A Common Semantic Basis for BDI Languages*

Louise A. Dennis⋆ Rafael H. Bordini† Berndt Farwer†
Michael Fisher⋆

⋆Department of Computer Science, University of Liverpool †Department of Computer Science, University of Durham

As the concept of an 'agent' becomes more popular, so the variety of programming languages based upon this concept increases. These *agent-based* programming languages range from minimal extensions of JAVA through to logic-based languages for 'intelligent' agents (Bordini et al. (2005a)). In our work, we are particularly concerned (at least initially) with approaches based on *rational agent theories*, primarily the *BDI theory* developed by Rao and Georgeff (1995). Such languages not only incorporate the autonomous behaviour required for the agent concept, but also provide sophisticated mechanisms for instigating, controlling, and reasoning about such behaviours.

Though programming languages based on the BDI approach (let us call these *BDI languages*) are increasingly popular, there are several problems, for example:

1. there are *too* many languages;

2. many of the languages are similar, yet subtly different – this makes it difficult for developers to learn more than one language, as they are not based on agreed notions/definitions; further, such differences make it difficult to identify precisely the general mechanisms and to transfer new techniques between languages; and

3. in spite of the fact that many BDI languages have logical semantics and utilise logical mechanisms, formal verification tools are rare.

This last aspect is particularly important, since BDI approaches are increasingly used in complex, critical applications such as space exploration (Muscettola et al. (1998); Clancey et al. (2003); Sierhuis (2006)).

In our work[1] we are attempting to design an intermediate language (called AIL– *Agent Infrastructure Layer*) for BDI-style programming languages. There are several motivations for this, including:

- providing a common semantic basis for a number of BDI languages, thus clarifying issues and aiding further programming language development;

- supporting formal verification by developing a *model-checker* optimised for checking AIL programs – existing BDI languages can have compilers for AIL so as to take advantage of its associated model-checker; and

- providing, potentially, a high-level virtual machine for efficient and portable implementation.

Rather than attempting to cover all BDI languages from the start, we have initially tackled some of the most popular. Thus, we have principally referred to the variant of AgentSpeak (Rao (1996)) used in *Jason* (Bordini et al. (2005b)) and 3APL (Dastani et al. (2005)) when designing the semantics for the AIL, but have also taken Jadex (Pokahr et al. (2005)) and (Concurrent) METATEM (Fisher (2005)) into account.

The current design for AIL, in the form of an extensive operational semantics, can be found in Dennis (2007) and a discussion in Dennis et al. (2007). In order to model a particular language in AIL it will be necessary to create an AIL compiler for that language. Sometimes it will prove possible to map only fragments of a given language into AIL. Our expectation is that large and useful fragments of most BDI-style agent programming languages will be translatable. In order to accommodate the main features of the primary BDI languages, AIL has some components with overlapping functionality.

In order to provide this semantics we needed to characterise the shared concepts of beliefs, goals, actions, and plans as well as accounting for common variations such as the use of events and deed stacks. Thus, our semantics develops a complex data structure to represent intentions associating events (which include outstanding goals) with stacks of deeds (which include belief updates) to be performed. A generalised notion of a plan is developed to operate on this data structure which captures many of the notions of plans available in the literature.

We have designed AIL aiming, in future work, not only to be able to accommodate a variety of languages but also to account for future developments of the existing languages. For example, most languages currently concentrate on

*Work supported by EPSRC grants EP/D054788 (Durham) and EP/D052548 (Liverpool).

[1]See *http://www.csc.liv.ac.uk/~michael/mcapl06* for details.

individual agents, so it is likely that those languages will be extended to include constructs to support the social level of multi-agent systems, particularly the notion of "organisations". AIL is therefore being designed with simple constructs which allow it to model many of the most obvious developments in this area. AIL's social organisations are currently based on METATEM's groups which flexibly allow the concepts of organisation and role to be captured (Fisher and Kakoudakis (1999)). The treatment of groups of agents as agents in their own right also provides a natural mechanism for introducing concepts of modularity into agent programs.

In the short term, planned work revolves around the implementation of AIL (in JAVA) and the provision of compilers for, at least, significant fragments of AgentSpeak and 3APL. In the longer term, the correctness of these compilers needs to be addressed and verification tools for AIL developed. In particular, we aim to use and extend the JPF model-checker (Visser et al. (2000)) so that AIL classes are treated as internal classes of JPF which should provide for efficient verification of agent programs written in various BDI languages.

References

Rafael H. Bordini, Mehdi Dastani, Jürgen Dix, and Amal El Fallah Seghrouchni, editors. *Multi-Agent Programming: Languages, Platforms and Applications*. Number 15 in Multiagent Systems, Artificial Societies, and Simulated Organizations. Springer-Verlag, 2005a.

Rafael H. Bordini, Jomi F. Hübner, and Renata Vieira. ***Jason*** and the golden fleece of agent-oriented programming. In Bordini et al. (2005a), chapter 1, pages 3–37.

William J. Clancey, Maarten Sierhuis, Charis Kaskiris, and Ron van Hoof. Advantages of Brahms for Specifying and Implementing a Multiagent Human-Robotic Exploration System. In *Proc. 16th International Florida Artificial Intelligence Research Society Conference (FLAIRS)*, pages 7–11. AAAI Press, 2003. ISBN 1-57735-177-0.

Mehdi Dastani, M. Birna van Riemsdijk, and John-Jules Ch. Meyer. Programming multi-agent systems in 3APL. In Bordini et al. (2005a), chapter 2, pages 39–67.

Louise A. Dennis. Agent Infrastructure Layer (AIL): Design and Operational Semantics v1.0. Technical Report ULCS-07-001, Department of Computer Science, University of Liverpool, 2007. Available from *http://www.csc.liv.ac.uk/research/techreports/*.

Louise A. Dennis, Rafael H. Bordini, Berndt Farwer, Michael Fisher, and Mike Wooldridge. A common semantic basis for BDI languages. In *Programming Multi-Agent Systems (ProMAS '07)*, 2007. To Appear.

M. Fisher. METATEM: The story so far. In *Proc. 3rd International Workshop on Programming Multiagent Systems (ProMAS)*, volume 3862 of *LNAI*, pages 3–22. Springer, 2005.

M. Fisher and T. Kakoudakis. Flexible Agent Grouping in Executable Temporal Logic. In *Proc. 12th International Symposium on Languages for Intensional Programming (ISLIP)*. World Scientific Press, 1999.

N. Muscettola, P. Pandurang Nayak, Barney Pell, and Brian Williams. Remote Agent: To Boldly Go Where No AI System Has Gone Before. *Artificial Intelligence*, 103(1-2):5–48, 1998.

Alexander Pokahr, Lars Braubach, and Winfried Lamersdorf. A Flexible BDI Architecture Supporting Extensibility. In *Proc. IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT)*, pages 379–385, 9 2005.

A. S. Rao and M. Georgeff. BDI Agents: from theory to practice. In *Proc. 1st International Conference on Multi-Agent Systems (ICMAS)*, pages 312–319, San Francisco, CA, June 1995.

Anand S. Rao. AgentSpeak(L): BDI Agents Speak Out in a Logical Computable Language. In *Proc. 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW)*, volume 1038 of *LNCS*, pages 42–55. Springer, 1996.

Maarten Sierhuis. Multiagent Modeling and Simulation in Human-Robot Mission Operations. (See *http://ic.arc.nasa.gov/ic/publications*), 2006.

Willem Visser, Klaus Havelund, Guillaume Brat, and SeungJoon Park. Model checking programs. In *Proceedings of the Fifteenth International Conference on Automated Software Engineering (ASE'00), 11-15 September, Grenoble, France*, pages 3–12. IEEE Computer Society, 2000.