Argumentation- vs. Proposal-based Negotiation: An Empirical Case Study on the Basis of Game-Theoretic Solution Concepts

Angelika Först[1], Achim Rettinger[1], and Matthias Nickles[2]

[1] Department of Informatics
Technical University Munich
85748 Garching, Germany
`angelika.foerst@gmail.com`
`rettinger@cs.tum.edu`
[2] Department of Computer Science
University of Bath
Bath BA2 7AY, UK
`m.l.nickles@bath.ac.uk`

Abstract. Recently, argumentation-based negotiation has been proposed as an alternative to classical mechanism design. The main advantage of argumentation-based negotiation is that it allows agents to exchange complex justification positions rather than just simple proposals. Its proponents maintain that this property of argumentation protocols can lead to faster and beneficial agreements when used for complex multiagent negotiation. In this paper, we present an empirical comparison of argumentation-based negotiation to proposal-based negotiation in a strategic two-player scenario, using a game-theoretic solution as a benchmark, which requires full knowledge of the stage games. Our experiments show that in fact the argumentation-based approach outperforms the proposal-based approach with respect to the quality of the agreements found and the overall time to agreement.

1 Introduction

Integration of individual entities into complex, open, and heterogeneous systems like the internet and peer-to-peer networks is ubiquitous. The potential of these systems is grounded in the interaction between their parts. Since they are often heterogeneous, interacting autonomous and intelligent agents [13] tend to have conflicting interests, but often they still can profit from coordinating their actions with other agents or even cooperating with each other. Hence, coordination techniques and mechanisms rapidly gain importance in the field of distributed artificial intelligence. Central to the concept of intelligent agents is their capability to reason about themselves and their environment. This aspect is usually not exploited by game-theoretic approaches [11] to automated negotiation and thus these approaches often lack flexibility. In recent years, argumentation-based negotiation [2] has been suggested as an approach to negotiation that takes advantage of the abilities of intelligent agents to reason about rich interaction scenarios where complex justification positions (and not just simple proposals) can be

exchanged [8, 9]. Therefore this approach is currently enjoying increasing popularity in the field of negotiation research. However, until today, only very few approaches exist in which the performance of argumentation-based negotiating agents, bargaining agents and game-theoretic solution concepts can actually be compared in a specific scenario.

The problem definition of this paper is driven mainly by two aspects: Firstly, many negotiation settings are well-researched and have been analysed using game-theoretic techniques. The merits are that optimal negotiation mechanisms and strategies can be provided for a broad range of problems, which are also used in real-world scenarios, e.g. auctions. But then, the applicability of such solutions is often restricted to specific situations. Secondly, the emerging field of argumentation-based negotiation endeavours to overcome some of the fundamental limitations of the game-theoretic approach, notably partial knowledge, inconsistent beliefs and bounded rationality. Substantial work has been done in this field, and a number of implementations have been realised (see [5] for recent theoretical and software approaches to argumentation-based negotiation). However, until today, very little work exists in which different approaches are implemented and the performance of argumentation-based negotiating agents, bargaining agents and game-theoretic solution concepts can actually be compared in a specific scenario. The objective of this paper is to examine the benefits of different types of negotiation in a complex and stochastic environment in which agents only dispose of partial, incomplete knowledge. For this purpose, a negotiation framework is implemented, together with negotiating agents using different negotiation mechanisms. The performance of our solution concepts is evaluated empirically by benchmarking their performance against a provably optimal solution borrowed from game theory that requires complete and fully observable information.

Our evaluation shows that the different negotiation mechanisms that were tests can be clearly ranked with respect to their performance. The upper benchmark is set by the employment of a game theoretic mediator with complete knowledge who discharges the agents from negotiation by computing the optimal outcome for them. If agents are bound to negotiate under incomplete knowledge, the argumentation-based approach is clearly favourable to bargaining with respect to a number of evaluation criteria.

This paper is structured as follows: In the next section we introduce the environment within which the negotiating agents are situated. In Section 3 we present our solution concepts in an abstract form. The verification of our working hypothesis was conducted through extensive empirical evaluation - Section 4 is dedicated to the presentation of the experimental setup, the main experimental results, and an interpretation of our findings. Section 5 concludes with a summary and suggestions for future work on the topic.

2   The Testbed

In the following, we describe the testbed used for the subsequent evaluation and comparison task. Our testbed is designed in a way that makes the negotiation scenario complex enough to draw meaningful conclusions while keeping the negotiation processes comprehensible and analyzable. In game theoretic terms our scenario is based on the most general framework of games, namely general sum stochastic games [6, 12]. In our case players additionally have to deal with incomplete and partially observable infor-

mation - as possessions of other players are not public - making it difficult to apply game theoretic solutions.

The scenario the agents are situated in is a production game. All players receive different kinds of resources. Each player tries to collect a certain number of resources of one type at a time to assemble products. By selling their products agents earn game points. The functionality of a player's resource store is equivalent to that of a FIFO (First In, First Out) queue. Hence, elements are added to one end of the queue (the *tail*), and taken off from the other (the *head*). The production unit however resembles a *stack* based on the LIFO (Last In, First Out) principle. Elements are added and removed only on one end. Thus, the game is called *Queue-Stack-Game*. One additional behaviour applies to the production units of this game. They can hold only one type of resources at a time and loose their previous content if new elements of a non-matching resource type are added.

Each round, every player is assigned a sequence of new resources, which are uniformly drawn from the available resource types. These elements are added in sequence to the tail of the queue. Next, a number of resources is taken off the head of the queue and added to the stack. As a consequence, the previous content of the stack might be lost if any of the new resources is of a non-matching type. To avoid this waste, players can negotiate with their peers and offer to give away resources of their queues. In doing so, they might be able to create sequences of identically typed resources of a certain length and thereby succeed in the game.

The following section decribes the rules and phases of the Queue-Stack-Game in detail.
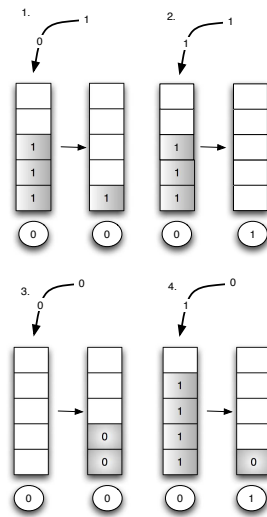
2.1 Production

There are a number of game parameters and restrictions that apply to the production process of the Queue-Stack-Game, which are listed here:

– Each agent can produce only one product at a time
– A product consists of a number of identically typed resources, this number being a game parameter, namely *stackCapacity*
– The types of resources and the order in which they are allocated to the producers are random. The number of resources each player receives per round is fixed though and is a parameter of the game, namely *getPerRound*
– The incoming sequence of resources cannot be altered by the agent before being added to the queue
– Each player is forced to input *pushPerRound* resources from the head of his queue into the production unit in each round
– If the type of any newly input resource does not match the type of the product being currently assembled, this product is spoiled and thrown away
– The players are admitted to remove elements of any types from their queue in order to give them to one of their fellow players
– If a player receives resources, he is allowed to arrange them in the desired order before they are immediately fed into the production unit

2.2 Allocation

Each round is divided into two phases, namely *allocation* and *negotiation*. In the *allocation* phase, *getPerRound* new random resources are enqueued in all players' resource stores. The resources allocated to the different players are independently generated. Subsequently, each agent is forced to remove the *pushPerRound*-first elements from the head of his queue and to push them onto the stack, maintaining their ordering. If

Fig. 1. Examples illustrating the behaviour of a player's stack when additional resources are pushed

the production unit already contains some elements and their type does not match the newly pushed resources, the old contents of the stack are wasted. Figure 1 illustrates four examples of feeding resources into the stack.

The examples show the state of the stack before and after new resources have been pushed. We assume two different types of resources, **0** and **1**. The number of game points owned in the current situation is shown underneath each stack. In situation (1) all elements of the stack are discarded when the **0** token is pushed, as the types do not match. The **0** token itself is also thrown away, when the next resource, a **1** token is pushed. In situation (2), the player has more luck. The two resources pushed complete the product, which the player can sell and thus is rewarded. The production unit is empty now, ready to accept new resources of any type. Situation (3) shows how resources are added to an empty stack. In example (4) the first of the pushed resources completes the stack, the player sells the completed product, earns a reward and the stack is emptied before the next resource is pushed.

2.3 Generating Possible Worlds

We will now formalise the notion of a state in the Queue-Stack-Game and outline the process of generating a set of possible worlds with respect to a particular state. A state s_c contains the following elements:

- The condition of the queue after resources have been removed, referred to as $queue(s_c)$
- the condition of the stack after transfer received from another player has been pushed, referred to as $stack(s_c)$,
- the number of rewards, $rewards(s_c)$,
- the set of resources received from another player, $get(s_c)$,
- the set of resources removed from the queue in order to be transferred to the other player, $give(s_c)$,
- $noWaste(s_c)$, a flag indicating whether elements of the stack were wasted when $get(s_c)$ was pushed,
- $earnedReward(s_c)$ a flag being set to 1 if a reward was earned when pushing $get(s_c)$ or 0 otherwise.

The queue of a state s_c can be generated by removing each possible subset of resources from the previous queue $queue(s_{c-1})$. The removed resources are $give(s)$. Which resources can be received from other players is not known to the agent, as he has no insight into his opponents' resource situation. So all possible combinations of resource types up to an arbitrary total amount are considered. As the resources can be pushed in any order, $get(s_c)$ is generated for each permutation of the received transfer. $stack(s_c)$ is the resulting stack, after $get(s_c)$ has been pushed. $rewards(s_c)$ is the number of rewards the agent possesses afterwards. $noWaste(s_c)$ and $earnedReward(s_c)$ are needed when calculating the utility for a state.

The deal that produced a state is implicit to the state. When we speak of the utility of a deal, we mean the utility of the state which results from execution of the deal.

2.4 Evaluating Possible Worlds – the Utility Function

We now need a numerical *utility function* which measures the quality of a state. A utility function u maps a state or a sequence of states to a real number [10]. The following criteria could be used to describe a "good" queue.

1. The more resources the agent possesses, the better.
2. Blocks of identically typed resources contained in the queue should be of maximum length; ideally, the length is a multiple of the number of resources needed to earn a reward.
3. Preferably, no elements of the stack should be wasted when resources are pushed.
4. Resources at the head of the queue which are to be pushed in the next round should carry more weight than resources at the back end of the queue.
5. As few resources as possible should be given to other players.
6. As many resources as possible should be received.

Equation 1 captures criteria 1 and 2. It computes the base utility for a state s. In any sequence of resources, each element is either of type **0** or **1**, or white and black, respectively. Single elements of identical type are indistinguishable. Resource sequences can thus be represented as a sequence of blocks containing identically typed resources. $b_i(r)$ is taken to denote the ith block of a sequence r. $amount(b_i(r))$ is the number of resources $b_i(r)$ contains and $type(b_i(r))$ denotes the type of resources in block $b_i(r)$. k denotes the number of blocks in r. $stackCapacity$ is the capacity of the stack, in other words, the number of resources required to obtain one unit of reward. The sequence of all resources that a player possesses is the concatenation of his stack and queue. Concatenation is represented by the "|" operator.

$$baseUtility(s) = \frac{1}{k} \sum_{i=0}^{k} \frac{amount(b_k(stack(s)|queue(s)))}{stackCapacity}$$

Each block is considered as a fraction of a complete stack. *stackCapacity* resources in a row are equivalent to one unit of reward. The equation computes the average reward that can be achieved.

Next, we will describe the course of one round of the Queue-Stack-Game. Each round is divided into two phases, namely *allocation* and *negotiation*. In the *allocation* phase, *getPerRound* new random resources are enqueued in all players' resource stores. The resources allocated to the different players are independently generated. Subsequently, each agent is forced to remove the *pushPerRound*-first elements from the head of his queue and to push them onto the stack, maintaining their ordering. For details on the allocation phase please see Appendix 2.2.

Having completed the *allocation* phase, the players enter the *negotiation* phase. The outcome of a successful negotiation is a *deal*, describing which sets of resources are to be exchanged between players. Hence, the agents engage in *practical reasoning*. The exchange of resources is the only means for agents to take action during the game. If a player chooses not to negotiate or not to agree to any deal proposed to him, his succeeding in the game entirely depends on the random resource sequence he is allocated. If players cannot find an agreement, the *default deal* is forced. The default deal entails no actions of the players, thus the resource situation of all players remains unchanged. The available locutions are *propose*, *reject*, *accept* and *inform*. The *negotiation protocol*, i.e. the communication rules are defined as follows:

1. The negotiation terminates immediately after an acceptance message is uttered by one of the participants.
2. The negotiation terminates with the default deal if a player quits the negotiation.
3. The players take turns in proposing deals. If a player cannot propose a new deal, he is forced either to accept a previously offered deal or to quit the negotiation.
4. All deals offered during the negotiation can be accepted at any point in time later on as long as they have not been rejected.
5. A counterproposal can be preceded by a critique and a rejection.

This protocol entails that agents have to receive up to three messages (*inform*, *reject*, *propose*) until they are allowed to respond.

After the outcome of the negotiation is set, the deal is executed. The resources each player receives from fellow players are pushed onto the stack, whereby the player himself can dictate the order in which they are to be pushed. Eventually, the players are rewarded if they were able to complete their stack and thus sold a product.

3 Three Approaches to the Game

3.1 Employing a Mediator

Our first approach to designing successful players involves the consultation of a trusted mediator. We assume the mediator does not take part in the game and is unbiased towards any of the players. The players truthfully reveal their resource situation and their utility function to the mediator. The mediator has thus perfect information of the players' private states. Using this knowledge, all possible *offers* per player can be computed. Here, offer refers to a subset of the queue which the owner offers to give to an fellow player. The space of all possible deals is thus the Cartesian product of each player's offer vector. Through the utility function each player assigns a utility value to each possible deal. By knowing the utility functions, the mediator can compute these values per deal and player.

The next task is to determine the *optimal* deal for both agents. We adapt the axioms of the Nash Bargaining Solution [7] to define optimality: *Pareto efficiency* (there is no other deal which improves the payoff of at least one agent without another agent being worse off), *Invariance* (utility functions only represent preferences over outcomes, the actual cardinalities of the utilities do not matter), *Independence of irrelevant alternatives* (if outcome o is the solution and other outcomes $o' \neq o$ are removed from the set of all possible outcomes, o still remains the solution) and *Symmetry* (the optimal solution remains the same as long as the set of utility functions is the same. Which player has which utility function does not influence the outcome.) According to the *Nash Bargaining Solution*, the optimal deal o^* is the deal that maximises the product of the players' utilities. Formally:

$$o^* = \arg\max_o [(u_1(o) - u_1(o_{default})) \times (u_2(o) - u_2(o_{default}))]$$

where n is the number of players and $u_i(o')$ is the utility which player i assigns to deal o'. The mediator chooses the deal from the set of all generated deals that satisfies this equation. He then proposes this deal to the players, whom we assume to accept.

The advantage of the mediator approach is obvious. The outcome is guaranteed to be Pareto efficient. Hence, it is impossible to find a deal where both players are better off. The Nash Bargaining Solution respects each player's interests as far as possible without being biased towards any particular player, and promotes fairness.

This approach has some shortcomings though. First of all, it requires the existence of a mediator whom the players trust, so they will reveal their utility functions and their resource situation. If the players are concerned with privacy issues in general and if they do not trust the mediator they might not agree to collaborate with that mediator. Furthermore, they might not be content with the solution found, because there are deals with which the individual would be better off. The individual players are not necessarily

interested in maximising the social welfare [4] but are only concerned with maximising their own profit. Additionally, the realisation of a mediator can be very complex and inefficient in real world scenarios.

The mediator will serve as a benchmark to which we compare the negotiation outcomes achieved by the argumentation-based agent described in the following sections.

3.2 Proposal- and Argumentation-based Negotiating Agents

In this section we describe two designs of agents, both capable to negotiate by exchanging proposals with negotiation partners. While the proposal-based negotiating agent's abilities are restricted to the exchange of proposals, the argumentation-based negotiating (ABN) agent can use arguments to *justify* his negotiation stance and to *critique* proposals he has received from fellow players. Arguments can be arbitrary logical formulae with literals taken from a given vocabulary.

Agent Architecture Overview The architecture of our argumentation-based agent follows the abstract architecture described in [8]. All incoming proposals are stored in a Proposal Database. If arguments are employed, these are stored as well. As a model of his environment, the agent maintains a set of possible worlds to which he will possibly agree. This set is continuously adapted during negotiation, i.e. possible worlds are removed after arguments or rejections of his proposals have been received and evaluated. According to his negotiation strategy, the agent then decides whether to accept or reject the last proposal. The ABN agent can then generate different types of arguments [3], in our case either a critique or a justification to inform his opponent why he is not inclined to accept the proposal. Of all generated arguments one is selected which will be uttered as a response. According to the adapted negotiation protocol (see Section 2.4), the agent cannot reply to every message received, but is bound to wait until he receives a proposal. So, not every incoming locution triggers an outgoing locution. The next sections describe the main components of the agent architecture in detail.

The Negotiation Strategy In this section, we describe the negotiation strategy both our agents pursue. Each agent generates a set of possible deals which he will propose to his opponent one after another, starting with the deal with highest utility, followed by deals with descending utility. This ensures that the opponent knows all deals which would yield higher utility for the proposing agent, before being given the chance to accept a new deal. On the other hand, an agent waits to accept until he is not able to make a proposal with higher utility himself. Thus, the use of this strategy aims to maximise the utility of the outcome for both players. Algorithm 1 shows the strategy of an agent using arguments in pseudo-code notation. Removing lines 14 to 18 yields the strategy of our proposal-based agent, who simply accepts or rejects deals without criticising them. Algorithm 1 is executed by the agents in every round of the game to determine the next locutions in the negotiation.

Deals received from the negotiation partner carry a j subscript, own proposals an i subscript. After agent j has offered deal $\delta_{j,r}$ to agent i in round r, agent i computes the best deal he is able to propose $\delta_{i,r+1}$ as a counterproposal (line 2). This is the deal with

the highest utility based on the current resource situation, which has not been offered yet. Additionally, the deal with higest utility of all deals received from agent j in earlier rounds ($\delta_{j,best}$) is determined (line 3). If no proposable deal could be found agent i terminates negotiation. Either by accepting $\delta_{j,best}$ if executing this deal improves the utility compared to the current situation in round r or by accepting the default deal and thus leaving the resource situation unchanged. If the best proposable deal $\delta_{i,r+1}$ has lower utility than the best deal received already $\delta_{j,best}$, agent i accepts this $\delta_{j,best}$. Otherwise, the agent has incentive to pursue negotiation and proposes the best deal possible $\delta_{i,r+1}$ (line 22). Furthermore the agent will reject the latest offer if it is not the deal with highest utility of all offers received so far (lines 19 to 20). Lines 14 to 17 show the generation of all possible arguments concerning the latest offer and selection of the best argument which is then uttered before making the counterproposal $\delta_{i,r+1}$.

In summary, it can be stated that the agent will accept the deal with the highest utility of all deals he was offered (*bestDealReceived*) when all deals left to propose have lower utility. He will withdraw from the negotiation and thus accept the default deal if he cannot make any more proposals, but has not received any offer whose utility exceeds that of the current situation. An explicit reject is stated with respect to the current offer if there is already another offer with higher utility.

Algorithm 1 Negotiation strategy

1: receive $\delta_{j,r}$
2: $\delta_{i,r+1} \leftarrow$ bestDealPossible()
3: $\delta_{j,best} \leftarrow$ bestDealReceived()
4: **if** $\delta_{i,r+1} = \perp$ **then**
5:     **if** utility($\delta_{i,default}$) $\geq$ utility($\delta_{j,best}$) **then**
6:         ACCEPT $\delta_{i,default}$
7:     **else**
8:         ACCEPT $\delta_{j,best}$
9:     **end if**
10: **else**
11:     **if** utility($\delta_{i,r+1}$) $<$ utility($\delta_{j,best}$) **then**
12:         ACCEPT $\delta_{j,best}$
13:     **else**
14:         allArguments[] $\leftarrow$ generateArguments($\delta_{j,r}$)
15:         **if** allArguments[] $\neq\perp$ **then**
16:             argument $\leftarrow$ selectBestArgument(allArguments[])
17:             INFORM argument
18:         **end if**
19:         **if** utility($\delta_{i,r}$) $<$ utility($\delta_{j,best}$) **then**
20:             REJECT $\delta_{j,r}$
21:         **end if**
22:         PROPOSE $\delta_{i,r+1}$
23:     **end if**
24: **end if**

Generating and Selecting Arguments Next, we explain how argument generation (line 14, "generateArguments") and argument selection (line 16, "selectBestArgument") is managed.

The negotiation language we designed contains just two basic elements. On the one hand, the statement $quit\_negotiation(agent)$, which an agent utters if he stops negotiating. On the other hand, $give(a, b, r, t)$ where a and b are agents, r is an amount of resources and t denotes a round of the game. The semantics of this statement is that agent a gives the resources r to agent b in round t. By combining statements using logical connectives, it is possible to create complex expressions with varying meaning. A deal, as it describes the exchange of resources between two players, consists of the conjunction of two statements:

$$give(a, b, r1, t) \land give(b, a, r2, t)$$

Arguments can serve two purposes in our approach: justification ("I cannot provide you with six white resources in the current round 13, because I only have four") or critique ("I reject your offer to give me four whites in exchange for three blacks in the current round 7, because I do not want to get four whites at all"). Each proposal is hence examined as to whether it contains one or more actions which either cannot be performed or are not desirable. An action is deemed not desirable if it is not contained in any deal considered in the agent's store of possible worlds. The argument generated then consists of the conjunction of the negated actions.

Here are two arguments which agent a sends to agent b. The following example corresponds to the above justification:

$$\neg give(a, b, four Whites, 13)$$
$$\land \neg give(a, b, five Whites, 13)$$
$$\land \neg give(a, b, six Whites, 13)$$

It states that the agent cannot give four, five or six white resources. A possible critique could be $\neg give(b, a, four Whites, 7)$. Agent a does not want to receive four white resources under any circumstances.

The question of which of all arguments generated is to be uttered is answered by one simple rule: If a justification was generated and it has not yet been uttered, it will be selected. Otherwise, the critique is selected.

Interpreting Arguments Now we will address the issue of how to evaluate incoming arguments. Arguments are statements about the opponent's mental attitude, i.e. his beliefs about possible worlds. Consisting of formulas on propositions of the form "*give(a, b, r, t)*", they describe the set of deals he might be willing to accept at all. Hence they are used to refine the set of possible offers. We assume our agents to be honest, so arguments are believed to be true. If an argument is received, its interpretation with respect to the current set of possible worlds is determined. The interpretation is a subset of the universe (the current set of possible worlds). This subset is then regarded as the new set of possible worlds. A set of possible worlds can be regarded as a logical formula of the

form

$$\underbrace{give(a, b, r_1, t) \land give(b, a, r_2, t)}_{deal_1} \lor \ldots$$

$$\ldots \lor \underbrace{give(a, b, r_n, t) \land give(b, a, r_m, t)}_{deal_k}$$

The r_i stand for arbitrary sets of resources. A r_i can appear in several deals.

Incoming arguments are transformed into a normal form, so that negations are pushed inward and all operators but $\lor$ and $\land$ are resolved. Then the subset of possible deals, which is denoted by the argument, is determined using the following inductive definitions in Table 1 where ϕ and ψ are any arguments.

Table 1. Interpretation of arguments as subsets of possible worlds

Argument	Interpretation
$give(a, b, r, t)$	set of all deals which include $give(a, b, r, t)$
$\neg give(a, b, r, t)$	set of all deals which do not include $give(a, b, r, t)$
$\phi \land \psi$	all deals which are elements of the intersection of the sets which are the interpretation of ϕ and ψ
$\phi \lor \psi$	all deals which are elements of the union of the sets which are the interpretation of ϕ and ψ

4 Evaluation

This section describes the empirical evaluation conducted to answer our central research questions: Do agents who use arguments in the negotiation perform better in our complex trading scenario than agents who are confined to exchanging proposals? Are agents using argumentation-based negotiation capable of reaching optimal deals? In the first section, we introduce the evaluation criteria for which data was gathered during the test runs. Section 4.2 describes the experimental setup. Finally, in Section 4.3, we evaluate our experimental findings with respect to our key research questions.

4.1 Evaluation Criteria

In our experimental evaluation, we consider a number of evaluation criteria which allow for measuring certain aspects of agent performance. The following subsections introduce these criteria one by one. Moreover, we outline why and to what extent we consider the criteria to be suitable metrics for evaluation.

– Rewards: Rewards earned over time are the most natural criterion for the Queue-Stack-Game, as this measure is used to determine the winner after one or more

rounds, and hence it also reflects the game-playing ability of any given agent strategy. The rewards a player has earned are influenced mainly by two factors. On the one hand, they depend on how lucky the player has been in terms of the resources that were randomly allocated to him. On the other hand, their number increases with the quality of the game strategy adopted, and in particular also with the quality of the chosen negotiation strategy. Hence, it is important to have several rounds in one game, so that the distribution of resources becomes fair.

– Social Welfare: Social welfare is a means of assessing the well-being of a society of agents as a whole, i.e. taking into account the well-being of all individual agents [1]. In literature, there are different measures for social welfare, e.g. *Egalitarian* or *Utilitarian* social welfare. We employed the Nash Bargaining Solution which is proven to promote pareto-optimal deals, which means that no other deal is preferred by every other agent. The mediator computes the optimal deal according to this measure. The deals achieved by each of the negotiation methods can be compared to this optimal deal and thus a "degree of optimality" can be established for each method. Furthermore, the deals achieved by ABN and by bargaining can be compared to each other with respect to the optimal deal. Social welfare is also an adequate indicator if all agents get better deals by arguing, or if, e.g., an increase of utility of Player1 can only be realised at Player2's expense, thus promoting unjust deals.

– Number of Communication Units: As a measure for the amount of communication, we define a communication unit for our negotiation language. Deals and arguments are generated by combining different statements of the form $give(a, b, r, t)$. We define this as one communication unit. Operators are not accounted for by this measure, i.e. a conjunctive expression has the same "value" as two atomic expressions. Similar to the other measures discussed above, the absolute number of communication units is meaningless in itself. However, counting the communication units allows for a comparison of the amount of communication across the different negotiation mechanisms. In the ABN approach, we distinguish communication units that were sent as part of proposals and messages that carried arguments.

– Number of Negotiation Steps: The number of steps the agents negotiate for per round is an apt measure to determine the speed of a negotiation style. Reaching an agreement in fewer steps is considered better, if the agreement is as good as the agreement that could have been reached within an unbounded number of negotiation rounds. Even if no agreement is reached, the less time and effort is invested to find out that no co-operation is possible or desired, the better.

– Number of Possible Worlds: Our agents maintain a changing number of possible worlds during negotiation. The absolute number of possible worlds is not interesting *per se*. Their number is highly dependent on the current resource situation of the agent, and can initially be very large. What we really want to show is that our agents are capable of reducing the number of possible worlds while negotiating. Examining the decrease in possible worlds over negotiation steps offers valuable clues on how the negotiation outcomes are achieved. This is because the number and the utility of the remaining possible worlds directly influence an agent's decision to accept or reject an offered deal.

Table 2. Frequency of data collection.

Test Criterion	Mediator	Player
Nash Bargaining Solution	end of round	n/a
Utilitarian Social Welfare	end of round	n/a
Egalitarian Social Welfare	end of round	n/a
Rewards	n/a	start and end of round
Utility	n/a	start and end of round
Communication Units (Proposals)	n/a	end of negotiation
Communication Units (Arguments)	n/a	end of negotiation
Negotiation Steps	n/a	end of negotiation
Possible Worlds	n/a	each negotiation-step

4.2 Experimental setup

In the following, we describe the tests that were run to generate the test data. The Queue-Stack-Game was played in three different settings. Scenario one comprises a meditator as described in Section 3.1 in addition to two players. In the second scenario, players can exchange deals but not arguments. Finally, in the third scenario, players are capable of exchanging arguments in addition to proposals. Ten consecutive games consisting of ten rounds per game are played in all three scenarios. The sequences of resources which are allocated to each player from the deck in each round are identical in all three settings.

What is left to chance is the random time the agents wait before uttering their initial proposal as soon as their *NegotiationBehaviour* is started. In summary, the course of the game in our three settings is solely dependent on the negotiation and its outcome. Hence, we can draw conclusions from the players' success in the game to their ability to negotiate.

The data for the empirical evaluation of the scenarios is logged by the players and the mediator, if existent, during the test runs. The criteria are logged with different frequency and in different phases of the game. Table 2 provides an overview of all logged test variables.

4.3 Evaluation of Experimental Results

In this section we evaluate and interpret our experiments with respect to the above mentioned test criteria.

Rewards Earned rewards measure an agent's success in the Queue-Stack-Game. The graphs in Figure 2 show the number of rewards earned by players Player1 (top) and Player2 (bottom) in every round of the game, respectively. The recurring decline of rewards is due to the start of a new game every ten rounds. That is, the agents are restarted with zero rewards in every eleventh round.

Figure 3 shows the average reward per round earned by both players in different test scenarios. Both agents perform best when guided by a mediator, which matches our initial expectations. Comparing the scenarios where the agents actually negotiate
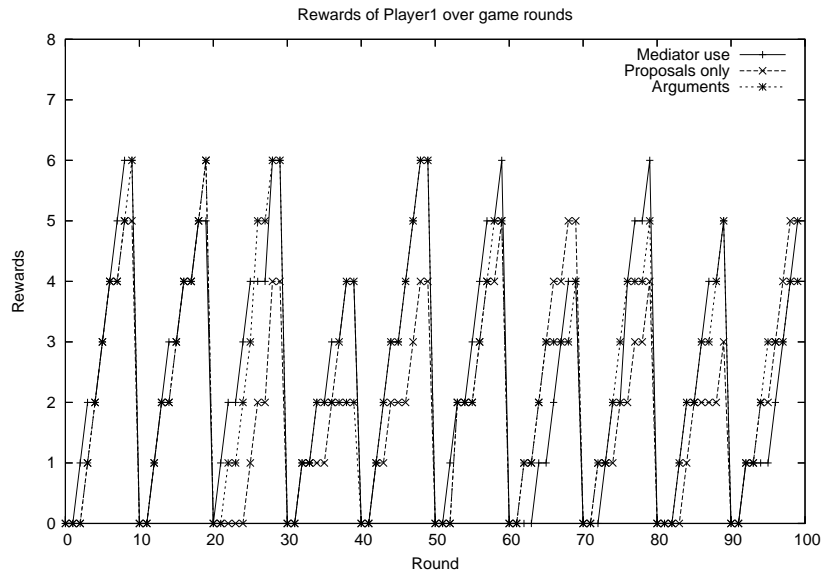
Fig. 2. Earned rewards over game rounds of Player1 for different test scenarios

with each other shows that both agents achieve better results when using arguments in addition to the proposal exchange.

Social welfare Figures 4 depict the average social welfare for each experimental setting, namely "Argumentation", "Proposal exchange" and "Mediator use". Different measures for social welfare were used, the most common being Utilitarian social welfare (sum of each player's payoff) and the Nash product (product of each player's payoff). We computed social welfare on the basis of actual rewards, not based on the utility of the negotiation outcome for reasons described above. As a matter of design, the mediator maximised the Nash product of the players' utilities, which are a heuristic for the expected reward. Quite naturally, this lead to the best average game results compared to the other mechanisms, considering any of our suggested measures. Likewise, it becomes apparent that the agents of the "Argumentation" scenario achieved the second best results and thus performed better than negotiating agents who were restricted to proposal exchange.

Table 3 summarises the percentage of games won by each player. The number of successful negotiations which ended with an agreement can be increased by 19,7 % from 66 to 79 of 100 by the use of arguments. Whereas in scenario 2 both agents accepted equally often, Player1 ended 9 more negotiations with an acceptance in scenario 3 whereas Player2 accepted in just two more negotiations.

Possible Worlds The agents in scenario 2 and 3 mainly differ in the way an agent's set of possible worlds is maintained. The exchange of arguments aims at the refinement
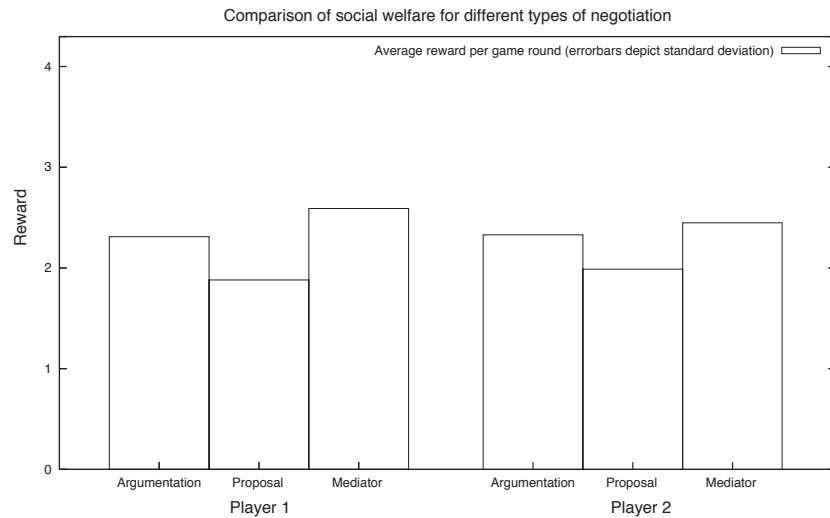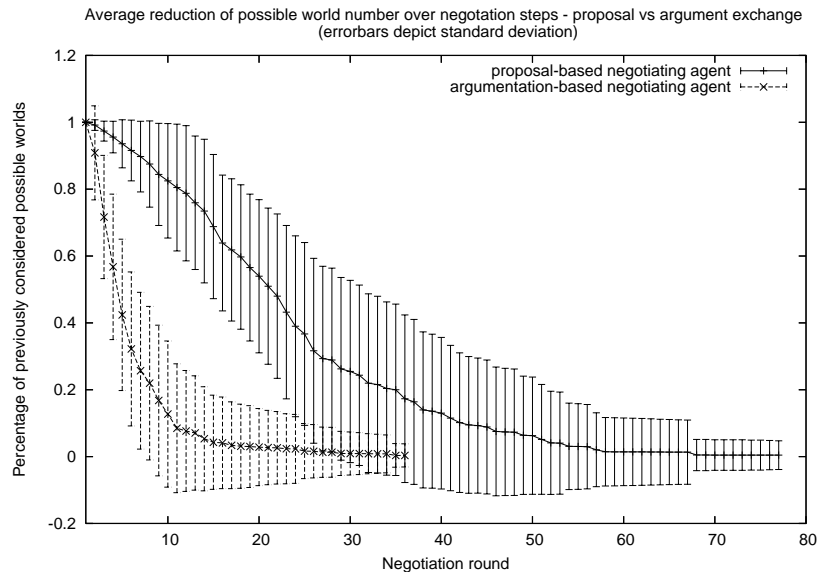
Fig. 3. Average rewards earned per game in different test scenarios. Left: Player1, Right: Player2
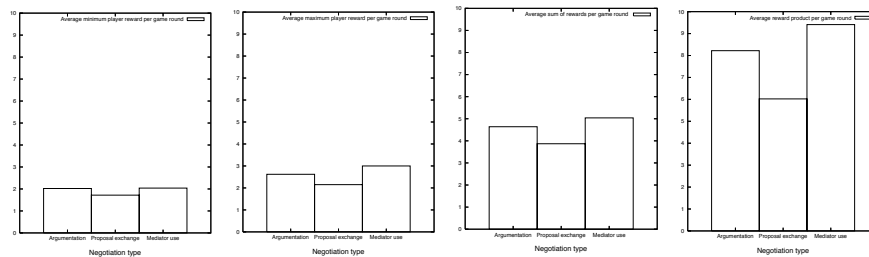
Fig. 4. Comparison of average social welfare based on earned rewards for "argumentation" (left), "proposal exchange" (middle), "mediator use" (right). Measures for social welfare (from top to bottom, left to right): minimum (egalitarian social welfare), maximum (elitist), sum (utilitarian social welfare), product (Nash product)

Table 3. Acceptance rates of negotiation scenarios

Scenario	Player1	Player2	No agreement
2: Proposals only	32%	34%	33%
3: With arguments	43%	36%	21%

Average reduction of possible world number over negotation steps - proposal vs argument exchange
(errorbars depict standard deviation)

Fig. 5. Player2: Average reduction of possible world number over negotation steps in scenario"proposal exchange" (continous errorbars) and "argument exchange" (dashed errorbars)

of the set of possible worlds, and thus the removal of worlds that are not acceptable to any of the agents. Hence, we look at how the number of possible worlds changes over negotiation rounds.

In Figure 5 the average decrease of possible worlds over rounds is plotted for Player2 for the two negotiation scenarios. The curve for agent Player1 is almost identical, we therefore omit it.

Comparing the plots of the two different test scenarios, one observation is obvious: The decrease of possible worlds proceeds much faster when arguments are used. After ten negotiation steps, agents in scenario 2 still maintain more than 80% of the worlds they initially considered possible on the average. By that time, agents in scenario 3 have removed over 80% of their initial worlds and maintain only less than 20% after ten steps. After termination of the negotiation process, ABN agents have eliminated about 65% of their initially possible worlds on the average, their counterparts in scenario 2 have been able to remove a mere 42%.

Negotiation Steps Considering the average number of negotiation steps, agents of the diferent scenarios needed to come to an agreement, the average of 39 steps of the scenario with argumentation lies clearly under the average of 76 of the scenario where only proposals are exchanged, see Figure 4.3. This means that when using arguments the negotiation terminates after little more than half of the steps required using pure proposal exchange in the average. This is due to the faster decrease of possible worlds in scenario 3, and it is also due to the fact that once the negotiation has started no arguments can be produced which entail an increase of possible worlds. In the few
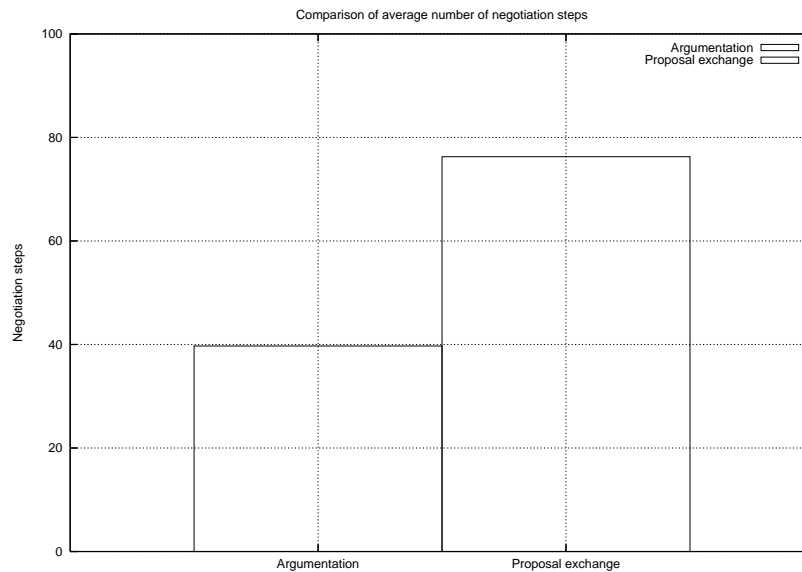
Fig. 6. Average number of negotiation steps: "argumentation" (left), "proposal exchange" (right)

cases where the ABN agents need more steps to come to an agreement, this can be still be justified by the better negotiation outcome these agents achieve compared to their proposal-exchanging counterparts.

Communication units The total number of communication units (in the sense defined above) averages 113.12 in scenario 2 and 44.01 in scenario 3. Latter number is composed of 12.87 units used on arguments and 31.14 units describing outcomes. Even the sum of proposals and arguments in scenario 3 does not get close to the average of units exchanged for proposals in scenario 2. This result is obtained by the richer semantics of the language which is used for argument exchange. The use of logic allows for using concise descriptions of subsets of possible worlds. If the negotiation language is restricted to deals and a set of possible deals is to be encoded, there is no alternative to enumerating the elements of this set. As the elements are deals and each deal equals two communication units, the number of units needed to encode a set is twice the cardinality of the set. Using logic this number still constitutes the worst case, but due to dependencies between different deals which contain identical actions, a subset of possible worlds can usually be encoded with fewer communication units. Hence, the use of logic as negotiation language allows for the reduction of communication overhead.

4.4 Summary

In this section we introduced the evaluation criteria that were considered in the conducted experiments. We then described the experimental setup for the data generation.
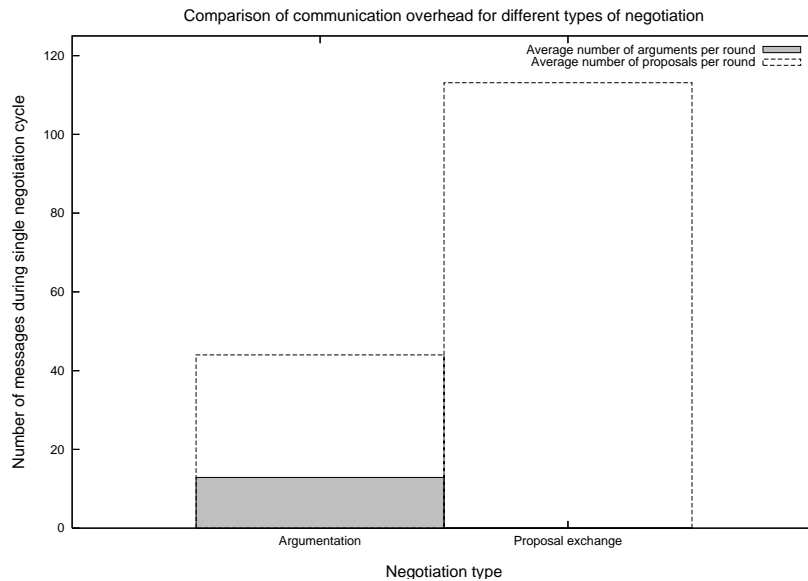
Fig. 7. Average number of communication units sent during negotiation

Furthermore, we were able to show through a thorough analysis of experimental results that additional use of arguments during negotiation in the Queue-Stack-Game not only drastically reduces the duration and communication overhead of negotiation, but also that the quality of the achieved agreements is higher (in the sense that the resulting deals cause a higher increase in agents' payoffs and thus they perform better in the game). This is due to the refinement of the set of possible worlds by exchanging arguments which accompany the rejection of deals. Not only is the actually rejected deal eliminated from the opponent's set of possible worlds, but so is also every deal that shares the undesired aspects that caused the rejection of the explicitly proposed deal.

5 Conclusion and Future Work

In this work we presented the implementation of three different negotiation mechanisms in an environment which is only partially observable, i.e. the state and the preferences of an agent's peer are not known to him, and which incorporates stochastic elements, i.e. subsequent states are not solely dependent on the actions which are carried out by agents. Two agents are randomly assigned resources which they can use in a specific manner to earn rewards. In most cases agents need to exchange resources with their opponent to be successful. Hence, the agents need to come to an agreement about which type of resources and how many of them they want to exchange. We approached this problem from three different angles.

Our first solution was the employment of a trustworthy mediator, toward whom the agents disclose their preferences and resource situation. Using this complete information, the mediator can compute the optimal solution and dictate the outcome of the

negotiation. Our second solution comprised agents who engaged in bargaining. They were restricted to a simple exchange of proposals to come to an agreement. Then, in our third scenario we provided the negotiating agents with the additional capability to accompany proposals or rejection of proposals with arguments. These arguments can either be a detailed critique of an previously received proposal, telling the opponent exactly which aspects of the proposal are undesirable. Or, an argument can carry information about the sender's negotiation stance and thus explain why a proposed deal cannot be fulfilled by the sender.

We extensively tested these solution concepts in identical experimental settings, allowing for a detailed comparison of the performance of the three negotiation mechanisms. As expected, agents perform best when consulting a mediator. When actually engaging in negotiation with each other, the use of arguments proves beneficial in various ways. Not only decrease communication overhead and duration of negotiation significantly, but agents simultaneously reach better agreements. Hence we were able to verify our working hypothesis, that the use of arguments enables better deals in an generic example scenario with partial, incomplete knowledge compared to negotiation that is purely based on proposal exchange.

A number of aspects could not be addressed and were beyond the scope of this paper. The following is a list of issues that could be the basis for future research:

- Although agents receive information about the internal state of their opponent, they do not actually try to create a model of their opponent, which they could use over several rounds. This aspect gains importance if agents are allowed to cheat. From the offers the opponent has made his resource situation could at least be inferred partially or inconsistencies in his offers and arguments could be detected.
- In the light of potentially agents that are not trustworthy it is necessary to reassess the process of argument evaluation. If the truthfulness of the arguments cannot be taken for granted, it is not advisable to accept all implications of the arguments without examining whether one believes the argument or not.
- Commitment to future actions is not considered as potential part of an agreement, even though the design of the negotiation language would allow it.
- Our agents have not been equipped with the ability to learn or plan, two essential aspects of intelligent agents.
- Also the use of the mediator could be reassessed. Players might not be required to execute the deal they have been advised to perform. They could bear that deal in mind and engage in negotiation nonetheless, leaving open which agreement they will pursue. Again, this problem is within the scope of research on computational trust.

By our strong efforts to keep our implementation generic in the choice of the tools and design we hope to contribute to the further investigation of these important issues.

References

1. J. Eatwell, M. Milgate, and P. Newman, editors. *The New Palgrave: A Dictionary of Economics*, volume 2. Macmillan, London, 1987.
2. N. R. Jennings, S. Parsons, P. Noriega, and C. Sierra. On argumentation-based negotiation. In *Proceedings of the International Workshop on Multi-Agent Systems*, Boston, USA, 1998.
3. S. Kraus. Automated negotiation and decision making in multiagent environments. In *EASSS*, pages 150–172, 2001.
4. A. R. Lomuscio, M. Wooldridge, and N. R. Jennings. A classification scheme for negotiation in electronic commerce. *Lecture Notes in Computer Science*, 1991, 2001.
5. N. Maudet, S. Parsons, and I. Rahwan. Argumentation in multi-agent systems: Context and recent developments. In *Proceedings of the AAMAS International Workshop on Argumentation in Multi-Agent Systems (ArgMAS), Hakodate, Japan, Lecture Notes in Artificial Intelligence, Volume 4766*. Springer, 2007.
6. C. Murray and G. Gordon. Multi-robot negotiation: Approximating the set of subgame perfect equilibria in general sum stochastic games. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*. MIT Press, Cambridge, MA, 2007.
7. J. F. Nash. The bargaining problem. *Econometrica*, 18(2):155–162, 1950.
8. I. Rahwan, S. Ramchurn, N. Jennings, P. McBurney, S. Parsons, and L. Sonenberg. Argumentation-based negotiation, 2004.
9. I. Rahwan, L. Sonenberg, and P. McBurney. Bargaining and argument-based negotiation: Some preliminary comparisons. In *Proceedings of the AAMAS Workshop on Argumentation in Multi-Agent Systems*, New York, 2004.
10. S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003.
11. T. W. Sandholm. Distributed rational decision making. In G. Weiß, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, pages 201–258. MIT Press, Cambridge, MA, USA, 1999.
12. X. Wang and T. Sandholm. Reinforcement learning to play an optimal nash equilibrium in team markov games. In S. T. S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 1571–1578. MIT Press, Cambridge, MA, 2003.
13. G. Weiß, editor. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1999.