A Note on Modeling Agent Systems
by Graph Transformation

P. Knirsch and H.-J. Kreowski

Fachbereich 3 Mathematik - Informatik
Universität Bremen
Postfach 33 04 40
D-28334 Bremen, Germany
{knirsch,kreo}@informatik.uni-bremen.de

Abstract. Specifying and programming with the help of agent systems is gaining more and more interest, especially in the field of distributed and reactive systems. In this paper, we propose a formal model of agent systems based on graph transformation. It is quite natural to visualize systems, especially system states, by means of graphs. Hence, it is also appropriate to specify those systems using graphs. Accordingly, changes from one state into another can be modeled by graph transformation.

1 Introduction

Agent systems are an upcoming framework in software engineering. They originally stem from the research field of artificial intelligence. Marvin Minsky stated in his innovative book “The society of mind” (cf. [Min85]) that the human mind is built up of entities (neurons), each capable of a limited number of functionalities. He assumes that from their interaction and cooperation intelligence emerges. When first introducing agent systems, researchers tried to imitate and utilize this effect. Later on, it appeared that agent systems have lots of others and even more significant benefits: they are flexible and have a semantically well defined and therefore efficient communication. Additionally it seems to be a well suited paradigm for designing distributed, open, and reactive systems. Especially this becomes increasingly important concerning software products acting in global networks, like the internet. As networks, i.e., systems, are suitably represented by means of graphical representations, we will go on and specify agents and agent systems by graphs and the actions they perform by modifying those graphs systematically by means of graph transformation.

The paper is organized as follows: next we are going to introduce concepts of agent systems in Section 2 and of graph transformation in Section 3, which presents formalisms needed in the following. After that, in Section 4, we give a first notion of what an agent system is, this being defined formally later on. Graphs have the benefit that their semantics is the graph itself. Hence, an easy semantics of a graph based agent system is intuitive to define. This is done in Section 5. Afterwards we examine a larger example that demonstrates the usefulness of our approach as well as of agent systems in general. Finally, we draw some conclusions and identify work to be done in the future.
2 Agent Systems

Without getting too involved with the theory of agent systems, we can state that it is a system consisting of agents, i.e., computational entities that interact and communicate in a certain environment. For a far more elaborated introduction into this theory see [Bra97], [W394]. Agents allow modular descriptions of very complex, dynamic, and distributed systems. They are closely related to objects but concentrate more on communication than on typing. A main feature of these agent systems is the idea of delegation: An agent is delegated to fulfill a task, which it pursues subsequently.

In this context several attributes that describe its kind of behavior can be assigned to the agent:

- **autonomous**: once an agent gets its tasks, it is not controlled by other entities,
- **reactive**: agents react to changes in their environment,
- **proactive**: an agent is proactive, if it is able to initiate interactions with other agents on its own,
- **cooperative**: agents work together to reach a common aim,
- **rational**: agents behave rationally, i.e., they perform only actions they consider to be optimal in the respective situation,
- **intelligent**: a fuzzy attribute meaning that an agent is capable of doing smart things like learning, reasoning, resolving conflicts, etc.

Based on reactivity which is a main characteristic, each of these agents is capable of sensing the environment and reacting appropriately, always pursuing its specific goal. The changes made by agents can be recognized by other agents. This observable agents’ external behavior corresponds to a functional abstraction within the whole system because the agents’ internal functionalities can be far more complex. Hence, with the help of agent systems very complex systems can be modeled modularly. If the agents’ goals are chosen accordingly, problems can be solved concurrently. In this case we speak of cooperative agents. In some existing applications, especially in the field of electronic commerce, agents behave competitively. Electronic markets, supply chains, and automated negotiations are only a few examples of scenarios in which agent systems gain an increasing importance, because those systems are characterized by high dynamics within their component structure, by complex interaction between the participants, and they are naturally distributed. These systems’ characteristics are investigated in [Bra97] and [RM97]. When designing or investigating those systems, graphical notations are often used to facilitate the understanding of their structure and partially of their behavior. Therefore, it seems very suitable to use graphs as a means to specify the system’s structure and graph transformation for the system evolving.

Up to now, many attempts in specifying agent systems have been made. They often use specification languages like CSP, Z, or modal logics in order to specify the essential agent components. Yet, there are not many attempts towards formal frameworks that cover all aspects of agenthood and make it possible to give a simple notion of what an agent system really is, i.e., what its syntax and its