Implementing Cognitive Agents in 3APL

M. Dastani F. de Boer F. Dignum W. van der Hoek
M. Kroese J. J. Meyer

Institute of Information and Computing Sciences
Utrecht University

Abstract

3APL is a programming language for implementing cognitive agents. We demonstrate a Java implementation of the 3APL interpreter developed at the Institute of Information and Computing Sciences at Utrecht University. The presented work is a part of a NWO research project. More information on this project can be found at `http://www.cs.uu.nl/3apl/`. The demonstration, which takes less than 30 minutes, consists of executing several 3APL programs. The aim of this demonstration is to explain various aspects of the 3APL such as the architecture of the programmable cognitive agents, the 3APL programming constructs, the working of the interpreter, the expressiveness of the 3APL programming language, and the ease with which cognitive agents can be programmed.

1 3APL Architecture

The programmable software agent that we demonstrate is a cognitive agent that has a mental state consisting of beliefs, basic actions, goals, a set of so-called practical reasoning rules for revising goals, and an interpreter. The beliefs represent the robot's general world knowledge as well as its knowledge about the surrounding environment. The basic actions that the agent can perform may be sensory, physical, and mental actions. The practical reasoning rules can be applied to revise actions that are blocked, goals that are not achievable, to optimize goals, or to generate reactive behavior. Finally, the interpreter determines the flow of control among the abovementioned ingredients. This cognitive architecture, which we call the 3APL architecture, is illustrated in Figure 1.

2 Implementation of the 3APL Interpreter

The abstract specification language for 3APL agents is already presented in [1]. In order to implement a 3APL agent, a set of programming constructs is proposed for each of the 3APL modules. An interpreter is implemented in Java to execute programmed 3APL agents. The implemented interpreter consists of a deliberation cycle through which practical reasoning rules and goals are selected and

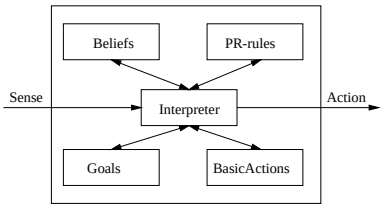

Figure 1: *The 3APL architecture.*

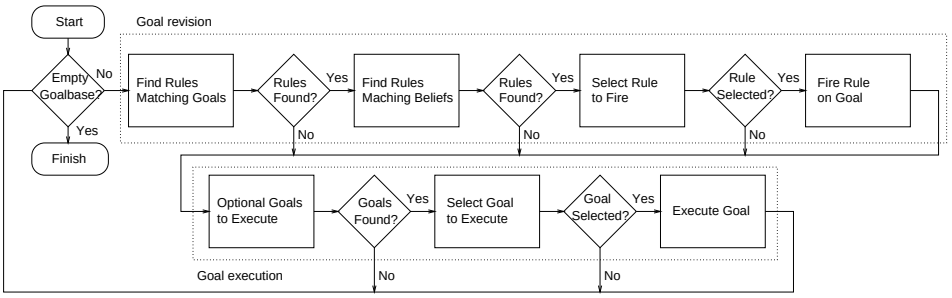

Figure 2: *The deliberation cycle of the 3APL interpreter.*

subsequently applied and executed. This deliberation cycle, which is illustrated in Figure 2, consists of two phases: a goal revision and a goal execution phase. In the goal revision phase (the upper part of the deliberation cycle in Figure 2), a subset of practical reasoning rules is selected on the basis of their applicability to goals and beliefs. Then, from this subset a practical reasoning rule is chosen and applied to the goal to which it is applicable. This application revises one of the robot's goals. In the goal execution phase (the lower part of the deliberation cycle in Figure 2), a subset of goals that can be executed is selected and from this subset a goal is chosen and executed.

In particular, we have defined a JAVA class which we call 3APL class. This class has private attributes for each of the 3APL modules such as belief base, goal base, basic actions and practical reasoning rules. This class has at least two methods, one method for creation of the 3APL object through which the a programmer can set various 3APL modules, and one method for running the object. The latter method corresponds to the deliberation cycle of 3APL.

References

[1] Koen V. Hindriks, Frank S. De Boer, Wiebe Van der Hoek, and John-Jules Ch. Meyer. Agent programming in 3apl. *Autonomous Agents and Multi-Agent Systems*, 2(4):357–401, 1999.