Formal Modelling of Reactive Agents as an Aggregation of simple Behaviours

Petros Kefalas

Dept. of Computer Science, CITY Liberal Studies,
Affiliated Institution of the University of Sheffield
13 Tsimiski Street, Thessaloniki 546 24, Greece
kefalas@city.academic.gr

Abstract. Agents, as highly dynamic systems, are concerned with three essential factors: (i) a set of appropriate environmental stimuli, (ii) a set of internal states, and (iii) a set of rules that relates the previous two and determines what the agent state will change to if a particular stimulus arrives while the agent is in a particular state. Although agent-oriented software engineering aims to manage the inherent complexity of software systems, there is still no evidence to suggest that any proposed methodology leads towards correct systems. In the last few decades, there has been a strong debate on whether formal methods can achieve this goal. In this paper, we show how a formal method, namely X-machines, can deal successfully with agent modelling. The X-machine possesses all those characteristics that can lead towards the development of correct systems. X-machines are capable of modelling both the changes that appear in an agent's internal state as well as the structure of its internal data. In addition, communicating X-machines can model agents that are viewed as an aggregation of different behaviours. The approach is practical and disciplined in the sense that the designer can separately model the individual behaviours of an agent and then describe the way in which these communicate. The effectiveness of the approach is demonstrated through an example of a situated, behaviour-based agent.

1 Introduction

An agent is an encapsulated computer system that is situated in some environment and that is capable of flexible, autonomous action in that environment in order to meet its design objectives [1]. Agents, as highly dynamic systems, are concerned with three essential factors: (i) a set of appropriate environmental stimuli or inputs, (ii) a set of internal states of the agent, and (iii) a set of rules that relate the two above and determines what the agent state will change to if a particular stimulus arrives while the agent is in a particular state.

Although *agent-oriented software engineering* aims to manage the inherent complexity of software systems [2], there is still no evidence to suggest that any methodology proposed so far leads towards correct systems. In the last few decades, there has been a strong debate on whether *formal methods* can achieve this goal. Academics and practitioners adopted extreme positions either for or against formal