A Formal Approach to Protocols and Strategies for (Legal) Negotiation [*]

Guido Governatori, Marlon Dumas, Arthur H.M. ter Hofstede and Phillipa Oaks

Cooperative Information Systems Research Centre
Queensland University of Technology
GPO Box 2434
Brisbane QLD 4001, Australia

{governat, dumas, arthur, oaks}@icis.qut.edu.au

ABSTRACT

We propose a formal and executable framework for expressing protocols and strategies for automated (legal) negotiation. In this framework a party involved in a negotiation is represented through a software agent composed of four modules: (i) a communication module which manages the interaction with the other agents; (ii) a control module; (iii) a reasoning module specified as a defeasible theory; and (iv) a knowledge base which bridges the control and the reasoning modules, while keeping track of past decisions and interactions. The choice of defeasible logic is justified against a set of desirable criteria for negotiation automation languages. Moreover, the suitability of the framework is illustrated through two case studies.

Keywords. Automated Legal Negotiation, Auctions, Defeasible Logic, Software Agents.

1. INTRODUCTION

What is legal negotiation? With negotiation we mean a process involving at least two parties aimed at reaching an agreement that is acceptable by the parties involved. In this perspective a legal negotiation is either a negotiation about legal content or a negotiation where legal issues have to be taken into account.

Legal negotiations are by far more common than one might expect. For example any agreement can be conceived as a form of private contract, and contracts are subject to the current (relevant) legislation. Moreover, there are several kinds of negotiation and some of them are typical of legal areas (for example pleading, cf. [10]) while others are more frequent in economics, commerce and related fields, but, in the end, even these cases have to be considered with a normative eye.

In addition to affecting its content, legal issues can also affect the negotiation process on itself. Indeed, any negotiation is guided by a protocol, which describes the rules of the dispute, that is, how the parties exchange their offers, and how and when the negotiation can go on or terminate. Protocols can themselves be seen as agreements, that can take the form of a contract, and are thereby subject to law. Thus, for example, an auction is a one-to-many negotiation governed by the rules of the auction itself. In public auctions the rules (procedures) are established by law (cf., for example the articles 576 and 581 of the Italian Code of Civil Procedure), while private auctions are self-regulated: the auction house states the rules under which the auction will be conducted (yet, these rules are subject to the law).

It should be clear that one of the main normative aspects of negotiation concerns the definition of the protocols (rules) under which negotiations are carried out.

Negotiation protocols can be classified into one-to-one (e.g., bargaining, divorce), one-to-many (e.g., tenders, English auction), and many-to-many (double auctions) [35]. This classification can be further refined depending on whether the negotiation is carried out in a single step (i.e., sealed-bid auctions), or iteratively (open-cry auctions). Figure 1 provides a classification schema that captures these two dimensions.

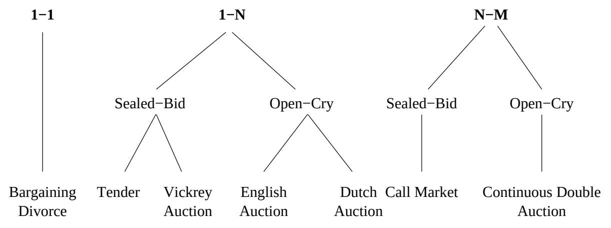

Figure 1: Taxonomy of negotiation protocols

Other classification dimensions orthogonal to the above ones include:

- Single issue vs. multiple issues. Legal negotiations often involve several interdependent issues. Auctions involving multiple items [30] or multiple units of an item are another example of multiple-issue negotiations.
- Direct vs. mediated. A mediator is a neutral party (e.g., a judge in a pleading), as opposed to a representative of one of the parties (e.g., a lawyer). Legal negotiations are often mediated.

[*]This work was funded by the ARC Grant "Self-Describing Transactions Operating in a Large, Heterogeneous, Distributed Environment" involving QUT and GBST Pty Ltd.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ICAIL-2001 St. Louis, Missouri USA
Copyright 2001 ACM 1-58113-368-5/01/0005 $5.00.

- Anonymous vs. identity-open. Auction houses often carry out anonymous or at least semi-anonymous auctions.

This brief overview shows that protocols are manifold; it would thus be difficult, if not impossible, to capture all of them in a single universal framework. Still, we believe that a logic-based approach could cope easily with most of them. Moreover, this same kind of approach could be used to specify the behaviour of the parties involved in a negotiation, that is, their strategies. Interestingly, expressing both the protocols and the strategies in a logic-based framework, makes it possible to reason about their internal and their mutual consistencies.

With this motivation, this paper aims at assessing the feasibility and suitability of a logic-based approach to legal negotiation, and to develop a framework for specifying protocols and strategies in this setting.

The paper is organised as follows: in Section 2 we introduce a list of desiderata that an automated negotiation system should satisfy (Section 2.1) and we describe an agent based architecture for such systems (Section 2.2). In Section 2.3 we defend Defeasible Logic against the desiderata of Section 2.1, and then we introduce Defeasible Logic formally (Section 3). In Section 4 we develop two case studies in order to illustrate how the framework works. We conclude the paper with a discussion on related work (Section 5) and directions for future research (Section 6).

2. RATIONALE AND APPROACH

2.1 Desiderata

Before choosing one or several languages for the specification of protocols and strategies for automated negotiation, it is important to establish a set of criteria that such languages need to satisfy. The criteria presented below are inspired from those formulated by [15] in the context of techniques for information modelling. They encompass several well-known principles of language design.

Firstly, a language for specifying negotiation protocols and strategies needs to be *formal*, in the sense that its syntax and its semantics should be properly defined. This ensures that the protocols and strategies can be interpreted unambiguously (both by machines and human beings) and that they are both *predictable* and *explainable*. In addition, a formal foundation is a prerequisite for verification or validation purposes.

Secondly, a language for negotiation automation should be *conceptual*. This, following the well-known *Conceptualization Principle* of [13], effectively means that the language should allow their users to focus only and exclusively on aspects related to negotiation, without having to deal with any aspects related to their implementation. As stated in [13], examples of conceptually irrelevant aspects are e.g. aspects of (external or internal) data representation, physical data organisation and access, as well as all aspects related to platform heterogeneity (e.g., message-passing formats).

Thirdly, in order to ease the interpretation of protocols and strategies and to facilitate their documentation, the language should be *comprehensible*. Comprehensibility can be achieved by ensuring that formal and intuitive meaning are as much in line as possible, and by offering structuring mechanisms (e.g., decomposition). These structuring mechanisms often lead to *modularity*, which in our setting means that a slight modification to a protocol or strategy should concern only a specific part of its specification. Closely related to its comprehensibility, the language that we aim for should be *suitable*, that is, it should offer concepts close to those involved in negotiation. It is worth noting that the suitability criterion is content dependent and it can be specialised in several sub-fields according to the nature of the phenomenon at hand and the aim of the intended modelling.

As we are interested in the actual execution of the negotiation process, the negotiation automation language should be *executable*. Furthermore, in the setting of an open environment such as the Internet, the execution of the language expressions should exhibit acceptable performances even for complex strategies involving many issues, and for a great number of participants (i.e., the execution performances should be *scalable*).

Finally, a negotiation automation language should be sufficiently *expressive*, that is, it should be able to precisely capture all the existing negotiation protocols, as well as a wide spectrum of negotiation strategies.

2.2 Architecture

Following a classical agent-based approach to automated negotiation, we view a negotiation process as a set of software agents which interact in order to reach an agreement. Agents participating in a negotiation can interact directly in the case of one-to-one negotiation, or through a mediator (or broker) in the case of a multi-party negotiation. The mediator may itself be considered as a software agent guided by a set of rules.

Following an abstract architecture for agents with memory proposed in [34] and other previous work, each of the software agents is composed of four modules:

1. A communication module responsible of receiving and sending messages to the other agents, while ensuring that these messages satisfy the constraints imposed by the protocol.

2. A memory which contains the history of the past decisions and interactions of the agent, including its current intentions.

3. A reasoning module which encodes the decision-making part of the agent.

4. A control module which coordinates the interactions between all the other modules.

In the remainder, we choose to express the control and the reasoning modules of an agent as defeasible theories, and consequently the memory as a knowledge base of strict and defeasible facts. This choice is defended in the next subsection. In this paper, we do not address the issue of designing the communication module. Still, we believe that defeasible logic could be also applied to this end. Indeed, one of the roles of the communication module is to check that the incoming and outgoing messages conform with the protocol. Hence, if the protocol is specified as a set of strict and defeasible rules, these rules can be used to derive a partial specification of the communication module.

A negotiation is a discussion between parties for the purpose of reaching an agreement (cf., for example, [6]). This suggests representing a negotiation as a dialogue between parties, this dialogue is articulated in progressive stages, where the parties make offers, reject or accept offers, or propose counter-offers. In the context of a negotiation involving two parties –let us call them the Proponent and the Opponent– the architecture can be depicted as in Figure 2.

A bilateral negotiation is therefore modeled through three sequences of defeasible theories. The first sequence records the evolution of the protocol, while the second and the third theories are used to store the knowledge bases or defeasible theories (DT) of the two parties. Graphically, a bilateral negotiation can be depicted as follows

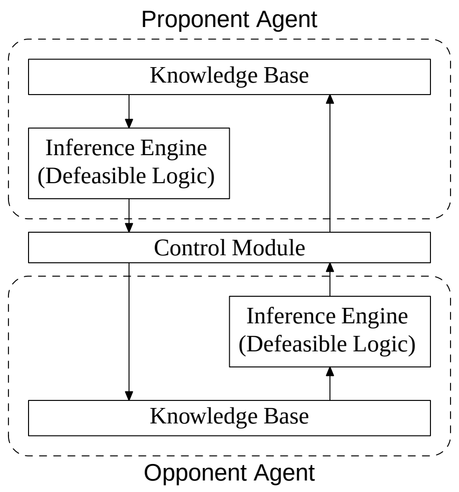

Figure 2: System architecture for bilateral negotiation

Stage 1	Protocol_1	Proponent DT_1	Opponent DT_1
Stage 2	Protocol_2	Proponent DT_2	Opponent DT_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$
Stage n	Protocol_n	Proponent DT_n	Opponent DT_n
$\vdots$	$\vdots$	$\vdots$	$\vdots$

AGREEMENT / END OF NEGOTIATION

Under this model, we can think of at least four kinds of negotiation strategies.

- Single fixed theory: a party uses a single defeasible theory through the whole negotiation, which is evaluated using new data that becomes available during the negotiation.
- Fixed sequence of theories; here a party fixes a sequence of theories for the whole negotiation.
- Parameterised theories: a party defines a set of rules that can be triggered or modified according to the stage of the negotiation.
- Revision of theories: a party modifies the actual theory from stage to stage according to the result of the previous stage.

In the case studies presented in section 4, a single fixed theory approach is used.

Information privacy is an important aspect of a negotiation. As long as the protocol allows it, the parties do not have to disclose every piece of information they have. Thus, we partition the defeasible theory of a party into two parts: the public part, whose conclusions have to be disclosed to the other party, and a private part. The Proponent computes its theory obtaining a set of conclusions and the public conclusions are passed to the Opponent through the communication modules of the parties. The Opponent then uses these facts to supplement its knowledge base, and re-computes its theory. According to the result of this last computation we can have three possible results: the Opponent accepts the Proponents offer and the negotiation is terminated successfully; the Opponent rejects the Proponents offer, makes a counter offer and the negotiation is continued (i.e., we pass to the next stage); or the Opponent rejects the offer, but the two parties cannot converge on an agreement so there is no point in negotiating, and the negotiation is terminated with a failure.

In the context of legal bilateral negotiation, we have to consider also external rules, for example laws regulating the trade at hand or the rules of negotiation, thus the we have to define the legal outcome, where each agent has to consider not only its own knowledge base but it has to take into account also the defeasible theory corresponding to the relevant jurisprudence. It is worth noting that legal issues have to be considered at two levels. They can and must be involved in the negotiation phase (e.g., in the evaluation of the offers and counteroffers); and subsequently, they are used to verify the legality and validity of the reached agreement.

The above architecture is also applicable for multi-party negotiations, where there are one or several sellers, and one or several buyers. In this setting, the architecture can be depicted as in Figure 3.

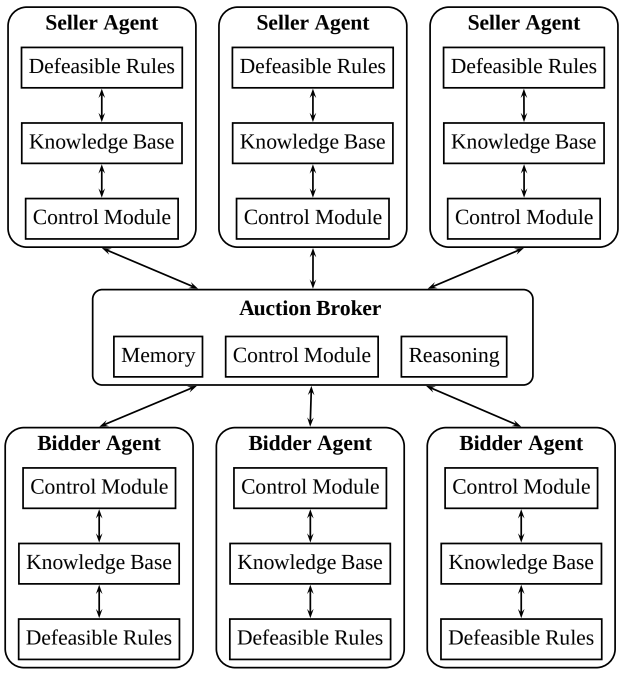

Figure 3: System architecture for auctions.

Conceptually, each time that a bidding agent is notified of a new event regarding the auction, the logic program is invoked. This program reads the knowledge base, attempts to deduce new facts or refute existing ones, and updates the knowledge base accordingly. Depending on the new state of the knowledge base, the agent determines whether it should submit a bid immediately, wait for some further event, or retract from the auction. The same process is carried out by the sellers if applicable.

2.3 Choice of Formalism

There are several reasons why Defeasible Logic can be considered an appropriate tool to formalize legal negotiation. Let us examine some of them.

A negotiation can be thought of as a dialogue between parties concerning the resolution of a common dispute (cf. [6]). In this perspective some authors (cf. [27, 18, 26]) suggested the use of argumentation-based reasoning formalisms to characterise it, while others proposed the use of argumentation to supplement negotiation procedures (cf. [22]). In [11, 12], it was shown that Defeasible Logic can be characterised by argumentation semantics, thus the formal semantics of Defeasible Logic is in line with the argumentative nature of legal negotiations. Moreover, given the close connection between derivations in Defeasible Logic and arguments, Defeasible Logic offers then a predictable and explainable formalism for legal negotiation.

The defeasibility of normative reasoning is a very well established phenomenon with many facets. Consequently a plethora of non-monotonic systems have been proposed to capture it, but, unfortunately, most of the proposed non-monotonic systems have different and sometimes incompatible intuitions. In [3, 4] it was shown that Defeasible Logic is flexible enough to deal with several intuitions of non-monotonic reasoning in a modular way applying the paradigm of [20]. Moreover, Nute [23, 24] has proposed to extend Defeasible Logic with deontic operators to capture normative phenomena, while [2] shows how regulations can be represented conceptually in Defeasible Logic.

Undoubtedly, it is in the context of legal negotiations in a strict sense –negotiations governed by statutory norms– that the suitability of Defeasible Logic as a protocol specification language finds its utmost degree. Indeed, the procedure for the negotiation could leave space to open exceptions, and then be subject to defeasibility, as it is the case of the protocol presented at the end of section 3.

Regarding strategy specification, most of the current systems adopt a quantitative approach based on utility functions [28]. Very often, it is not easy to find the right utility functions for a given set of negotiation issues, especially in situations where one needs to express preferences without attaching a metric to them. Moreover, utility functions are mostly used to determine preferences that can otherwise be expressed in a more comprehensible and suitable way through (defeasible) rules and priorities among these rules. For this reason, we believe that in the context of legal negotiations, defeasible logic is more suitable than, or at least is complementary to, strategy specification approaches purely based on utility functions.

Last but not least the complexity of Defeasible Logic is linear [19], and existing implementations are able to deal with non trivial theories consisting of over 100,000 propositional rules [21], offering thus an executable and scalable system.

3. BASICS OF DEFEASIBLE LOGIC

In this section we describe Defeasible Logic formally. Defeasible logic is a sceptical formalism, meaning that it does not support contradictory conclusions. Instead it seeks to resolve differences. In cases where there is some support for concluding A but also support for concluding *not* A ($\neg A$), the logic does not conclude either of them (thus the name "sceptical"). If the support for A has priority over the support for $\neg A$ then A would be concluded. Sceptical reasoning is appropriate for the study of normative reasoning.

A set of norms (rules) will be represented as a defeasible theory. A defeasible theory, i.e., a knowledge base in Defeasible Logic, consists of six different kinds of knowledge: facts, strict rules, defeasible rules, defeaters, a superiority relation, and a specification of conflicting literals.

Facts denote simple pieces of information that are deemed to be true regardless of other knowledge items. A typical fact is that John is a minor: $minor(John)$.

Briefly, strict and defeasible rules are represented, respectively, by expressions of the form $A_1,\dots,A_n \to B$ and $A_1,\dots,A_n \Rightarrow B$, where $A_1,\dots,A_n$ is a possibly empty set of prerequisites and B is the conclusion of the rule.

Strict rules are rules in the classical sense: whenever the premises of a rule are given, we are allowed to apply the rule and get a conclusion. When the premises are indisputable (e.g., facts) then so is the conclusion. An example of a strict rule is "every minor is a person". Written formally:

$$minor(X) \to person(X).$$

It is worth noting that, technically, facts can be represented as strict rules with an empty antecedent. However, this is against the very same idea of non-monotonic reasoning, and the difference between facts and strict rules with empty antecedents is a pragmatic one: facts capture a snapshot of the case at hand, and they may change from case to case, while rules are the description of the scenario under analysis, and are not depended on the case at hand.

Defeasible rules are rules that can be defeated by contrary evidence. An example of such a rule is "every person has the capacity to perform legal acts to the extent that the law does not provide otherwise"; written formally:

$$person(X) \Rightarrow hasLegalCapacity(X).$$

The idea is that if we know that someone is a person, then we may conclude that he/she has legal capacity, *unless there is other evidence suggesting that he/she has not*.

Defeaters are a special kind of rules. They are used to prevent conclusions not to support them. For example

$$WeakEvidence \rightsquigarrow \neg guilty$$

This rule states that if pieces of evidence are assessed as weak, then they can prevent the derivation of a "guilty" verdict; on the other hand they cannot be used to support a "not guilty" conclusion.

The *superiority relation* among rules is used to define priorities among rules, that is, where one rule may override the conclusion of another rule. For example, given the defeasible rules

$$\begin{array}{rl} r: & person(X) \Rightarrow hasLegalCapacity(X) \\ r': & minor(X) \Rightarrow \neg hasLegalCapacity(X) \end{array}$$

which contradict one another, no conclusive decision can be made about whether a minor has legal capacity. But if we introduce a superiority relation $\succ$ with $r' \succ r$, then we can indeed conclude that the minor does not have legal capacity.

It turns out that we only need to define the superiority relation over rules with contradictory conclusions. Also notice that a cycle in the superiority relation is counter-intuitive from the knowledge representation perspective. In the above example, it makes no sense to have both $r \succ r'$ and $r' \succ r$. Consequently, the defeasible logic we discuss requires an acyclic superiority relation.

For each literal p we define the set of *p-Complementary literals* ($C(p)$), that is, the set of literals that cannot hold when p does. Let us consider an example: suppose we have the predicates *innocent* and *guilty*. Here, we define, for any constant a, $C(innocent(a)) = \{\neg innocent(a), guilty(a)\}$. We know that, under the usual interpretation of the predicates they cannot be true at the same time for one and the same individual. We stipulate that the negation of a literal is always complementary to the literal, thus, in the rest of the paper, when we give a set of p-Complementary literals we shall omit the negation of the literal.

Now we present formally defeasible logics. A *rule* r consists of its *antecedents* (or *body*) $A(r)$ which is a finite set of literals, an arrow, and its *consequent* (or *head*) $C(r)$ which is a literal. There are three kinds of arrows, $\to$, $\Rightarrow$ and $\rightsquigarrow$ which correspond, respectively, to strict rules, defeasible rules and defeaters. Where the body of a rule is empty or consists of one formula only, set notation may be omitted in examples.

Given a set R of rules, we denote the set of all strict rules in R by R_s, the set of strict and defeasible rules in R by R_{sd}, the set of defeasible rules in R by R_d, and the set of defeaters in R by R_{dft}.

$R[q]$ denotes the set of rules in R with consequent q, and $R[C(q)]$ denotes the set of rules in R whose consequent is in $C(q)$.

A *defeasible theory* D is a structure

$$D = (F, R, \succ, C)$$

where F is a finite set of facts, R is a finite set of rules, $\succ$ is a binary relation over R, and C is a function mapping a literal to a set of literals.

A *conclusion* of D is a tagged literal, where a tag is either ∂ or Δ, that may have positive or negative polarity.

$+\Delta q$ which is intended to mean that q is definitely provable in D (i.e., using only strict rules).

$-\Delta q$ which is intended to mean that we have proved that q is not definitely provable in D.

$+\partial q$ which is intended to mean that q is defeasibly provable in D.

$-\partial q$ which is intended to mean that we have proved that q is not defeasibly provable in D.

Basically a conclusion B is supported if there is a rule whose conclusion is B, the prerequisites are either supported or given in the case at hand, and a stronger rule whose conclusion is *not B* has prerequisites that fail to be supported.

Provability is based on the concept of a *derivation* (or *proof*) in $D = R$. A derivation is a finite sequence $P = (P(1), \ldots, P(n))$ of tagged literals satisfying four conditions (which correspond to inference rules for each of the four kinds of conclusion). In the following $P(1..i)$ denotes the initial part of the sequence P of length i.

```
$+\Delta$:
If $P(i+1) = +\Delta q$ then
    $\exists r \in R_s[q]$
        $\forall a \in A(r) : +\Delta a \in P(1..i)$
$-\Delta$:
If $P(i+1) = -\Delta q$ then
    $\forall r \in R_s[q]$
        $\exists a \in A(r) : -\Delta a \in P(1..i)$
```

The definition of Δ describes just forward chaining of strict rules. For a literal q to be definitely provable we need to find a strict rule with head q, of which all antecedents have been definitely proved previously. And to establish that q cannot be proven definitely we must establish that for every strict rule with head q there is at least one antecedent which has been shown to be non-provable.

Now we turn to the more complex case of defeasible provability. Before giving its formal definition we provide the idea behind such a notion. A defeasible proof of a literal p consists of three phases. In the first phase either a strict or defeasible rule is put forth in order to support a conclusion p; then we consider an attack on this conclusion using the rules for its negation $\neg p$. The attack fails if each rule for $\neg p$ is either discarded (it is possible to prove that part of the antecedent is not defeasibly provable) or if we can provide a stronger counterattack, that is, if there is an applicable strict or defeasible rule stronger than the rule attacking p. It is worth noting that defeaters cannot be used in the last phase.

```
$+\partial$:
If $P(i+1) = +\partial q$ then either
(1) $+\Delta q \in P(1..i)$ or
(2)  (2.1) $\exists r \in R_{sd}[q] \forall a \in A(r) : +\partial a \in P(1..i)$ and
     (2.2) $\forall p \in C(q) - \Delta p \in P(1..i)$ and
     (2.3) $\forall s \in R[C(q)]$ either
         (2.3.1) $\exists a \in A(s) : -\partial a \in P(1..i)$ or
         (2.3.2) $\exists t \in R_{sd}[q]$ such that
             $\forall a \in A(t) : +\partial a \in P(1..i)$ and $t \succ s$

$-\partial$:
If $P(i+1) = -\partial q$ then
(1) $-\Delta q \in P(1..i)$ and
(2)  (2.1) $\forall r \in R_{sd}[q] \; \exists a \in A(r) : -\partial a \in P(1..i)$ or
     (2.2) $\exists p \in C(q)$ such that $+\Delta p \in P(1..i)$ or
     (2.3) $\exists s \in R[C(q)]$ such that
         (2.3.1) $\forall a \in A(s) : +\partial a \in P(1..i)$ and
         (2.3.2) $\forall t \in R_{sd}[q]$ either
             $\exists a \in A(t) : -\partial a \in P(1..i)$ or $t \not\succ s$
```

Let us work through the condition for $+\partial$, an analogous explanation holds for $-\partial$. To show that q is provable defeasibly we have two choices: (1) We show that q is already definitely provable; or (2) we need to argue using the defeasible part of D as well. In particular, we require that there must be a strict or defeasible rule with head q which can be applied (2.1). But now we need to consider possible "attacks", that is, reasoning chains in support of a complementary of q. To be more specific: to prove q defeasibly we must show that every complementary literal is not definitely provable (2.2). Also (2.3) we must consider the set of all rules which are not known to be inapplicable and which have head in $C(q)$ (note that here we consider defeaters, too, whereas they could not be used to support the conclusion q; this is in line with the motivation of defeaters. Essentially each such rule s attacks the conclusion q. For q to be provable, each such rule s must be counterattacked by a rule t with head q with the following properties: (i) t must be applicable at this point, and (ii) t must be stronger than s. Thus each attack on the conclusion q must be counterattacked by a stronger rule.

To explain the mechanism of defeasible derivations –showing at the same time the appropriateness of Defeasible Logic for normative reasoning– we consider rule 162 of the Australian Civil Aviation Regulations 1988: "When two aircraft are on converging headings at approximately the same height, the aircraft that has the other on its right shall give way, except that (a) power-driven heavier-than-air aircraft shall give way to airships, gliders and balloons; ..." This norm can be represented in defeasible logic as follows:

$$r_1 : \neg \mathit{rightOfWay}(Y,X) \Rightarrow \mathit{rightOfWay}(X,Y)$$
$$r_2 : \mathit{onTheRightOf}(X,Y) \Rightarrow \mathit{rightOfWay}(X,Y)$$

The first rule states that, given two aircraft, if one of the aircraft does not have right of way then the other aircraft does, while the second states that the aircraft X has right of way over the aircraft Y if X is on the right of Y;

$$r_3 : \; \mathit{powerDriven}(X), \neg \mathit{powerDriven}(Y) \Rightarrow \neg \mathit{rightOfWay}(X,Y)$$

The idea of the above rules is that a power-driven aircraft does nothave right of way over a non-power-driven one.

$$r_4 : \mathit{balloon}(X) \to \neg \mathit{powerDriven}(X)$$
$$r_5 : \mathit{glider}(X) \to \neg \mathit{powerDriven}(X)$$

r_4 and r_5 classify balloons and gliders as non-power-driven aircraft and

$$r_6 : \; \Rightarrow \mathit{powerDriven}(X)$$

assumes aircraft to be power-driven unless further information is given. The superiority relation is determined as follows: $r_3 \succ r_2$ because r_3 is an exception to r_2 and, by specificity

$$r_4 \succ r_6 \qquad r_5 \succ r_6$$

Moreover $r_1 \succ r_3$.

Let us examine the following cases: 1) two aircraft of the same type (power-driven, non-power-driven) are converging 2) a power-driven aircraft and a non-power-driven aircraft are converging. In the first case we can apply r_2 since the prerequisites of r_3 do not hold and we cannot prove the antecedent of r_1.

In the second case we can apply r_3 given that both of the two prerequisites hold, and after the conclusion of r_3 has been established we can apply r_1 to derive that the non-power-driven aircraft has the right of way over the power-drive one.

As stated in Section 2.3, Defeasible Logic as a protocol specification language, finds its utmost degree of suitability in the context of legal negotiations in a strict sense, i.e., negotiations governed by statutory norms. For example, consider an auction consisting of two bidding rounds, in which the winners are the n highest bids of the second round, unless there is no offer in the second round or there is no second round, in which case the winners are the n highest bids of the first round.[1]

To describe this protocol we need the following predicates:

- $\mathit{firstRoundOffer}(X,Y)$ ($\mathit{secondRoundOffer}(X,Y)$): the bidder X has offered Y at the first (second) round;
- $\mathit{bid}(X,Y)$: the actual bid for the bidder X is Y.

At this point this scenario can be captured by the following defeasible rules:

$$\begin{aligned} r_1 &: \mathit{firstRoundOffer}(X,Y) \Rightarrow \mathit{bid}(X,Y)\\ r_2 &: \mathit{secondRoundOffer}(X,Y) \Rightarrow \mathit{bid}(X,Y)\end{aligned}$$

where $r_2 \succ r_1$ and the complementary literals are thus defined

$$C(\mathit{bid}(X,Y)) = \{\mathit{bid}(X,Z) | Y \neq Z\}$$

All that remain is to determine the highest n bids, but this can be easily done with a simple inspection of the conclusions with form $+\partial \mathit{bid}(X,Y)$.

4. CASE STUDIES

In this section we present in detail two examples of the application of Defeasible Logic to automated (legal) negotiation scenarios. In the first example (Section 4.1) we consider a simple case of negotiation: single issue bargaining. In the second example (Section 4.2) we examine how to model an English auction automated bidder in Defeasible Logic.

4.1 Bargaining

In this scenario we have a buyer that wants to buy a house and negotiates the price with a seller. Here we describe a simple defeasible theory capturing his/her strategy, as well as the normative rules to which this strategy is subject.

For the sake of simplicity, we use a strategy consisting of a fixed single theory. The buyer strategy is to offer first the minimum price, then wait for the counteroffer; if the counteroffer is less than the maximum price, the buyer will accept it, otherwise the maximum price will be offered. The rules corresponding to this simple strategy are:

s_1: $\mathit{price}(X) \Rightarrow \mathit{offer}(X)$

s_2: $\mathit{counteroffer}(X), \mathit{maxPrice}(Y), X < Y \Rightarrow \mathit{accept}$

s_3: $\mathit{counteroffer}(X), \mathit{maxPrice}(Y), X > Y \Rightarrow \mathit{offer}(Y)$

[1]This protocol is very similar to that used in Italy for the auction for the concession of the UMTS radio frequencies.

where $s_3 \succ s_1$ and

$$C(\mathit{offer}(X)) = \{\mathit{offer}(Y) | Y \neq X\}$$

This last condition states that at a given point in time, only one offer can be made.

At this point we need a set of rules to establish the initial and the maximum prices.

r_1: $\mathit{cash}(X), \mathit{finance}(Y), Z = X + Y \Rightarrow \mathit{maxPrice}(Z)$

r_2: $\Rightarrow \mathit{housePrice}(p1)$

r_3: $\Rightarrow \mathit{appliances}(p2)$

r_4: $\mathit{brandNewAppliances} \Rightarrow \mathit{appliances}(p3)$

r_5: $\mathit{maxPrice}(X), Y > X \leadsto \neg \mathit{totalPrice}(Y)$

r_6: $\mathit{housePrice}(X), \mathit{appliances}(Y), \mathit{tax}(W), Z = X + Y + W$
$\Rightarrow \mathit{totalPrice}(Z)$

Where $r_4 \succ r_3$ and $r_5 \succ r_6$. The complementary literals of the predicates *appliances* and *totalPrice* are defined as follows:

$$\begin{aligned} C(\mathit{appliances}(X)) &= \{\mathit{appliances}(Y) | X \neq Y\}\\ C(\mathit{totalPrice}(X)) &= \{\mathit{totalPrice}(Y) | X \neq Y\}\end{aligned}$$

Let us explain shortly the rules: r_1 says that the maximum price is the sum of the available cash and the finance that can be obtained, while rule r_5 states that the price inclusive of taxes and appliances (according to rule r_6) should not exceed the maximum price. Installed appliances are sometimes sold with the house, the buyer needs appliances and will pay a given price $p2$ for them (r_3), but will pay a higher price $p3$ if they are brand new (r_4).

During the negotiation, external considerations have to be taken into account in order to determine the price. For example, from July 1, GST (Good and Service Tax) has been introduced in Australia, thus, if the settlement date occurs after such a date, the transaction will be subject to GST (rule t_1 below), but only if the house is brand new (rule t_2). However, in such a case, first-home buyers are entitled to a rebate (rule t_3) unless they are not permanent residents or Australian citizens (t_4). Rules t_5–t_6 below are used in determining the amount of taxes, based on whether the 7k rebate is applied or not.

t_1: $\mathit{afterJuly1} \Rightarrow \mathit{GST}$

t_2: $\neg \mathit{brandNew} \Rightarrow \neg \mathit{GST}$

t_3: $\mathit{firstHome} \Rightarrow \mathit{rebate}$

t_4: $\neg \mathit{resident} \Rightarrow \neg \mathit{rebate}$

t_5: $\mathit{housePrice}(X), \mathit{GST}, Y = .1X \Rightarrow \mathit{tax}(Y)$

t_6: $\mathit{housePrice}(X), \mathit{GST}, \mathit{rebate}, Y = .1X, Y < 7k \Rightarrow \mathit{tax}(0)$

t_7: $\mathit{housePrice}(X), \mathit{GST}, \mathit{rebate}, Y = .1X, Y > 7k, W = Y - 7k$
$\Rightarrow \mathit{tax}(W)$

where $t_2 \succ t_1$, $t_4 \succ t_3$, $t_6 \succ t_5$, $t_7 \succ t_5$ and $t_7 \succ t_6$. Moreover

$$C(\mathit{tax}(X)) = \{\mathit{tax}(Y) | X \neq Y\}$$

4.2 English auction

The *English auction* is perhaps one of the most popular one-to-many negotiation mechanisms. In its simplest form, it serves to select a buyer for an item and to establish its price (multi-party single-issue negotiation). There are many variants of the English auction (see [1] for a survey). The variant that is currently in use within the biggest Internet trading communities (e.g. *eBay* [8] and *TradeOut* [32]) may be roughly described as follows. The seller starts by setting a *reservation price*, which may or may not be announced to the bidders. He also sets a *timing constraint*, which may be either expressed as a firm deadline, as a maximum duration between two successive bids, or as both. Potential buyers then issue increasingly higher bids. The *increment* between one bid and the next is constrained to be greater than a given threshold. The auction stops when the timing constraint is violated, i.e. either the deadline is reached, or no bid is registered for longer than the established maximum duration. The last bidder then buys the item at the price of the last bid. If no bid is issued at or above the reservation price, the item is not sold.

We assume that the role of the seller in an English auction is implemented by an *auction broker*. This system registers the parameters of the auction, publishes them, processes incoming bids, and continuously makes accessible the auction's status. The control module of the auction broker is described as a task structure in Figure 4. This task structure can straightforwardly be translated into a defeasible theory.

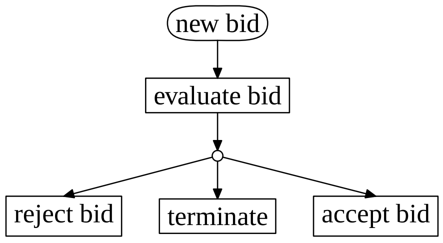

Figure 4: Auction broker control module

On the other hand, each bidder is represented through a software agent whose control and reasoning module capture the bidder's strategy. To illustrate how a bidder's strategy is expressed using defeasible logic, we consider the following scenario. Mark wishes to participate in the auction of an item. He doesn't know exactly how much the item is worth, but he thinks that its value lies somewhere within two bounds L and U. He is keen not to over-value the item, so he decides to assume at the beginning of the auction that the item is worth L, and to eventually increase his valuation whenever one of the following two situations occurs: (a) at least three bids above his current valuation have been registered, or (b) somebody has bid more than 20% of his current valuation. As soon as one of these conditions is met, Mark will raise his valuation by the minimum possible amount that allows him to stay in the auction. However, he will never accept to valuate the item above U. As it is usual in the case of English auctions, Mark will start by bidding some minimum amount (i.e. the reservation price), and if needed, he will subsequently overbid the other participants' bids by the minimum increment, as long as the resulting bid is less than his current valuation. In the eventuality where the auction's deadline is too close and that he does not hold the current highest bid, he will bid his current valuation instead of just over-bidding by the minimum increment.

Formally, the parameters, status, and history of the auction, are modeled through the following predicates and constants:

- Constant *minIncrement* denotes the minimum amount by which the bidders are allowed to overbid.
- Constant *initialBid* denotes the minimum amount of the first acceptable bid. This is not the same as the reservation price which is usually greater. In fact, we assume in the sequel that the bidders do not know the reservation price.
- Predicate $timeRemaining(T)$ provides the time remaining before the end of the auction.
- Predicate $highestBid(X)$ provides the current highest bid.
- Predicate $bidsAbove(X,N)$ holds if N bids above amount X have been registered. This predicate actually provides an aggregated view of the history of the auction, which is available as a predicate called $priceQuotes(L)$, where L is a list of pairs $\langle$time, bid$\rangle$. The rules for deriving predicate *bidsAbove* out of predicate *priceQuotes* are all strict (i.e., not defeasible), and therefore we omit them in the sequel.

Meanwhile, the parameters of Mark's strategy are modeled by the following constants and predicates:

- Constant *timeThreshold* is the duration to the deadline, below which Mark estimates that he should bid his valuation instead of just over-bidding by the minimum increment.
- Constant *significantBidders* is the number of bidders that should bid above Mark's current valuation before he considers raising it.
- Constant *significantIncrement* is the amount (expressed as a percentage), that another bidder should bid above Mark's current valuation before he considers raising it.
- Constant *maxValuation* is self-explainable.
- Predicates $submitBid(X)$ states that a bid of amount X should be submitted. At the beginning of the auction, the agent's knowledge base contains the fact $submitBid(initialBid)$.
- Predicate $valuation(X)$ provides to the current valuation.
- $newValuation(X)$ states that the valuation has been raised and should now be X. When the bidding agent finds a fact of the form $newValuation(x)$ in its knowledge base, it immediately deletes it, and updates predicate $valuation(X)$ accordingly (i.e., $+\partial valuation(X)$ will hold subsequently).
- Predicate $myBid(X)$ gives the amount of the last accepted bid issued by the bidder. At the beginning of the auction $myBid(0)$ holds.

The rules modeling the strategy are:

r_1: $myBid(X), highestBid(Y), valuation(Z),$
$X < Y, Y + minIncrement < Z,$
$timeRemaining(T), T > timeThreshold$
$\Rightarrow submitBid(Y + minIncrement)$

r_2: $myBid(X), highestBid(Y), valuation(Z),$
$X < Y, Y + min\_increment < Z,$
$timeRemaining(T), T \leq timeThreshold$
$\Rightarrow submitBid(Z)$

r_3: $myBid(X), highestBid(Y), newValuation(Z),$
$X < Y, Y + minIncrement = W, W < Z,$
$timeRemaining(T), T > timeThreshold$
$\Rightarrow submitBid(W)$

r_4: $myBid(X), highestBid(Y), newValuation(Z),$
$X < Y, Y + minIncrement < Z,$
$timeRemaining(T), T \leq timeThreshold,$
$\Rightarrow submitBid(Z)$

r_5: $valuation(X), bidsAbove(X,N),$
$N \geq significantBidders, highestBid(Y)$
$\Rightarrow newValuation(Y + minIncrement)$

r_6: $valuation(X), highestBid(Y),$
$Y > (1 + significantIncrement) \times X$
$\Rightarrow newValuation(X + minIncrement)$

r_7: $Y > maxValuation \rightsquigarrow \neg newValuation(Y).$

The superiority relation between these rules is defined as follows: r_4 and r_5 have precedence over both r_1 and r_2.

The sets of complementary literals state that there can only be one amount to bid, and one new valuation, i.e.

- $C(submitBid(X)) = \{submitBid(Y) \mid Y \neq X\}$
- $C(valuation(X)) = \{valuation(Y) \mid Y \neq X\}$
- $C(newValuation(X)) = \{newValuation(Y) \mid Y \neq X\}$

Rules r_1 through r_4 model the bidding strategy. Rule r_1 states that if there is enough time remaining and the agent's current bid is not the highest, it should be increased by the minimum increment, provided that the current valuation allows so. Rule r_2 states that if the deadline is close and the bidder does not hold the item, a bid of the amount of the current valuation should be submitted immediately. Rules r_3 and r_4 are similar to r_1 and r_2, except that they apply only when the valuation is raised. Rules r_5 through r_7 determine if the valuation should be raised: rule r_5 and r_6 model the two conditions under which the valuation should be raised, while rule r_7 is a defeater modeling the fact that the bidder is under no circumstances willing to valuate the item above a given amount. The use of this defeater provides a strong modularity to the defeasible program. If for instance the user wanted to modify the above strategy with a statement of the form *"raise the valuation if the reservation price has not been met and the highest bid is above my current valuation"*, then (s)he just have to extend the corresponding logic program with the following rule :

r_9: $reservationNotMet, valuation(X),$
$highestBid(Y), Y > X$
$\Rightarrow newValuation(Y + minIncrement)$
($r_9 \succ r_2$ and $r_9 \succ r_3$)

without having to worry whether the reservation price is greater than his/her maximum valuation or not (raising the valuation above its maximum is prevented by rule r_7).

The control module of the bidding agent is described as a task structure in Figure 5[2]. At the beginning of the process, the agent activates the defeasible inference engine, which replies with a defeasible fact of the form $submitBid(X)$. Accordingly, the agent contacts the auctioneer to submit a bid of amount X. The auctioneer

[2]Again, this task structure can straightforwardly be translated into a defeasible theory.

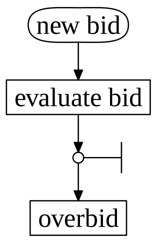

Figure 5: Bidder Control Module

replies by stating whether the bid is accepted or rejected. If the bid is accepted, the bidding agent introduces the facts $myBid(X)$ and $highestBid(X)$ into the knowledge base, waits until the auction broker notifies it that another participant has over-bidden, and then triggers the inference engine again. If on the other hand the submitted bid is rejected (i.e., another participant bided the same or a higher amount before the submitted bid was processed), the auction broker provides the current price quote (i.e., the highest bid). The agent then updates predicates *highestBid* and *bidsAbove* to reflect this, and triggers the inference engine to determine if a new bid should be submitted. If no bid is to be submitted, the agent waits for notifications of new bids issued by the other participants, before updating the knowledge base and triggering the inference engine again.

5. RELATED WORK

Bellucci and Zeleznikow [6] propose a legal negotiation support system tailored for Australian family law. Such a system is based on bi-directional fuzzy cognitive map, and it relies essentially on numerical (fuzzy) values given to the negotiation issues by the parties involved. Then those values are used to determine the order of decomposition and allocation of the content of the dispute. Finally trade-off (fuzzy) relationships are used in resolving the conflicts. It is clear that such an approach does not meet some of the desiderata listed in Section 2.1: in general it is not easy to define the appropriate value functions and the corresponding trade-off relationship, and numerical functions are not easily explainable, thus this system does not seem a natural and intuitive approach to the problem. Secondly, it does not take into account legal issues nor deal with the defeasibility of normative reasoning.

The architecture of another legal negotiation support system is outlined in [22]. The main feature is a user interface which uses Toulmin's diagram to represent the status of the negotiation. Additionally it supports several defeasible reasoning mechanisms (prioritized logic programming), and an adviser. However, the reasoning modules are used only for negotiation contents and not to describe protocols, but, as we have seen, in a legal context, protocols for negotiation can be defeasible. Finally, the advisor module is based on utility functions, and we have argued that this approach lacks suitability and comprehensibility.

The Michigan AuctionBot [25, 35] is an auction management server supporting the creation, location and enactment of different kinds of auctions. Users can manually interact with the system through an HTML-based interface, or alternatively, they can develop their own arbitrarily complex bidding agents, and connect them to the auction manager through a TCP/IP-level API. This API is generic enough to deal with several kinds of auctions (e.g. English, Dutch, double, etc.) through a common set of primitives. The AuctionBot has recently been used to host an trading agent competition [31], as has another similar platform called FishMarket [7, 16].

Unlike our work, neither FishMarket nor AuctionBot, address the issue of specifying bidding agent strategies.

More recently, yet another auction management server called *eAuctionHouse* has been released [30]. An interesting feature of eAuctionHouse is that it supports combinatorial auctions, that is, auctions in which a bidder may place bids on combinations of items, and may even issue simultaneous bids for several combinations, while ensuring that only one of his bids will win. eAuctionHouse also supports mobile agents that can issue bids on behalf of a user. However, the user is not allowed to specify his own bidding strategy: he has to choose between a set of predefined ones.

[17] classifies techniques for designing negotiation strategies into 3 categories: game-theoretic, heuristic and argumentation-based. The first approach, models a negotiation situation as a game, and attempts to find dominant strategies for a each participant by applying game theory techniques. For simple negotiation protocols, and under very specific assumptions about the rationality and attitude toward risk of each participant, this kind of analysis leads to simple and optimal strategies. However, very often these results do not generalize easily to more complex situations. In heuristic-based approaches on the other hand, a strategy consists of a family of tactics (i.e. a method for generating counter-offers), and a set of rules for selecting a particular tactic depending on the stage of the negotiation. The strategies for bargaining and bidding that we discussed in the previous section, can be seen as belonging to this family. Interestingly, our case studies show that defeasible logic is suitable to express both the tactics, and the rules for tactic selection. Finally, argumentation-based approaches extend heuristic ones, by introducing communication performatives such as threats (e.g., “this is my last offer”), rewards (e.g., “if you accept this offer, I will be more clever in our next negotiation”), etc. These performatives can be modeled in our approach through defeasible predicates shared by the participants of a negotiation. Concretely, in the bargaining case study of Section 4.1, in addition to having a predicate *offer*, the theory should also include predicates *threat*, *reward*, etc. as well as derivation rules for these predicates.

The use of logic-based formalisms with rule prioritisation in the context of automated negotiations has been studied in [9] and [29].

Garcia et al. [9] suggests to use a “variant” of defeasible logic to express strategies for agents trading over stock markets. The “variant” of defeasible logic that the authors consider, does not support an explicit ranking of the rules within a theory, but rather derives this ranking through a specificity criterion over arguments. Roughly speaking, an argument is more specific than (and therefore can defeat) another argument, if it takes into account more information. In addition to the fact that this approach allows “tie-breaks” between defeasible arguments, it may sometimes lead to counter-intuitive situations. Indeed, the semantics given by a user to the concept of “more information”, may potentially not be in line with that given by the defeasible logic’s inference method.

Reeves et al. [29], use Corteous Logic Programs (CLP) [14] to express knowledge about user preferences, constraints, and negotiation structures. The authors do not address the issue of specifying negotiation strategies directly, but rather that of deriving the set of negotiation structures that have to be carried out in order to transform a contract template into an executable contract. Interestingly, Defeasible Logic Programming (DLP) is more expressive than CLP [5], in the sense that it allows the user to express stratified theories. Hence, the question remains open whether it is possible to express useful contract templates in DLP that cannot be expressed in CLP.

6. CONCLUSION

In this paper we have proposed a qualitative logic-based approach to (automated) legal negotiation, and we have shown that it meets a set of desiderata that a negotiation framework should satisfy. In particular, we believe that the use of Defeasible Logic as the inference engine of the proposed architecture makes our approach “suitable” for legal negotiation in so far as it allows to capture effectively the defeasibility of law in a very close way.

It is worth noting that the present approach offers a great deal of flexibility when compared with other approaches, we have already argued that other approaches, and in particular those based on utility functions, fail to satisfy some of the criteria listed in section 2.1. Still, for some problems, very well understood and well behaving utility functions have been defined; in such cases it may be useful to combine the two approaches. Usually utility functions are applied to evaluate offers; thus, if one of the party is willing to concede on some issues, if the opponent party has conceded on other issues, then several utility functions can be defined with different weight for the conceded attributes. For example, let f_1, f_2 and f_3 be utility functions defined over a set of attributes. We can supplement the given functions with a set of defeasible rules to choose the most appropriate utility function according to the solved issues, for example:

$$\begin{aligned} r_1 &:\Rightarrow f_1 \\ r_2 &: \mathit{concession}_2 \Rightarrow f_2 \\ r_3 &: \mathit{concession}_3 \Rightarrow f_3 \end{aligned}$$

The meaning of the above rules is as follows: usually the offer is evaluated using f_1, however if some concessions are given by the counter-party, utility functions f_2 or f_3 are used instead.

One of the advantages of this method is that it is possible to formulate trade-off strategies in an easy way, and utility functions can be simplified.

7. REFERENCES

[1] Agorics Inc. A survey of auctions. `http://www.agorics.com/new.html`, 1996.

[2] Grigoris Antoniou, David Billington, Guido Governatori, and Michael Maher. On the modeling and analysis of regulations. In *Proceedings of the Australian Conference on Information Systems*, 1999.

[3] Grigoris Antoniou, David Billington, Guido Governatori, and Michael Maher. A flexible framework for defeasible logics. In *Proc. American National Conference on Artificial Intelligence (AAAI-2000)*, pages 401–405. AAAI/MIT Press, 2000.

[4] Grigoris Antoniou, David Billington, Guido Governatori, Michael Maher, and Andrew Rock. A family of defeasible reasoning logics and its implementation. In Werner Horn, editor, *ECAI 2000. Proceedings of the 14th European Conference on Artificial Intelligence*, Amsterdam, 2000. IOS Press.

[5] Grigoris Antoniou, Micheal J. Maher, and David Billington. Defeasible logic versus logic programming without negation as failure. *Journal of Logic Programming*, 41(1):45–57, 2000.

[6] Emilia Bellucci and John Zeleznikow. AI techniques for modelling legal negotiation. In *ICAIL-99*, pages 108–116. ACM Press, 1999.

[7] Instituto de Investigación en Inteligencia Artificial (IIIA). The FishMarket project. http://www.iiia.csic.es/Projects/fishmarket.

[8] eBay. Home page. `http://www.ebay.com`.

[9] A. Garcia, D. Gollapally, P. Tarau, and G. Simari. Deliberative stock market agents using Jinni and defeasible logic programming. In *Proc. of the ECAI Workshop on Engineering Societies in the Agents' World*, Berlin, Germany, August 2000. Springer Verlag.

[10] Thomas F. Gordon. *The Pleadings Game*. Kluwer Academic Press, Dordrecht, 1995.

[11] Guido Governatori and Michael J. Maher. An argumentation-theoretic characterization of defeasible logic. In Werner Horn, editor, *ECAI 2000. Proceedings of the 14th European Conference on Artificial Intelligence*, Amsterdam, 2000. IOS Press.

[12] Guido Governatori, Michael J. Maher, Grigoris Antoniou, and David Billington. Argumentation semantics for defeasible logics. In Riichiro Mizoguchi and John Slaney, editors, *PRICAI 2000: Topics in Artificial Intelligence*, volume 1886 of *LNAI*, pages 27–37, Berlin, 2000. Springer-Verlag.

[13] J.J. van Griethuysen, editor. *Concepts and Terminology for the Conceptual Schema and the Information Base*. Publ. nr. ISO/TC97/SC5/WG3-N695, ANSI, 11 West 42nd Street, New York, NY 10036, 1982.

[14] Benjamin N. Grosof. Prioritized conflict handling for logic programs. In J. Maluszynski, editor, *Proc. Int. Logic Programming Symposium*, pages 197–211. MIT Press, 1996.

[15] A.H.M. ter Hofstede. *Information Modelling in Data Intensive Domains*. PhD thesis, University of Nijmegen, Nijmegen, The Netherlands, 1993.

[16] IIIA, ISOCO and UPC (organizers). AMECIII trading agents' tournament. `http://www.iiia.csic.es/Projects/fishmarket/agents2000`, June 2000.

[17] N.R. Jennings, Parsons S, C. Sierra, and P. Faratin. Automated negotiation. In *Proc. of the Conference on Pratical Applications of Intelligent Agents and Multi-agent Systems (PAAM)*, Manchester, UK, 2000.

[18] Ronald Loui. Process and policy: resource-bounded non-demonstrative reasoning. *Computational Intelligence*, 14:1–38, 1998.

[19] Michael J. Maher. Propositional defeasible logic has linear complexity. Technical report, Department of Mathematical and Computer Sciences, Loyola University, Chicago, 2000.

[20] Michael J. Maher and Guido Governatori. A semantic decomposition of defeasible logic. In *Proc. American National Conference on Artificial Intelligence (AAAI-99)*, pages 299–305. AAAI Press, 1999.

[21] Michael J. Maher, Andrew Rock, Grigoris Antoniou, David Billignton, and Timothy Miller. Efficient defeasible reasoning systems. In *Proc. International Conference on Tool in Artificial Intelligence*. EEEI Computer Society Press, 2000.

[22] Katsumi Nitta, Masato Shibasaki, Akira Yamazaki, and Yoshiaki Yasumura. A legal negotiation support system. In *ICAIL-99*, pages 132–133. ACM Press, 1999.

[23] Donald Nute. Apparent obligation. In Donald Nute, editor, *Defeasible Deontic Logic*, pages 287–316. Kluwer, 1997.

[24] Donald Nute. Norms, priorities and defeasibility. In Paul McNamara and Henry Prakken, editors, *Norms, Logics and Information Systems. New Studies in Deontic Logic*, pages 83–100. IOS Press, Amsterdam, 1998.

[25] The University of Michigan Artificial Intelligence Laboratory. The Michigan Internet AuctionBot. http://auction.eecs.umich.edu.

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

[27] Henry Prakken. Relating protocols for dynamic dispute with logics for defeasible argumentation. *Synthese*, 127:187–219, 2001.

[28] Howard Raiffa. *The art and science of negotiation*. Harvard University Press, Cambridge, MA, 1982.

[29] Daniel M. Reeves, Michael P. Wellman, Benjamin N. Grosof, and Hoi Y. Chan. Automated negotiation from declarative contract descriptions. In *Seventeenth National Conference on Artificial Intelligence, Workshop on Knowledge-Based Electronic Markets(KBEM)*, Austin, Texas, July 30–31 2000. AAAI, AAAI Press.

[30] T. Sandholm. eMediator: A next generation electronic commerce server. In *International Conference on Autonomous Agents (AGENTS),*, Barcelona, Spain, June 2000.

[31] Betsy Strother. TAC: A Trading Agent Competition. *ACM SIGeCom Exchanges*, 1(1), August 2000.

[32] TradeOut. Home page. `http://www.tradeout.com`.

[33] G. Weiss, editor. *Multiagent Systems: A Modern Introduction to Distributed Artificial Intelligence*. MIT Press, 1999.

[34] M. Wooldridge. Intelligent agents. In Weiss [33].

[35] P.R. Wurman, M.P. Wellman, and W.E. Walsh. The Michigan Internet AuctionBot: A configurable auction server for human and software agents. In *Proc. of the 2nd Int. Conference on Autonomous Agents*. ACM Press, May 1998.