Semantic Issues in the Verification of Agent Communication Languages

Michael Wooldridge (m.j.wooldridge@qmw.ac.uk)
Department of Electronic Engineering, Queen Mary and Westfield College, London E1 4NS, United Kingdom

Abstract. This article 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 defined. 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

One of the main reasons why multi-agent systems are currently a major area of research and development activity is that they are seen as a key enabling technology for the Internet-wide electronic commerce systems that are widely predicted to emerge in the near future [20]. 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 safely communicate with one-another 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) [30, 19]. 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 [19, p46]). In this article, 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 3, 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 4. The basic idea is to show how demonstrating conformance to an ACL semantics can be seen as a verification problem in the standard software engineering sense [7]. Demonstrating

that a program semantically complies to a standard involves showing that the program satisfies the specification given by the semantics. If the semantics are logical, then demonstrating compliance thus reduces to a proof problem. We discuss the practical implications of these definitions in section 4.1. In section 5, we give examples of some ACLs, and show that some of these are verifiable, while others are not. In section 6, we discuss an alternative approach to verification, in which verification is done via model checking rather than proof. Finally, in section 7, we discuss the implications of our results, with emphasis on future directions for work on verifiable ACLs.

2. Background

Current techniques for developing the semantics of ACLs trace their origins to speech act theory. In this section, we give a brief overview of this work.

2.1. Speech Acts

The theory of speech acts is generally recognised as having begun in the work of the philosopher John Austin [4]. Austin 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. It may seem strange to think of utterances changing the world in the way that physical actions do. If we pick up a block from a table (to use an overworked but traditional example), then the world has changed in an obvious way. But how does speech change the world? Austin gave as paradigm examples declaring war and saying "I now pronounce you man and wife". Stated in the appropriate circumstances, these utterances clearly change the state of the world in a very tangible way[1].

Austin identified a number of *performative verbs*, which correspond to various different types of speech acts. Examples of such performative verbs are *request*, *inform*, and *promise*. In addition, Austin distinguished three different aspects of speech acts: the *locutionary act*, or act of making an utterance (e.g., saying "Please make some tea"), the *illocutionary act*, or action performed in saying something (e.g., "He requested me to make some tea"), and *perlocution*, or effect of the act (e.g., "He got me to make tea").

[1] Notice that when referring to the effects of communication, we are ignoring "pathological" cases, such as shouting while on a ski run and causing an avalanche. Similarly, we will ignore "microscopic" effects (such as the minute changes in pressure or temperature in a room caused by speaking).

Austin referred to the conditions required for the successful completion of performatives as *felicity conditions*. He recognized three important felicity conditions:

1. a) There must be an accepted conventional procedure for the performative.
   b) The circumstances and persons must be as specified in the procedure.
2. The procedure must be executed correctly and completely.
3. The act must be sincere, and any *uptake* required must be completed, insofar as is possible.

Austin's work was refined and considerably extended by Searle, in his 1969 book *Speech Acts* [38]. Searle identified several properties that must hold for a speech act performed between a hearer and a speaker to succeed, including normal I/O conditions, preparatory conditions, and sincerity conditions. For example, consider a *request* by SPEAKER to HEARER to perform ACTION:

1. *Normal I/O conditions.* Normal I/O conditions state that HEARER is able to hear the request, (thus must not be deaf, ...), the act was performed in normal circumstances (not in a film or play, ...), etc.
2. *Preparatory conditions.* The preparatory conditions state what must be true of the world in order that SPEAKER correctly choose the speech act. In this case, HEARER must be able to perform ACTION, and SPEAKER must believe that HEARER is able to perform ACTION. Also, it must not be obvious that HEARER will do ACTION anyway.
3. *Sincerity conditions.* These conditions distinguish sincere performances of the request; an insincere performance of the act might occur if SPEAKER did not really want ACTION to be performed.

Searle also gave a five-point typology of speech acts:

1. *Representatives.* A representative act commits the speaker to the truth of an expressed proposition. The paradigm case is *informing*.
2. *Directives.* A directive is an attempt on the part of the speaker to get the hearer to do something. Paradigm case: *requesting*.
3. *Commissives.* Commit the speaker to a course of action. Paradigm case: *promising*.

4. *Expressives.* Express some psychological state (e.g., gratitude). Paradigm case: *thanking.*

5. *Declarations.* Effect some changes in an institutional state of affairs. Paradigm case: *declaring war.*

2.2. Speech Acts in Artificial Intelligence

In the late 1960s and early 1970s, a number of researchers in artificial intelligence (AI) began to build systems that could plan how to autonomously achieve goals [2]. Clearly, if such a system is required to interact with humans or other autonomous agents, then such plans must include *speech* actions. This introduced the question of how the properties of speech acts could be represented such that planning systems could reason about them. Cohen and Perrault [15] gave an account of the semantics of speech acts by using techniques developed in AI planning research [18]. The aim of their work was to develop a theory of speech acts:

> "[B]y modelling them in a planning system as operators defined ... in terms of speakers and hearers beliefs and goals. Thus speech acts are treated in the same way as physical actions". [15]

The formalism chosen by Cohen and Perrault was the STRIPS notation, in which the properties of an action are characterised via pre- and post-conditions [18]. The idea is very similar to Hoare logic [24]. Cohen and Perrault demonstrated 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.

Consider the *Request* act. The aim of the *Request* act will be for a speaker to get a hearer to perform some action. Figure 1 defines the *Request* act. Two preconditions are stated: the "cando.pr" (can-do pre-conditions), and "want.pr" (want pre-conditions). The cando.pr states that for the successful completion of the *Request*, two conditions must hold. First, the speaker must believe that the hearer of the *Request* is able to perform the action. Second, the speaker must believe that the hearer also believes it has the ability to perform the action. The want.pr states that in order for the *Request* to be successful, the speaker must also believe it actually wants the *Request* to be performed. If the pre-conditions of the *Request* are fulfilled, then the *Request* will be successful: the result (defined by the "effect" part of the definition) will be that the hearer believes the speaker believes it wants some action to be performed.

$\underline{Request(S, H, \alpha)}$		
PRECONDITIONS	CANDO.PR	$(S\ BELIEVE\ (H\ CANDO\ \alpha)) \wedge$ $(S\ BELIEVE\ (H\ BELIEVE\ (H\ CANDO\ \alpha)))$
	WANT.PR	$(S\ BELIEVE\ (S\ WANT\ requestInstance))$
EFFECT		$(H\ BELIEVE\ (S\ BELIEVE\ (S\ WANT\ \alpha)))$
$\underline{CauseToWant(A_1, A_2, \alpha)}$		
PRECONDITIONS	CANDO.PR	$(A_1\ BELIEVE\ (A_2\ BELIEVE\ (A_2\ WANT\ \alpha)))$
	WANT.PR	$\times$
EFFECT		$(A_1\ BELIEVE\ (A_1\ WANT\ \alpha))$

Figure 1. Definitions from the Plan-Based Theory of Speech Acts

While the successful completion of the *Request* ensures that the hearer is aware of the speaker's desires, it is not enough in itself to guarantee that the desired action is actually performed. This is because the definition of *Request* only models the illocutionary force of the act. It says nothing of the perlocutionary force. What is required is a *mediating act*. Table 1 gives a definition of *CauseToWant*, which is an example of such an act. By this definition, an agent will come to believe it wants to do something if it believes that another agent believes it wants to do it. This definition could clearly be extended by adding more pre-conditions, perhaps to do with beliefs about social relationships or power structures.

Using these ideas, and borrowing a formalism for representing the mental state of agents that was developed by Robert Moore [31], Douglas Appelt was able to implement a system that was capable of planning to perform speech acts [3].

2.3. Speech Acts as Rational Action

While the 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 [13]. The foundation upon which they built this model of rational action was their theory of intention, described in [12]. The for-

mal theory is too complex to describe here, but as a flavour, here is the Cohen-Levesque definition of *requesting*, paraphrased in English [13, p241]:

> A request is an attempt on the part of *spkr*, by doing e, to bring about a state where, ideally, (i) *addr* intends α, (relative to the *spkr* still having that goal, and *addr* still being helpfully inclined to *spkr*), and (ii) *addr* actually eventually does α, or at least brings about a state where *addr* believes it is mutually believed that it wants the ideal situation.

Actions in the Cohen-Levesque framework were modelled using techniques adapted from dynamic logic [23].

2.4. Agent Communication Languages: KQML and FIPA

Throughout the 1980s and 1990s, interest in multi-agent systems developed rapidly [6, 41]. An obvious problem in multi-agent systems is how to get agents to communicate with one-another — the interoperability issue referred to in the introduction. To this end, in the early 1990s, the DARPA Knowledge Sharing Effort (KSE) began to develop the Knowledge Query and Manipulation Language (KQML) and the associated Knowledge Interchange Format (KIF) as a common framework via which multiple expert systems (cf. agents) could exchange knowledge [33, 30].

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 KIF was also defined, to express such content. KIF is essentially classical first-order predicate logic, recast in a LISP-like syntax.

To better understand the KQML language, consider the following example [30, p354]:

```
(ask-one
   :content  (PRICE IBM ?price)
   :receiver stock-server
   :language LPROLOG
   :ontology NYSE-TICKS
)
```

The intuitive interpretation of this message is that the sender is asking about the price of IBM stock. The performative is `ask-one`, which an agent will use to ask a question of another agent where exactly one reply

is needed. The various other components of this message represent its attributes. The most important of these is the `:content` field, which specifies the message content. In this case, the content simply asks for the price of IBM shares. The `:receiver` attribute specifies the intended recipient of the message, the `:language` attribute specifies that the language in which the content is expressed is called `LPROLOG` (the recipient is assumed to "understand" `LPROLOG`), and the final `:ontology` attribute defines the terminology used in the message.

Formal definitions of the syntax of KQML and KIF were developed by the KSE, but KQML lacked any formal semantics until Labrou and Finin's [26]. These semantics were presented using a pre- and post-condition notation, closely related to Cohen and Perrault's plan-based theory of speech acts [15]. These pre- and post-conditions were specified by Labrou and Finin using a logical language containing modalities for belief, knowledge, wanting, and intending. However, Labrou and Finin recognised that any commitment to a particular semantics for this logic itself would be contentious, and so they refrained from giving it a semantics. However, this rather begs the question of whether their semantics are actually well-founded. We return to this issue later.

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 [14], the most important of which being that, the language 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, which appear to be important if agents are to *coordinate* their actions with one-another [25].

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 [19][2]. 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. Here is an example of a FIPA ACL message (from [19, p10]):

```
(inform
   :sender   agent1
```

[2] FIPA simply refer to their ACL as "ACL", which can result in confusion when discussing ACLs in general. To avoid ambiguity, we will always refer to "the FIPA ACL".

```
   :receiver agent2
   :content  (price good2 150)
   :language sl
   :ontology hpl-auction
)
```

Even a superficial glance confirms that the FIPA ACL is similar to KQML; the relationship is discussed in [19, pp68–69].

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 [13], but in particular on Sadek's enhancements to this work [9]. 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's 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 rational effect of a messages is its purpose: what an agent will be attempting to achieve in sending the message (cf. perlocutionary act). 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.

To illustrate the FIPA approach, we give an example of the semantics of the FIPA `inform` performative [19, p25]:

$$\begin{array}{l} \langle i, \mathit{inform}(j,\varphi)\rangle \\ \quad \text{FP:} \quad B_i\varphi \wedge \neg B_i(\mathit{Bif}_j\varphi \vee U_j\varphi) \\ \quad \text{RE:} \quad B_j\varphi \end{array} \tag{1}$$

The B_i is a modal connective for referring to the beliefs of agents (see e.g., [21]); Bif is a modal connective that allows us to express whether an agent has a definite opinion one way or the other about the truth or falsity of its parameter; and U is a modal connective that allows us to represent the fact that an agent is "uncertain" about its parameter. Thus an agent i sending an *inform* message with content φ to agent j will be respecting the semantics of the FIPA ACL if it believes φ, and it it not the case that it believes of j either that j believes whether φ is true or false, or that j is uncertain of the truth or falsity of φ.

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" [19, p46], and identify it as an area of future work. (Checking that an implementation respects the *syntax* of an ACL like KQML or FIPA 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 [19], and no indication is given of how it might be done. In short, the aim of the remainder of this article 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 such verification.

3. 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 $\mathcal{L}_{\mathcal{C}}$. This ACL may be KQML together with KIF [26], it may be the FIPA-97 communication language [19], or some other proprietary language. The exact nature of $\mathcal{L}_{\mathcal{C}}$ is not important for our purposes. The only requirements that we place on $\mathcal{L}_{\mathcal{C}}$ are that it has a well-defined *syntax* and a well-defined *semantics*. The syntax identifies a set $wff(\mathcal{L}_{\mathcal{C}})$ of *well-formed formulae* of $\mathcal{L}_{\mathcal{C}}$ — syntactically acceptable constructions of $\mathcal{L}_{\mathcal{C}}$. Since we usually think of formulae of $\mathcal{L}_{\mathcal{C}}$ as being *messages*, we use μ (with annotations: $\mu', \mu_1, \ldots$) to stand for members of $wff(\mathcal{L}_{\mathcal{C}})$.

The semantics of $\mathcal{L}_{\mathcal{C}}$ are assumed to be defined in terms of a second language $\mathcal{L}_{\mathcal{S}}$, 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 $\mathcal{L}_{\mathcal{S}}$. This formula defines what FIPA [19, 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.

The idea of defining the semantics of one language in terms of another might seem strange, but the technique is common in computer science:

- when Hoare-logic style semantics are given for programming languages, the semantics of a program written in, for example, PASCAL or C are defined in terms of a second language — that of classical first-order logic [24];
- an increasingly common approach to defining the semantics of many programming languages is to give them a *temporal semantics*, whereby the semantics of a program in a language such as C or PASCAL are defined as a formula of temporal logic [28].

Note that in this article 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.

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

$$[\![\_]\!]_{\mathcal{C}} : \mathit{wff}(\mathcal{L}_{\mathcal{C}}) \rightarrow \mathit{wff}(\mathcal{L}_{\mathcal{S}})$$

which maps a single message μ of $\mathcal{L}_{\mathcal{C}}$ to a single formula $[\![\mu]\!]_{\mathcal{C}}$ of $\mathcal{L}_{\mathcal{S}}$, which represents the semantics of μ. Note that the "sincerity condition" $[\![\mu]\!]_{\mathcal{C}}$ 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.

To make the idea concrete, recall the FIPA semantics of *inform* messages, given in (1), above. In our framework, we can express the FIPA semantics as

$$[\![\langle i, \mathit{inform}(j, \varphi)\rangle]\!]_{\mathcal{C}} = B_i\varphi \wedge \neg B_i(\mathit{Bif}_j\varphi \vee U_j\varphi)$$

It should be obvious how this corresponds to the FIPA definition.

In order that the semantics of $\mathcal{L}_{\mathcal{C}}$ be well-defined, we must also have a semantics for our semantic language $\mathcal{L}_{\mathcal{S}}$ itself. While there is no reason in principle why we should not define the semantics of $\mathcal{L}_{\mathcal{S}}$ in terms of a further language $\mathcal{L}_{\mathcal{S}'}$, (and so on), we assume without loss

of generality that the semantics of $\mathcal{L}_{\mathcal{S}}$ are given with respect to a class $mod(\mathcal{L}_{\mathcal{S}})$ of *logical models* for $\mathcal{L}_{\mathcal{S}}$. More precisely, the semantics of $\mathcal{L}_{\mathcal{S}}$ will be defined via a *satisfaction relation* "$\models_{\mathcal{S}}$", where

$$\models_{\mathcal{S}} \subseteq wff(\mathcal{L}_{\mathcal{S}}) \times mod(\mathcal{L}_{\mathcal{S}}).$$

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

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

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

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

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 $\mathcal{L}_{\mathcal{C}}$. 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 $\mu \in \mathcal{L}_{\mathcal{C}}$ 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 $\mathcal{L}_{\mathcal{S}}$. 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

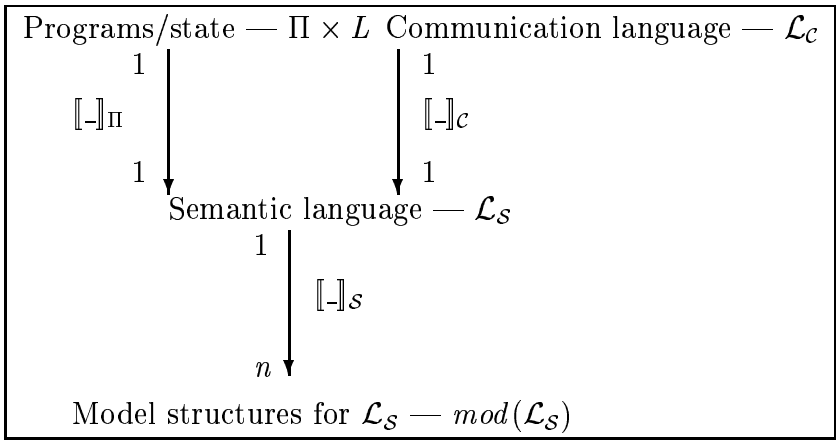

Figure 2. The components of an agent communication framework.

$\mathcal{L}_\mathcal{S}$ 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 $\mathcal{L}_\mathcal{C}$ — otherwise there is no point of reference between the two. Formally then, a semantics for agent program/state pairs is a function

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

The relationships between the various formal components introduced above are summarised in Figure 2. 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:*

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

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*,* $\mathcal{L}_\mathcal{C} = \langle wff(\mathcal{L}_\mathcal{C}), [\![\_]\!]_\mathcal{C} \rangle$ *is a communication language,* $\mathcal{L}_\mathcal{S} = \langle wff(\mathcal{L}_\mathcal{S}), [\![\_]\!]_\mathcal{S} \rangle$ *is a semantic language, and* $[\![\_]\!]_\Pi$ *is a semantics for* Π*.*

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.

4. Verifiability Defined

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 leads to the following definition.

DEFINITION 2. *Suppose*

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

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

$$[\![[\![\pi_i, l]\!]_\Pi]\!]_\mathcal{S} \subseteq [\![[\![\mu]\!]_\mathcal{C}]\!]_\mathcal{S}.$$

Note that the problem could equivalently have been phrased in terms of logical consequence: $(\pi_i, l) \models_f \mu$ iff $[\![\mu]\!]_\mathcal{C}$ is an $\mathcal{L}_\mathcal{S}$-logical consequence of $[\![\pi_i, l]\!]_\Pi$. If we had a sound and complete proof system $\vdash_\mathcal{S}$ for $\mathcal{L}_\mathcal{S}$, then we could similarly have phrased it as a proof problem: $(\pi_i, l) \models_f \mu$ iff $[\![\pi_i, l]\!] \vdash_\mathcal{S} [\![\mu]\!]_\mathcal{C}$. The first approach, however, is probably the most general.

Using this definition, we can define what it means for a communication framework to have a verifiable semantics.

DEFINITION 3. *An agent communication framework*

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

is 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 $|f| \times |\pi| \times |\mu| \times |l|$.*

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. Our ideal, when setting out an agent communication framework f, should clearly be to construct f such that it is practically verifiable. However, practical verifiability is quite a demanding property, as we shall see in section 5. In the following subsection, we examine the implications of these definitions.

4.1. 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 $\mathcal{L}_\mathcal{C}$, with a well-defined syntax and semantics, where the semantics are given in terms of $\mathcal{L}_\mathcal{S}$, a semantic language. Again, this is not problematic: we have such a language $\mathcal{L}_\mathcal{C}$ 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.

Put simply, the FIPA semantics are given in terms of mental states, and since we do not understand how such states can be systematically attributed to programs, we cannot verify that such programs respect the semantics. More precisely, 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 ([8]) is *considerably* behind what would be required for ACL verification. Other researchers have also recognised this difficulty [39, 34].

Note that it *is* possible to choose a semantic language $\mathcal{L}_S$ 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 [29]. 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 [17]. 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 5.)

Another issue is the high computational complexity of the verification process itself [32]. 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 $\mathcal{L}_S$ 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.

To sum up, it is entirely possible to define a communication language $\mathcal{L}_C$ with semantics in terms of a language $\mathcal{L}_S$. However, giving a *program semantics* for a semantic language (such as that of FIPA-97) with *ungrounded* semantics is a serious unsolved problem.

5. Example Frameworks

To illustrate the idea of verification, as introduced above, in this section we will consider a number of progressively richer agent communica-

tion frameworks. For each of these frameworks, we discuss the issue of verifiability, and where possible, characterise the complexity of the verification problem.

5.1. Example 1: Classical Propositional Logic.

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:

$$\forall \varphi \in wff(\mathcal{L}_{\mathcal{C}}), \forall l \in L_i, send(\pi_i, \varphi, l) \text{ iff } \varphi \in l \tag{2}$$

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 $[\![\_]\!]_\Pi$ and $[\![\_]\!]_\mathcal{C}$ 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 (2) 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 highly unlikely to be possible[3].

5.2. 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

[3] In fact, f_2 will be practically verifiable if and only if P = NP, which is regarded as extremely unlikely [32].

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. There is an impartial, objective procedure we can apply to obtain a declarative representation of the "knowledge" implicit within an arbitrary program, in the form of Fagin-Halpern-Moses-Vardi knowledge theory [17]. To check whether an agent is respecting the semantics of the communication language, we simply check whether the information in the message sent by the agent is a logical consequence of the knowledge implicit within the agent's state, which we obtain using the tools of knowledge theory.

In what follows, we assume all sets are finite. 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 (see [17, pp103–114] for more details). 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 [17] 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). Thus $ks^+(\pi_i, l)$

represents the set of propositions that the agent i knows are true when it is in state l; and $ks^-(\pi_i, l)$ represents the set of propositions that i knows are false when it is in state l.

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

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

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 $[\![\_]\!]_\mathcal{S}$ 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. *Framework 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 $[\![\_]\!]_\Pi$ and $[\![\_]\!]_\mathcal{C}$ 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 can be done in 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 the $\mathcal{L}_0$ logical consequence problem

$$\bigwedge_{p_j \in ks^+(\pi_i, l)} p_j \wedge \bigwedge_{p_k \in ks^-(\pi_i, l)} \neg p_k \models [\![\mu]\!]_\mathcal{S}$$

This problem 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. We note in closing that it is straightforward to extend framework f_2 to allow a much richer agent communication language (including requesting, informing, and commissives) [40].

5.3. 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 [19]. Following the discussion in section 4.1, 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 f_3 verification problem is obviously undecidable. But of course the problem is worse than this, since as the discussion in section 4.1 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, mentalistic semantics.

6. Verification via Model Checking

The problem of verifying whether an agent implements the semantics of a communication language has thus far been presented as one of determining logical consequence, or, equivalently, as a proof problem. Readers familiar with verification from theoretical computer science will recognise that this corresponds to the "traditional" approach to verifying that a program satisfies a specification. Other considerations aside, a significant drawback to proof theoretic verification is the problem of computational complexity. As we saw above, even if the semantic language is as impoverished as classical propositional logic, verification will be co-NP-complete. In reality, logics for verification must be considerably more expressive than this.

Problems with the computational complexity of verification logics led researchers in theoretical computer science to investigate other approaches to formal verification. The most successful of these is *model*

checking [27, 22, 10]. The idea behind model checking is as follows. Recall that in proof theoretic verification, to verify that a program π_i has some property φ when in state l, we derive the theory of that program $[\![\pi_i, l]\!]_\Pi$ and attempt to establish $[\![\pi_i, l]\!]_\Pi \vdash \varphi$, i.e., that property φ is a theorem of the theory $[\![\pi_i, l]\!]_\Pi$. In temporal semantics, for example [28, 29], $[\![\pi_i, l]\!]_\Pi$ is a temporal logic formula such that the models of this formula correspond to all possible runs of the program π_i.

In contrast, model checking approaches work as follows. To determine whether or not π_i has property φ when in state l, we proceed as follows:

- Take π_i, l, and from them generate a model $M_{\pi_i,l}$ that encodes all the possible computations of π.
- Determine whether or not $M_{\pi_i,l} \models \varphi$, i.e., whether the formula φ is valid in $M_{\pi_i,l}$; the program π_i has property φ in state l just in case the answer is "yes".

In order to encode all computations of the program, the model generated in the first stage will be a *branching time* temporal model [16]. Intuitively, each branch, (or path), through this model will correspond to one possible execution of the program. Such a model can be generated automatically from the text of a program in a typical imperative programming language.

The main advantage of model checking over proof theoretic verification is in complexity: model checking using the branching time temporal logic CTL [11] can be done in time $O(|\varphi| \times |M|)$, where $|\varphi|$ is the size of the formula to be checked, and $|M|$ is the size of the model (i.e., the number of states it contains) [16]. Model-checking approaches have recently been used to verify finite-state systems with up to 10^{120} states [10].

Using a model checking approach to conformance testing for ACLs, we would define the program semantics as a function

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

which assigns to every program/state pair an $\mathcal{L}_\mathcal{S}$-model, which encodes the properties of that program/state pair. Verifying that $(\pi_i, l) \models_f \mu$ would involve checking whether $[\![\pi_i, l]\!]_\Pi \models_\mathcal{S} [\![\mu]\!]_\mathcal{C}$, i.e., whether the sincerity condition $[\![\mu]\!]_\mathcal{C}$ was valid in model $[\![\pi_i, l]\!]_\Pi$.

The comparative efficiency of model checking is a powerful argument in favour of the approach. Algorithms have been developed for (propositional) belief-desire-intention logics that will take a model and

a formula and will efficiently determine whether or not the formula is satisfied in that model [35, 5]. These belief-desire-intention logics are closely related to those used to give a semantics to the FIPA-97 ACL. However, there are two unsolved problems with such an approach.

The first problem is that of developing the program semantics $[\![\_]\!]_\Pi$. We have procedures that, given a program, will generate a branching temporal model that encode all computations of that program. However, these are *not* the same as models for belief-desire-intention logics. Put simply, the problem is that we do not yet have any techniques for systematically assigning beliefs, desires, intentions, and uncertainties (as in the FIPA-97 SL case [19]) to arbitrary programs. This is again the problem of *grounding* that we referred to above. As a consequence, we cannot do the first stage of the model checking process for ACLs that have (ungrounded) FIPA-like semantics.

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.

To summarise, model checking approaches appear to have considerable advantages over proof-theoretic approaches to verification with respect to their much reduced computational complexity. However, as with proof-theoretic approaches, the problem of ungrounded ACL semantics remains a major problem, with no apparent route of attack. Also, the problem of model checking with quantified logics is an as-yet untested area. Nevertheless, model checking seems a promising direction for ACL conformance testing.

7. 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 article 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 $\mathcal{L}_{\mathcal{S}}$ used to give a semantics to the communication language. $\mathcal{L}_{\mathcal{S}}$ is 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 mentalistic 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. Model checking approaches appear to be a promising alternative.

In addition, the article has given examples of agent communication frameworks, some of which are verifiable by this definition, others of which, (including the FIPA-97 ACL [19]), are not.

The results of this article 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 article 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 important. However, it may well be that we need new ways of thinking about the semantics and verification of such standards. A number of promising approaches have recently appeared in the literature [39, 34, 40]. One approach that can work effectively in certain cases is *mechanism design* [36]. The basic idea is that in certain multi-agent scenarios (auctions are a well-known example), it is possible to design an interaction protocol so that the dominant strategy for any participating agent is to tell the truth. Vickrey's mechanism is probably the best-known example of such a technique [37]. In application domains where such techniques are feasible, they can be used to great effect. However, most current multi-agent applications do not lend themselves to such techniques. While there is therefore great potential for the application of mechanism design in the long term, in the short term it is unlikely to play a major role in agent communication standards.

References

1. Abadi, M.: 1987, 'Temporal Logic Theorem Proving'. Ph.D. thesis, Computer Science Department, Stanford University, Stanford, CA 94305.
2. Allen, J. F., J. Hendler, and A. Tate (eds.): 1990, *Readings in Planning*. Morgan Kaufmann Publishers: San Mateo, CA.
3. Appelt, D. E.: 1985, *Planning English Sentences*. Cambridge University Press: Cambridge, England.
4. Austin, J. L.: 1962, *How to Do Things With Words*. Oxford University Press: Oxford, England.
5. Benerecetti, M., F. Giunchiglia, and L. Serafini: 1999, 'A model checking algorithm for multiagent systems'. In: J. P. Müller, M. P. Singh, and A. S. Rao (eds.): *Intelligent Agents V — Proceedings of the Fifth International Workshop on Agent Theories, Architectures, and Languages (ATAL-98)*, Lecture Notes in Artificial Intelligence. Springer-Verlag, Heidelberg.
6. Bond, A. H. and L. Gasser (eds.): 1988, *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann Publishers: San Mateo, CA.
7. Boyer, R. S. and J. S. Moore (eds.): 1981, *The Correctness Problem in Computer Science*. Academic Press.
8. Brafman, R. I. and M. Tennenholtz: 1997, 'Modeling Agents as Qualitative Decision Makers'. *Artificial Intelligence* **94**(1-2), 217–268.
9. Bretier, P. and D. Sadek: 1997, '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 (eds.): *Intelligent Agents III (LNAI Volume 1193)*. pp. 189–204.
10. Clarke, E., O. Grumberg, and D. Long: 1994, 'Verification tools for finite-state concurrent systems'. In: J. W. de Bakker, W. P. de Roever, and G. Rozenberg (eds.): *A Decade of Concurrency — Reflections and Perspectives (LNCS Volume 803)*. Springer-Verlag: Berlin, Germany, pp. 124–175.
11. Clarke, E. M. and E. A. Emerson: 1981, 'Design and Synthesis of Synchronization Skeletons Using Branching Time Temporal Logic'. In: D. Kozen (ed.): *Logics of Programs — Proceedings 1981 (LNCS Volume 131)*. pp. 52–71.
12. Cohen, P. R. and H. J. Levesque: 1990a, 'Intention is Choice with Commitment'. *Artificial Intelligence* **42**, 213–261.
13. Cohen, P. R. and H. J. Levesque: 1990b, 'Rational Interaction as the basis for Communication'. In: P. R. Cohen, J. Morgan, and M. E. Pollack (eds.): *Intentions in Communication*. The MIT Press: Cambridge, MA, pp. 221–256.
14. Cohen, P. R. and H. J. Levesque: 1995, 'Communicative Actions for Artificial Agents'. In: *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*. San Francisco, CA, pp. 65–72.
15. Cohen, P. R. and C. R. Perrault: 1979, 'Elements of a Plan Based Theory of Speech Acts'. *Cognitive Science* **3**, 177–212.
16. Emerson, E. A.: 1990, 'Temporal and Modal Logic'. In: J. van Leeuwen (ed.): *Handbook of Theoretical Computer Science*. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, pp. 996–1072.
17. Fagin, R., J. Y. Halpern, Y. Moses, and M. Y. Vardi: 1995, *Reasoning About Knowledge*. The MIT Press: Cambridge, MA.
18. Fikes, R. E. and N. Nilsson: 1971, 'STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving'. *Artificial Intelligence* **5**(2), 189–208.

19. FIPA: 1997, 'Specification Part 2 — Agent Communication Language'. The text refers to the specification dated 23 October 1997.
20. Guilfoyle, C., J. Jeffcoate, and H. Stark: 1997, *Agents on the Web: Catalyst for E-Commerce*. London: Ovum Ltd.
21. Halpern, J. Y. and Y. Moses: 1992, 'A Guide to Completeness and Complexity for Modal Logics of Knowledge and Belief'. *Artificial Intelligence* **54**, 319–379.
22. Halpern, J. Y. and M. Y. Vardi: 1991, 'Model Checking versus Theorem Proving: A Manifesto'. In: V. Lifschitz (ed.): *AI and Mathematical Theory of Computation — Papers in Honor of John McCarthy*. Academic Press, pp. 151–176.
23. Harel, D.: 1984, 'Dynamic Logic'. In: D. Gabbay and F. Guenther (eds.): *Handbook of Philosophical Logic Volume II — Extensions of Classical Logic*. D. Reidel Publishing Company: Dordrecht, The Netherlands, pp. 497–604. (Synthese library Volume 164).
24. Hoare, C. A. R.: 1969, 'An Axiomatic Basis for Computer Programming'. *Communications of the ACM* **12**(10), 576–583.
25. Jennings, N. R.: 1993, 'Commitments and Conventions: The Foundation of Coordination in Multi-Agent Systems'. *The Knowledge Engineering Review* **8**(3), 223–250.
26. Labrou, Y. and T. Finin: 1997, 'Semantics and Conversations for an Agent Communication Language'. In: *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*. Nagoya, Japan, pp. 584–591.
27. Lichtenstein, O. and A. Pnueli: 1984, 'Checking that Finite State Concurrent Programs Satisfy Their Linear Specification'. In: *Proceedings of the Eleventh ACM Symposium on the Principles of Programming Languages*. pp. 97–107.
28. Manna, Z. and A. Pnueli: 1992, *The Temporal Logic of Reactive and Concurrent Systems*. Springer-Verlag: Berlin, Germany.
29. Manna, Z. and A. Pnueli: 1995, *Temporal Verification of Reactive Systems — Safety*. Springer-Verlag: Berlin, Germany.
30. Mayfield, J., Y. Labrou, and T. Finin: 1996, 'Evaluating KQML as an Agent Communication Language'. In: M. Wooldridge, J. P. Müller, and M. Tambe (eds.): *Intelligent Agents II (LNAI Volume 1037)*. Springer-Verlag: Berlin, Germany, pp. 347–360.
31. Moore, R. C.: 1990, 'A Formal Theory of Knowledge and Action'. In: J. F. Allen, J. Hendler, and A. Tate (eds.): *Readings in Planning*. Morgan Kaufmann Publishers: San Mateo, CA, pp. 480–519.
32. Papadimitriou, C. H.: 1994, *Computational Complexity*. Addison-Wesley: Reading, MA.
33. Patil, R. S., R. E. Fikes, P. F. Patel-Schneider, D. McKay, T. Finin, T. Gruber, and R. Neches: 1992, 'The DARPA Knowledge Sharing Effort: Progress Report'. In: C. Rich, W. Swartout, and B. Nebel (eds.): *Proceedings of Knowledge Representation and Reasoning (KR&R-92)*. pp. 777–788.
34. Pitt, J. and E. H. Mamdani: 1999, 'A Protocol-Based Semantics for an Agent Communication Language'. In: *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*. Stockholm, Sweden.
35. Rao, A. S. and M. P. Georgeff: 1993, 'A Model-Theoretic Approach to the Verification of Situated Reasoning Systems'. In: *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93)*. Chambéry, France, pp. 318–324.

36. Rosenschein, J. S. and G. Zlotkin: 1994, *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers.* The MIT Press: Cambridge, MA.
37. Sandholm, T.: 1999, 'Distributed Rational Decision Making'. In: G. Weiss (ed.): *Multiagent Systems.* The MIT Press: Cambridge, MA, pp. 201–258.
38. Searle, J. R.: 1969, *Speech Acts: An Essay in the Philosophy of Language.* Cambridge University Press: Cambridge, England.
39. Singh, M.: 1998, 'Agent Communication Languages: Rethinking the Principles'. *IEEE Computer* pp. 40–47.
40. Wooldridge, M.: 1999, 'Verifying that Agents Implement a Communication Language'. In: *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99).* Orlando, FL, pp. 52–57.
41. Wooldridge, M. and N. R. Jennings: 1995, 'Intelligent Agents: Theory and Practice'. *The Knowledge Engineering Review* **10**(2), 115–152.