Contract-based Negotiation via Argumentation (preliminary report)

Daniela Carbogim and David Robertson
Division of Informatics, University of Edinburgh
80 South Bridge, Edinburgh, UK, EH1 1HN
{danielac,dr}@dai.ed.ac.uk

Abstract

Negotiation is often described as the process by which agents come to a mutually acceptable agreement about some subject. This definition is quite broad, as this process can be viewed from many different angles. Here we consider negotiation from the perspective of contracts. We present a general model of contract-based negotiation and propose a logic programming-based argumentation framework to capture and formalise this model.

Keywords: contract-based negotiation, argumentation, logic programming, autonomous agents.

1 Introduction

Negotiation is often described as the process by which agents come to a mutually acceptable agreement about some subject [12, 13, 17]. This definition is very broad and this process can be considered from many different angles. In particular, there has been a recent interest in applying argumentation systems to capture negotiation processes, as negotiation often involves the exchange of arguments between agents.

In [13], a negotiation framework was developed based on the Logic of Argumentation [9] to deal with conflicting objectives and to coordinate joint approaches between autonomous agents. Negotiation was defined as a protocol for the exchange of messages containing proposals, counter proposals, critiques and explanations.

This is one view of the problem, which is particularly useful in forming agreements between agents. However, we are interested in another aspect of negotiation, which is concerned with adjusting the terms of an agreement rather than forming an agreement. This is the main difference between contract-based and protocol-based negotiation approaches. In many cases, agreements are regulated by contracts. Contract-based negotiation processes allow the terms of contracts to be adjusted according to the agents' internal theories.

In [16], Sierra et al. proposed a model of negotiation based on contracts, which were represented as collections of issues (variables) whose values needed to be set. Through negotiation, an agent would propose values within a range that was acceptable to it in order to obtain an assignment that would suit every participant. In our work, however, we focus on contract-based negotiation from a broader perspective. Contracts are defined as *objects* that can be adjusted based on reasoned arguments by the agents involved in the agreement. This type of negotiation actually corresponds to the way humans deal with contracts in many cases.

The contributions of this paper are presenting a general model of contract-based negotiation (section 2) and proposing a logic programming based argumentation framework (section 3) to capture and formalise this model (section 4).

2 A Model of Contract-based Negotiation

In this section we characterise the type of negotiation in which we are interested: contract-based negotiation. We assume here that contracts are objects which regulate agreements between two autonomous agents – a *consumer* (or client) and a *producer* (or server) – about the supply of a particular product or service. The purpose of contract-based negotiation is to adjust the terms of this agreement so that it is acceptable for both parties involved. Note that, even though this type of negotiation process is carried out by exactly two parties, an agent can be involved in different processes simultaneously. Also, an agent can play distinct roles in distinct processes, being a producer in some contexts and a consumer in others.

The process of contract-based negotiation can be described as follows. Initially, one of the parties proposes a binding contract to regulate the agreement between them; without loss of generality, we can assume that the producer makes this first proposal. This contract is now the *object of negotiation* between producer and consumer. It can be seen as a set of formulae stating the conditions for accomplishing the agreement.

The consumer receives the contract from the producer and analyses it. If it agrees with the clauses, then the process of negotiation is over. More interestingly, the consumer might have reasons to believe that this particular contract will not be successfully completed. In this case, the consumer sends it back to the producer with the appropriate criticisms. The producer then tries to adapt some of the clauses in that particular contract in order to make it more acceptable, sending it back again to the consumer for further analysis.

The process of adjusting the contract continues until there are no more criticisms (i.e. it is acceptable for both producer and consumer) or until one of the parties withdraw. This process is similar to the kind of negotiation that humans perform in many situations involving contracts.

2.1 An Example

Let us consider an example to illustrate the generic contract-based negotiation model described above. Let a, b be autonomous agents such that a has agreed to supply a particular product p to b. A contract-based negotiation process will be carried out by a and b in order to adjust the clauses of this agreement so that it is acceptable to both parties.

Being the producer, a proposes an initial contract to b stating the conditions for success or failure of the arrangement between them. These conditions might include the delivery of the product by a itself within the stipulated time and the appropriate payment for it by the consumer b. If we assume an underlying Horn-clause resolution-based system, as well as a model of time based on integer time points, such contract could be represented as the set of formulae below.

$$
\begin{array}{l}
\mathit{contract\_completion}(X,P,D) \leftarrow \\
\quad \mathit{producer}(X,P) \wedge \\
\quad \mathit{consumer}(Y,P) \wedge \\
\quad \mathit{delivered}(X,Y,P,D) \wedge \\
\quad \mathit{delivery\_date}(X,Y,P,D_d) \wedge \\
\quad D \leq D_d \\
\quad \mathit{appropriate\_payment}(Y,X,P) \\[6pt]
\mathit{contract\_failure}(X,P,D) \leftarrow \\
\quad \mathit{producer}(X,P) \wedge \\
\quad \mathit{consumer}(Y,P) \wedge \\
\quad \mathit{delivered}(X,Y,P,D) \wedge \\
\quad \mathit{delivery\_date}(X,Y,P,D_d) \wedge \\
\quad (D > D_d \vee \mathit{not\_appropriate\_payment}(Y,X,P)) \\[6pt]
\mathit{delivered}(X,Y,P,D) \leftarrow \\
\quad \mathit{delivery\_date}(X,Y,P,D) \\[6pt]
\mathit{delivery\_date}(X,Y,P,D) \leftarrow \\
\quad \mathit{starting\_date}(X,Y,P,D_s) \wedge \\
\quad \mathit{production\_time}(X,P,T_p) \wedge \\
\quad D \ \mathit{is} \ D_s + T_p
\end{array}
$$

$$
\begin{array}{l}
\mathit{appropriate\_payment}(Y,X,P) \leftarrow \\
\quad \mathit{pay}(Y,X,P,V) \wedge \\
\quad \mathit{cost}(X,Y,P,V) \\[6pt]
\mathit{not\_appropriate\_payment}(Y,X,P) \leftarrow \\
\quad \mathit{pay}(Y,X,P,V_p) \wedge \\
\quad \mathit{cost}(X,Y,P,V) \\
\quad V_p < V \\[6pt]
\mathit{pay}(Y,X,P,V) \leftarrow \\
\quad \mathit{cost}(X,Y,P,V) \\[6pt]
\mathit{cost}(X,Y,P,V) \leftarrow \\
\quad \mathit{price}(X,P,V) \\[6pt]
\mathit{producer}(a,p) \\
\mathit{consumer}(b,p) \\
\mathit{starting\_date}(a,b,p,1) \\
\mathit{production\_time}(a,p,6) \\
\mathit{price}(a,p,10)
\end{array}
$$

For a contract to be acceptable to an agent we mean that it has reasons to believe that the contract will be successful and does not have reasons to believe that it will fail. That is, from the contract we can only conclude *contract_completion*, but not *contract_failure*, and the agent cannot think of any other issue that could cause failure to be derived. In this sense the contract above is acceptable to the producer a because it is consistent with its internal theory (a proposed it) and from it we can only derive *contract_completion*.

Note that the predicate *contract_failure* corresponds to the negation of *contract_completion*. In fact, here we do not represent negation explicitly, but in terms of intuitively contradicting predicates. An advantage of doing so is that we do not commit to classical negation properties, such as the *excluded middle* or the *non-contradiction* principles, allowing conflicting arguments to be derived.

The producer a then sends this contract to the consumer b, who will investigate within its internal theory if there are any grounds for deriving *contract_failure*. For instance, b might believe that a will take longer to actually deliver the product p, so it adds the following clause to the contract.

$$\begin{array}{l} delivered(a, b, p, D) \leftarrow \\ \qquad starting\_date(a, b, p, D_s) \wedge \\ \qquad production\_time(a, p, T_p) \wedge \\ \qquad D \ is \ D_s + T_p + 5 \end{array}$$

The clause above can be seen as a criticism to the initially proposed contract. By adding this formula to it, we can also conclude that the contract will fail. Clearly this is an undesirable situation. The consumer b sends this "extended" version back to a, who tries to reconcile the criticism with the original clauses. It does so by accommodating its own preferences and internal beliefs in order to allow only *contract_completion* to be derived.

A new version of the contract will then feature changes based on b's criticism. In this particular case, a can adapt the clause that specifies the delivery date of product p in order to cope with an expected delay in delivery.

$$\begin{array}{l} delivery\_date(a, b, p, D) \leftarrow \\ \qquad starting\_date(a, b, p, D_s) \wedge \\ \qquad production\_time(a, p, T_p) \wedge \\ \qquad D \ is \ D_s + T_p + 5 \end{array}$$

Note that the clause above specifically addresses the delivery of product p by a to b, as opposed to the more generic formula proposed initially. The latter may still be valid in general, but not in this particular case, where the more specific clause is preferred.

In this way, a guarantees that the conditions for failure are no longer satisfied. Now a sends this version back again to b, who can either agree with it or provide some more reasoned criticism.

Suppose now that b believes that it should have a discount (e.g. of 20%) on the price of p. It states this by adding the following clause to the contract, which says that b will pay a smaller amount for this product than the actual price established by a. I.e., according to the current version of the contract, b will not be paying a appropriately.

$$\begin{array}{l} pay(b, a, p, V) \leftarrow \\ \qquad price(a, p, V_p) \wedge \\ \qquad V \ is \ V_p - 0.2 * V_p \end{array}$$

Conditions for contract failure are again satisfied and the contract is sent back to a for further adjustment. This time a can alter the clause that specifies the cost of product p for the consumer b as follows.

$$\begin{array}{l} cost(a, b, p, V) \leftarrow \\ \qquad price(a, p, V_p) \wedge \\ \qquad V \ is \ V_p - 0.2 * V_p \end{array}$$

This version of the contract is acceptable in the sense that we can no longer satisfy the conditions for failure. The producer sends it again for b's scrutiny, but this time b cannot find any more reasons for not accomplishing the agreement successfully. Finally, b agrees with this proposal. The final binding contract that sets the terms for the supply of p by a to b is then described below:

```
contract_completion(X, P, D) ←
        producer(X, P) ∧
        consumer(Y, P) ∧
        delivered(X, Y, P, D) ∧
        delivery_date(X, Y, P, Dd) ∧
        D ≤ Dd
        appropriate_payment(Y, X, P)

contract_failure(X, P, D) ←
        producer(X, P) ∧
        consumer(Y, P) ∧
        delivered(X, Y, P, D) ∧
        delivery_date(X, Y, P, Dd) ∧
        (D > Dd ∨ not_appropriate_payment(Y, X, P))

delivered(X, Y, P, D) ←
        delivery_date(X, Y, P, D)

delivery_date(a, b, p, D) ←
        starting_date(a, b, p, Ds) ∧
        production_time(a, p, Tp) ∧
        D is Ds + Tp + 5

appropriate_payment(Y, X, P) ←
        pay(Y, X, P, V) ∧
        cost(X, Y, P, V)

not_appropriate_payment(Y, X, P) ←
        pay(Y, X, P, Vp) ∧
        cost(X, Y, P, V)
        Vp < V

pay(Y, X, P, V) ←
        cost(X, Y, P, V)

cost(a, b, p, V) ←
        price(a, p, Vp) ∧
        V is Vp − 0.2 * Vp

producer(a, p)
consumer(b, p)
starting_date(a, b, p, 1)
production_time(a, p, 6)
price(a, p, 10)
```

This example illustrates the process of contract-based negotiation we are interested in formalising. We now investigate how an argumentation framework can provide the necessary mechanisms for implementing it.

3 A General Framework for Argumentation

In this section we outline a generic argumentation framework that we will use to formalise the model of negotiation presented in section 2.

This approach was proposed in [3] as a generic architecture for formal argumentation systems that captures argument dynamics. It can be seen as an extension of Kowalski & Toni's Abstract Argumentation Framework [8, 1, 7], as well as Dung's Argumentation Framework [4], in the sense that these formalisms were used as a starting point in developing a more detailed architecture for implementing adversarial argument. The idea is to identify different notions involved in dynamic argumentation and define them separately, thus providing a clearer picture of how arguments can be generated and evaluated, and also allowing a larger class of arguments to be formally represented.

3.1 The Theory

A theory is a set of sentences based on a monotonic proof system $(\mathcal{L}, R)$ that allows conclusions to be derived via the inference rules in R. The provability relation defined by this system is denoted by $\vdash$. In this architecture, a theory $\mathcal{T} \subseteq \mathcal{L}$ simply defines the object we argue about. This is the level where arguments are generated.

Definition 3.1 *Let $(\mathcal{L}, R)$ be the proof system underlying a theory $\mathcal{T} \subseteq \mathcal{L}$. If $\alpha \in \mathcal{L}$ can be inferred from a set of formulae $\Gamma \subseteq \mathcal{T}$, then we say that $\Gamma \vdash \alpha$ is an argument for α.*

In the pragmatic level arguments are "weaker" than proofs because contradicting arguments may coexist within a theory. The idea of contradicting arguments is based on the more fundamental notion of conflicting sentences in the language. Because this depends on the underlying logic, we assume that conflict between sentences is a primitive notion represented here by a special predicate $conflicts \subseteq \mathcal{L} \times \mathcal{L}$, in a similar way as presented in [4, 13].

Definition 3.2 *We say that an argument $\Gamma \vdash \alpha$ attacks an argument $\Theta \vdash \beta$ if $conflicts(\alpha', \beta')$.*

After characterising the concepts of argument and attack, we now define *when* and *how* attacks are generated. This requires an additional theory.

3.2 The Criticism Theory

The idea behind a criticism theory is to provide potential points of attack to arguments within the architecture itself. These possible attacks can guide the generation of arguments in the theory $\mathcal{T}$, but are defined separately from it by means of an additional theory $\mathcal{T}_c$.

A criticism theory $\mathcal{T}_c$ is again based on a monotonic proof system $(\mathcal{L}_c, R_c)$ that defines a provability relation $\vdash_c$. The notion of argument can be defined analogously to definition 3.1 in $\mathcal{T}_c$ by means of $\vdash_c$. The intuitive idea behind this additional theory is that arguments from $\mathcal{T}_c$ are mapped onto the theory in order to generate attacking arguments in $\mathcal{T}$.

Note that the provability relations from theory and criticism theory might be different (e.g. $\vdash$ can be deductive and $\vdash_c$ abductive), and this is one reason for defining them separately. Also note that this dissociation is not always clear, in particular because the theories might clash when the underlying proof systems are equivalent. However, making this dissociation explicit allows for a larger class of argument systems to be described [3].

Given a theory $\mathcal{T}$ based on $(\mathcal{L}, R)$ and a criticism theory $\mathcal{T}_c$ based on $(\mathcal{L}_c, R_c)$, let us formally present the mechanism for mapping arguments from $\mathcal{T}_c$ to $\mathcal{T}$. This mechanism is based on the following twofold mapping, which essentially relates consequences from both theories:

- $equiv : L \subseteq \mathcal{T} \to \mathcal{L}_c$ relates the underlying languages by identifying which conclusions in the criticism theory correspond to the conclusions in the theory.
- $map : L_c \subseteq \mathcal{T}_c \to 2^{\mathcal{L}}$ identifies what sentences need to be added in the theory to derive the corresponding conclusion of an argument from the criticism theory. This idea is in line with [17, 13], where agents that argue and negotiate should have their language only partially defined in order to allow new concepts to be introduced.

In particular, if $\mathcal{T}$ and $\mathcal{T}_c$ are based on the same proof system, this mapping corresponds to the identity function.

The process of generating an attack in $\mathcal{T}$ from an argument in $\mathcal{T}_c$ is described below and illustrated in figure 1.

- Let $\Gamma \vdash \alpha$ be an argument for α in $\mathcal{T}$ and $\overline{\alpha} \in \mathcal{L}$ such that $conflicts(\alpha, \overline{\alpha})$.
- If $equiv(\overline{\alpha}) = \beta$, where $\beta \in \mathcal{L}_c$, then β is a possible point of attack on α.
- Let $\Delta \vdash_c \beta$ be an argument for β in $\mathcal{T}_c$.
- If $\Delta = \{\delta_i | 1 \leq i \leq n\}$ and $S_i = map(\delta_i)$ for $1 \leq i \leq n$, then $S = \bigcup_i^n S_i$ is the set of sentences that need to be added to the theory so that we can derive an argument for $\overline{\alpha}$ from it.
- Consider the extended theory $\mathcal{T}' = \mathcal{T} \cup S$. Because the proof system on which the theory is based is monotonic, the addition of new clauses allows new consequences to be derived without compromising the original ones[1]. Therefore, we now have that argument (ii) below attacks argument (i) according to definition 3.2.

  (i) $\Gamma \vdash \alpha$, where $\Gamma \subseteq \mathcal{T} \subseteq \mathcal{T}'$; and
  (ii) $\Gamma' \vdash \overline{\alpha}$, where $\Gamma' \subseteq \mathcal{T}'$.

Therefore, $\mathcal{T}_c$ can be seen as a source of attacks to arguments in $\mathcal{T}$.

It is interesting to point that both $\mathcal{T}$ and $\mathcal{T}_c$ are monotonic, but the development of the argument is non-monotonic in the sense that certain arguments are preferred over others. This is in line with the work in [13], where "the process of building arguments is monotonic, but the process of coming

[1] We assume that we cannot derive negations in our proof system in order to avoid inconsistent theories.

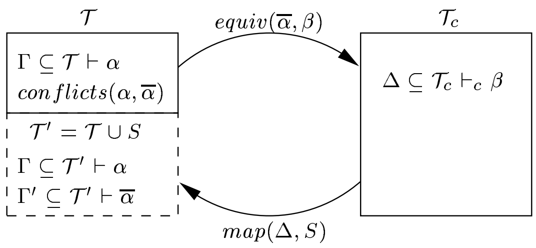

Figure 1: Generating attacks based on a criticism theory.

to conclusions using arguments is non-monotonic", and also with Prakken's view that "conflict is a matter of debate and should be dealt by an Argumentation Framework rather than by the underlying logic" [14].

The last step in describing the necessary elements for generating dynamic arguments is to account for the notion of priorities.

3.3 The Control Module

In human argument we often apply extra information to control the generation and termination of arguments. This information usually involves the notion of *defeat*, where conflict among arguments is decided based on certain priorities or preferences [15, 18, 2].

The control module defines this notion by specifying some comparative measure for arguments and a criterion based on this measure for adjudicating between conflicting arguments. It has already been argued that this criterion is domain specific [6, 15], and in this framework it also depends on the choice of priority measures.

There are many ways one can capture this notion of comparative measures among arguments, such as defining a strict partial ordering on the set of defeasible rules [15], or adopting the specificity principle [18]. In our architecture, this can also be done in different ways, e.g. by prioritising consequences in the theory or by prioritising arguments from the criticism theory.

To allow adaptation separately from the rest of the framework, we characterise the control module as a meta-level component which treats theory and/or criticism theory as sources of information and propagates priority measures through them accordingly. This architecture is similar to the type of layered meta-interpreter described in [19] and is sketched in figure 2.

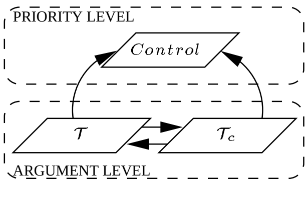

Figure 2: Architecture overview.

These three components – theory, criticism theory and control module – provide what is needed in order to generate and compare arguments (in the sense of definition 3.1). Therefore, they constitute the core of our framework and may be compared to what Dung called *argument generation unit* [4]. However, the interesting point about this architecture is that it captures the mechanisms for constructing dynamic arguments and for determining if a conclusion is accepted on the basis of such arguments. This can be seen as the *argument processing unit* specified by Dung [4], and it will be described next in terms of our three-component architecture.

3.4 Describing Argument Dynamics

In [11], Loui argues that what makes beliefs rational is not only their relations to other beliefs, but also the way in which they are constructed. Here we assume that arguments are constructed based on dialectical procedures. I.e. two parties are involved, namely a proponent P and an opponent O, whose roles are to establish and invalidate a particular claim, respectively. To fulfill their objectives, they take turns in advancing arguments that defeat the argument previously presented by the other party.

We now formally define the idea of dynamic argument in a similar way as presented in [15, 11]:

Definition 3.3 *Let P, O be the parties involved and α be a sentence in the theory $\mathcal{T}$. A dynamic argument about α is a finite sequence $< P_1, Arg_1 >, ..., < P_n, Arg_n >$, $n \geq 1$, such that:*

- *$P_1 = P$. If $P_i = P$, $i \geq 1$, then $P_{i+1} = O$ and vice versa.*
- *Arg_i, $1 \leq i \leq n$, is a minimal argument[2] in $\mathcal{T}$, possibly derived based on a criticism theory $\mathcal{T}_c$. In particular, Arg_1 is an argument for α.*
- *Arg_i defeats Arg_{i-1}, $1 < i \leq n$.*
- *P_{n+1} cannot derive an argument that defeats Arg_n.*

Therefore, P_n is considered the "winning" party; i.e. its goal regarding α has been met.

This definition summarises what is needed to describe argument dynamics based on the argumentation framework in [3]. One of the major differences between this proposal and other formal argumentation approaches (especially for representing non-monotonic reasoning) is that it allows changes to be made to the theory itself based on reasoned criticism arguments. Moreover, the source of these criticisms is explicitly defined, as well as the priority criteria for comparing arguments.

Some issues in definition 3.3 could be further elaborated, such as resource-bounded termination as opposed to natural termination [11], or strategies for choosing possible attacks. However, here we restrict ourselves to a more simplistic approach for the sake of clarity.

Acceptability of a conclusion can then be defined in terms of dynamic arguments.

Definition 3.4 *A sentence α is said to be acceptable if we can construct a dynamic argument about α where the proponent is the winning party.*

4 Formalising Contract-based Negotiation via Argumentation

In this section, we present an approach that uses the framework for argumentation described in section 3 to formalise the generic model of contract-based negotiation of section 2.

Intuitively, negotiation is a dynamic process that can involve the exchange of arguments. Hence our idea to formalise it as a dynamic argument. The three-component argumentation architecture of [3] provides means to describe such arguments. Moreover, it provides the notion of arguing about "something", where this "something" is actually represented by a theory $\mathcal{T}$ and can be changed based on reasoned arguments advanced by the participants. These aspects connect directly to the idea of arguing about a contract.

[2] We restrict ourselves to minimal arguments to avoid unnecessary and infinite repetition.

In summary, we want to instantiate the architecture in section 3 in order to obtain a formal argumentation system for constructing dynamic arguments about contracts between producers and consumers (or between proponents and opponents).

4.1 Negotiation Processes as Dynamic Arguments

Let a, b be two autonomous agents – a producer and a consumer, respectively – that will negotiate the terms of a contract regarding the supply of a particular product. In an argumentation context, this contract is represented by a theory $\mathcal{T}$ based on a monotonic proof system $(\mathcal{L}, R)$. Essentially, the theory consists of a set of formulae which contains, in particular, conflicting sentences that correspond to success and failure in achieving the terms of the agreement. The goal is to adjust $\mathcal{T}$ according to the beliefs of a and b. These adjustments are done as we generate a dynamic argument about the successful completion of the contract.

Let $\mathcal{T}_a$ and $\mathcal{T}_b$ represent the internal theories of agents a and b, respectively. Within each agent's theory we assume that there is a module that accounts for contract manipulation and negotiation which is based on a monotonic proof system. In particular, let $\mathcal{T}_{a_c} \subseteq \mathcal{T}_a$, $\mathcal{T}_{b_c} \subseteq \mathcal{T}_b$ be such subsets of agents a and b's internal theories. For the sake of clarity, we assume that the systems underlying $\mathcal{T}$, $\mathcal{T}_{a_c}$ and $\mathcal{T}_{b_c}$ are equivalent. However, we make no further assumptions about an agent's theory, in particular about the way it represents or revises beliefs. In fact, the rest of this theory does not even need to be logical.

In this process, the role of the criticism theory is played by the sub-theories or modules $\mathcal{T}_{a_c}$ and $\mathcal{T}_{b_c}$. From the consumer's perspective, the criticism theory allows it to present arguments that attack the successful completion of the contract. From the point of view of the producer, this module allows it to present arguments that attack failure, reestablishing success as the consequence of that contract. This idea is illustrated in figure 3. The mappings allow producer and consumer to relate the consequences in the contract to consequences in their own theories, and then map their own preferences, arguments and reasons in terms of clauses in the contract.

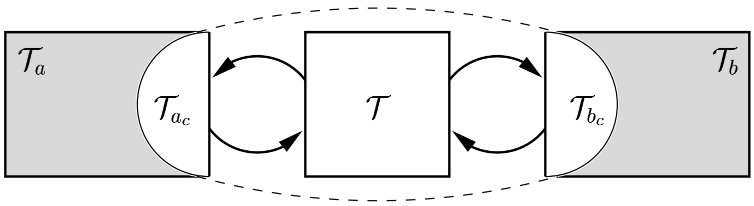

Figure 3: Contract-based negotiation in the argumentation framework.

With respect to preferences, priority measures can be defined from two perspectives. On one hand, agents can have preferences based on utility functions, values [5] or explicitly represented by means of a special meta-predicate [17]. These are specified within the agents' theories and they might indicate, for instance, that some criticisms are not strong enough to be pointed out in the contract. On the other hand, we can also have criteria for preferring one clause over another in a contract in case of conflicting consequences. In any case, these notions of preference conform to the architecture in section 3.3.

It is important to note that contracts are defined and altered in terms of agents' beliefs. In this sense, they are similar to KQML messages, where the use of performatives is described in terms of the agents' cognitive states [10]. However, differently from KQML, contracts are structures – or objects – which are manipulated by agents and used to test whether certain properties are satisfied or not.

In this way, we have instantiated the following components in the argumentation architecture: a theory that represents the contract; and a criticism theory and a control module for each agent. These are the necessary components for generating and comparing arguments about the contract. Based on these, the negotiation process can be implemented as a dynamic argument in the sense of definition 3.3. In that context, P and O represent the producer a and the consumer b, respectively. At each step, agents can adjust the clauses in the contract or withdraw from the agreement. Note that if a particular agent withdraws from the process, then the notion of *winning party* from definition 3.3 no longer obtains.

By instantiating the architecture in this way we can implement argumentation systems for generating dynamic arguments that correspond to the contract-based negotiation model presented in section 2. In particular, the example from section 2.1 could be generated by this system if the agent criticism (sub)theories and the contract were appropriately instantiated based on the same underlying proof system, namely a Horn-clause resolution-based system.

5 Conclusion

In this paper we have considered negotiation between autonomous agents from a different perspective, based on contracts rather than protocols. In particular, we define contracts as *objects* that can be adjusted via an argumentation process carried out by the involved agents. This approach differs from protocol-based formalisms in the sense that they are useful for adapting the terms of an agreement rather than forming an agreement.

We have proposed a generic model that characterises this type of approach to negotiation and used an argumentation framework to formalise it. This framework was proposed in [3] and it has been used to specify argumentation systems in single-agent domains. Here we have investigated how it can also be applied in a multi-agent context.

Acknowledgements. The first author is supported by the Brazilian Government through CNPq, grant 200074/97-0. The second author is supported by an EPSRC Advanced Fellowship.

References

[1] A. Bondarenko, P. M. Dung, R. A. Kowalski, and F. Toni. An abstract, argumentation-theoretic approach to default reasoning. *Artificial Intelligence*, 93(1–2):63–101, 1997.

[2] G. Brewka. Well-founded semantics for extended logic programs with dynamic preferences. *Journal of Artificial Intelligence Research*, 4:19–36, 1996.

[3] D. V. Carbogim, D. S. Robertson, and J. R. Lee. Extending the abstract argumentation framework to describe argument dynamics. DAI Research Paper 940, Artificial Intelligence, Division of Informatics, University of Edinburgh, 1999.

[4] P. M. 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.

[5] J. Fox and S. Parsons. Arguing about beliefs and actions. In A. Hunter and S. Parsons, editors, *Applications of Uncertainty Formalisms*, volume 1455 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1998.

[6] K. Konolige. Defeasible argumentation in reasoning about events. In Z. W. Ras and L. Saitta, editors, *Proceedings of the Third International Symposium on Methodologies for Intelligent Systems (ISMIS 1988)*, pages 380–390, Turin, Italy, 1988.

[7] R. Kowalski and F. Toni. Argument and reconciliation. In *Proceedings of the Workshop of Applications of Logic Programming to Legal Reasoning at the International Symposium on Fifth Generation Computer Systems (ICOT94)*, Tokyo, 1994.

[8] R. Kowalski and F. Toni. Abstract argumentation. *Artificial Intelligence and Law*, 4(3–4):275–296, 1996. Special Issue on Logical Models of Argumentation, H. Prakken and G. Sartor, editors.

[9] P. Krause, S. Ambler, M. Elvang-Goransson, and J. Fox. A logic of argumentation for reasoning under uncertainty. *Computational Intelligence*, 11(1):113–131, 1995.

[10] Y. Labrou and T. Finin. A semantics approach for KQML – a general purpose communication language for software agents. In *Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM94)*, pages 447–455, Gaithersburg, USA, 1994.

[11] R. P. Loui. Process and policy: resource-bounded non-demonstrative reasoning. *Computational Intelligence*, 14(1):1–38, 1998.

[12] S. Parsons and N. R. Jennings. Negotiation through argumentation – a preliminary report. In *Proceedings of 2nd International Conference on Multi-Agents System*, pages 267–274, Kyoto, Japan, 1997.

[13] S. Parsons, C. Sierra, and N. R. Jennings. Agents that reason and negotiate by arguing. *Journal of Logic and Computation*, 8(3):261–292, 1998.

[14] H. Prakken. From logic to dialectics in legal argument. In *Proceedings of the Fifth International Conference on Artificial Intelligence and Law*, pages 165–174, Washington DC, USA, 1995. ACM Press.

[15] H. Prakken and G. Sartor. Argument-based extended logic programming with defeasible priorities. *Journal of Applied Non-classical Logics*, 7(1):25–75, 1997. Special issue on Handling Inconsistency in Knowledge Systems.

[16] C. Sierra, P. Faratin, and N. R. Jennings. A service-oriented negotiation model between autonomous agents. In *Proceedings of 8nd European Workshop on Modeling Autonomous Agents in a Multi-Agent World*, Ronneby, Sweden, 1997.

[17] C. Sierra, N. R. Jennings, P. Noriega, and S. Parsons. A framework for argumentation-based negotiation. In *Proceedings of 4th International Workshop on Agent Theories, Architectures and Languages (ATAL'97)*, pages 167–182, Rhode Island, USA, 1997.

[18] G. R. Simari and R. P. Loui. A mathematical treatment of defeasible reasoning and its implementation. *Artificial Intelligence*, 53(2–3):125–157, 1992.

[19] L. U. Yalcinalp and L. Sterling. Uncertainty reasoning in Prolog with layered meta-interpreters. In *Proceedings of the 7th IEEE Conf. in AI Applications*, 1991.