Model-Checking Auctions, Coalitions and Trust[1]

Matt Webster* Louise Dennis* Michael Fisher*
*Department of Computer Science, University of Liverpool, Liverpool, L69 3BX, UK
{matt,louised,mfisher}@liverpool.ac.uk

Abstract

In this paper we tackle the verification of auction models that are at the heart of many market-based multi-agent systems. Specifically, we program auctions with auctions and trust in a BDI-based programming language and then use agent model checking to verify logical properties concerning beliefs within the multi-agent system.

1 Auctions and Verification in Multi-Agent Systems

The basic idea of an *auction* (Klemperer, 2004) is at the heart of many multi-agent scenarios. Not only are auctions central to e-commerce applications, but are implicit within many *market-based* approaches to agent computation. These include resource-allocation, telecommunications, electricity supply management, agent mobility, task allocation and scheduling. However, although much work has been carried out on the deep analysis of auction mechanisms, such as through formal mechanism design (Wooldridge et al., 2007), the analysis of *implementations* of auction mechanisms has lagged behind. While there has been *some* work on the formal verification of auction implementations, such as (Doghri, 2008), this has been lacking an agent perspective. Thus, the more sophisticated agent aspects such as *goals*, *intentions*, *beliefs* and *deliberation* have not been verified within an auction context. It is this we aim to tackle in this paper.

We have been developing techniques for verifying multi-agent systems over a number of years. The agents involved are rational (and, so, are capable of "intelligent" autonomous behaviour) and are represented in a high-level language describing their beliefs, intentions, etc. In particular, we have previously looked at the verification of BDI languages such as GWENDOLEN, and both homogeneous and heterogeneous multi-agent systems (Dennis and Fisher, 2008). In order to carry out formal verification, formal descriptions of both the system being considered and the properties to be verified are required. Producing formal descriptions for multi-agent systems is well understood, with the BDI approach to rational agency being particularly popular (Rao and Georgeff, 1995). Our approach, based on *model checking*, is outlined in (Bordini et al., 2008; Webster et al., 2009), and is available from `http://sourceforge.net/projects/mcapl`.

A multi-agent program, originally programmed in some agent programming language is executed in an interpreter via an interface to the *Agent Infrastructure Layer (AIL)*. Each agent uses AIL data structures to store its internal state comprising, for instance, a belief base, a plan library, a current intention, and a set of further intentions (as well as other temporary state information). They also use the specific interpreter for the agent programming language built using AIL classes and methods. Therefore, the AIL streamlines the development and verification of heterogenous multi-agent systems. The interpreter defines the reasoning cycle for the agent programming language which interacts with the model checker, essentially notifying it when a new state is reached that is relevant for verification. This allows the model checker, AJPF (*Agent JPF*), to create a Java product automata from the program and a property defined in the AJPF property specification language.

The product automata runs in the JPF virtual machine (see `http://javapathfinder.sourceforge.net`). This is a Java virtual machine specially designed to maintain backtrack points and explore, for instance, all possible thread scheduling options (that can affect the result of the verification). The JPF model checker is extensible and configurable, which allows us to optimise its performance for AIL-based systems, for instance AJPF has a specialised *Property Listener* which terminates execution of the product automata if it detects that the property has been, and will remain, satisfied for the rest of the run.

1.1 Example: Auction with Coalitions and Trust

A basic sealed-bid English auction scenario was extended to include the possibility of *coalitions* (Sandholm and Lesser, 1997). In our model, a coalition is when several agents collaborate by pooling their bid resources in order to win the auction. For example, if three agents x, y, z bid 100, 150 and 200 respectively, then z ought to win every time. However, if x and y form a coalition, their collective bid of 250 will be enough to win the auction.

[1]Work funded by EPSRC project "Model Checking Agent Programming Languages" (EP/D052548).

Our simple coalition scenario consists of five agents: one auctioneer, four bidders. At the start of the auction all bidders submit a bid and the auctioneer announces the winner. One of the agents is a coalition-forming agent: if it loses the first auction, it then tries to form a coalition with an agent it trusts. After forming a coalition the agent bids again. Then, if its coalition is successful in winning the auction, it stops. However, if its coalition is unsuccessful then it no longer believes that it can trust the other agent in the coalition, and will try to form another coalition with another agent it trusts.

For illustrative purposes a specification for the coalition-forming agent is laid out below in the GWENDOLEN agent programming language; the complete specification is given in (Webster et al., 2009).

AGENT: ag2
Initial Beliefs: $my\_name(ag2), trust(ag4), trust(ag5)$
Initial Goals: $+!_p bid$
Plans:

$\downarrow^{Ag}$ **tell**$(B) : \top$ <- $+B$

$+!_p bid : my\_name(Name) \wedge \neg\uparrow^{ag1}$ **tell**$(bid(150,Name))$ <- $\uparrow^{ag1}$ **tell**$(bid(150,Name))$

$$+win(A) : \left[\begin{array}{l} my\_name(Name) \wedge \neg win(Name) \wedge trust(Ag) \\ \wedge \neg formed\_coalition(Ag') \wedge \neg\uparrow^{Ag} \textbf{tell}(coalition(Name)) \end{array} \right] \text{ <- } \left[\begin{array}{l} \uparrow^{Ag} \textbf{tell}(coalition(Ag)); \\ +formed\_coalition(Ag) \end{array} \right]$$

$$+win(A) : \left[\begin{array}{l} my\_name(Name) \wedge \neg win(Name) \wedge trust(Ag) \\ \wedge formed\_coalition(Ag') \wedge \neg\uparrow^{Ag} \textbf{tell}(coalition(Name)) \end{array} \right] \text{ <- } \left[\begin{array}{l} \uparrow^{Ag} \textbf{tell}(coalition(Ag)); \\ +formed\_coalition(Ag); \\ -trust(Ag') \end{array} \right]$$

$+agree(A,X) : \top$ <- $\uparrow^{ag1}$ **tell**$(bid(150+X, ag2))$

The GWENDOLEN agent specification was converted automatically into a lower-level Java/AIL program and verified that $\Diamond \mathsf{B}(a_1, win)$ where a_1 forms a coalition with one of the trusted agents after losing the auction to a_2, i.e. eventually the agent believes that a_1 will win. If it chooses a_4 first, it wins and stops. If it chooses a_3 first, it loses the auction and distrusts that agent, trying subsequently with a_4. It then wins the auction. Verification took place in 20m 30s using 22 MB on a Windows XP PC with an Intel Core 2 Duo (E6750 @ 2.66GHz) CPU and 2 GB of RAM.

2 Concluding Remarks

In this paper we have discussed the verification by model-checking of agent-based auction software. We have shown that it is a realistic proposition to model-check the properties of interesting multi-agent implementations within a reasonable time. For example, as we reach the situation where agents form coalitions together, based on a dynamic notion of trust, in order to compete with other agents, then we are not far from realistic agent systems. Clearly, for bigger scenarios improved efficiency will be required (and, indeed, this is something we are actively working on), but the examples implemented and verified in this paper show that small, but non-trivial, market-based multi-agent systems can be automatically verified.

References

R. H. Bordini, L. A. Dennis, B. Farwer, and M. Fisher. Automated Verification of Multi-Agent Programs. In *Proc. 23rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 69–78, 2008.

L. A. Dennis and M. Fisher. Programming Verifiable Heterogeneous Agent Systems. In Koen Hindriks, Alexander Pokahr, and Sebastian Sardina, editors, *Sixth International Workshop on Programming in Multi-Agent Systems (ProMAS'08)*, Estoril, Portugal, May 2008.

Ines Doghri. Formal verification of WAHS: an autonomous and wireless P2P auction handling system. In *Proc. 8th International Conference on New Technologies in Distributed Systems (NOTERE)*, pages 1–10, New York, NY, USA, 2008. ACM. doi: http://doi.acm.org/10.1145/1416729.1416754.

Paul Klemperer. *Auctions: Theory and Practice*. Princeton University Press, Princeton, USA, 2004.

A. S. Rao and M. Georgeff. BDI Agents: From Theory to Practice. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS)*, pages 312–319, San Francisco, CA, June 1995.

Tuomas Sandholm and Victor R. Lesser. Coalitions Among Computationally Bounded Agents. *Artificial Intelligence*, 94(1-2), 1997.

Matt Webster, Louise Dennis, and Michael Fisher. Model-Checking Auctions, Coalitions and Trust. Technical Report ULCS-09-004, Department of Computer Science, University of Liverpool, 2009. http://www.csc.liv.ac.uk/research/techreports/tr2009/ulcs-09-004.pdf.

Michael Wooldridge, Thomas Ågotnes, Paul E. Dunne, and Wiebe van der Hoek. Logic for Automated Mechanism Design - A Progress Report. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, pages 9–. AAAI Press, 2007.