Languages for Negotiation

Michael Wooldridge and **Simon Parsons**[1]

Abstract. This paper considers the use of logic-based languages for multi-agent negotiation. We begin by motivating the use of such languages, and introducing a formal model of logic-based negotiation. Using this model, we define two important computational problems: the success problem (given a particular negotiation history, has agreement been reached?) and the guaranteed success problem (does a particular negotiation protocol guarantee that agreement will be reached?) We then consider a series of progressively more complex negotiation languages, and consider the complexity of using these languages. We conclude with a discussion on related work and issues for the future.

1 Introduction

Negotiation has long been recognised as a central topic in multi-agent systems [7, 5]. Much of this interest has arisen through the possibility of automated trading settings, in which software agents bargain for goods and services on behalf of some end-user [6].

One obstacle currently preventing the vision of agents for electronic commerce from being realised is the lack of standardised agent communication languages and protocols to support negotiation. To this end, several initiatives have begun, with the goal of developing such languages and protocols. Most activity in this area is currently focused on the FIPA initiative [2]. The FIPA community is developing a range of agent-related standards, of which the centrepiece is an agent communication language known as "ACL". This language includes a number of performatives explicitly intended to support negotiation [2, pp17–18].

Our aim in this paper is to consider the use of languages like FIPA's ACL for negotiation. In particular, we focus on the use of *logical* languages for negotiation. The use of logic for negotiation is not an arbitrary choice. For example, logic has proved to be powerful tool with which to study the expressive power and computational complexity of database query languages [3]. We believe it will have similar benefits for the analysis of negotiation languages.

In the following section, we introduce a general formal framework for logic-based negotiation. In particular, we define the concept of a negotiation history, and consider various possible definitions of what it means for negotiation to succeed on such a history: we refer to this as the *success* problem. In section 4, we define *protocols* for negotiation, and consider the problem of when a particular protocol guarantees that agreement between negotiation participants will be reached: we refer to this as the *guaranteed success* problem. In section 5, we consider three progressively more complex languages for negotiation. We begin with propositional logic, and show that, for this language, the guaranteed success problem is in the second tier of the polynomial hierarchy (it is Π_2^p-complete, and hence unlikely to be tractable even if we were given an oracle for NP-complete problems). We then present two further negotiation languages, which are more suited to electronic commerce applications; the second of these is in fact closely based on the negotiation primitives provided in the FIPA agent communication standard [2]. We show that the success problem for these languages is provably intractable (they have double exponential time lower bounds). We conclude by briefly discussing related work and issues for future work.

[1] Department of Computer Science, University of Liverpool, Liverpool L69 7ZF, United Kingdom. Email {M.J.Wooldridge, S.D.Parsons}@csc.liv.ac.uk

2 Preliminaries

We begin by assuming a non-empty set $Ag = \{1, \ldots, n\}$ of *agents*. These agents are the negotiation participants, and it is assumed they are negotiating over a finite set $\Omega = \{\omega, \omega', \ldots\}$ of *outcomes*. For now, we will not be concerned with the question of exactly what outcomes are, or whether they have any internal structure — just think of outcomes as possible states of affairs.

Each agent $i \in Ag$ is assumed to have preferences with respect to outcomes, given by a partial pre-order $\succeq_i \subseteq \Omega \times \Omega$. Following convention, we write $\omega \succeq_i \omega'$ to mean $(\omega, \omega') \in \succeq_i$.

Negotiation proceeds in a series of rounds, where at each round, every agent puts forward a proposal. A proposal is a *set of outcomes*, that is, a subset of Ω. The intuition is that in putting forward such a proposal, an agent is asserting that any of these outcomes is acceptable.

In practice, the number of possible outcomes will be prohibitively large. To see this, consider that in a domain where agents are negotiating over n attributes, each of which may take one of m values, there will be m^n possible outcomes. This means it will be impractical for agents to negotiate by explicitly enumerating outcomes in the proposals they make. Instead, we assume that agents make proposals by putting forward a formula of a *logical negotiation language* — a language for describing deals. In much of this paper, we will be examining the implications of choosing different negotiation languages, and in order to compare them, we must make certain general assumptions. The first is that a negotiation language $\mathcal{L}$ is associated with a set $wff(\mathcal{L})$ of *well-formed formulae* — syntactically acceptable constructions of $\mathcal{L}$. Next, we assume that $\mathcal{L}$ really *is* a logical language, containing the usual connectives of classical logic: "$\wedge$" (and), "$\vee$" (or), "$\neg$" (not), "$\Rightarrow$" (implies), and "$\Leftrightarrow$" (iff) [1, p32]. In addition, $\mathcal{L}$ is assumed to have a Tarskian satisfaction relation "$\models_\mathcal{L}$", which holds between outcomes Ω and members of $wff(\mathcal{L})$. We write $\omega \models_\mathcal{L} \varphi$ to indicate that outcome $\omega \in \Omega$ satisfies formula $\varphi \in wff(\mathcal{L})$. The classical connectives of $\mathcal{L}$ are assumed to have standard semantics, so that, for example, $\omega \models_\mathcal{L} \varphi \wedge \psi$ iff both $\omega \models_\mathcal{L} \varphi$ and $\omega \models_\mathcal{L} \psi$. If $\varphi \in wff(\mathcal{L})$, then we denote by $[\![\varphi]\!]_\mathcal{L}$ the set of outcomes that satisfy φ, that is, $[\![\varphi]\!]_\mathcal{L} = \{\omega \mid \omega \models_\mathcal{L} \varphi\}$.

As we noted above, negotiation proceeds in a series of rounds, where at each round, every agent puts forward a formula of $\mathcal{L}$ representing the proposal it is making. A single round is thus characterised by a tuple $\langle \varphi_1, \ldots, \varphi_n \rangle$, where for each $i \in Ag$, the formula $\varphi_i \in wff(\mathcal{L})$ is agent i's proposal. Let R be the set of all possible rounds. We use $r, r', \ldots$ to stand for members of R, and denote agent i's proposal in round r by $r(i)$.

A *negotiation history* is a finite sequence of rounds $(r_0, r_1, \ldots, r_k)$. Let $H = R^*$ be the set of all possible negotiation histories. We use $h, h', \ldots$ to stand for members of H. If $u \in I\!N$, then we denote the u'th round in history h by $h(u)$. Thus $h(0)$ is the first round in h, $h(1)$ is the second, and so on.

3 Types of Agreement

Given a particular negotiation history, an important question to ask is whether or not agreement has been reached with respect to this history. For many negotiation scenarios, this problem is far from trivial: it may well not be obvious to the negotiation participants that they have in fact made mutually acceptable proposals.

In fact, we can identify several different types of agreement condition, which may be used in different negotiation scenarios. It is assumed that the negotiation participants will settle on the agreement condition to be used before the actual negotiation process proper begins. The selection of an agreement condition is thus a *meta-negotiation* issue, which falls outside the scope of our work.

To understand what agreement means in our framework, it is helpful to view a negotiation history as a matrix of $\mathcal{L}$-formulae, as follows.

$$\begin{array}{cccc} \varphi_1^0 & \varphi_1^1 & \cdots & \varphi_1^k \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_n^0 & \varphi_n^1 & \cdots & \varphi_n^k \end{array}$$

In this matrix, φ_i^u is the proposal made by agent i in round $u \in I\!N$. The simplest type of agreement is where "all deals are still valid" — once an agent has made a proposal, then this proposal remains valid throughout negotiation. (One important implication of such agreement is that since all previous offers are still valid, it makes no sense for agents to make more restrictive proposals later in negotiation: we emphasise that our formal approach does not depend on this assumption — other types of agreement are possible, as we demonstrate below.)

In this case, determining whether agreement has been reached means finding at least one outcome $\omega \in \Omega$ such that every agent i has made a proposal $\varphi_i^{u_i}$ where $\omega \models_{\mathcal{L}} \varphi_i^{u_i}$. In other words, agreement will have been reached if every agent i has made a proposal $\varphi_i^{u_i}$ such that $[\![\varphi_1^{u_1}]\!]_{\mathcal{L}} \cap \cdots \cap [\![\varphi_n^{u_n}]\!]_{\mathcal{L}} \neq \emptyset$. This will be the case if the formula $\varphi_1^{u_1} \wedge \cdots \wedge \varphi_n^{u_n}$ is satisfiable. Given a history h, expressed as a matrix as above, agreement has been reached iff the following formula is satisfiable:

$$\bigwedge_{i \in Ag} \left(\bigvee_{u_i \in \{0, \ldots, k\}} \varphi_i^{u_i} \right) \tag{1}$$

Given a history $h \in H$, we denote the formula (1) for h by φ_h. We refer to the problem of determining whether agreement has been reached in some history h as the *success problem*. Note that the success problem can trivially be reduced to the satisfiability problem for the negotiation language using only polynomial time.

An obvious variant of this definition is where prior negotiation history is disregarded: the only proposals that matter are the most recent. Agreement will be reached in such a history iff the conjunction of proposals made on the final round of negotiation is satisfiable. The success condition is thus:

$$\bigwedge_{i \in Ag} \varphi_i^{|h-1|} \tag{2}$$

A third possible definition of agreement is that agents must converge on "equivalent" proposals. Such agreement is captured by the following condition.

$$\varphi_1^{|h-1|} \Leftrightarrow \cdots \Leftrightarrow \varphi_n^{|h-1|} \tag{3}$$

4 Protocols

Multi-agent interactions do not generally take place in a vacuum: they are governed by *protocols* that define the "rules of encounter" [7]. Put simply, a protocol specifies the proposals that each agent is allowed to make, as a function of prior negotiation history. Formally, a protocol π is a function $\pi : H \to \wp(R)$ from histories to sets of possible rounds. One important requirement of protocols is that the number of rounds they allow on any given history should be at most polynomial in the size of the negotiation scenario. The intuition behind this requirement is that otherwise, a protocol could allow an exponential number of rounds — since an exponential number of rounds could not be enumerated in practice, such protocols could never be implemented in any realistic domain.

We will say a history is *compatible* with a protocol if the rounds at each step in the history are permitted by the protocol. Formally, history h is compatible with π if the following conditions hold:

1. $h(0) \in \pi(\epsilon)$ (where ϵ is the empty history); and
2. $h(u) \in \pi((h(0), \ldots, h(u-1)))$ for $1 \leq u < |h|$.

Now, what happens if $\pi(h) = \emptyset$? In this case, protocol π says that there are no allowable rounds, and we say that negotiation has *ended*. The end of negotiation does not imply that the process has succeeded, but rather simply that the protocol will not permit it to continue further.

Notice that negotiation histories can in principle be unrealistically long. To see this, suppose that the set Ω of outcomes is finite. Then every agent has $2^{|\Omega|}$ possible proposals, meaning that even if an agent never makes the same proposal twice, negotiation histories can be exponentially long. We say protocol π is *efficient* if it guarantees that negotiation will end with a history whose length is polynomial in the size of Ω and Ag. Efficiency seems a reasonable requirement for protocols, as exponentially long negotiation histories could never be practical.

When we create an agent interaction protocol, we attempt to *engineer* the protocol so that it has certain desirable properties [7, pp20-22]. For example, we might aim to engineer the protocol so that it ensures any agreement is socially efficient (Pareto optimal), that the protocol is computationally simple, and so on.

In this paper, we will be concerned with just one property of protocols: whether or not they *guarantee success*. We will say a protocol π guarantees success if every negotiation history compatible with π ends with agreement being reached. Protocols that guarantee success are frequently desirable, for obvious reasons.

Before proceeding, we need to say something about how protocols are *represented* or *encoded*. (This is a technical matter that is important when we come to consider some decision problems later in the paper.) We will assume that (efficient) protocols are represented as a two-tape Turing machine: the machine takes as input a representation of prior negotiation history on its first tape, and writes as output the set of possible subsequent rounds on the second tape. We will further assume that the Turing machine requires time polynomial in the size of $|Ag \times \Omega|$ in order to carry out this computation.

5 Example Negotiation Languages

Example 1: Classical Propositional Logic. For the first example, we will assume that agents are negotiating over a domain that may be characterised in terms of a finite set of attributes, each of which may be either true ($\top$) or false ($\bot$). An outcome is thus an assignment of true or false to every attribute. The proposals possible in this kind of language are exactly the kind of outcomes typically considered in decision theory. For example, in the classic "oil wildcatter" problem agents might be involved in a negotiation about which of two oil fields to drill in, and proposals might be of the form:

- $drillField_A \wedge \neg drillField_B$
- $\neg drillField_A \wedge drillField_B$

The obvious language with which to express the properties of such domains is classical propositional logic, which we will call $\mathcal{L}_0$. The set $wff(\mathcal{L}_0)$ contains formulae constructed from a finite set of proposition symbols $\Phi = \{p, q, r, \ldots\}$ combined into formulae using the classical connectives "$\neg$" (not), "$\wedge$" (and), "$\vee$" (or), and so on. It is easy to see that the success problem for $\mathcal{L}_0$ histories will be NP-complete. More interesting is the fact that we can establish the complexity of the guaranteed success problem for $\mathcal{L}_0$. (In what follows, we assume some familiarity with complexity theory [4].)

Theorem 1 *The guaranteed success problem for efficient $\mathcal{L}_0$ protocols is complete for Π_2^p.*

Proof: We need to prove that: (i) the problem is in Π_2^p, and (ii) the problem is Π_2^p hard. To establish membership of Π_2^p, we define a Π_2^p alternating Turing machine M that accepts efficient $\mathcal{L}_0$ protocols which guarantee success, and rejects all others. The input to M will be an efficient $\mathcal{L}_0$ protocol π. The machine M runs the following algorithm:

1. universally select all histories h compatible with π;
2. existentially select an outcome ω;
3. accept if $\omega \models_{\mathcal{L}_0} \varphi_h$, otherwise reject.

Step (1) uses universal alternation to generate each history compatible with π; step (2) uses existential alternation to establish whether or not that history is successful; step (3) forces the machine to accept if every history compatible with the protocol is successful, and reject otherwise. At step (1), the histories selected will be at most polynomial in the size of Ω and Ac. Observe that the machine has just two alternations, a universal followed by an existential, and hence M is indeed a Π_2^p alternating Turing machine.

To show that the problem is Π_2^p hard, we reduce the $\text{QBF}_{2,\forall}$ problem — this is the quintessential Π_2^p complete problem [4, p96]. An instance of $\text{QBF}_{2,\forall}$ is given by a quantified boolean formula with the following structure:

$$\forall x_1, \ldots, x_k \; \exists y_1, \ldots, y_l \;\; \varphi(x_1, \ldots, x_k, y_1, \ldots, y_l)$$

Such a formula is true if for all assignments that we can give to boolean variables $x_1, \ldots, x_k$, there is some assignment we can give to boolean variables $y_1, \ldots, y_l$ such that $\varphi(x_1, \ldots, x_k, y_1, \ldots, y_l)$ is true. Here is an example of such a formula.

$$\forall x_1 \exists x_2 [(x_1 \vee x_2) \wedge (x_1 \vee \neg x_2)] \tag{4}$$

Formula (4) in fact evaluates to false. (If x_1 is false, there is no value we can give to x_2 that will make the body of the formula true.)

To reduce an instance (1) of $\text{QBF}_{2,\forall}$ to the $\mathcal{L}_0$ guaranteed success problem, we create an agent for each $\exists$-variable and $\forall$-variable in the QBF formula, and an additional agent for the body $\varphi(x_1, \ldots, x_k, y_1, \ldots, y_l)$. We then construct a protocol π so that:

- the agent corresponding to the body initially proposes $\varphi(x_1, \ldots, x_k, y_1, \ldots, y_l)$, and proposes "false" thereafter;
- each $\exists$-variable agent corresponding to y_i initially proposes $y_i \Leftrightarrow \top$, then $y_i \Leftrightarrow \bot$, and "$\bot$" thereafter;
- the nth $\forall$-variable agent proposes "$\bot$" until round n, then on round n is allowed to make two proposals, $y_n \Leftrightarrow \top$ and $y_n \Leftrightarrow \bot$, and proposes "$\bot$" thereafter.

The set of negotiation histories allowed by this protocol for example (4) can be described as follows.

agent for body:	$(x_1 \vee x_2) \wedge (x_1 \vee \neg x_2)$	$\bot$
agent for $\exists$-variable x_2:	$x_2 \Leftrightarrow \top$	$x_2 \Leftrightarrow \bot$
agent for $\forall$-variable x_1:	$\{x_1 \Leftrightarrow \top, x_1 \Leftrightarrow \bot\}$	$\bot$

The set notation in the third row denotes the proposals this agent is allowed to make at that step. The input formula will be true just in case every negotiation history compatible with this protocol is successful. Further, any negotiation history generated in this way will be polynomial in the number of clauses and the number of boolean variables in the original $\text{QBF}_{2,\forall}$ formula, and the reduction can clearly be done in polynomial time. Hence any instance of $\text{QBF}_{2,\forall}$ can be reduced in polynomial time to the problem of determining whether or not an efficient $\mathcal{L}_0$ protocol guarantees success, and we are done. $\square$

Note that Π_2^p-complete problems are generally reckoned to be worse than, say, co-NP-complete or NP-complete problems, although the precise status of such problems in the relation to these classes is not currently known for sure [4]. Theorem 1 should therefore be regarded as an extremely negative result.

An obvious question to ask is whether the complexity of the guaranteed success problem can be reduced in some way. There are two main factors that lead to the overall complexity of the problem: the complexity of the underlying negotiation language, and the "branching factor" of the protocol. It is possible to prove that if we chose a negotiation language whose satisfiability problem was in P, then the complexity of the corresponding guaranteed success problem would be reduced one level in the polynomial hierarchy — in fact it would be co-NP-complete (i.e., Π_1^p-complete).

With respect to the branching factor of the protocol, suppose we have a *deterministic* $\mathcal{L}_0$ protocol π — one in which $|\pi(h)| \leq 1$ for all $h \in H$. Since such protocols generate only one history, then it is not hard to see that the corresponding guaranteed success problem will be NP-complete. Of course, determinism is a far too restrictive property to require of realistic protocols.

Example 2: A Language for Electronic Commerce. Propositional logic is a simple and convenient language to analyse, but is unlikely to be useful for many realistic negotiation domains. In this example, we focus on somewhat more realistic *e-commerce* scenarios, in which agents negotiate to reach agreement with respect to some financial transaction [6]. We present a negotiation language $\mathcal{L}_1$ for use in such scenarios.

We begin by defining the outcomes that agents are negotiating over. The idea is that agents are trying to reach agreement on the values of a finite set $V = \{v_1, \ldots, v_m\}$ of *negotiation issues* [8, pp181–182], where each issue has a natural number value. An outcome $\omega \in \Omega$ for such a scenario is thus a function $\omega : V \to \mathbb{N}$, which assigns a natural number to each issue.

In order to represent the proposals that agents make in such a scenario, we use a subset of first-order logic. We begin by giving some examples of formulae in this subset.

- $(price = 20) \wedge (warranty = 12)$
  "the price is \$20 and the warranty is 12 months"
- $(15 \leq price \leq 20) \wedge (warranty = 12)$
  "the price is between \$15 and \$20 and the warranty is 12 months"
- $(price + warrantyCost \leq 2000)$
  "price plus warranty is less than \$2000"

Formally, $\mathcal{L}_1$ is the subset of first-order logic containing: a finite set V of variables, (with at least one variable for each negotiation issue); a set C of constants, one for each natural number; the binary addition function "$+$"; the equality relation "$=$"; and the less-than relation "$<$".

There is both good news and bad news about $\mathcal{L}_1$: the good news is that it is decidable; the bad news is that it is *provably* intractable. In fact, we can prove that $\mathcal{L}_1$ has a double exponential time lower bound. In what follows, $TA[t(n), a(n)]$ is used to denote the class of problems that may be solved by an alternating Turing machine using at most $t(n)$ time and $a(n)$ alternations on inputs of length n [4, p104].

Theorem 2 *The success problem for $\mathcal{L}_1$ is complete for $\bigcup_{k>0} TA[2^{2^{n^k}}, n]$.*

Proof: Follows from the fact that $\mathcal{L}_1$ formulae may be reduced in linear time to formulae of Presburger arithmetic and *vice versa* [1, p250]. Presburger arithmetic is a subset of first-order logic containing equality, the successor function $S : \mathbb{N} \to \mathbb{N}$ and constant $\mathbf{0}$, the less than relation "$<$", and the addition function "$+$". Formulae of Presburger arithmetic are interpreted over a structure $\langle \mathbb{N}, 0, S, <, + \rangle$, where the components of this structure have the obvious meaning. Since the problem of deciding whether a formula of Presburger arithmetic is true is complete for $\bigcup_{k>0} TA[2^{2^{n^k}}, n]$, (see e.g., [4, p105]), and this complexity class is closed under polynomial time reductions, the result follows easily. ▫

The details of the class $TA[t(n), a(n)]$ are perhaps not very important for the purposes of this example. The crucial point is that any algorithm we care to write that will solve the general $\mathcal{L}_1$ success problem will have *at least* double exponential time complexity. It follows that such an algorithm is highly unlikely to be of any practical value. With respect to the guaranteed success problem for $\mathcal{L}_1$, we note that since the success problem gives a lower bound to the corresponding guaranteed success problem, the $\mathcal{L}_1$ guaranteed success problem will be at least $\bigcup_{k>0} TA[2^{2^{n^k}}, n]$ hard.

Example 3: A negotiation meta-language. The language used in the previous example is suitable for stating deals, and is thus sufficient for use in scenarios in which agents negotiate by just trading such deals. However, as discussed in [8], the negotiation process is more complex for many scenarios, and agents must engage in persuasion to get the best deal. Persuasion requires more sophisticated dialogues, and, as a result, richer negotiation languages. One such language, based on the negotiation primitives provided by the FIPA ACL [2], and related to [8], includes the illocutions shown in Table 1[2]. In this table, φ is a formula of a language such as $\mathcal{L}_0$ or $\mathcal{L}_1$. In this sense, the language which includes the illocutions is a *meta-language* for negotiation — a language for talking about proposals. For the rest of this example, we will consider a language $\mathcal{L}_2$ which consists of exactly those illocutions in Table 1, where φ is a formula in $\mathcal{L}_1$.

Illocution	Meaning
$request(i, j, \varphi)$	a request from i to j for a proposal based on φ
$offer(i, j, \varphi)$	a proposal of φ from i to j
$accept(i, j, \varphi)$	i accepts proposal φ made by agent j
$reject(i, j, \varphi)$	i rejects proposal of φ made by agent j
$withdraw(i, j)$	i withdraws from negotiation with j

Table 1. Illocutions for the negotiation language $\mathcal{L}_2$.

These illocutions work as follows. There are two ways in which a negotiation can begin, either when one agent makes an *offer* to another, or when one makes a *request* to another. A request is a semi-instantiated offer. For example, the following illocution

$$request(i, j, (price = ?) \wedge (warranty = 12))$$

is interpreted as "If I want a 12 month warranty, what is the price?".

Proposals are then traded in the usual way, with the difference that an agent can reply to a proposal with a *reject*, explicitly saying that a given proposal is unacceptable, rather than with a new proposal. Negotiation ceases when one agent *accept*s an offer or *withdraw*s from negotiation. Note that this protocol assumes two agents are engaged in the negotiation. (Many-many negotiations are handled in [8] by many simultaneous two-way negotiations.)

To further illustrate the use of $\mathcal{L}_2$, consider the following short negotiation history between two agents negotiating over the purchase of a used car:

1. $request(a, b, (price \leq 4000) \wedge (model = ?) \wedge (age = ?))$
2. $offer(b, a, (price = 3500) \wedge (model = Escort) \wedge (age = 8))$
3. $reject(a, b, (price = 3500) \wedge (model = Escort) \wedge (age = 8))$
4. $offer(b, a, (price = 3900) \wedge (model = Golf) \wedge (age = 6))$
5. $offer(a, b, (price = 3200) \wedge (model = Golf) \wedge (age = 6))$
6. $offer(b, a, (price = 3400) \wedge (model = Golf) \wedge (age = 6))$
7. $accept(a, b, (price = 3400) \wedge (model = Golf) \wedge (age = 6))$

Broadly speaking, the illocutions in $\mathcal{L}_2$ are syntactic sugar for the kinds of proposal that we have discussed above: we can map them into $\mathcal{L}_1$ and hence into the framework introduced in section 2. To do this we first need to extend the condition for agreement. In the case where we have two agents, a and b negotiating, the agreement condition we use is a combination of (2) and (3):

$$(\varphi_a^{|h-1|} \wedge \varphi_b^{|h-1|}) \wedge (\varphi_a^{|h-1|} \Leftrightarrow \varphi_b^{|h-1|}) \quad (5)$$

Thus the agents must not only make mutually satisfiable proposals on the final round, they must make equivalent proposals. Given this, we can prove the following result.

[2] Note that the language proposed in [8] also includes illocutions which include the reason for an offer. We omit discussion of this facility here. We also omit the timestamp from the illocutions.

Theorem 3 *The augmented success problem for $\mathcal{L}_2$ is complete for $\bigcup_{k>0} TA[2^{2^{n^k}}, n]$.*

Proof: The result follows from Theorem 2 and the fact that we can define a linear time transformation between $\mathcal{L}_2$ and $\mathcal{L}_1$ histories, which preserves the conditions of success. We will in fact define a mapping which translates from $\mathcal{L}_2$ illocutions to $\mathcal{L}_1$ formulae — the mapping can be easily be extended to histories. Three $\mathcal{L}_2$ illocutions can be re-written directly:

- $\mathit{offer}(i, j, \varphi)$ becomes a proposal φ;
- $\mathit{accept}(i, j, \varphi)$ becomes a proposal φ which matches the last proposal;
- $\mathit{reject}(i, j, \varphi)$ becomes a proposal $\neg\varphi$.

These illocutions then fit precisely into the framework defined above, and success occurs in precisely the same situation — when (5) is satisfiable — once the last proposal, the one which makes (5) satisfiable, is echoed by the second agent. The remaining two illocutions can be captured by:

- $\mathit{request}(i, j, \varphi)$ becomes a proposal φ in which some attributes are of the form $(\mathit{value}_{min} \leq \mathit{attribute} \leq \mathit{value}_{max})$;
- $\mathit{withdraw}(i, j)$ becomes "$\perp$".

A proposal "$\perp$" immediately makes (5) unsatisfiable, and the negotiation terminates, exactly as one would expect of a *withdraw*. A proposal in which some attributes A_i are of the form $(\mathit{value}_{min} \leq \mathit{attribute} \leq \mathit{value}_{max})$ and others A_j have more restricted values leads immediately to the satisfiability of (5) if the response is a proposal which agrees on the A_j and has any value for the A_i (since these will agree with the intervals $[\mathit{value}_{min}, \mathit{value}_{max}]$). Since the transformation will clearly be linear in the size of the history, the result follows. □

There is also the question of whether success can be guaranteed when negotiating in $\mathcal{L}_2$, and this, of course, depends upon the protocol used. Table 2 gives the protocol used in [8]. We will call this $\pi_{\mathcal{L}_2}$.

Agent i says	Agent j replies
$\mathit{request}(i, j, \varphi_i^u)$	$\mathit{offer}(j, i, \varphi_j^u)$
$\mathit{offer}(i, j, \varphi_i^u)$	$\mathit{offer}(j, i, \varphi_j^u)$, or $\mathit{accept}(j, i, \varphi_j^u)$, or $\mathit{reject}(j, i, \varphi_i^u)$, or $\mathit{withdraw}(j, i)$
$\mathit{reject}(i, j, \varphi)$	$\mathit{offer}(j, i, \varphi_j^u)$ or $\mathit{withdraw}(j, i)$
$\mathit{accept}(i, j, \varphi_j^{u-1})$	end of negotiation
$\mathit{withdraw}(i, j)$	end of negotiation

Table 2. The protocol $\pi_{\mathcal{L}_2}$ for $\mathcal{L}_2$ at the uth step of the negotiation.

Clearly this protocol can lead to negotiations which never terminate (since it is possible for agents to trade the same pair of unacceptable offers for ever). However, it is not unreasonable to insist that conditions are placed upon the protocol in order to ensure that this does not happen and that negotiations eventually terminate. One such condition is that agents make concessions at each stage, that is, that each offer made by an agent is less preferable to that agent than any of its predecessors. Under this condition, and assuming that agents withdraw once φ drops below some threshold, we have:

Theorem 4 *Protocol $\pi_{\mathcal{L}_2}$ guarantees success.*

Proof: Consider an ongoing negotiation. If we can show that the negotiation terminates, then success is guaranteed. Now, if i generates the illocution $\mathit{withdraw}(i, j)$ or $\mathit{accept}(i, j, \varphi)$, negotiation terminates. If i generates $\mathit{reject}(i, j, \varphi)$, then either j withdraws and the negotiation terminates after the next step, or j responds with an offer. Similarly, if i generates $\mathit{offer}(i, j, \varphi)$, either the negotiation terminates after the next step, or j issues an offer or a reject. A reject will, of course, generate a withdrawal or an offer. Thus the only way that the negotiation can continue is through the exchange of offers, albeit offers interspersed with *reject*s. Since both agents always concede, any offer an agent makes will be less acceptable to it than the previous offer it made, and so, after making a number of offers, the value of the deal being offered will fall beneath the threshold. At this point the agent will withdraw, and the negotiation will terminate. □

One simple scenario which is captured by $\pi_{\mathcal{L}_2}$ is that in which one agent, i say, rejects every offer made by the other, j, until suitable concessions have been gained. Of course, provided that the end-point is acceptable for j, there is nothing wrong with this — and if the concession j is looking for are too severe, then j will withdraw before making an acceptable offer.

6 Discussion

This paper has identified two important computational problems in the use of logic-based languages for negotiation — the problem of determining if agreement has been reached in a negotiation, and the problem of determining if a particular negotiation protocol will lead to an agreement. Both these problems are computationally hard. In particular the paper showed the extent of the problems for some languages that could realistically be used for negotiations in electronic commerce. This effort is thus complementary to work on defining such languages. Obvious future lines of work are to consider the impact of these results on the design of negotiation languages and protocols, and to extend the work to cover more complex languages. In particular, we are interested in extending the analysis to consider the use of argumentation in negotiation [8].

Acknowledgements: This work was supported by the EPSRC under grant GR/M07076.

REFERENCES

[1] H. B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, 1972.

[2] FIPA. Specification part 2 — Agent communication language, 1999. The text refers to the specification dated 16 April 1999.

[3] N. Immerman. *Descriptive Complexity*. Springer-Verlag: Berlin, Germany, 1999.

[4] D. S. Johnson. A catalog of complexity classes. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science Volume A: Algorithms and Complexity*, pages 67–161. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1990.

[5] S. Kraus. Negotiation and cooperation in multi-agent environments. *Artificial Intelligence*, 94(1-2):79–98, July 1997.

[6] P. Noriega and C. Sierra, editors. *Agent Mediated Electronic Commerce (LNAI Volume 1571)*. Springer-Verlag: Berlin, Germany, 1999.

[7] J. S. Rosenschein and G. Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*. The MIT Press: Cambridge, MA, 1994.

[8] C. Sierra, N. R. Jennings, P. Noriega, and S. Parsons. A framework for argumentation-based negotiation. In M. P. Singh, A. Rao, and M. J. Wooldridge, editors, *Intelligent Agents IV (LNAI Volume 1365)*, pages 177–192. Springer-Verlag: Berlin, Germany, 1998.