{C}'_j$. Pick $g(\mathcal{O}_i) = \Sigma$ and apply case 2a of Definition 6.5, then $\mathcal{P}_{i+1} = \mathcal{P}_i$, $\mathcal{O}_{i+1} = \mathcal{O}_i - \{\Sigma\}$, $\mathcal{D}_{i+1} = \mathcal{D}_i$, $\mathcal{C}_{i+1} = \mathcal{C}_i$. The only change in both derivations is the removal of Σ from the opponent's set. Thus, trivially by hypothesis, $\langle \mathcal{P}_{i+1}, \mathcal{O}_{i+1}, \mathcal{D}_{i+1}, \mathcal{C}_{i+1} \rangle \approx \langle \mathcal{P}'_{j+1}, \mathcal{O}'_{j+1}, A'_{j+1}, \mathcal{C}'_{j+1} \rangle$.

5. If the player choice is the opponent, $\Sigma \in \mathcal{O}'_j$ is selected $f(\Sigma) = \sigma$, $\sigma \in \mathcal{A}$, $\sigma \notin A_i$ and

$\sigma \notin \mathcal{C}'$, then, by applying case 2ic of the *dd*s, $\mathcal{P}'_{j+1} = \mathcal{P}'_j \cup \{c(\sigma)\}$, $\mathcal{O}'_{j+1} = \mathcal{O}'_j - \{\Sigma\}$, $A'_{j+1} = A'_j \cup (\mathcal{A} \cap \{c(\sigma)\})$, $\mathcal{C}'_{j+1} = \mathcal{C}'_j \cup \{\sigma\}$. Pick $g(\mathcal{O}_i) = \Sigma$ and choose $\sigma \in \Sigma$ in applying case 2b of Definition 6.5, then $\mathcal{P}_{i+1} = \mathcal{P}_i \cup \{c(\sigma)\}$, $\mathcal{O}_{i+1} = \mathcal{O}_i - \{\Sigma\}$, $\mathcal{D}_{i+1} = \mathcal{D}_i \cup (\{c(\sigma)\} \cap (\mathcal{A} \cup \mathcal{R}^p))$, $\mathcal{C}_{i+1} = \mathcal{C}_i \cup \{\sigma\}$. Since $\mathcal{P}_i = \mathcal{P}'_j$ by hypothesis, trivially, $\mathcal{P}_{i+1} = \mathcal{P}'_{j+1}$. Since $\mathcal{O}_i = \mathcal{O}'_j$ by hypothesis, trivially, $\mathcal{O}_{i+1} = \mathcal{O}'_{j+1}$. Since $\mathcal{D}_i \cap \mathcal{A} = A'_j$ by hypothesis, trivially, $\mathcal{D}_{i+1} \cap \mathcal{A} = A'_{j+1}$. Since $\mathcal{C}_i = \mathcal{C}'_j$ by hypothesis, trivially, $\mathcal{C}_{i+1} = \mathcal{C}'_{j+1}$. Thus $\langle \mathcal{P}_{i+1}, \mathcal{O}_{i+1}, \mathcal{D}_{i+1}, \mathcal{C}_{i+1} \rangle \approx \langle \mathcal{P}'_{j+1}, \mathcal{O}'_{j+1}, A'_{j+1}, \mathcal{C}'_{j+1} \rangle$.

6. Given our assumption of the player choice and selection function, the case where 2ii of the *dd*s is applied will not arise except following on immediately from when case 1i is applied, which is covered by point 1 above.

By construction, $\langle \mathcal{P}_n, \mathcal{O}_n, \mathcal{D}_n, \mathcal{C}_n \rangle \approx \langle \mathcal{P}'_m, \mathcal{O}'_m, A'_m, \mathcal{C}'_m \rangle$ and hence $\mathcal{D}_n \cap \mathcal{A} = A'_m$. Thus a defence set derived by the *dd* procedure (consisting of assumptions only) is contained in the defence set derived by the *fdd* procedure (consisting of facts and assumptions). □

Appendix E

Definition of the Control Procedures of Chapter 5 by means of AABA

We give in this section an example definition of the elements needed to ensure that dialogue instances, as defined in Section 5, are appropriately sequenced and interleaved. We define these here using AABA for consistency with Chapter 7, though the use of AABA for this is not essential. We define first in Section E.1 the components that make up the dialogical status of the agent thereby completing the representation of our (resource reallocation) agents as an AABA framework (see Chapter 7) and then in Section E.2 we present the "control procedures" (see Chapter 5) based on this. The dialogical status and control procedures are defined here to correspond to the (non-AABA) definitions of Appendix B.

E.1 Dialogical Status as AABA framework

We provide in this section definitions for $\mathcal{L}_d$, $\mathcal{A}_d$, $\mathcal{R}_{c,d}$ and $\mathcal{R}_x^{p,d}$ which pertain to the dialogical status of some agent x, thus obtaining the full AABA framework $\mathcal{AF}_{x,d} = \langle \mathcal{L} \cup \mathcal{L}_d, \mathcal{R}_c \cup \mathcal{R}_{c,d}, \mathcal{R}_x^p \cup \mathcal{R}_x^{p,d}, \mathcal{A} \cup \mathcal{A}_d, c \rangle$, where $\mathcal{L}$, $\mathcal{R}_c$, $\mathcal{R}_x^p$ and $\mathcal{A}$ are as defined in Section 7.2.

The non-assumptions that make up $\mathcal{L}_d$ are as follows:

Definition E.1 (Non-assumptions in $\mathcal{L}_d$) *The non-assumptions $\mathcal{L}_d \setminus \mathcal{A}_d$ are (instances of) the following schemata, for $X, Y \in names(AS)$, $Z \in 2^{names(AS)}$, $R \in \mathcal{R}$, $Response \in \{acceptProposal, rejectProposal\}$:*

- *$(\neg)\ offered(X, R)$, standing for "agent X has (not) presently offered resource R to an agent as part of a request or proposal dialogue instance, neither as initiator nor responder";*
- *$(\neg)\ awaitingResponse(X)$, standing for "$X$ is (not) awaiting a response from some (any) agent, which may be following an earlier request, proposal or accept_proposal dialogue move it has sent";*
- *$initiatedDialogue(X, Z,$`Actions`$)$, standing for "X has initiated a request or proposal dialogue instance directed at a set of agents Z regarding* `Actions` *and has not yet received responses from all agents Z";*
- *$acceptedProposal(X, Y,$`Actions`$)$, standing for "X has accepted a proposal sent to it from Y regarding* `Actions` *and is awaiting confirmation";*
- *$receivedResponse(X, Y,$`Actions`$, Response)$, standing for "X has received response of the form $Response$ from Y following a proposal dialogue move it sent regarding* `Actions`*".*

The latter three sentences in Definition E.1 are intended as *facts* to be added to and removed from $\mathcal{R}_x^{p,d}$ as x engages in dialogue instances. The other sentences in $\mathcal{L}_d$ are intended as inferences and will be given meaning in terms of $\mathcal{R}_{c,d}$ below.

The assumptions $\mathcal{A}_d$ that make up $\mathcal{L}_d$ are as follows:

Definition E.2 (Assumptions in $\mathcal{A}_d$ and their contraries) *The assumptions $\mathcal{A}_d$ (and their associated contraries) in the language $\mathcal{L}_d$ are all instances of the following schemata, for $X \in names(AS)$, $R \in Res$:*

- *$asm(\neg awaitingResponse(X))$, standing for "assume that agent X is not awaiting any responses from any agents following earlier dialogue moves it may have sent",*

with $c(asm(\neg awaitingResponse(X))) = \{awaitingResponse(X)\}$;

- $asm(\neg offered(X, R))$: *standing for "assume that X has not presently offered R to any agent as part of a request or proposal dialogue instance, neither as initiator nor responder",*

  with $c(asm(\neg offered(X, R))) = \{offered(X, R)\}$.

Each of the non-assumptions (as well as the assumptions) that make up $\mathcal{L}_d$ are given meaning for our negotiation policy in terms of the inference rules in $\mathcal{R}_{c,d}$, as follows:

Definition E.3 (Inference rules in $\mathcal{R}_{c,d}$) *The set of inference rules $\mathcal{R}_{c,d}$ for agent $x \in names(AS)$ are all instances of the following schemata, for $R \in \mathcal{R}$:*

$$awaitingResponse(x) \leftarrow initiatedDialogue(x, _, _)$$
$$awaitingResponse(x) \leftarrow acceptedProposal(x, _, \texttt{Actions}),$$
$$\texttt{give(}_\texttt{,}R\texttt{,}x\texttt{)} \in \texttt{Actions}, useful(R, x)$$
$$\neg awaitingResponse(x) \leftarrow asm(\neg awaitingResponse(x))$$
$$offered(x, R) \leftarrow initiatedDialogue(x, _, \texttt{Actions}), \texttt{give(}x\texttt{,}R\texttt{,}_\texttt{)} \in \texttt{Actions}$$
$$offered(x, R) \leftarrow acceptedProposal(x, _, \texttt{Actions}), \texttt{give(}x\texttt{,}R\texttt{,}_\texttt{)} \in \texttt{Actions}$$
$$\neg offered(x, R) \leftarrow asm(\neg offered(x, R))$$

$\mathcal{R}_{c,d}$ contains inference rules representing that x is *awaiting a response* from some agent(s) if either it has initiated a request or proposal dialogue instance and has not yet received responses from all responder agents (first rule), or it has accepted a proposal in which it is to receive a *useful* resource and has not yet received a confirm or disconfirm from the initiatior agent (second rule), as well as a general-purpose inference rule (third rule) imposing a form of *closed world assumption* for when x is *not awaiting a response*. Also, inference rules representing that x has offered to give away a resource R if either it "remembers" initiating a dialogue instance or accepting a proposal as such (fourth and fifth rules), as well as a general-purpose inference rule (sixth rule) imposing a form of *closed world assumption* for those resources x has not offered to give away.

E.2 Control Procedures

We now provide a definition of the "control procedures" occurring in the communication rules of our negotiation policy for an agent x in terms of the AABA framework $\mathcal{AF}_{x,d}$ defined above. The definitions of the procedures below correlate with the definitions of Appendix B.

The `valid(initiateDialogue(Actions))` procedure, which correlates with Definition B.1, is as follows:

Definition E.4 (`valid(initiateDialogue(Actions))`) *The procedure evaluates to true if the following conditions hold:*

- $\models_{\mathcal{AF}_x} \neg awaitingResponse(x)$*; and*
- *for all* `give(`x`,`R`,_ ) ∈ Actions`*, it is the case that* $\models_{\mathcal{AF}_x} \neg offered(x, R)$.

And evaluates to false otherwise.

The `valid(accept(`Y`,Actions))` procedure, which correlates with Definition B.2, is as follows:

Definition E.5 (`valid(accept(Actions))`) *The procedure evaluates to true if the following conditions hold:*

- *for all resources R for which the condition* $\models_{\mathcal{AF}_x} offered(x, R)$ *holds, it is the case that* $\models_{\mathcal{AF}_x} \neg useful(x, R)$*; and*
- *for all* `give(`x`,`R`,_ ) ∈ Actions`*, it is the case that* $\models_{\mathcal{AF}_x} \neg offered(x, R)$.

And evaluates to false otherwise.

The `valid(receiveResponse(`Y`,Actions))` procedure, which correlates with Definition B.3, is as follows:

Definition E.6 (`valid(receiveResponse(`Y`,Actions))`) *The procedure evaluates to true if the condition* $\models_{\mathcal{AF}_x}$ *initiatedDialogue*$(x, Z,$`Actions`$)$ *holds, and* $Y \in Z$*. The procedure evaluates to false otherwise.*

The `valid(confirm(`Z`,Actions))` procedure, which correlates with Definition B.4, is as follows:

Definition E.7 (`valid(confirm(`Z`,Actions)))` *The procedure evaluates to true if the condition* $\models_{\mathcal{AF}_x}$ *initiatedDialogue*$(x, Z,$`Actions`$)$ *holds, and for all* $Y \in Z$*, it is the case that* $\models_{\mathcal{AF}_x}$ *receivedResponse*$(x, Y,$`Actions`$,$ *acceptProposal*$)$*. The procedure evaluates to false otherwise.*

The `valid(disconfirm(`Z`,Actions))` procedure, which correlates with Definition B.5, is as follows:

Definition E.8 (`valid(disconfirm(`Z`,Actions)))` *The procedure evaluates to true if the following conditions hold:*

- $\models_{\mathcal{AF}_x}$ *initiatedDialogue*$(x, Z',$`Actions`$)$*; and*
- *for all* $Y \in Z'$, $\models_{\mathcal{AF}_x}$ *receivedResponse*$(x, Y,$`Actions`$, _)$*; and*
- $Z = \{Y \mid \models_{\mathcal{AF}_x} receivedResponse(x, Y,$`Actions`$, acceptProposal)\} \subset Z'$.

And evaluates to false otherwise.

The `valid(receiveConfirm(`Y`,Actions))` procedure, which correlates with Definition B.6, is as follows:

Definition E.9 (`valid(receiveConfirm(`Y`,Actions)))` *The procedure evaluates to true if the condition* $\models_{\mathcal{AF}_x}$ *acceptedProposal*$(x, Y, Actions)$ *holds, and evaluates to false otherwise.*

The `do(initiateDialogue(`Z`,Actions))` procedure, which correlates with Definition B.7, is as follows:

Definition E.10 (`do(initiateDialogue(`Z`,Actions))`) x *adds* $initiatedDialogue(x, Z,$ `Actions`$)$ *to* $\mathcal{R}_x^{p,d}$.

The `do(accept_proposal(`Y`,Actions))` procedure, which correlates with Definition B.8, is as follows:

Definition E.11 (`do(accept_proposal(`Y`,Actions))`) x *adds* $acceptedProposal(x, Y,$ `Actions`$)$ *to* $\mathcal{R}_x^{p,d}$.

The `do(receiveResponse(`Y`,Actions,`$Response$`))` procedure, which correlates with Definition B.9, is as follows:

Definition E.12 (`do(receiveResponse(`Y`,Actions,`$Response$`))`) *x updates its state according to the value of Response, as follows:*

- *if* $Response \in \{agree, refuse\}$*, then* $initiatedDialogue(x, \{Y\},$`Actions`$)$ *is removed from* $\mathcal{R}_x^{p,d}$*;*
- *otherwise, if* $Response \in \{acceptProposal, rejectProposal\}$*, then* $receivedResponse(x, Y,$ `Actions`$, Response)$ *is added to* $\mathcal{R}_x^{p,d}$.

The `do(confirm(Actions))` procedure, which correlates with Definition B.10, is as follows:

Definition E.13 (`do(confirm(Actions))`) x *removes* $initiatedDialogue(x, Z,$`Actions`$)$ *from* $\mathcal{R}_x^{p,d}$*, and for all* $Y \in Z$*, removes* $receivedResponse(x, Y,$`Actions`$, _\,)$ *from* $\mathcal{R}_x^{p,d}$.

The `do(receiveConfirm(`Y`,Actions))` procedure, which correlates with Definition B.11, is as follows:

Definition E.14 (`do(receiveConfirm(`Y`,Actions))`) x *removes* $acceptedProposal(x, Y,$ `Actions`$)$ *from* $\mathcal{R}_x^{p,d}$.

Bibliography

[AF03] Martyn Fletcher Agent and Martyn Fletcher. The story of the holonic packing cell. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, AAMAS 2003, New York, NY, USA, 2003. ACM.

[AMP00] Leila Amgoud, Nicolas Maudet, and Simon Parsons. Modelling dialogues using argumentation. In *ICMAS 2000: Proceedings of the Fourth International Conference on MultiAgent Systems*, pages 31–38. IEEE Press, 2000.

[APM00] Leila Amgoud, Simon Parsons, and Nicolas Maudet. Arguments, dialogue, and negotiation. In *ECAI 2000: Proceedings of the 14th European Conference on Artificial Intelligence*, pages 338–342. IOS Press, 2000.

[AS99] Martin Andersson and Tuomas Sandholm. Time-quality tradeoffs in reallocative negotiation with combinatorial contract types. In *AAAI 1999: Proceedings of the 16th national conference on Artificial intelligence*, pages 3–10. American Association for Artificial Intelligence, 1999.

[Aus62] John Austin. *How to do things with words*. Oxford University Press, 1962.

[AvH04] Grigoris Antoniou and Frank van Harmelen. *A Semantic Web Primer*. MIT Press, 2004.

[BCG07] Fabio Luigi Bellifemine, Giovanni Caire, and Dominic Greenwood. *Developing Multi-agent Systems with JADE*. Wiley-Blackwell, 2007.

[BH07] Elizabeth Black and Anthony Hunter. A generative inquiry dialogue system. In *AAMAS 2007: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pages 1–8. ACM, 2007.

[BH08a] Philippe Besnard and Anthony Hunter. *Elements of Argumentation*. MIT Press, 2008.

[BH08b] Elizabeth Black and Anthony Hunter. Using enthymemes in an inquiry dialogue system. In *AAMAS 2008: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, pages 437–444. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[BHS03] Martin D. Beer, Richard Hill, and Andrew Sixsmith. Deploying an agent-based architecture for the management of community care. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, AAMAS 2003, pages 932–933, New York, NY, USA, 2003. ACM.

[Cas97] Cristiano Castelfranchi. To be or not to be an agent. In *Proceedings of the Workshop on Intelligent Agents III, Agent Theories, Architectures, and Languages*, pages 37–39, London, UK, 1997. Springer-Verlag.

[CDE+06] Yann Chevaleyre, Paul E. Dunne, Ulle Endriss, Jrme Lang, Michel Lematre, Nicolas Maudet, Julian Padget, Steve Phelps, Juan A. Rodrguez-aguilar, and Paulo Sousa. Issues in multiagent resource allocation. *Informatica*, 30(1):3–31, 2006.

[CGG+06] Vasa Curcin, Moustafa Ghanem, Yike Guo, Kostas Stathis, and Francesca Toni. Building next generation service-oriented architectures using argumentation agents. In *Proceedings of the Third International Conference on Grid Services Engineering and Management*, GCEM 2006, September 2006.

[CL90] Philip R. Cohen and Hector J. Levesque. Rational interaction as the basis for communication. In *Intentions in Communication*, pages 221–256. MIT Press, 1990.

[CSS10] Peter Cramton, Yoav Shoham, and Richard Steinberg. *Combinatorial Auctions*. MIT Press, 2010.

[DC08] Paul E. Dunne and Yann Chevaleyre. The complexity of deciding reachability properties of distributed negotiation schemes. *Theoretical Computer Science*, 396:113–144, May 2008.

[DKT06] Phan Minh Dung, Robert A. Kowalski, and Francesca Toni. Dialectic proof procedures for assumption-based, admissible argumentation. *Artificial Intelligence*, 170(2):114–159, 2006.

[DKT09] Phan Minh Dung, Robert A. Kowalski, and Francesca Toni. Assumption-based argumentation. In *Argumentation in Artificial Intelligence*, pages 199–218. Springer, 2009.

[DMT07] Phan Minh Dung, Paolo Mancarella, and Francesca Toni. Computing ideal sceptical argumentation. *Artificial Intelligence, Special Issue on Argumentation in Artificial Intelligence*, 171(10–15):642–674, 2007.

[DMWB06] Sylvie Doutre, Peter McBurney, Michael Wooldridge, and William Barden. Information-seeking agent dialogs with permissions and arguments. In *AAMAS 2006: Proceedings of the 5th international joint conference on Autonomous agents and multiagent systems*. ACM, 2006.

[Dun95] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–358, 1995.

[EMST03] Ulrich Endriss, Nicholas Maudet, Fariba Sadri, and Francesca Toni. Protocol conformance for logic-based agents. In *IJCAI 2003: Proceedings of the 18th international joint conference on Artificial intelligence*, pages 679–684. Morgan Kaufmann Publishers Inc., 2003.

[EMST06] Ulle Endriss, Nicolas Maudet, Fariba Sadri, and Francesca Toni. Negotiating socially optimal allocations of resources. *Journal of Artificial Intelligence Research*, 25(1):315–348, 2006.

[Far00] Peyman Faratin. *Automated Service Negotiation Between Autonomous Computational Agents*. PhD thesis, Queen Mary and Westfield College, Department of Electronic Engineering, 2000.

[FB07] Frank Fiedrich and Paul Burghardt. Agent-based systems for disaster management. *Commun. ACM*, 50:41–42, March 2007.

[ffipa] FIPA: The foundation for intelligent physical agents. http://www.fipa.org/.

[FG97] Stan Franklin and Art Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In *Proceedings of the Workshop on Intelligent Agents III, Agent Theories, Architectures, and Languages*, pages 21–35, London, UK, 1997. Springer-Verlag.

[FIP98] FIPA. Specification part 2 — agent communication language. Technical report, Foundation for Physical Agents, 1998.

[FWJ02] Shaheen S. Fatima, Michael Wooldridge, and Nicholas R. Jennings. Multi-issue negotiation under time constraints. In *AAMAS 2002: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 143–150. ACM, 2002.

[FWW+93] Tim Finin, Jay Weber, Gio Wiederhold, Mike Genesereth, Rich Fritzson, Don McKay, Stu Shapiro, Jim McGuire, Richard Pelavin, and Chris Beck. Specification of the kqml agent-communication language, 1993.

[Gae09] Dorian Gaertner. *Argumentation and Normative Reasoning*. PhD thesis, Imperial College London, 2009.

[GIRS09] Andrey Glaschenko, Anton Ivaschenko, George Rzevski, and Petr Skobelev. Multi-agent real time scheduling system for taxi companies. In *Proceedings of the Eighth*

Interntional Conference on Autonomous Agents and Multiagent Systems, AAMAS 2009. Curran Associates, Inc., May 2009.

[GMR97] Matteo Golfarelli, Dario Maio, and Stefano Rizzi. A task-swap negotiation protocol based on the contract net paradigm. Technical report, Università di Bologna, 1997.

[GPW07] Thomas F. Gordon, Henry Prakken, and Douglas Walton. The carneades model of argument and burden of proof. *Artificial Intelligence*, 171(10–15):875–896, 2007.

[GS04] Alejandro J. Garcia and Guillermo R. Simari. Defeasible logic programming: An argumentative approach. *Theory and Practice of Logic Programming*, 4(2):95–138, 2004.

[GT08] Dorian Gaertner and Francesca Toni. Hybrid argumentation and its properties. In *COMMA 2008: Proceeding of the 2008 conference on Computational Models of Argument*, pages 183–195. IOS Press, 2008.

[Hal07] Kasper Hallenborg. Decentralized scheduling of baggage handling using multi-agent technologies. In *Multiprocessor Scheduling, Theory and Applications*, pages 381–404. I-Tech Education and Publishing, 2007.

[HDM10] Nabila Hadidi, Yannis Dimopoulus, and Pavlos Moraitis. Argumentative alternating offers. In *AAMAS 2010: Proceedings of the 9th International Conference on Autonomous Agents and MultiAgent Systems*, pages 441–448, 2010.

[HP02] Dorothea Herreiner and Clemens Puppe. A simple procedure for finding equitable allocations of indivisible goods. *Social Choice and Welfare*, 19(2):415–430, 2002.

[HT07] Adil Hussain and Francesca Toni. Bilateral agent negotiation with information-seeking. In *EUMAS 2007: Proceedings of the Fifth European Workshop on Multi-Agent Systems*, 2007.

[HT08] Adil Hussain and Francesca Toni. On the benefits of argumentation for negotiation (preliminary version). In *EUMAS 2008: Proceedings of the Sixth European Workshop on Multi-Agent Systems*, 2008.

[HT09] Adil Hussain and Francesca Toni. Assumption-based argumentation for communicating agents. In *The Uses of Computational Argumentation Symposia, as part of the AAAI 2009 Fall Symposium Series*, 2009.

[Hun07] Anthony Hunter. Real arguments are approximate arguments. In *AAAI 2007: Proceedings of the 22nd national conference on Artificial intelligence*, pages 66–71. AAAI Press, 2007.

[JDC08] Clement Jonquet, Pascal Dugenie, and Stefano Cerri. Service-based integration of grid and multi-agent systems models. In Ryszard Kowalczyk, Michael Huhns, Matthias Klusch, Zakaria Maamar, and Quoc Vo, editors, *Service-Oriented Computing: Agents, Semantics, and Engineering*, volume 5006 of *Lecture Notes in Computer Science*, pages 56–68. Springer Berlin / Heidelberg, 2008.

[Jen01] Nicholas R. Jennings. An agent-based approach for building complex software systems. *Communications of the ACM*, 44(4):35–41, 2001.

[JFL+01] N. R. Jennings, P. Faratin, A. R. Lomuscio, S. Parsons, C. Sierra, and M. Wooldridge. Automated negotiation: Prospects, methods and challenges. *Group Decision and Negotiation*, 10(2):199–215, 2001.

[Kar06] N. C. Karunatillake. *Argumentation-Based Negotiation in a Social Context*. PhD thesis, School of Electronics and Computer Science, University of Southampton, 2006.

[KJRM09] Nishan C. Karunatillake, Nicholas R. Jennings, Iyad Rahwan, and Peter McBurney. Dialogue games that agents play within a society. *Artificial Intelligence*, 173(9–10):935–981, 2009.

[Kra01] Sarit Kraus. *Strategic Negotiation in Multiagent Environments*. MIT Press, 2001.

[Kri09] Vijay Krishna. *Auction Theory*. Academic Press, 2009.

[KSE98] Sarit Kraus, Katia Sycara, and Amir Evenchik. Reaching agreements through argumentation: A logical model and implementation. *Artificial Intelligence*, 104(1–2):1–69, 1998.

[KT01] Hiroaki Kitano and Satoshi Tadokoro. Robocup rescue: A grand challenge for multi-agent and intelligent systems. *AI Magazine*, 22:39–52, 2001.

[LM08] Michael Luck and Peter McBurney. Computing as interaction: agent and agreement technologies. In V. Marik, editor, *Proceedings of the 2008 IEEE International Conference on Distributed Human-Machine Systems*, 2008.

[LMSW05] M. Luck, P. McBurney, O. Shehory, and S. Willmott. *Agent Technology: Computing as Interaction (A Roadmap for Agent Based Computing)*. AgentLink, 2005.

[MBP05] S. S. Manvi, M. N. Birje, and Bhanu Prasad. An agent-based resource allocation model for computational grids. *Multiagent and Grid Systems*, 1:17–27, January 2005.

[MHP02] Peter Mcburney, David Hitchcock, and Simon Parsons. The eightfold way of deliberation dialogue. *International Journal of Intelligent Systems*, 22(1):95–132, 2002.

[MHVB00] David Montana, Jose Herrero, Gordon Vidaver, and Garrett Bidwell. A multiagent society for military transportation scheduling. *Journal of Scheduling*, 3:225–246, 2000.

[MLF96] James Mayfield, Yannis Labrou, and Tim Finin. Evaluation of kqml as an agent communication language. In *Intelligent Agents II Agent Theories, Architectures, and Languages*, pages 347–360. Springer Berlin, 1996.

[Mou88] Hervé Moulin. *Axioms of Cooperative Decision Making*. Cambridge University Press, 1988.

[MP02] Peter McBurney and Simon Parsons. Games that agents play: A formal framework for dialogues between autonomous agents. *Journal of Logic, Language and Information*, 11(3):315–334, 2002.

[MP05] Peter McBurney and Simon Parsons. Locutions for argumentation in agent interaction protocols. In Rogier M. van Eijk, Marc-Philippe Huget, and Frank Dignum, editors, *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 3*, Lecture Notes in Computer Science, pages 209–225. Springer Berlin, Germany, 2005.

[MP09] Peter McBurney and Simon Parsons. Dialogue games for agent argumentation. In *Argumentation in Artificial Intelligence*, pages 261–280. Springer, 2009.

[MVEPA03] Peter Mcburney, Rogier M. Van Eijk, Simon Parsons, and Leila Amgoud. A dialogue game protocol for agent purchase negotiations. *Autonomous Agents and Multi-Agent Systems*, 7:235–273, November 2003.

[NMJ08] Antoine Nongaillard, Philippe Mathieu, and Brigitte Jaumard. A multi-agent resource negotiation for the utilitarian social welfare. In *Engineering Societies in the Agents World IX: 9th International Workshop, ESAW 2008*, pages 208–226. Springer-Verlag, 2008.

[PFPS+92] Ramesh Patil, Richard Fikes, Peter Patel-Schneider, Donald P McKay, Tim Finin, Thomas Gruber, and Robert Neches. The darpa knowledge sharing effort: Progress report. In *Proceedings of the Third International Conference on Principles Of Knowledge Representation And Reasoning*. Morgan Kaufman, 1992.

[PHD+07] Philippe Pasquier, Ramon Hollands, Frank Dignum, Iyad Rahwan, and Liz Sonenberg. An empirical study of interest-based negotiation. In *ICEC 2007: Proceedings of the ninth international conference on Electronic commerce*, pages 339–348. ACM, 2007.

[PM99a] Jeremy Pitt and Abe Mamdani. Designing agent communication languages for multi-agent systems. In *MAAMAW 1999: Proceedings of the 9th European Work-*

shop on Modelling Autonomous Agents in a Multi-Agent World, pages 102–114. Springer-Verlag, 1999.

[PM99b] Jeremy Pitt and Abe Mamdani. A protocol-based semantics for an agent communication language. In *IJCAI 1999: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 486–491. Morgan Kaufmann Publishers Inc., 1999.

[Pra05] Henry Prakken. Coherence and flexibility in dialogue games for argumentation. *Journal of Logic and Computation*, 15(6):1009–1040, 2005.

[Proa] SICStus Prolog. http://www.sics.se/isl/sicstuswww/site/index.html.

[Prob] PrologBeans. http://www.sics.se/sicstus/docs/latest3/html/prologbeans/.

[PSJ98] Simon Parsons, Carles Sierra, and Nick Jennings. Agents that reason and negotiate by arguing. *Journal of Logic and Computation*, 8(3):261–292, 1998.

[RFM+09] Sarvapali Ramchurn, Alessandro Farinelli, Kathryn Macarthur, Mariya Polukarov, and Nick Jennings. Decentralised coordination in robocup rescue. *The Computer Journal*, 53(9):1447–1461, January 2009.

[RGS02] Sonia V. Rueda, Alejandro J. Garcia, and Guillermo R. Simari. Argument-based negotiation among bdi agents. *Journal of Computer Science and Technology*, 2(7):1–8, 2002.

[RPSD07] Iyad Rahwan, Philippe Pasquier, Liz Sonenberg, and Frank Dignum. On the benefits of exploiting hierarchical goals in bilateral automated negotiation. In *ArgMAS 2007: Proceedings of the 4th international conference on Argumentation in multi-agent systems*, pages 18–30. Springer-Verlag, 2007.

[RRJ+03] Iyad Rahwan, Sarvali D. Ramchurn, Nicholas R. Jennings, Peter McBurney, Simon Parsons, and Liz Sonenberg. Argumentation-based negotiation. *The Knowledge Engineering Review*, 18(4):343–375, 2003.

[RS09] Iyad Rahwan and Guillermo R. Simari, editors. *Argumentation in Artificial Intelligence*. Springer, 2009.

[RSD03] Iyad Rahwan, Liz Sonenberg, and Frank Dignum. Towards interest-based negotiation. In *AAMAS 2003: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 773–780. ACM, 2003.

[RZ94] Jeffrey S. Rosenschein and Gilad Zlotkin. *Rules of Encounter*. MIT Press, 1994.

[San98] Tuomas W. Sandholm. Contract types for satisficing task allocation: I theoretical results. In *AAAI Spring Symposium: Satisficing Models*, pages 68–75, 1998.

[San99] Tuomas W. Sandholm. *Multiagent systems: a modern approach to distributed artificial intelligence*, chapter Distributed rational decision making, pages 201–258. MIT Press, 1999.

[Sea69] John Searle. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, 1969.

[Sin98] Munindar P. Singh. Agent communication languages: Rethinking the principles. *Computer*, 31(12):40–47, 1998.

[SJNP97] Carles Sierra, Nick R. Jennings, Pablo Noriega, and Simon Parsons. A framework for argumentation-based negotiation. In *ATAL 1997: Proceedings of the 4th International Workshop on Intelligent Agents IV, Agent Theories, Architectures, and Languages*, pages 177–192. Springer-Verlag, 1997.

[Smi80] Reid G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104–1113, 1980.

[STT01] Fariba Sadri, Francesca Toni, and Paolo Torroni. Dialogues for negotiation: Agent varieties and dialogue sequences. In *ATAL 2001: Revised Papers from the 8th International Workshop on Intelligent Agents VIII*, pages 405–421. Springer-Verlag, 2001.

[TGK+08] Francesca Toni, Mary Grammatikou, Stella Kafetzoglou, Leonidas Lymberopoulos, Symeon Papavassileiou, Dorian Gaertner, Maxime Morge, Stefano Bromuri, Jarred Mcginnis, Kostas Stathis, Vasa Curcin, Moustafa Ghanem, and Li Guo. The argugrid platform: An overview. In Jrn Altmann, Dirk Neumann, and Thomas Fahringer, editors, *Proceedings of the 5th international workshop on Grid Economics and Business Models*, volume 5206 of *Lecture Notes in Computer Science*, pages 217–225, Berlin, Heidelberg, 2008. Springer-Verlag.

[Ton07] Francesca Toni. E-business in argugrid. In Daniel Veit and Jrn Altmann, editors, *Proceedings of the 4th international conference on Grid economics and business models*, volume 4685 of *Lecture Notes in Computer Science*, pages 164–169, Berlin, Heidelberg, 2007. Springer-Verlag.

[Ton08] Francesca Toni. Assumption-based argumentation for epistemic and practical reasoning. *Computable Models of the Law: Languages, Dialogues, Games, Ontologies*, pages 185–202, 2008.

[Tou58] Stephen E. Toulmin. *The Uses of Argument*. Cambridge University Press, 1958.

[vEGH02] Frans H. van Eemeren, Rob Grootendorst, and Francisca Snoeck Henkemans. *Argumentation: Analysis, Evaluation, Presentation*. Lawrence Erlbaum Associates Inc, 2002.

[vVP05] Jelle van Veenen and Henry Prakken. A verifiable protocol for arguing about rejections in negotiation. In *AAMAS 2005: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 1165–1166. ACM, 2005.

[Wal95] Douglas N. Walton. *Argumentation Schemes for Presumptive Reasoning*. Lawrence Erlbaum Associates Inc, 1995.

[WJ95] Michael Wooldridge and Nicholas R. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10:115–152, 1995.

[WK95] Douglas N. Walton and Erik C. W. Krabbe. *Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning*. State University of New York Press, 1995.

[Woo00] Michael J. Wooldridge. *Reasoning About Rational Agents*. The MIT Press, 2000.

[Woo09] Michael Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley and Sons, 2nd Edition, 2009.

[WP00] Michael Wooldridge and Simon Parsons. Languages for negotiation. In *ECAI 2000: Proceedings of the Fourteenth European Conference on Artificial Intelligence*, pages 393–400. Springer-Verlag, 2000.

[WVI04] Steven Wollkind, John Valasek, and Thomas R. Ioerger. Automated conflict resolution for air traffic management using cooperative multiagent negotiation. In *AIAA Guidance, Navigation, and Control Conference*, pages 2004–4992, 2004.