Adding Knowledge Updates to 3APL

Vivek Nigam* and João Leite

CENTRIA, New University of Lisbon,Portugal
vivek.nigam@gmail.com, jleite@di.fct.unl.pt

Abstract. 3APL is a widely known multi-agent programming language. However, when to be used in certain domains and environments, 3APL has some limitations related to its simplistic update operator that only allows for updates to the extensional part of the belief base and its lack of a language with both default and strong negation to enable the representation and reasoning about knowledge with the open and closed world assumptions. In this paper, we propose to address these issues by modifying the belief base of 3APL to be represented by Dynamic Logic Programming, an extension of Answer-Set Programming that allows for the representation of knowledge that changes with time.

1 Introduction

In the past few years, several agent architectures and agent programming languages have been proposed. Among them we can find, for example, 3APL [9,12], FLUX [22], IMPACT [10], DALI [8], JASON [5] and Minerva [14,18]. For a survey on some of these systems, as well as others, see [6,7,20].

In this paper, we take a closer look at 3APL, one of the existing systems that has recently received an increasing amount of attention, and propose some enhancements to its language and semantics.

3APL is a logic based programming language for implementing cognitive agents that follows the classical BDI architecture where agents have beliefs (B), intentions (I) and desires (D) to guide their actions. The semantics of 3APL agents is defined by a *transition system* composed of *transition rules*. The use of 3APL provides the agent programmer with a very *intuitive* and simple way to define agents. The programmer can declaratively specify the *beliefs* (represented by Horn Clauses) and *goals* (represented by conjunctions of atoms) of agents, how they *build plans* to achieve such goals, and reason with their beliefs. Furthermore, communication between agents can be done in an elegant way by modifying the beliefs of agents, allowing for the possibility of reasoning with the transferred messages. Despite all these interesting properties, 3APL, when to be used in certain domains and environments, has some limitations that serve as our motivation to propose the modifications presented in this paper. These limitations, in our opinion, are:

* Supported by the Alβan Program, the European Union Programme of High Level Scholarships for Latin America, no. E04M040321BR.

R.H. Bordini et al. (Eds.): ProMAS 2006, LNAI 4411, pp. 165–181, 2007.

1. Limited belief updates - The mechanism used by 3APL to update agent's beliefs is quite limited. Such updates in 3APL amount to the simple addition and removal of facts in the agent's belief base. It is not difficult to find a situation where this type of belief update is insufficient. Consider an agent with a belief base containing the rule $believe(santa\_claus) \leftarrow mother\_said(santa\_claus)$, and the fact $mother\_said(santa\_claus)$. This agent can be seen as a child agent that believes in everything its mother says, in this case it believes in *santa claus*, because its mother said so ($mother\_said(santa\_claus)$). Furthermore, consider that the agent evolves and discovers that in fact, *santa claus* doesn't exist, even though its mother said so. Since 3APL only allows for updates to the extensional part of the belief base (i.e. its set of facts), it is not possible to achieve the desired semantics, where $believe(santa\_claus)$ is false and $mother\_said(santa\_claus)$ is true, by the mere addition and retraction of facts. Note that it is not possible to remove the fact $believe(santa\_claus)$ because there is none to be removed, and if the fact $mother\_said(santa\_claus)$ is removed it would be change the belief base in an undesired way. To obtain the desired effect, updates in the intensional part of the knowledge base (i.e. its set of rules) are required;

2. Limited expressive power of negative information - 3APL allows for the use of one form of negation, namely *negation by finite failure*. It has been shown that the use of default negation (*not*) provides good expressive power to a language. Furthermore, the use of both default and strong negations ($\neg$), concurrently, such as in Answer-Set Programming [11], allows for easy ways to reason with both the *closed* and *open world assumptions*. For example, in the classical car-train cross, where the car should pass the cross if its sure that the train is not coming, it is necessary to reason with the open world assumption, where strong negation plays a key role ($\neg train$). On the other hand, to represent a cautious agent that would move if it believes that a place is not safe ($not\ safe$), the use of default negation is more adequate;

In this paper, we will use Dynamic Logic Programming (DLP) [19,2,14], an extension of Answer Set Programming, to address these limitations stated above. We propose to represent the 3APL agent's *belief base* as a Dynamic Logic Program.

According to the paradigm of *DLP*, knowledge is given by a series of theories, encoded as generalized logic programs[1], each representing distinct states of the world. Different states, sequentially ordered, can represent different time periods, thus allowing DLP to represent knowledge that undergoes successive updates. Since individual theories may comprise mutually contradictory as well as overlapping information, the role of *DLP* is to employ the mutual relationships among different states to determine the declarative semantics for the combined theory comprised of all individual theories at each state. Intuitively, one can add, at the end of the sequence, newer rules (arising from new or reacquired knowledge) leaving to *DLP* the task of ensuring that these rules are in force, and that previous ones are valid (by inertia) only so far as possible, i.e. that

[1] Logic programs with default and strong negation both in the body and head of rules.