Verifiable Semantics for Agent Communication Languages

Michael Wooldridge
Queen Mary and Westfield College
Department of Electronic Engineering
London E1 4NS, United Kingdom
M.J.Wooldridge@qmw.ac.uk

Abstract

This paper examines the issue of developing semantics for agent communication languages. In particular, it considers the problem of giving a verifiable *semantics for such languages — a semantics where conformance (or otherwise) to the semantics could be determined by an independent observer. These problems are precisely defined in an abstract formal framework. Using this framework, a number of example agent communication frameworks are examined with respect to the problem of verifying conformance. A discussion is then presented, of the various options open to designers of agent communication languages, with respect the problem of verifying conformance.*

1. Introduction

Multi-agent systems are currently a major area of research and development activity. One of the main reasons for this level of interest is that multi-agent systems are seen as a key enabling technology for the Internet-wide electronic commerce systems that are widely predicted to emerge in the near future [11]. If this vision of large-scale, open multi-agent systems is to be realised, then the fundamental problem of *inter-operability* must be addressed. It must be possible for agents built by different organisations, using different hardware and software platforms, to communicate via a common language with a universally agreed semantics.

The inter-operability requirement has led to the development of several standardised *agent communication languages* (ACLs) [16, 10]. However, to gain acceptance, particularly for sensitive applications such as electronic commerce, it must be possible to determine whether or not any system that claims to *conform* to an ACL standard actually does so. We say that an ACL standard is *verifiable* if it enjoys this property. Unfortunately, verifiability has to date received little attention by the standards community (although it *has* been recognised as an issue [10, p46]). In this paper, we establish a simple formal framework that allows us to precisely define what it means for an ACL to be verifiable. This framework is defined in section 2, following a brief discussion of the background to this work. We then formally define what it means for an ACL to be verifiable in section 3, and discuss the practical implications of these definitions in section 3.1. In section 4, we give examples of some ACLs, and show that some of these are verifiable, while others are not. Finally, in section 5, we discuss the implications of our results, with emphasis on future directions for work on verifiable ACLs.

Background

Current activities in the area of ACL semantics trace their origins to the work of Austin [2]. He noted that a certain class of natural language utterances — hereafter referred to as *speech acts* — had the characteristics of *actions*, in the sense that they change the state of the world in a way analogous to physical actions. Austin's work was refined and considerably extended by Searle, in his 1969 book *Speech Acts* [18]. Searle attempted to derive the "necessary and sufficient" conditions for the successful performance of speech acts, and gave a five-point typology of various different classes of speech acts.

Speech acts were introduced to the Artificial Intelligence (AI) community largely through the work of Cohen and Perrault, who in [8] gave an account of the semantics of speech acts by using techniques developed in AI planning research. They showed how the pre- and post-conditions of speech acts such as *request* could be represented in a multi-modal logic containing operators for describing the *beliefs*, *abilities*, and *wants* of the participants in the speech act.

While this plan-based theory of speech acts was a major step forward, it was recognised that a theory of speech acts should be rooted in a more general theory of *rational action*. This observation led Cohen and Levesque to develop a theory in which speech acts were modelled as actions performed by rational agents in the furtherance of their intentions [6].

In the early 1990s, the DARPA Knowledge Sharing Effort (KSE) began to develop the Knowledge Query and Manipulation Language (KQML) as a common framework via which multiple expert systems (cf. agents) could exchange knowledge [16]. KQML is essentially an "outer" language for messages: it defines a simple LISP-like format for messages, and 41 *performatives*, or message types, that define the intended meaning of a message. Example KQML performatives include `ask-if` and `tell`. The *content* of messages was not considered part of the KQML standard, but a separate "Knowledge Interchange Format" (KIF) was also defined, to express such content. KIF is essentially classical first-order predicate logic. Formal definitions of the syntax of KQML and KIF were developed by the KSE, but KQML lacked any formal semantics until [14]. Partly because of the lack of any other obvious standard communication language, and partly because of the availability of free software libraries to implement KQML messaging, the take-up of KQML by the multi-agent systems community was significant. However, Cohen and Levesque (among others) criticized KQML on a number of grounds [7], the most important of which being that, at that time, the language had no formal semantics, and was missing an entire class of performatives — *commissives*, by which one agent makes a commitment to another. As Cohen and Levesque point out, it is difficult to see how many multi-agent scenarios could be implemented without commissives.

In 1995, the Foundation for Intelligent Physical Agents (FIPA) began its work on developing standards for agent systems. The centrepiece of this initiative is the development of an ACL [10]. This ACL is superficially similar to KQML: it defines an "outer" language for messages, it defines 20 performatives (such as `inform`) for defining the intended interpretation of messages, and it does not mandate any specific language for message content. In addition, the concrete syntax for FIPA ACL messages closely resembles that of KQML. The FIPA ACL has been given a formal semantics, in terms of a Semantic Language (SL). The approach adopted for defining these semantics draws heavily on [6], but in particular on Sadek's enhancements to this work [4]. SL is a quantified multi-modal logic, which contains modal operators for referring to the *beliefs*, *desires*, and *uncertain beliefs* of agents, as well as a simple dynamic logic-style apparatus for representing agent actions. The semantics of the FIPA ACL map each ACL message to a formula of SL, which defines a constraint that the sender of the message must satisfy if it is to be considered as conforming to the FIPA ACL standard. FIPA refer to this constraint as the *feasibility* condition. The semantics also map each message to an SL-formula which defines the *rational effect* of the action — the "purpose" of the message: what an agent will be attempting to achieve in sending the message. However, in a society of autonomous agents, the rational effect of a message cannot (and should not) be guaranteed. Hence conformance does not require the recipient of a message to respect the rational effect part of the ACL semantics — only the feasibility condition.

FIPA recognise that "demonstrating in an unambiguous way that a given agent implementation is correct with respect to [the semantics] is not a problem which has been solved" [10, p46], and identify it as an area of future work. (Checking that an implementation respects the *syntax* of the ACL standard is, of course, trivial.) If an agent communication language such as FIPA's ACL is ever to be widely used — particularly for such sensitive applications as electronic commerce — then such conformance testing is obviously crucial. However, the problem of conformance testing (*verification*) is not actually given a concrete definition in [10], and no indication is given of how it might be done. In short, the aim of this paper is to unambiguously define what it means for an agent communication language such as that defined by FIPA to be verifiable, and then to investigate the issues surrounding verification.

2. Agent Communication Frameworks

In this section, we present an abstract framework that allows us to precisely define the verifiable ACL semantics problem. First, we will assume that we have a set $Ag = \{1, \ldots, n\}$ of *agent names* — these are the unique identifiers of agents that will be sending messages to one another in a system.

We shall assume that agents communicate using a communication language $\mathcal{L}_C$. This ACL may be KQML together with KIF [14], it may be the FIPA-97 communication language [10], or some other proprietary language. The exact nature of $\mathcal{L}_C$ is not important for our purposes. The only requirements that we place on $\mathcal{L}_C$ are that it has a well-defined *syntax* and a well-defined *semantics*. The syntax identifies a set $wff(\mathcal{L}_C)$ of *well-formed formulae* of $\mathcal{L}_C$ — syntactically acceptable constructions of $\mathcal{L}_C$. Since we usually think of formulae of $\mathcal{L}_C$ as being *messages*, we use μ (with annotations: $\mu', \mu_1, \ldots$) to stand for members of $wff(\mathcal{L}_C)$.

The semantics of $\mathcal{L}_C$ are assumed to be defined in terms of a second language $\mathcal{L}_S$, which we shall call the *semantic language*. The idea is that if an agent sends a message, then the meaning of sending this message is defined by a formula of $\mathcal{L}_S$. This formula defines what FIPA [10, p48] refer to as the *feasibility pre-condition* — essentially, a constraint that the sender of the message must satisfy in order to be regarded as being "sincere" in sending the message. For example, the feasibility pre-condition for an *inform* act would typically state that the sender of an inform must believe the content of the message, otherwise the sender is not being sincere. Note that in this paper we are not concerned with the *effects* that messages have on recipients. This is because although the "rational effect" of a message on its recipient is the reason that the sender will send a message (e.g., agent i informs agent j of φ because i wants j to believe φ), the sender can have no guarantee that the recipient will even receive the message, still less that it will have the intended effect. The key to our notion of semantics is therefore what properties must hold of the *sender* of a message, in order that it can be considered to be sincere in sending it.

Note that our approach, of defining the semantics of $\mathcal{L}_C$ in terms of another language $\mathcal{L}_S$, is quite normal: the semantics of KQML have been defined in terms of a modal language, containing operators for referring to the beliefs and wants of agents [14], and the semantics of the FIPA-97 ACL are defined in terms of the quantified multi-modal logic SL [10, pp45–55].

Formally, the semantics of the ACL $\mathcal{L}_C$ are given by a function

$$[\![\_]\!]_C : wff(\mathcal{L}_C) \to wff(\mathcal{L}_S)$$

which maps a single message μ of $\mathcal{L}_C$ to a single formula $[\![\mu]\!]_C$ of $\mathcal{L}_S$, which represents the semantics of μ. Note that the "sincerity condition" $[\![\mu]\!]_C$ for message μ acts in effect like a *specification* (in the software engineering sense), which must be satisfied by any agent that claims to conform to the semantics. Verifying that an agent program conforms to the semantics is thus a process of checking that the program satisfies this specification.

In order that the semantics of $\mathcal{L}_C$ be well-defined, we must also have a semantics for our semantic language $\mathcal{L}_S$ itself. While there is no reason in principle why we should not define the semantics of $\mathcal{L}_S$ in terms of a further language $\mathcal{L}_{S'}$, (and so on), we assume without loss of generality that the semantics of $\mathcal{L}_S$ are given with respect to a class $mod(\mathcal{L}_S)$ of *logical models* for $\mathcal{L}_S$. More precisely, the semantics of $\mathcal{L}_S$ will be defined via a *satisfaction relation* "$\models_S$", where

$$\models_S \subseteq wff(\mathcal{L}_S) \times mod(\mathcal{L}_S).$$

By convention, if $M \in mod(\mathcal{L}_S)$ and $\varphi \in wff(\mathcal{L}_S)$ then we write $M \models_S \varphi$ to indicate that $(\varphi, M) \in \models_S$. If $M \models_S \varphi$, then we read this as "φ is *satisfied* (or equivalently, is *true*) in M". The meaning of a formula φ of $\mathcal{L}_S$ is then the set of models in which φ is satisfied. We define a function

$$[\![\_]\!]_S : wff(\mathcal{L}_S) \to \wp(mod(\mathcal{L}_S))$$

such that if $\varphi \in wff(\mathcal{L}_S)$, then $[\![\varphi]\!]_S$ is the set of models in which φ is satisfied:

$$[\![\varphi]\!]_S = \{M \mid M \in mod(\mathcal{L}_S) \text{ and } M \models_S \varphi\}.$$

Agents are assumed to be *implemented* by *programs*, and we let Π stand for the set of all such agent programs. For each agent $i \in Ag$, we assume that $\pi_i \in \Pi$ is the program that implements it. For our purposes, the *contents* of Π are not important — they may be JAVA, C, or C++ programs, for example. At any given moment, we assume that a program π_i may be in any of a set L_i of *local states*. The local state of a program is essentially just a snapshot of the agent's memory at some instant in time. As an agent program π_i executes, it will perform operations (such as assignment statements) that modify its state. Let $L = \bigcup_{i \in Ag} L_i$ be the set of all local states. We use l (with annotations: $l', l_1, \ldots$) to stand for members of L.

One of the key activities of agent programs is *communication*: they send and receive messages, which are formulae of the communication language $\mathcal{L}_C$. We assume that we can identify when an agent emits such a message, and write $send(\pi_i, \mu, l)$ to indicate the fact that agent $i \in Ag$, implemented by program $\pi_i \in \Pi$, sends a message $\mu \in \mathcal{L}_C$ when in state $l \in L_i$.

We now define what we mean by the *semantics* of an agent program. Intuitively, the idea is that when an agent program π_i is in state l, we must be able to characterise the properties of the program as a formula of the semantic language $\mathcal{L}_S$. This formula is the *theory* of the program. In theoretical computer science, the derivation of a program's theory is the first step to reasoning about its behaviour. In particular, a program theory is the basis upon which we can *verify* that the program satisfies its specification. Formally, a program semantics is a function that maps a pair consisting of an agent program and a local state to a formula $\mathcal{L}_S$ of the semantic language. Note that the semantics of Π *must* be defined in terms of the same semantic language that was used to define the semantics of $\mathcal{L}_C$ — otherwise there is no point of reference between the two. Formally then, a semantics for agent program/state pairs is a function

$$[\![\_]\!]_\Pi : \Pi \times L \to wff(\mathcal{L}_S).$$

The relationships between the various formal components introduced above are summarised in Figure 1. We now collect these various components together and define what we mean by an *agent communication framework*.

Definition 1 *An* agent communication framework *is a* $(2n+4)$*-tuple:*

$$\langle Ag, \pi_1, \ldots, \pi_n, L_1, \ldots, L_n, \mathcal{L}_C, \mathcal{L}_S, [\![\_]\!]_\Pi \rangle$$

where $Ag = \{1, \ldots, n\}$ *is a non-empty set of agents,* $\pi_i \in \Pi$ *is an agent program,* L_i *is the set of local states of* π_i*,* $\mathcal{L}_C = \langle wff(\mathcal{L}_C), [\![\_]\!]_C \rangle$ *is a communication language,* $\mathcal{L}_S = \langle wff(\mathcal{L}_S), [\![\_]\!]_S \rangle$ *is a semantic language, and* $[\![\_]\!]_\Pi$ *is a semantics for* Π.

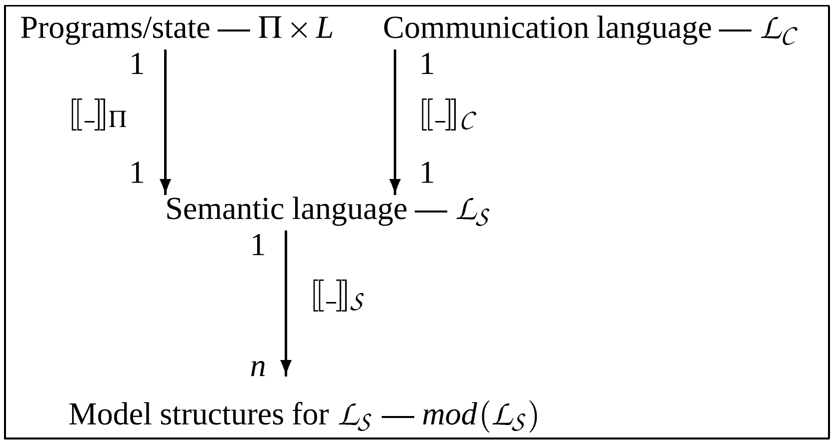
Programs/state — $\Pi \times L$
Communication language — $\mathcal{L}_C$
1
$[\![\_]\!]_\Pi$
1
1
$[\![\_]\!]_C$
1
Semantic language — $\mathcal{L}_S$
1
$[\![\_]\!]_S$
n
Model structures for $\mathcal{L}_S$ — $mod(\mathcal{L}_S)$

Figure 1. The components of an agent communication framework.

We let F be the set of all such agent communication frameworks, and use f (with annotations: $f', f_1, \ldots$) to stand for members of F.

3. Verifiable Communication Frameworks

We are now in a position to define what it means for an agent program, in sending a message while in some particular state, to be respecting the semantics of a communication framework. Recall that a communication language semantics defines, for each message, a *constraint*, or *specification*, which must be satisfied by the sender of the message if it is to be considered as satisfying the semantics of the communication language. The properties of a program when in some particular state are given by the program semantics, $[\![\_]\!]_\Pi$. This naturally leads to the following definition.

Definition 2 *Suppose*

$$f = \langle Ag, \pi_1, \ldots, \pi_n, L_1, \ldots, L_n, \mathcal{L}_C, \mathcal{L}_S, [\![\_]\!]_\Pi \rangle$$

is an agent communication framework, and that $send(\pi_i, \mu, l)$ *for some* $i \in Ag$, $\mu \in wff(\mathcal{L}_C)$, *and* $l \in L_i$. *Then* i *is said to respect the semantics of framework* f *(written* $(\pi_i, l) \models_f \mu$*) iff* $[\![[\![\pi_i, l]\!]_\Pi]\!]_S \subseteq [\![[\![\mu]\!]_C]\!]_S$.

Note that the problem could equivalently have been phrased in terms of logical consequence: $(\pi_i, l) \models_f \mu$ iff $[\![\mu]\!]_C$ is an $\mathcal{L}_S$-logical consequence of $[\![\pi_i, l]\!]_\Pi$. Using this definition, we can define what it means for a communication framework to have a verifiable semantics.

Definition 3 *An agent communication framework*

$$f = \langle Ag, \pi_1, \ldots, \pi_n, L_1, \ldots, L_n, \mathcal{L}_C, \mathcal{L}_S, [\![\_]\!]_\Pi \rangle$$

is said to be verifiable *iff it is a decidable question whether* $(\pi_i, l) \models_f \mu$ *for arbitrary* π_i, l, μ.

The intuition behind verifiability is as follows: if an agent communication framework enjoys this property, then we can determine whether or not an agent is respecting the framework's communication language semantics whenever it sends a message. If a framework is verifiable, then we know that it is possible *in principle* to determine whether or not an agent is respecting the semantics of the framework. But a framework that is verifiable *in principle* is not necessarily verifiable *in practice*. This is the motivation behind the following definition.

Definition 4 *An agent communication framework* $f \in F$ *is said to be* practically verifiable *iff it is decidable whether* $(\pi_i, l) \models_f \mu$ *in time polynomial in the size of* f, π, μ, *and* l.

If we have a practically verifiable framework, then we can do the verification in polynomial time, which implies that we have at least some hope of doing automatic verification using computers that we can envisage today.

3.1. What does it mean to be Verifiable?

If we had a verifiable agent communication framework, what would it look like? Let us take each of the components of such a framework in turn. First, our set *Ag* of agents, implemented by programs π_i, (where these programs are written in an arbitrary programming language). This is straightforward: we obviously have such components today. Next, we need a communication language $\mathcal{L}_C$, with a well-defined syntax and semantics, where the semantics are given in terms of $\mathcal{L}_S$, a semantic language. Again, this is not problematic: we have such a language $\mathcal{L}_C$ in both KQML and the FIPA-97 language. Taking the FIPA case, the semantic language is SL, a quantified multi-modal logic with equality. This language in turn has a well defined syntax and semantics, and so next, we must look for a program semantics $[\![\_]\!]_\Pi$. At this point, we encounter problems.

The semantics of SL are given in the normal modal logic tradition of Kripke (possible worlds) semantics, where each agent's "attitudes" (belief, desire, ...) are characterised as relations holding between different states of affairs. Although Kripke semantics are attractive from a mathematical perspective, it is important to note that they are not connected in any principled way with computational systems. That is, for any given $\pi_i \in \Pi$, (where π_i is, say, a JAVA program), there is no known way of attributing to that program an SL formula (or, equivalently, a set of SL models), which characterises it in terms of beliefs, desires, and so on. Because of this, we say that SL (and most similar logics with Kripke semantics) are *ungrounded* — they have no concrete computational interpretation. In other words, if the semantics of $\mathcal{L}_{\mathcal{S}}$ are ungrounded (as they are in the FIPA-97 SL case), then we have no semantics for programs — and hence an unverifiable communication framework. Although work *is* going on to investigate how arbitrary programs can be ascribed attitudes such as beliefs and desires, the state of the art ([3]) is *considerably* behind what would be required for ACL verification.

Note that it *is* possible to choose a semantic language $\mathcal{L}_{\mathcal{S}}$ such that a principled program semantics $[\![\_]\!]_\Pi$ can be derived. For example, temporal logic has long been used to define the semantics of programming languages [15]. A temporal semantics for a programming language defines for every program a temporal logic formula characterising the meaning of that program. Temporal logic, although ultimately based on Kripke semantics, is firmly *grounded* in the histories traced out by programs as they execute — though of course, standard temporal logic makes no reference to attitudes such as belief and desire. Also note that work in *knowledge theory* has shown how *knowledge* can be attributed to computational processes in a systematic way [9]. However, this work gives no indication of how attitudes such as desiring or intending might be attributed to arbitrary programs. (We use techniques from knowledge theory to show how a grounded semantics can be given to a communication language in Example 2 of section 4.)

Another issue is the computational complexity of the verification process itself. Ultimately, determining whether an agent implementation is respecting the semantics of a communication framework reduces to a logical proof problem, and the complexity of such problems is well-known. If the semantic language $\mathcal{L}_{\mathcal{S}}$ of a framework f is equal in expressive power to first-order logic, then f is of course not verifiable. For quantified multi-modal logics, (such as that used by FIPA to define the semantics of their ACL), the proof problem is often much harder than this — proof methods for quantified multi-modal logics are very much at the frontiers of theorem-proving research (cf. [1]). In the short term, at least, this complexity issue is likely to be another significant obstacle in the way of ACL verification.

4. Example Frameworks

Example 1: Classical Propositional Logic. In this section, we define three communication frameworks, and investigate whether or not they are verifiable. For our first example, we define a simple agent communication framework f_1 in which agents communicate by exchanging formulae of classical propositional logic. The intuitive semantics of sending a message φ is that the sender is *informing* other agents of the truth of φ. An agent sending out a message φ will be respecting the semantics of the language if it "believes" (in a sense that we precisely define below) that φ is true; an agent will not be respecting the semantics if it sends a message that it "believes" to be false. We also assume that agent programs exhibit a simple behaviour of sending out all messages that they believe to be true. We show that framework f_1 is verifiable, and that in fact every agent program in this framework respects the semantics of f_1.

Formally, we must define the components of a framework f_1:

$$f_1 = \langle Ag, \pi_1, \ldots, \pi_n, L_1, \ldots, L_n, \mathcal{L}_{\mathcal{C}}, \mathcal{L}_{\mathcal{S}}, [\![\_]\!]_\Pi \rangle$$

These components are as follows. First, Ag is some arbitrary non-empty set — the contents are not significant. Second, since agents communicate by simply exchanging messages that are simply formulae of classical propositional logic, $\mathcal{L}_0$, we have $\mathcal{L}_{\mathcal{C}} = \mathcal{L}_0$. Thus the set $wff(\mathcal{L}_0)$ contains formulae made up of the proposition symbols $\Phi = \{p, q, r, \ldots\}$ combined into formulae using the classical connectives "$\neg$" (not), "$\wedge$" (and), "$\vee$" (or), and so on.

We let the semantic language $\mathcal{L}_{\mathcal{S}}$ also be classical propositional logic, and define the $\mathcal{L}_{\mathcal{C}}$ semantic function $[\![\_]\!]_{\mathcal{C}}$ simply as the identity function: $[\![\varphi]\!]_{\mathcal{C}} = \varphi$, for all $\varphi \in \mathcal{L}_{\mathcal{C}}$. The semantic function $[\![\_]\!]_{\mathcal{S}}$ for $\mathcal{L}_{\mathcal{S}}$ is then the usual propositional denotation function — the definition is entirely standard, and so we omit it in the interests of brevity.

An agent i's state l_i is defined to be a set of formulae of propositional logic, hence $L_i = \wp(wff(\mathcal{L}_0))$. An agent i's program π_i is assumed to simply implement the following rule:

$$\forall \varphi \in wff(\mathcal{L}_{\mathcal{C}}), \forall l \in L_i, send(\pi_i, \varphi, l) \text{ iff } \varphi \in l \qquad (1)$$

In other words, an agent program π_i sends a message μ when in state l iff μ is present in l. The semantics of agent programs are then defined as follows:

$$[\![\pi_i, \{\varphi_0, \varphi_1, \ldots, \varphi_k\}]\!]_\Pi = \varphi_0 \wedge \varphi_1 \wedge \cdots \wedge \varphi_k.$$

In other words, the meaning of a program in state l is just the conjunction of formulae in l. The following theorem sums up the key properties of this simple agent communication framework.

Theorem 1 *(1) Framework f_1 is verifiable. (2) Every agent in f_1 does indeed respect the semantics of f_1.*

Proof: For (1), suppose that $send(\pi_i, \mu, l)$ for arbitrary π_i, μ, $l = \{\varphi_0, \varphi_1, \ldots, \varphi_k\}$. Then π_i is respecting the semantics for f_1 iff

$$[\![[\![\pi_i, \{\varphi_0, \varphi_1, \ldots, \varphi_k\}]\!]_\Pi]\!]_\mathcal{S} \subseteq [\![[\![\mu]\!]_\mathcal{C}]\!]_\mathcal{S}$$

which by the f_1 definitions of $[\![\text{-}]\!]_\Pi$ and $[\![\text{-}]\!]_\mathcal{C}$ reduces to $[\![\varphi_0 \wedge \varphi_1 \wedge \cdots \wedge \varphi_k]\!]_\mathcal{S} \subseteq [\![\mu]\!]_\mathcal{S}$. But this is equivalent to showing that μ is an $\mathcal{L}_0$-logical consequence of $\varphi_0 \wedge \varphi_1 \wedge \cdots \wedge \varphi_k$. Since $\mathcal{L}_0$ logical consequence is obviously a decidable problem, we are done. For (2), we know from equation (1) that $send(\pi_i, \mu, l)$ iff $\mu \in l$. Since μ is clearly a logical consequence of l if $\mu \in l$, we are done. □

An obvious next question is whether f_1 is *practically* verifiable, i.e., whether verification can be done in polynomial time. Here, observe that verification reduces to a problem of determining logical consequence in $\mathcal{L}_0$, which reduces to a test for $\mathcal{L}_0$-validity, and hence in turn to $\mathcal{L}_0$-unsatisfiability. Since the $\mathcal{L}_0$-satisfiability problem is well-known to be NP-complete, we can immediately conclude the following.

Theorem 2 *The f_1 verification problem is co-NP-complete.*

Note that co-NP-complete problems are ostensibly *harder* than merely NP-complete problems, from which we can conclude that *practical* verification of f_1 is unlikely to be possible.

Example 2: Grounded Semantics for Propositional Logic. One could argue that Example 1 worked because we made the assumption that agents explicitly maintain databases of $\mathcal{L}_0$ formulae: checking whether an agent was respecting the semantics in sending a message φ amounted to determining whether φ was a logical consequence of this database. This was a convenient, but, as the following example illustrates, unnecessary assumption. For this example, we will again assume that agents communicate by exchanging formulae of classical propositional logic $\mathcal{L}_0$, but we make no assumptions about their programs or internal state. We show that despite this, we can still obtain a verifiable semantics, because we can *ground* the semantics of the communication language in the states of the program.

As in Example 1, we set both the communication language $\mathcal{L}_\mathcal{C}$ and the semantic language $\mathcal{L}_\mathcal{S}$ to be classical propositional logic $\mathcal{L}_0$. We require some additional definitions. Let the set G of *global states* of a system be defined by $G = L_1 \times \cdots \times L_n$. We use g (with annotations: $g_1, g', \ldots$) to stand for members of G. We assume that we have a vocabulary $\Phi = \{p, q, \ldots\}$ of primitive propositions to express the properties of a system. In addition, we assume it is possible to determine whether or not any primitive proposition $p \in \Phi$ is true of a particular global state or not. We write $g \models p$ to indicate that p is true in state g. Next, we define a relation $\sim_i \subseteq G \times L_i$ for each agent $i \in Ag$ to capture the idea of *indistinguishability*. The idea is that if an agent i is in state $l \in L_i$, then a global state $g = \langle l'_1, \ldots, l'_n \rangle$ is indistinguishable from the state l that i is currently in (written $g \sim_i l$) iff $l = l'_i$. Now, for any given agent program π_i in local state l, we define the *positive knowledge set* of π_i in l, (written $ks^+(\pi_i, l)$) to be the set of propositions that are true in all global states that are indistinguishable from l, and the *negative knowledge set* of π_i in l, (written $ks^-(\pi_i, l)$) to be the set of propositions that are false in all global states that are indistinguishable from l. Formally,

$$ks^+(\pi_i, l) = \{p \mid p \in \Phi \text{ and } \forall g \in G, g \sim_i l \text{ implies } g \models p\}$$
$$ks^-(\pi_i, l) = \{p \mid p \in \Phi \text{ and } \forall g \in G, g \sim_i l \text{ implies } g \not\models p\}$$

Readers familiar with epistemic logic [9] will immediately recognise that this construction is based on the definition of knowledge in distributed systems. The idea is that if $p \in ks^+(\pi_i, l)$, (respectively, $p \in ks^-(\pi_i, l)$), then given the information that i has available in state l, p must necessarily be true (respectively, false).

The $\mathcal{L}_\mathcal{C}$ semantic function $[\![\text{-}]\!]_\mathcal{C}$ is defined to be the identity function again, so $[\![\varphi]\!]_\mathcal{C} = \varphi$. For the program semantics, we define

$$[\![\pi_i, l]\!]_\Pi = \bigwedge_{p_j \in ks^+(\pi_i, l)} p_j \quad \wedge \bigwedge_{p_k \in ks^-(\pi_i, l)} \neg p_k.$$

The formula $[\![\pi_i, l]\!]_\Pi$ thus encodes the *knowledge* that the program π_i has about the truth or falsity of propositions Φ when in state l. The $\mathcal{L}_\mathcal{S}$ semantic function $[\![\text{-}]\!]_\mathcal{S}$ is assumed to be the standard $\mathcal{L}_0$ semantic function, as in Example 1. An agent will thus be respecting the semantics of the communication framework if it sends a message such that this message is guaranteed to be true in all states indistinguishable from the one the agent is currently in. This framework has the following property.

Theorem 3 *If f_2 is finite, then f_2 is verifiable.*

Proof: Suppose that $send(\pi_i, \mu, l)$ for arbitrary π_i, μ, l. Then π_i is respecting the semantics for f_2 iff $[\![[\![\pi_i, l]\!]_\Pi]\!]_\mathcal{S} \subseteq [\![[\![\mu]\!]_\mathcal{C}]\!]_\mathcal{S}$ which by the f_2 definitions of $[\![\text{-}]\!]_\Pi$ and $[\![\text{-}]\!]_\mathcal{C}$ reduces to $[\![\bigwedge_{p_j \in ks^+(\pi_i, l)} p_j \wedge \bigwedge_{p_k \in ks^-(\pi_i, l)} \neg p_k]\!]_\mathcal{S} \subseteq [\![\mu]\!]_\mathcal{S}$. Computing G will take time $O(|L_1 \times \cdots \times L_n|)$; computing $\sim_i$ can be done in time $O(|L_i| \times |G|)$; and given G and $\sim_i$, computing $ks^+(\pi_i, l)$ and $ks^-(\pi_i, l)$ can be done in time $O(|\Phi| \times |G|)$. Once given $ks^+(\pi, l)$ and $ks^-(\pi, l)$, determining whether $[\![\bigwedge_{p_j \in ks^+(\pi_i, l)} p_j \wedge \bigwedge_{p_k \in ks^-(\pi_i, l)} \neg p_k]\!]_\mathcal{S} \subseteq [\![\mu]\!]_\mathcal{S}$ reduces to a problem of determining $\mathcal{L}_0$ logical consequence, which is obviously decidable. □

Since f_2 verification reduces to $\mathcal{L}_0$ logical consequence checking, we can use a similar argument to that used for

Theorem 2 to show the problem is in general no more complex than f_1 verification:

Theorem 4 *The f_2 verification problem is co-NP-complete.*

Note that the main point about this example is the way that the semantics for programs were *grounded* in the states of programs. In this example, the communication language was simple enough to make the grounding easy. More complex communication languages with a similarly grounded semantics are possible. Framework f_2 can be extended to allow agents to communicate in propositional epistemic logic, for example [9].

Example 3: The FIPA-97 ACL. For the final example, consider a framework f_3 in which we use the FIPA-97 ACL, and the semantics for this language defined in [10]. Following the discussion in section 3.1, it should come as no surprise that such a framework is not verifiable. It is worth spelling out the reasons for this. First, since the semantic language SL is a quantified multi-modal logic, with greater expressive power than classical first order logic, it is clearly undecidable. (As we noted above, the complexity of the decision problem for quantified modal logics is often much harder than for classical predicate logic [1].) So the formal problem of verification is for f_3 is obviously undecidable. But of course the problem is worse than this, since as the discussion in section 3.1 showed, we do not have any idea of how to assign a program semantics for semantic languages like SL, because these languages have an ungrounded semantics.

5. Discussion

If agents are to be as widely deployed as some observers predict, then the issue of inter-operation — in the form of standards for communication languages — must be addressed. Moreover, the problem of determining conformance to these standards must also be seriously considered, for if there is no way of determining whether or not a system that claims to conform to a standard does indeed conform to it, then the value of the standard itself must be questioned. This paper has given the first precise definition of what it means for an agent communication framework to be verifiable, and has identified some problematic issues for verifiable communication language semantics, the most important of which being that:

- We must be able to characterise the properties of an agent program as a formula of the language $\mathcal{L}_S$ used to give a semantics to the communication language. $\mathcal{L}_S$ if often a multi-modal logic, referring to (in the FIPA-97 case, for example) the beliefs, desires, and uncertainties of agents. We currently have very little idea about systematic ways of attributing such descriptions to programs — the state of the art is *considerably* behind what would be needed for anything like practical verification, and this situation is not likely to change in the near future.
- The computational complexity of logical verification, (particularly using quantified multi-modal languages), is likely to prove a major obstacle in the path of practical agent communication language verification.

In addition, the paper has given examples of agent communication frameworks, some of which are verifiable by this definition, others of which, (including the FIPA-97 ACL [10]), are not.

An obvious response to this paper is to ask is whether its negative results are an artifact of the way the verification problem has been defined. In particular, readers familiar with verification techniques from mainstream computer science might wonder whether a *model checking* approach to verification is more appropriate [12, 5]. Using such an approach, we would define the program semantics as a function

$$[\![\_]\!]_\Pi : \Pi \times L \to mod(\mathcal{L}_S)$$

which assigns to every program/state pair an $\mathcal{L}_S$-model, which encodes the properties of that program/state pair. Verifying that $(\pi_i, l) \models_f \mu$ would involve checking whether $[\![\pi_i, l]\!]_\Pi \models_S [\![\mu]\!]_C$, i.e., whether the sincerity condition $[\![\mu]\!]_C$ was satisfied in model $[\![\pi_i, l]\!]_\Pi$. The main perceived advantage of model checking approaches are with respect to efficiency: model checking is often much simpler than theorem proving, and as a consequence model-checking approaches have been used to verify significant finite-state systems (with up to 10^{120} states [5]). Moreover, model checking algorithms have been developed for (propositional) belief-desire-intention logics [17], which are somewhat similar to those used to give a semantics to the FIPA ACL. However, there are two major problems with such an approach. The first is that of developing the program semantics $[\![\_]\!]_\Pi$: as described above, we do not yet have any techniques for systematically assigning models representing beliefs, desires, and uncertainties (as in the FIPA-97 SL case [10]) to arbitrary programs. The second problem is that model checking approaches have been shown to be useful for systems that can be represented as *finite state* models using *propositional* temporal logics. If the verification logic allows arbitrary quantification, (or the system to be verified is not finite state), then a model checking approach is unlikely to be practicable.

A possible alternative approach to verification would be a form of *operational semantics*. The idea is best explained by analogy with testing compilers to determine compliance with programming language standards. Although programming languages often have a formal (denotational, algebraic, ...) semantics, compilers are invariably tested by subjecting them to a barrage of test cases and checking the results against what would be expected of a conformant compiler. Could some similar approach be adopted to test ACL standard compliance? It is difficult to see how. For what realistic tests could we construct to determine whether some agent intended to x, when we do not ourselves agree on what intending x is?

Perhaps a more realistic alternative would be not to try to actually *test* agent programs to determine whether they comply with the semantics, but to elevate messages to the status of *legally binding contracts*. Thus if some agent i promised to do x by a certain date, and failed to do x by this date, then i would be liable to penalties, in exactly the same way that contractors who fail to meet their obligations are liable in human societies. There are several arguments against this approach. The legal issues, for one thing, are an as-yet untested area (but see [13]). In addition, the approach seems to imply a *centralised* third party responsible for checking and if necessary enforcing the legal constraints. Centralisation is clearly at odds with the goal of building loosely coupled distributed systems. Finally, the *practicality* of "policing" Internet-wide multi-agent systems in this way must be questionable. However, the scheme is almost certainly practicable for smaller-scale systems. In the medium term, it may be the only option available.

Finally, note that the results of this paper could be interpreted as negative, in that they imply that verification of conformance to ACLs using current techniques is not likely to be possible. However, the paper should emphatically *not* be interpreted as suggesting that standards — particularly, standardised ACLs — are unnecessary or a waste of time. If agent technology is to achieve its much vaunted potential as a new paradigm for software construction, then such standards are *essential*. However, it may well be that we need new ways of thinking about the semantics and verification of such standards.

References

[1] M. Abadi. *Temporal Logic Theorem Proving*. PhD thesis, Computer Science Department, Stanford University, Stanford, CA 94305, 1987.

[2] J. L. Austin. *How to Do Things With Words*. Oxford University Press: Oxford, England, 1962.

[3] R. I. Brafman and M. Tennenholtz. Modeling agents as qualitative decision makers. *Artificial Intelligence*, 94(1-2):217–268, July 1997.

[4] P. Bretier and D. Sadek. A rational agent as the kernel of a cooperative spoken dialogue system: Implementing a logical theory of interaction. In J. P. Müller, M. Wooldridge, and N. R. Jennings, editors, *Intelligent Agents III (LNAI Volume 1193)*, pages 189–204. Springer-Verlag: Berlin, Germany, 1997.

[5] E. Clarke, O. Grumberg, and D. Long. Verification tools for finite-state concurrent systems. In J. W. de Bakker, W. P. de Roever, and G. Rozenberg, editors, *A Decade of Concurrency — Reflections and Perspectives (LNCS Volume 803)*, pages 124–175. Springer-Verlag: Berlin, Germany, 1994.

[6] P. R. Cohen and H. J. Levesque. Rational interaction as the basis for communication. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication*, pages 221–256. The MIT Press: Cambridge, MA, 1990.

[7] P. R. Cohen and H. J. Levesque. Communicative actions for artificial agents. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 65–72, San Francisco, CA, June 1995.

[8] P. R. Cohen and C. R. Perrault. Elements of a plan based theory of speech acts. *Cognitive Science*, 3:177–212, 1979.

[9] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge*. The MIT Press: Cambridge, MA, 1995.

[10] The Foundation for Intelligent Physical Agents. FIPA '97 Specification part 2 — Agent communication language. The text refers to the specification dated 10th October 1997.

[11] C. Guilfoyle, J. Jeffcoate, and H. Stark. *Agents on the Web: Catalyst for E-Commerce*. Ovum Ltd, London, Apr. 1997.

[12] J. Y. Halpern and M. Y. Vardi. Model checking versus theorem proving: A manifesto. In V. Lifschitz, editor, *AI and Mathematical Theory of Computation — Papers in Honor of John McCarthy*, pages 151–176. Academic Press, 1991.

[13] C. E. A. Karnow. Liability for distributed artificial intelligences. *Berkeley Technology Law Journal*, 11(1):147–204, 1996.

[14] Y. Labrou and T. Finin. Semantics and conversations for an agent communication language. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*, pages 584–591, Nagoya, Japan, 1997.

[15] Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems — Safety*. Springer-Verlag: Berlin, Germany, 1995.

[16] J. Mayfield, Y. Labrou, and T. Finin. Evaluating KQML as an agent communication language. In M. Wooldridge, J. P. Müller, and M. Tambe, editors, *Intelligent Agents II (LNAI Volume 1037)*, pages 347–360. Springer-Verlag: Berlin, Germany, 1996.

[17] A. S. Rao and M. P. Georgeff. A model-theoretic approach to the verification of situated reasoning systems. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 318–324, Chambéry, France, 1993.

[18] J. R. Searle. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press: Cambridge, England, 1969.