Proving Properties of Open Agent Systems

Frank Guerin and Jeremy Pitt
Intelligent and Interactive Systems, Department of Electrical & Electronic Engineering,
Imperial College of Science, Technology & Medicine, Exhibition Road, London, SW7 2BT.
{f.guerin,j.pitt}@ic.ac.uk

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Distributed Artificial Intelligence—*Multiagent systems*; D.2.4 [**Software**]: Software/Program Verification[Model checking]

General Terms

Verification, Economics.

1. MOTIVATION

In an open agent system the constituent agents are developed and owned by different individuals or organisations who may have conflicting interests. Hence the internals (i.e. program and state) of agents are not public and so notions of trust and deception are relevant. It is proposed that such systems will be used in scenarios where legally binding contracts are made and money is exchanged by the agents on behalf of their owners. Not surprisingly, agent owners can be expected to be reluctant to delegate tasks involving potentially detrimental outcomes to an agent unless they can be assured that the system has certain desirable properties. It may be a requirement, for example, that an agent cannot profit from lying to its peers. Solutions from game theory and economics [1] allow us to design *mechanisms* for interactions (a set of public rules governing an interaction) which provide incentives for participants to behave as we desire, for example to tell the truth; mechanisms can be designed to have properties such as individual rationality, incentive compatibility and stability. Having chosen a suitable mechanism, we can implement it for an agent system as a *protocol* and prove that these properties hold for the agent system.

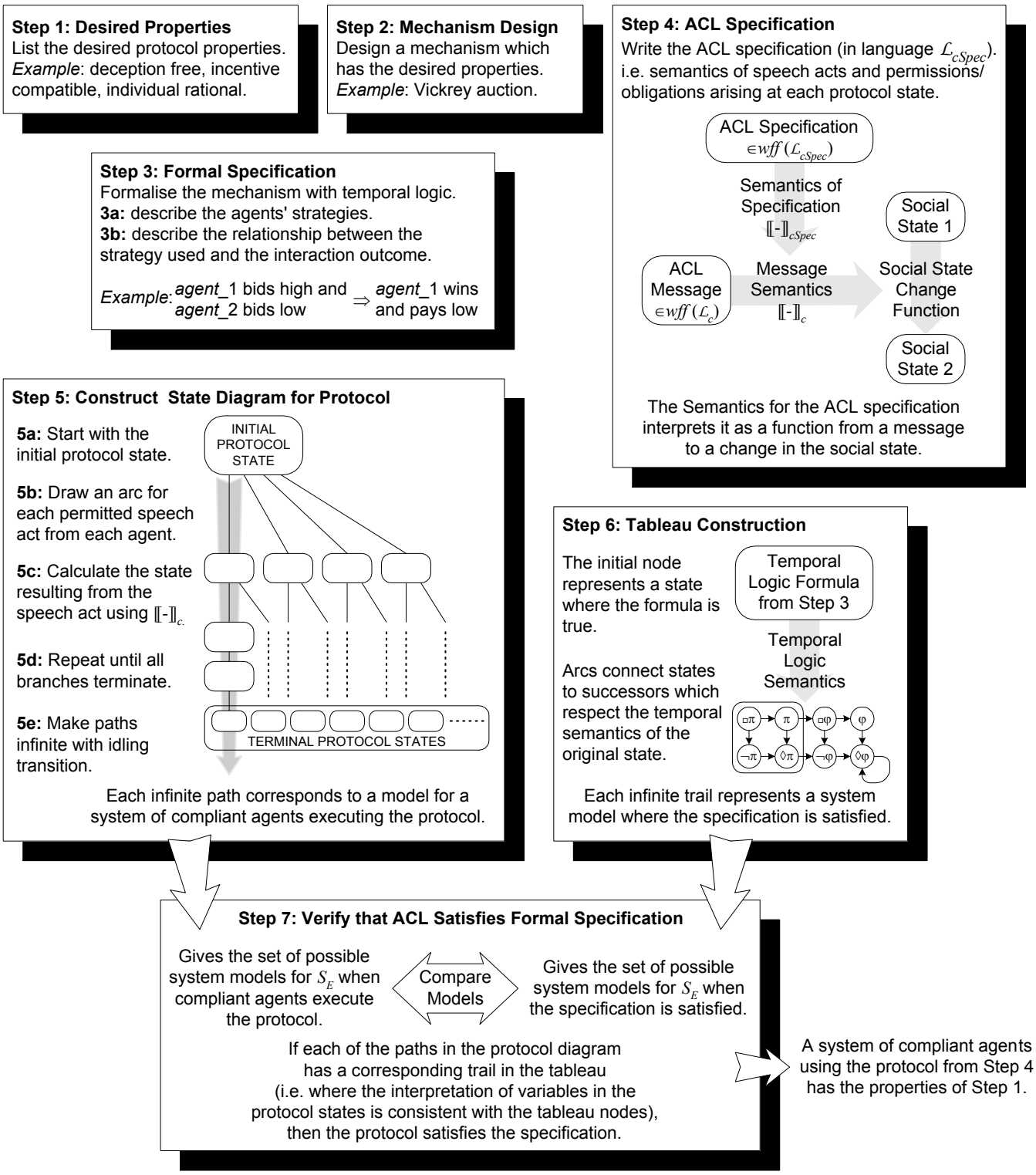
2. FORMAL FRAMEWORK

To prove that a system of agents has the properties of a chosen mechanism we must formalise both the system and the mechanism's properties. We formalise the system as a *fair transition system* [4] which has a set of system variables, an initial condition, a set of transitions by which a system state can change and fairness requirements. We may not have access to the internals of any agent; hence our system variables only describe externally observable phenomena: communication channels and *social* state variables. The system has two transitions: the idling transition (does nothing) and the environmental transition (allows a channel to be modified by adding or removing a message). We call this system an *external* system S_E; states of S_E are *social states* of our agents. A model of S_E is an infinite sequence of states; it is a computation of S_E if the first state of the model satisfies the initial condition and each state is accessible from the previous one by one of the transitions and it meets the fairness requirements. Our system allows any message to be sent in a transition, so we cannot guarantee anything about its behaviour; we need to constrain the messages that can be sent; we do this with an *agent communication language* or ACL. In previous work [2] we described how an ACL specification language (call this $\mathcal{L}_{cSpec}$) can be used to specify an ACL which defines the semantics of communication in terms of a change to the social state; it can also define protocols that constrain the messages that agents can send. Since the ACL is based on social states, states of the protocol correspond directly to states of our system S_E.

We design a protocol that constrains the agents in the same way as the mechanism we have chosen. We then need to prove that agents following this protocol do have the property the mechanism is supposed to have. We formalise the mechanism's properties as a temporal logic formula which describes the actions agents may take and the consequences. Using the model checking approach, we identify the set of possible computations (system models) which could be produced by compliant agents executing the protocol and we then compare these with the set of models where the mechanism's temporal logic formula is satisfied. Since our proof relies on agents being compliant, there must be some means of enforcement so that rogue agents will not be permitted to damage the system's properties; at least there should be a method for determining if agents are misbehaving at runtime. This is possible because a social language is used [5]; we can observe agent actions and compare them with the permissions and obligations arising in social states.

Our framework allows a mechanism to be specified in the form of a protocol and made public so that agent designers can inspect it. We have provided a procedure by which they can verify the properties of the protocol. This work has the potential to increase the range of applications in which agent owners may be willing to delegate to their embedded counterparts. For more detailed information please see [3].

AAMAS'02, July 15-19, 2002, Bologna, Italy.



3. REFERENCES

[1] K. Binmore. *Fun and Games: A text on Game theory.* D.C. Heath and Co., Lexington, Massachusetts, 1992.

[2] F. Guerin and J. Pitt. Denotational semantics for agent communication languages. In *Autonomous Agents 2001, Montreal*, pages 497–504. ACM Press, 2001.

[3] F. Guerin and J. Pitt. IIS Technical Report TRS020015. http://www.iis.ee.ic.ac.uk/reports, 2002.

[4] Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems (Safety)*. Springer-Verlag, 1995.

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