Temporal Logics for Normative Agent Communication Protocols

Ulle Endriss
Department of Computing, Imperial College London (UK)
ue@doc.ic.ac.uk

ABSTRACT

We sketch how to express typical features of agent communication protocols in a simple temporal logic and show that conformance verification at runtime reduces to a generalised form of model checking.

Categories and Subject Descriptors

F.4.1 [**Theory of Computation**]: Mathematical Logic—*Temporal logic*; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Theory, Verification

Keywords

Model checking, Protocol conformance

1. INTRODUCTION

Communication is a central issue in multiagent systems research. While much work has been devoted to so-called mentalistic models of communication, where communicative acts are specified in terms of agents' beliefs and intentions, recently a number of authors have argued for a *convention-based* approach to agent communication languages [4, 5]. Mental attitudes are useful to explain *why* agents may behave in certain ways, but (being non-verifiable for an outside observer) they cannot serve as a basis for specifying the *norms* and *conventions* of interaction required for building open systems that allow for meaningful communication. In the convention-based approach, *protocols* specifying the rules of interaction play a central role.

A protocol specifies under what circumstances a given dialogue (between two or more agents) should be considered *legal* (*i.e.* conformant to the social rules governing the system to which the protocol applies). We are going to restrict our attention to protocols that impose constraints on turn-taking (*who* is allowed to speak?) and, in particular, the *performatives* of dialogue moves (rather than their application-specific *content*). An example for a dialogue performative would be *accept* and a protocol rule might specify that this performative may only be used in response to an earlier *propose*-move. In the context of negotiation, for instance, the content of such a move would have to include an exact specification of the actual deal being proposed. The language used for describing this deal would depend on the concrete application in question and we cannot expect to be able to devise a general-purpose language for the specification of protocol rules relating to such content items.

A variety of mechanisms for the specification of protocols have been put forward in the literature; good examples are deterministic finite automata. In the present paper we advocate the use of temporal logic for the specification of convention-based communication protocols. More specifically, we are going to use *propositional linear temporal logic* (PLTL), which is probably the most intuitive of the standard temporal logics [3]. Formulas of PLTL are evaluated over the non-negative integers (as every integer, or *time point*, will correspond to a turn in a finite dialogue, we are going to evaluate formulas over initial segments of the non-negative integers only). The language of PLTL can express statements such as $\bigcirc\varphi$ (φ is true at the next time point), $\Diamond\varphi$ (φ is true at some future time point), and φ UNTIL ψ (ψ is true at some future time point and φ remains true until then).

2. MODELS AND DIALOGUES

Given a model $\mathcal{M}$ and a formula φ, the *model checking* problem is the problem of deciding whether φ is true at every point in $\mathcal{M}$. In the sequel, we are going to formulate the problem of checking conformance of a dialogue to a protocol as a (variant of the) model checking problem.

We are going to use a special class of PLTL models to represent dialogues between agents and PLTL formulas to specify protocols. For every agent A referred to in the protocol under consideration, we assume that the set $\mathcal{L}$ of propositional letters includes a special proposition $turn(A)$ and that there are no other propositions of this form in $\mathcal{L}$. Furthermore, we assume that the set of performatives in our communication language is a subset of $\mathcal{L}$, and that $\mathcal{L}$ includes the special proposition INITIAL. We say that a model *represents* a dialogue iff it meets the following conditions: (1) INITIAL is true at point 0 and at no other $t > 0$; (2) exactly one proposition of the form $turn(\_)$ is true at any point $t > 0$; (3) exactly one performative is true at any point $t > 0$.

AAMAS'05, July 25-29, 2005, Utrecht, Netherlands.

An actual dialogue determines a *partial* model: It fixes the frame as well as the valuation for INITIAL and the propositions in $\mathcal{L}$ corresponding to turn-assignments and performatives, but it does not say anything about any of the other propositional letters that we may have in our language $\mathcal{L}$ (e.g. to represent dialogue states). We can *complete* a given partial model by arbitrarily fixing the valuation for the remaining propositional letters. Every possible way of completing a dialogue model in this manner givens rise to a different PLTL model, *i.e.* a dialogue typically corresponds to a whole classes of models.

This is why we cannot use standard model checking (which applies to single models) to decide whether a given dialogue satisfies a formula encoding a protocol. Instead, the reasoning problem we are interested in is this:

> *Given a partial model $\mathcal{M}$ (induced by a dialogue) and a formula φ (the specification of a protocol), is there a full model $\mathcal{M}'$ completing $\mathcal{M}$ such that φ is true at every point in $\mathcal{M}'$?*

In other words, we have to decide whether the partial description of a model can be completed in such a way that model checking would succeed.

The above problem is known as the *generalised model checking* problem and has been studied by Bruns and Godefroid [1]. In fact, the problem addressed by these authors is slightly more general than ours, as they do not work with a fixed frame and distinguish cases where all complete instances of the partial model validate the formula from those where there exists at least one such instance. Generalised model checking may be regarded as a combination of satisfiability checking and model checking in the usual sense. If there are no additional propositions in $\mathcal{L}$, then generalised model checking reduces to standard model checking. If we can characterise the class of all models representing a given dialogue by means of a formula ψ, then φ and ψ can be used to construct a formula that is satisfiable (has got a model) iff that dialogue conforms to the protocol given by φ.

3. EXAMPLES

A wide range of communication protocols studied in the multiagent systems literature have been modelled using deterministic finite automata. This class of protocols can be represented using a fragment of PLTL where the only temporal operator required is the next-operator $\bigcirc$. If our language $\mathcal{L}$ includes a propositional letter of the form $state(i)$ for every state i, then we can describe the *state transition function* of an automaton by means of formulas such as this:

$$state(5) \wedge \bigcirc propose \rightarrow \bigcirc state(3)$$

This formula expresses that uttering a *propose*-move in state 5 takes us to state 3. Next we have to specify the range of *legal follow-ups* for every dialogue state. For example, from state 3 we may only allow for two kinds of moves:

$$state(3) \rightarrow \neg \bigcirc \neg (accept \vee reject)$$

Observe that we use the *weak* variant of the next-operator $\neg \bigcirc \neg$ (rather than just $\bigcirc$). Otherwise the above formula would not be satisfied in a model representing an *unfinished* (but otherwise legal) dialogue. In case we want model checking to succeed only for *complete* dialogues, we can add the formula $state(i) \rightarrow \bigcirc \top$ to the specification for every non-final state i. We can formulate similar rules, using the propositional letters of the form $turn(\_)$, to regulate a protocol's turn-taking policy.

For many purposes, purely automata-based protocols are not sufficient. For instance, they do not support the specification of general *future obligations* (or *commitments*) on the communicative behaviour of an agent. This is an important feature of many protocols proposed in the literature. In the context of an auction protocol, for example, we may say that, by opening an auction, an auctioneer acquires the obligation to close that auction again at some later stage. Suppose these actions can be performed by making a dialogue move with the performatives *open-auction* and *end-auction*, respectively. Again, we do not want model checking to fail just because a dialogue has not yet been completed; that is, a specification such as $open\text{-}auction \rightarrow \Diamond end\text{-}auction$ would be too simplistic. Instead, we may use the following formula:

$$open\text{-}auction \rightarrow \text{PENDING} \wedge (\text{PENDING UNLESS } end\text{-}auction)$$

Here φ UNLESS ψ a shorthand for $(\varphi \text{ UNTIL } \psi) \vee \Box\varphi$. The new proposition PENDING is used to mark time points at which there are still obligations to be met. A model representing a dialogue where *open-auction* has been uttered, but *end-auction* has not, will satisfy this rule. However, PENDING will be true at its very last time point. If we want to check whether a dialogue does not only not violate any rules but also fulfils all obligations, then we can run generalised model checking with a specification including the additional formula $\text{PENDING} \rightarrow \bigcirc \top$.

4. CONCLUSION

We have argued that temporal logic can be used to specify convention-based agent communication protocols in a simple and elegant manner. In particular, we have indicated how to use PLTL to specify both very simple automata-based protocols and protocols involving dialogue obligations. Of course, using this logic to express the kinds of properties we have considered in our examples is not new, but the application of this technique to the specification of conversational conventions is both novel and, we believe, very promising. We have also identified generalised model checking as a tool for checking protocol conformance at runtime.

A full version of this paper is due to appear in the proceedings of the AAMAS Workshop on Agent Communication [2].

5. REFERENCES

[1] G. Bruns and P. Godefroid. Generalized model checking: Reasoning about partial state spaces. In *11th Intl. Conf. on Concurrency Theory*. Springer, 2000.

[2] U. Endriss. Temporal logics for representing agent communication protocols. In *AAMAS Workshop on Agent Communication*, 2005. To appear.

[3] D. Gabbay, I. Hodkinson, and M. Reynolds. *Temporal Logic: Mathematical Foundations and Computational Aspects*, volume 1. Oxford University Press, 1994.

[4] J. Pitt and A. Mamdani. A protocol-based semantics for an agent communication language. In *16th Intl. Joint Conf. on Artif. Intell.* Morgan Kaufmann, 1999.

[5] M. P. Singh. Agent communication languages: Rethinking the principles. *IEEE Computer*, 31(12):40–47, 1998.