Chapter 8

A Verification Logic for GOAL Agents

K.V. Hindriks

Abstract Although there has been a growing body of literature on verification of agents programs, it has been difficult to design a verification logic for agent programs that fully characterizes such programs and to connect agent programs to agent theory. The challenge is to define an agent programming language that defines a computational framework but also allows for a logical characterization useful for verification. The agent programming language GOAL has been originally designed to connect agent programming to agent theory and we present additional results here that GOAL agents can be fully represented by a logical theory. GOAL agents can thus be said to execute the corresponding logical theory.

K.V. Hindriks
Delft University of Technology, The Netherlands e-mail: k.v.hindriks@tudelft.nl

8.1 Introduction

As technology for developing agent systems is becoming more mature, the availability of techniques for verifying such systems also becomes more important. Such techniques do not only complement tools for debugging agent systems but may also be used to supplement the techniques available for debugging agents. For example, model checking techniques may be used to find *counter examples* that show that an agent system does not satisfy a particular property. A counter example produces a run of a system that violates a property and as such indicates what is wrong with an agent. Program model checking discussed in [66] is an approach that supports this type of verification. It involves the construction of a semantic model $M$ that correctly represents an agent’s execution and that can be used to check satisfaction of a property $\varphi$, i.e. $M \models \varphi$. The key problem that needs to be solved to be able to use model checking for verification concerns the *efficient* construction of (part of) a model $M$ from a given agent system that is sufficient for verifying this system.

Model checking is one approach to verifying agents. An alternative approach to verifying agents involves the use of *deduction*. This approach assumes that a *logical theory* of an agent is available. The task of verifying that an agent satisfies a particular property $\varphi$ amounts to deducing $\varphi$ from the given theory $T$, i.e. $T \vdash \varphi$. The key problem that needs to be solved to be able to use deduction as a verification tool concerns the construction of a corresponding *logical theory* $T$ from a given agent system. It is the goal of this chapter to introduce such a theory for the Goal agent programming language [221].

Verification techniques based on deduction have been widespread in Computer Science and have been provided for a broad range of programming languages. The programming constructs and the structure of programs in a programming language often naturally give rise to an associated *programming logic*. This has been particularly true for imperative programming languages but also for concurrent programming languages [23, 296, 298].

The verification approach presented here for Goal consists of two parts. First, an operational semantics that provides a model for executing agent programs is defined. This provides a computational framework that specifies how Goal agents are to be executed. Second, a logic for verification is introduced and it is shown that the logical semantics corresponds with the operational semantics. It is our aim in this chapter to stay as close as possible to the actual implementation of the Goal language, although we do abstract away a number of features that are present in the interpreter for the language and focus on single agents. In particular, we have provided a semantics for logic programs as part of the operational semantics to model the Prolog engine that is used in the implementation. Similarly, we have aimed for a verification logic which semantics corresponds in a precise sense with the operational semantics and can be used to fully characterize agent programs. As we will show, basic Goal agent programs discussed here may be mapped into a corresponding logical theory by means of a straightforward translation scheme that