

Prototyping 3APL in the Maude Term Rewriting Language

Extended Abstract

M. Birna van Riemsdijk¹ Frank S. de Boer^{1,2} Mehdi Dastani¹ John-Jules Ch. Meyer¹

Utrecht University¹
CWI, Amsterdam²
The Netherlands

[birna | frankb | mehdi | jj]@cs.uu.nl

ABSTRACT

This paper presents an implementation of (a simplified version of) the cognitive agent programming language 3APL in the Maude term rewriting language. We argue that Maude is very well suited for prototyping agent programming languages such as 3APL.

Categories and Subject Descriptors

D.3.1 [Programming Languages]: Formal Definitions and Theory; I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence—*Intelligent agents, languages and structures*; I.2.5 [Artificial Intelligence]: Programming Languages and Software

General Terms

Theory, Languages

Keywords

Agent programming languages, prototyping, term rewriting

1. INTRODUCTION

An important line of research in the agent systems field is research on agent programming languages [2]. This type of research is concerned with an investigation of what kind of programming constructs an agent programming language should contain, and what exactly the meaning of these constructs should be. In order to test whether these constructs indeed facilitate the programming of agents in an effective way, the programming language has to be implemented.

This can be done using Java, which was for example used for implementing the agent programming language 3APL [13]. Java has several advantages, such as its platform independence, its support for building graphical user interfaces, and the extensive standard Java libraries. A disadvantage is however that the translation of the formal semantics of an

agent programming language such as 3APL into Java is not very direct. It can therefore be difficult to ascertain that such an implementation is a faithful implementation of the semantics of the agent programming language, and experimenting with different language constructs and semantics can be quite cumbersome.

As an alternative to the use of Java, we discuss in this abstract¹ the usage of the Maude term rewriting language [6] for prototyping 3APL. Maude is based on the mathematical theory of rewriting logic. The language has been shown to be suitable both as a *logical* framework in which many other logics can be represented, and as a *semantic* framework, through which programming languages with an operational semantics can be implemented in a rigorous way [17]. We argue that, since agent programming languages such as 3APL have both a logical and a semantic component, Maude is very well suited for prototyping such languages. Further, it turns out that, since Maude is reflective, 3APL's meta-level reasoning cycle or deliberation cycle can be implemented very naturally in Maude.

An important advantage of Maude is that it can be used for verification as it comes with an LTL model checker [12]. This abstract does not further address model checking 3APL using Maude, since the usage of Maude's model checker is relatively easy, given the implementation of 3APL in Maude. The fact that Maude provides these verification facilities however, is an important, and in fact, our original, motivation for our effort of implementing 3APL in Maude.

2. IMPLEMENTATION OF 3APL IN MAUDE

3APL, which was first introduced by Hindriks [13], is a cognitive agent programming language. This means that it has explicit constructs for representing the high-level mental attitudes of an agent. The version of 3APL that we have implemented in Maude comes closest to the one presented in [25]. It is single-agent and builds on propositional logic. In this version, a 3APL agent has beliefs, a plan, and goals. Beliefs represent the current state of the world and information internal to the agent. Goals represent the desired state of the world, and plans are the means to achieve the goals. Further, a 3APL agent has rules for selecting a plan to achieve a certain goal given a certain belief, and it has rules for revising its plan during execution. Together, these elements may form the state or configuration of a 3APL

¹The full version of this paper has been submitted elsewhere.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...\$5.00.

agent. In order to be able to refer to an agent's beliefs and goals, 3APL incorporates logical languages (belief and goal query languages) with corresponding semantics. The formal operational semantics of 3APL is defined by means of a transition system [21].

"Maude is a formal declarative programming language based on the mathematical theory of rewriting logic [18]. Maude and rewriting logic were both developed by José Meseguer. Maude is a state-of-the-art formal method in the fields of algebraic specification [27] and modeling of concurrent systems. The Maude language specifies rewriting logic theories. Data types are defined algebraically by equations and the dynamic behavior of a system is defined by rewrite rules which describe how a part of the state can change in one step." [20]

The general idea of the implementation of 3APL in Maude, is that 3APL configurations are represented as terms in Maude, and the transition rules of 3APL are mapped onto rewrite rules of Maude. This idea is taken from [26], in which, among others, implementations in Maude of the operational semantics of a simple functional language and an imperative language are discussed. Further, the logical languages of 3APL and their corresponding semantics can be represented in Maude as equational theories. Finally, the meta-level reasoning cycle or deliberation cycle of 3APL [13, 8, 24] can also be modelled very naturally in Maude, since rewriting logic is *reflective* [7]. "Informally, a reflective logic is a logic in which important aspects of its metatheory can be represented at the object level in a consistent way, so that the object level representation correctly simulates the relevant metatheoretic aspects." [6, Chapter 10]

3. ADVANTAGES OF MAUDE

Based on our experience with the implementation of 3APL in Maude, we argue that Maude is well suited as a prototyping and analysis tool for logic-based cognitive agent programming languages. We observe that cognitive agent programming languages such as 3APL have a logical *as well as* a semantic component: the logical part consists of the belief and goal query languages, and the semantic part consists of the transition system. Since Maude supports both the logical and the semantic component, the implementation of languages like 3APL in Maude is very natural, and the integration of the two components is seamless.

We observe that the direct mapping of transition rules of 3APL into rewrite rules of Maude ensures a *faithful* implementation of the operational semantics of 3APL in Maude. This direct mapping is a big advantage compared with the implementation of a 3APL interpreter in a general purpose language such as Java, in which the implementation is less direct. In particular, in Java one needs to program a mechanism for applying the specified transition rules in appropriate ways, whereas in the case of Maude the term rewriting engine takes care of this. As another approach of implementing a cognitive agent programming language in Java, one might consider to implement the plans of the agent as methods in Java, which is for example done in the Jadex framework [22]. Since Java does not have support for revision of programs, implementing 3APL plans as methods in Java is not possible. We refer to [24] for a theoretical treatment of the issues with respect to semantics of plan revision.

A faithful implementation of 3APL's semantics in Maude

is very important with regard to our main original motivation for this work, i.e., to use the Maude LTL model checker to do formal verification for 3APL. The natural and transparent way in which the operational semantics of 3APL can be mapped to Maude, is a big advantage compared with the use of, e.g., the PROMELA language [14] in combination with the SPIN model checker [15].

SPIN is a generic verification system which supports the design and verification of asynchronous process systems. SPIN verification models are focused on proving the correctness of process interactions, and they attempt to abstract as much as possible from internal sequential computations. The language PROMELA is a high level language for specifying abstractions of distributed systems which can be used by SPIN, and it's main data structure is the message channel. In [3], an implementation of the cognitive agent programming language AgentSpeak(F) - the finite state version of AgentSpeak(L) [4] - in PROMELA is described, for usage with SPIN. Most of the effort is devoted to translating AgentSpeak(F) into the PROMELA data structures. It is shown how to translate the data structures of AgentSpeak(F) into PROMELA channels. It is however not shown that this translation is correct, i.e., that the obtained PROMELA program correctly simulates the AgentSpeak(F) semantics. In contrast with the correctness of the implementation of 3APL in Maude, the correctness of the AgentSpeak(F) implementation in PROMELA is not obvious, because of the big gap between AgentSpeak(F) data structures and semantics, and PROMELA data structures and semantics. In [12], it is shown that the performance of the Maude model checker is comparable with that of SPIN.

A further important advantage of Maude is that deliberation cycles can be programmed very naturally as strategies, using reflection. A related advantage is that a clear separation of the object-level and meta-level semantics can be maintained in Maude. A 3APL program can be executed without making use of a deliberation cycle, while it can equally easily be executed *with* a deliberation cycle.

While we have implemented a simplified version of 3APL in Maude, we argue that the features of Maude are very well suited to support the implementation of various extensions of this version of 3APL. Implementing these extensions is left for future research.

In particular, an extension of a single-agent to a *multi-agent* version will be naturally implementable, since, from a computational point of view, rewriting logic is intrinsically concurrent [17]. It was in fact the search for a general concurrency model that would help unify the heterogeneity of existing models, that provided the original impetus for the first investigations on rewriting logic [18].

Further, a more practically useful implementation will have to be *first-order*, rather than propositional. Although the implementation of a first-order version will be more involved, it can essentially be implemented in the same way as the current version, i.e., by mapping transition rules to rewrite rules. In [9], the transition rules for a first-order version of 3APL are presented. Configurations in this setting have an extra substitution component, which records the assignment of values to variables. An implementation of this version in Maude will involve extending the notion of a configuration with such a substitution component, as specified in the cited paper.

Finally, we aim to extend the *logical part* in various ways.

Regarding this propositional version, one could think of extending the belief base and goal base to arbitrary sets of propositional formulas, rather than just sets of atoms as we have used in the current implementation. Also, the belief and goal query languages could be extended to query arbitrary propositional formulas. The satisfaction relations for queries could then be implemented using, e.g., tableau methods as suggested in [17]. Further, when considering a first-order version of 3APL, the belief base can be implemented as a set of Horn clauses, or even as a Prolog program. Implementing the belief base as a Prolog program is possible in the current Java implementation of 3APL. How to define standard Prolog in rewriting logic has been described in [16]. Finally, we aim to experiment with the implementation of more sophisticated specifications of the goals and their accompanying satisfaction relations, as proposed in [23].

4. REFERENCES

- [1] J. V. Baalen, J. L. Caldwell, and S. Mishra. Specifying and checking fault-tolerant agent-based protocols using Maude. In *FAABS '00: Proc. of the First Int. Workshop on Formal Approaches to Agent-Based Systems-Revised Papers*, volume 1871 of *LNCS*, London, UK, 2001. Springer-Verlag.
- [2] R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni. *Multi-Agent Programming: Languages, Platforms and Applications*. Springer, Berlin, 2005.
- [3] R. H. Bordini, M. Fisher, C. Pardavila, and M. Wooldridge. Model checking AgentSpeak. In *Proc. of AAMAS'03*, 2003.
- [4] R. H. Bordini and A. F. Moreira. Proving the asymmetry thesis principles for a BDI agent-oriented programming language. *Electronic Notes in Theoretical Computer Science*, 70(5), 2002.
- [5] N. Boudiaf, F. Mokhati, M. Badri, and L. Badri. Specifying DIMA multi-agent models using Maude. In *Proc. of PRIMA'04*, volume 3371 of *LNCS*. Springer, 2005.
- [6] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. Maude manual (version 2.1.1). 2005.
- [7] M. Clavel and J. Meseguer. Reflection and strategies in rewriting logic. *Electronic Notes in Theoretical Computer Science*, 4:125–147, 1996.
- [8] M. Dastani, F. S. de Boer, F. Dignum, and J.-J. Ch. Meyer. Programming agent deliberation – an approach illustrated using the 3APL language. In *Proc. of AAMAS'03*, 2003.
- [9] M. Dastani, M. B. van Riemsdijk, F. Dignum, and J.-J. Ch. Meyer. A programming language for cognitive agents: goal directed 3APL. In *Programming multiagent systems, first int. workshop (ProMAS'03)*, volume 3067 of *LNAI*. Springer, Berlin, 2004.
- [10] M. Dastani, M. B. van Riemsdijk, and J.-J. Ch. Meyer. Programming multi-agent systems in 3APL. In R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors, *Multi-Agent Programming: Languages, Platforms and Applications*. Springer, Berlin, 2005.
- [11] F. Durán, S. Eker, P. Lincoln, and J. Meseguer. Principles of mobile Maude. In *Proc. of the Second Int. Symposium on Agent Systems and Applications and Fourth Int. Symposium on Mobile Agents*, volume 1882 of *LNCS*, 2000. Springer-Verlag.
- [12] S. Eker, J. Meseguer, and A. Sridharanarayanan. The Maude LTL model checker. In F. Gaducci and U. Montanari, editors, *Proc. of the 4th Int. Workshop on Rewriting Logic and Its Applications (WRLA 2002)*, volume 71 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 2002.
- [13] K. V. Hindriks, F. S. de Boer, W. van der Hoek, and J.-J. Ch. Meyer. Agent programming in 3APL. *Int. J. of AAMAS*, 2(4):357–401, 1999.
- [14] G. Holzmann. *Design and Validation of Computer Protocols*. Prentice Hall, New Jersey, 1991.
- [15] G. Holzmann. The model checker SPIN. *IEEE Trans. Software Engineering*, 23(5):279–295, 1997.
- [16] M. Kulas and C. Beierle. Defining standard Prolog in rewriting logic. In *Electronic Notes in Theoretical Computer Science*, volume 36. Elsevier Science Publishers, 2000.
- [17] N. Martí-Oliet and J. Meseguer. Rewriting logic as a logical and semantic framework. In J. Meseguer, editor, *Electronic Notes in Theoretical Computer Science*, volume 4. Elsevier Science Publishers, 2000.
- [18] J. Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theoretical Computer Science*, 96:73–155, 1992.
- [19] J. Meseguer. Membership algebra as a logical framework for equational specification. In *Proc. of the 12th Int. Workshop on Recent Trends in Algebraic Development Techniques*, 1997. Springer-Verlag.
- [20] P. C. Ölveczky. Formal modeling and analysis of distributed systems in Maude. Lecture Notes, 2005.
- [21] G. D. Plotkin. A Structural Approach to Operational Semantics. Technical Report DAIMI FN-19, University of Aarhus, 1981.
- [22] A. Pokahr, L. Braubach, and W. Lamersdorf. Jadex: a BDI reasoning engine. In R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors, *Multi-Agent Programming: Languages, Platforms and Applications*. Springer, Berlin, 2005.
- [23] M. B. van Riemsdijk, M. Dastani, and J.-J. Ch. Meyer. Semantics of declarative goals in agent programming. In *Proc. of AAMAS'05*, 2005.
- [24] M. B. van Riemsdijk, J.-J. Ch. Meyer, and F. S. de Boer. Semantics of plan revision in intelligent agents. In C. Rattray, S. Maharaj, and C. Shankland, editors, *Theoretical Computer Science*, pages 240–257. 2006. Special issue of Algebraic Methodology and Software Technology (AMAST'04).
- [25] M. B. van Riemsdijk, W. van der Hoek, and J.-J. Ch. Meyer. Agent programming in Dribble: from beliefs to goals using plans. In *Proc. of AAMAS'03*, 2003.
- [26] A. Verdejo and N. Martí-Oliet. Executable structural operational semantics in Maude. Technical report, Universidad Complutense de Madrid, Madrid, 2003.
- [27] M. Wirsing. Algebraic specification. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics. Elsevier, Amsterdam, 1990.