On the use of logic in negotiation*

Michael Wooldridge and Simon Parsons
Department of Computer Science
University of Liverpool, Liverpool L69 7ZF
United Kingdom
{m.j.wooldridge, s.d.parsons}@csc.liv.ac.uk

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, 4]. 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 [5].

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. We believe it will have similar benefits for the analysis of negotiation languages.

We consider two distinct aspects of logical negotiation languages:

- *Adequacy*: Which negotiation languages are appropriate for which application domains? Is a given language sufficiently expressive for a particular domain?
- *Simplicity*: Is a chosen negotiation language likely to be usable in practice?

Of course, there is a tradeoff between adequacy and simplicity: the more powerful a language is, the less tractable it becomes.

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 investigate how the complexity of the problem varies, depending on the choice of negotiation language and the properties of the protocol.

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.

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.

*This research was supported by the EPSRC under grant GR/M07076.

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:

$$R = \underbrace{wff(\mathcal{L}) \times \cdots \times wff(\mathcal{L})}_{|Ag| \text{ times}}$$

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.

Complexity Concepts and Classes

Throughout the paper, we make use of the terminology and tools of complexity theory. Although we provide a summary of main concepts from complexity theory that we use, we emphasise that a detailed presentation is beyond the scope of this paper. We refer the reader to [3, 6] for details. We start from the complexity classes P (of languages/problems that may be recognised/solved in deterministic polynomial time), and NP (of languages/problems that may be recognised/solved in non-deterministic polynomial time). If $\mathcal{C}_1$ and $\mathcal{C}_2$ are complexity classes, then we denote by $\mathcal{C}_1^{\mathcal{C}_2}$ the class of languages/problems that are in $\mathcal{C}_1$ assuming the availability of an oracle for languages/problems in $\mathcal{C}_2$ [6, pp415–417]. Thus, for example, NP^{NP} denotes the class of languages/problems that may be recognised/solved in non-deterministic polynomial time, assuming the presence of an oracle for languages/problems in NP. A language that is complete for NP^{NP} would thus be NP-complete even if we had "free" answers to NP-complete problems (such as propositional logic satisfiability). We define the *polynomial hierarchy* with reference to these concepts [6, pp423–429]. First, define

$$\Sigma_0^p = \Pi_0^p = \text{P}$$

Thus both Σ_0^p and Π_0^p denote the classes of languages/problems that may be recognised/solved in deterministic polynomial time. We then inductively define the remaining tiers of the hierarchy, as follows:

$$\Sigma_{u+1}^p = \text{NP}^{\Sigma_u^p} \qquad \Pi_{u+1}^p = \text{co-}\Sigma_{u+1}^p$$

Thus Σ_1^p is simply the class NP, and Π_1^p is the class co-NP, while $\Sigma_2^p = \text{NP}^{\text{NP}}$ and $\Pi_2^p = \text{co-NP}^{\text{NP}}$.

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.

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

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 clearly be reduced to the satisfiability problem for the negotiation language using only polynomial time. The types of agreement we consider in this paper are all variants of satisfiability. However, it is important to understand that (1) is *by no means* the only type of agreement that we can define.

A different type of agreement 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: where the proposals made agree on all particulars. Such agreement is captured by the following condition.

$$\varphi_1^{|h-1|} \Leftrightarrow \cdots \Leftrightarrow \varphi_n^{|h-1|} \quad (3)$$

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 $\pi : H \rightarrow \wp(R)$ 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]. Examples of such properties include:

- *Social efficiency*: by which we mean that any outcome is guaranteed to be Pareto optimal.
- *Stability*: by which we mean negotiation participants have no incentive to diverge from the protocol.
- *Simplicity*: by which we mean that agents do not have to work hard to determine the best strategy.

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 desirable, for obvious reasons. But consider the nature of this problem. In general, a protocol allows *branching* during the negotiation process. This branching arises because negotiation participants are allowed to make a number of proposals at each round. It is easy to see that the number of negotiation histories of length l compatible with a negotiation protocol with branching factor b will be b^l, that is, exponential in the length of the protocol. Thus determining whether or not a protocol guarantees success will intuitively involve solving an exponential number of individual success problems, which are themselves logical satisfiability problems. This suggests that the guaranteed success problem is likely to be computationally hard; in the next section, when we consider some concrete negotiation languages, we will see just how hard it actually is.

Before proceeding, however, 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

In this section, we present a series of progressively more complex negotiation languages and protocols. We begin with propositional logic. Although this logic is not appropriate for many negotiation domains, it is a useful starting point for our analysis, and the results we establish for propositional logic can be regarded as "lower bounds" for other, more expressive 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 $\mathcal{L}_0$.

Theorem 1 (From [9].) *The guaranteed success problem for efficient $\mathcal{L}_0$ protocols is Π_2^p-complete.*

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 [3]. 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. To make this concrete, let us consider the subset $\mathcal{L}_0^{HC}$ of $\mathcal{L}_0$ in which formulae are restricted to be conjunctions of *Horn clauses*. A clause is said to be Horn if it contains at most one positive (unnegated) literal. It is well known that there is a deterministic polynomial time algorithm for determining the satisfiability of $\mathcal{L}_0^{HC}$ [6, pp78–79]. We can prove the following.

Theorem 2 *The guaranteed success problem for efficient $\mathcal{L}_0^{HC}$ protocols is Π_1^p-complete.*

Proof: We need to prove that: (i) the problem is in Π_1^p, and (ii) the problem is Π_1^p hard. To establish membership of Π_1^p, we define a Π_1^p Turing machine M that accepts efficient $\mathcal{L}_0^{HC}$ protocols which guarantee success, and rejects all others. The input to M will be an efficient $\mathcal{L}_0^{HC}$ protocol π. The machine M runs the following algorithm:

1. universally select all histories h compatible with π;
2. accept if φ_h is satisfiable, otherwise reject.

Notice that the second step can be done in deterministic polynomial time. Since the algorithm requires a single, universal alternation, it is a Π_1^p algorithm.

To show that the problem is Π_1^p-hard, we show how the problem of deciding validity of $\mathcal{L}_0$ formulae can be reduced to determining whether or not an efficient $\mathcal{L}_0^{HC}$ protocol guarantees success. The input to the propositional validity problem is a formula φ, and we can assume without loss of generality that φ is in Conjunctive Normal Form (CNF). A CNF formula has the form

$$C_1 \wedge \cdots \wedge C_m$$

where each C_n $(1 \leq n \leq m)$ is a disjunction of literals. The idea is to create a protocol π so that:

- there is an agent corresponding to each clause C_n;
- there is a single agent for the set of atomic propositions Φ.

The protocol then works as follows:

- an agent corresponding to a clause $C_m = l_1 \vee \cdots \vee l_o$ proposes each of its literals it turn, and proposes "$\perp$" thereafter;
- the agent corresponding to the set of atomic propositions takes a single different proposition $p \in \Phi$ on each round, and on this round is allowed to make two proposals: $p \Rightarrow \perp$ and $\neg p \Rightarrow \perp$. When the set Φ is exhausted, the agent proposes $\perp$.

(Notice that the proposals made are in $\mathcal{L}_0^{HC}$.) The protocol ensures that negotiation ends when both the clause agents and the Φ agent can make no more non-$\perp$ proposals.

To see how the reduction works, consider an input formula

$$\varphi = \underbrace{(p \vee q)}_{C_1} \wedge \underbrace{(p \vee \neg q)}_{C_2}$$

that we wish to test for validity. Then the protocol that we create from this formula allows the following negotiation histories:

agent for C_1:	p	q
agent for C_2:	p	$\neg q$
Φ agent:	$\{p \Rightarrow \perp, \neg p \Rightarrow \perp\}$	$\{q \Rightarrow \perp, \neg q \Rightarrow \perp\}$

There will clearly be four histories of this protocol, each history corresponding to one possible valuation that could be given to the primitive propositions Φ. The protocol will clearly guarantee success just in case the input formula is valid. $\square$

Language	Protocol	
	Non-deterministic	Deterministic
$\mathcal{L}_0$	Π_2^p-complete	NP-complete
$\mathcal{L}_0^{HC}$	Π_1^p-complete	P

Table 1: The complexity of the guaranteed success problem for deterministic and non-deterministic protocols, using variants of propositional logic.

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 $h \in H$. Such a protocol is guaranteed to produce a single negotiation history. This allows us to easily establish the following result for deterministic $\mathcal{L}_0$ protocols.

Theorem 3 *The guaranteed success problem for efficient deterministic $\mathcal{L}_0$ protocols is* NP*-complete.*

Similarly, we can establish the following for $\mathcal{L}_0^{HC}$.

Theorem 4 *The guaranteed success problem for efficient deterministic $\mathcal{L}_0^{HC}$ protocols is in* P.

Table 1 summarises our complexity results for protocols based on propositional logic. We remark that determinism is a far too restrictive property to require of realistic protocols.

Before leaving this section, we present a simple example of a protocol that guarantees success for agents negotiating using a subset of propositional logic. We refer to the subset of propositional logic in question as $\mathcal{L}_0^*$. Formulae of $\mathcal{L}_0^*$ are constrained to take the form

$$l_0 \wedge \cdots \wedge l_m$$

where each l_m is a *literal*, that is, either an element of Φ or the negation of an element of Φ. If $\varphi \in \mathit{wff}(\mathcal{L}_0^*)$, then we denote the set of literals in φ by $\mathit{lit}(\varphi)$.

Given two formulae φ and ψ, we say that φ is a *concession* with respect to ψ if φ is "less constraining" than ψ, i.e., if $[\![\psi]\!]_{\mathcal{L}_0^*} \subset [\![\varphi]\!]_{\mathcal{L}_0^*}$. It should be easy to see that by this definition, φ is less constraining than ψ if φ is a subformula of ψ and $\mathit{lit}(\varphi) \subset \mathit{lit}(\psi)$. So, for example, $p \wedge \neg q$ represents a concession with respect to $p \wedge \neg q \wedge \neg r$, which in turn represents a concession with respect to $p \wedge \neg q \wedge \neg r \wedge s$.

We now turn to the protocol in question. The protocol is simply the monotonic concession protocol of Rosenschein and Zlotkin [7, pp40–41], recast into our logical framework. This protocol is defined by the following two rules:

1. on the first round, every agent is allowed to propose any formula of $\mathcal{L}_0^*$;
2. on round u $(u > 0)$, one agent $i \in Ag$ must make a proposal φ_i^u such that φ_i^u represents a concession with respect to φ_i^{u-1}; every other agent puts forward the same proposal that it put forward on round $u - 1$.

It is not difficult to prove the following.

Theorem 5 *The monotonic concession protocol for $\mathcal{L}_0^*$ guarantees success. Moreover, it guarantees that agreement will be reached in $O(|Ag \times \Phi|)$ rounds.*

Note that when using this protocol, the obvious strategy for an agent to play in order to ensure that negotiation concludes as quickly as possible involves conceding literals that *clash* with those of other negotiation participants. For example, in a two agent negotiation scenario, suppose that agent 1 proposed $p \wedge \neg q$ and agent 2 proposed $p \wedge q$. Then the obvious concession for agent 2 to make would involve proposing p subsequently. Otherwise, successful termination would require a further negotiation round.

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 [5]. 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 $\omega : V \to I\!N$, 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"
- $\exists n \cdot warranty \geq 12$
  "the warranty is longer than 12 months"

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 [3, p104].

Theorem 6 (From [9].) *The success problem for $\mathcal{L}_1$ is complete for $\bigcup_{k>0} TA[2^{2^{n^k}}, n]$.*

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.

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 2: Illocutions for the negotiation language $\mathcal{L}_2$.

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 2[1]. 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 2, where φ is a formula in $\mathcal{L}_1$.

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):

[1] 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.

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

Table 3: The protocol $\pi_{\mathcal{L}_2}$ for $\mathcal{L}_2$ at the uth step of the negotiation.

$$(\varphi_a^{|h-1|} \wedge \varphi_b^{|h-1|}) \wedge (\varphi_a^{|h-1|} \Leftrightarrow \varphi_b^{|h-1|}) \qquad (4)$$

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.

Theorem 7 (From [9].) *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 6 and the fact that we can define a polynomial time reduction 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:

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

These illocutions then fit precisely into the framework defined above, and success occurs in precisely the same situation — when (4) is satisfiable — once the last proposal, the one which makes (4) satisfiable, is echoed by the second agent. The remaining two illocutions can be captured by:

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

A proposal "$\perp$" immediately makes (4) 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 $(value_{min} \leq attribute \leq value_{max})$ and others A_j have more restricted values leads immediately to the satisfiability of (4) 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 $[value_{min}, value_{max}]$). 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 3 gives the protocol used in [8]. We will call this $\pi_{\mathcal{L}_2}$.

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 — this is essentially the monotonic concession protocol discussed easier. Under this condition, and assuming that agents withdraw once φ drops below some threshold, we have:

Theorem 8 (From [9].) *Protocol $\pi_{\mathcal{L}_2}$ guarantees success.*

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, and the main contribution of this paper was to show quite how hard they are. 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].

References

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

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

[3] 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.

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

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

[6] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley: Reading, MA, 1994.

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

[8] Carles Sierra, Nick R. Jennings, Pablo Noriega, and Simon 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.

[9] M. Wooldridge and S. Parsons. Languages for negotiation. In *Proceedings of the Fourteenth European Conference on Artificial Intelligence (ECAI-2000)*, Berlin, Germany, 2000.