Using the $\pi$-Calculus to Model Multiagent Systems *

Albert C. Esterline and Toinette Rorie**

Department of Computer Science/NASA ACE
North Carolina A&T State University
Greensboro, NC 27411
esterlin@ncat.edu and rorie@lucent.com

Abstract. We present a formal framework that uses the $\pi$-calculus for modeling multiagent systems. A process algebra in general is a term algebra used as an abstract programming language that stresses the composition of processes by a small set of process operators. The $\pi$-calculus in particular allows one to express systems of processes that have changing communication structure. We explicate the agent abstraction as a $\pi$-calculus process that persists through communication actions. Our principal task here is to show how the $\pi$-calculus can be used to model certain aspects that have already been specified for a major multiagent system. We also sketch how a $\pi$-calculus framework supports development activities in this context, and we suggest how various general aspects of multiagent systems may be modeled in this framework.

1 Introduction

Wooldridge [20] defines an agent as a computer system capable of autonomous action. An intelligent agent in particular is capable of flexible autonomous action, where flexibility involves reactivity (the ability to perceive its environment and to respond to changes in it), pro-activeness (aiming at goals by taking the initiative), and social ability (since agents must cooperate to achieve goals). Autonomy distinguishes agents from objects since the decision whether to execute an action lies with the object invoking the method in an abject system but with the agent receiving the request in an agent system. This paper focuses on multiagent systems, which ”provide an infrastructure specifying communication and interaction protocols” and are typically open (with no ”centralized designer”) [7]. Abstractly, the open nature of these environments entails that smaller systems (agents) can be composed to form larger systems (multiagent systems), where composition involves coordinated concurrent activity. Coordination is also critical and, in the case of self-interested agents, a major challenge.

* This research was supported by grant NAG 5-4102, ”Formal Foundations of Agents,” from NASA Goddard Space Flight Center. Special thanks are due to the Goddard personnel, especially Walt Truszkowski.
** Current address: Lucent Technologies, 200 Lucent Lane, Cary, NC 27511

© Springer-Verlag Berlin Heidelberg 2001
In modeling multiagent systems, we use the *agent abstraction*: the notion of an autonomous, flexible computational entity in an open environment that specifies communication protocols. This notion provides higher-level abstractions for complex systems, leading to simpler development techniques that alleviate the complexity of multiagent systems [17]. A *model* is (partially) expressed by a specification when our goal is system development. A (modeling) *framework* is a notation along with directions on using it to express (and hence to construct) models. This paper suggests some aspects of a formal framework for modeling such systems (without actually developing the framework) and, in particular, formally explicates the agent abstraction. Formal modeling naturally lags quick-and-dirty approaches to system design, but once a field has been defined in practice, formal modeling offers rigor, eliminates ambiguity, and allows one to abstract away inessential details.

Among concurrency formalisms, the most abstract are (temporal and modal) logics, used to describe or specify the communication behavior of processes. More concrete are process algebras, which are term algebras used as abstract concurrent programming languages that stress the composition of processes by a small set of process operators. The most concrete are concurrency automata (paradigmatically Petri nets, but also, for example, Statecharts), which describe processes as concurrent and interacting machines with all the details of their operational behavior. We have used modal logics to formalize the transaction abstraction for multiagent systems [9,18], high-level Petri nets to model agent behavior [2], and Statecharts to express multiagent plans [21]. These formalisms do not directly represent agents as complex entities, and they are not compositional. The possible exception is Statecharts, which have been given compositional, process-algebraic semantics; Statecharts have the advantages and disadvantages of a visual formalism, and there is some promise in using them to supplement process algebras.

There are several reasons to consider an agent as a process (as understood in concurrency theory, i.e., an entity designed for a possibly continuous interaction with its users) and to model multiagent systems with a process-algebraic framework. Compositionality allows process terms for agents to be combined simply to form a term for a multiagent system. Communication protocols required of a multiagent environment can be formulated straightforwardly in process algebras, which are often used for formalizing protocols. The basic action in a process algebra is communication across an interface with a *handshake*. Two processes performing a handshake must be prepared at the time to perform complementary actions (usually thought of as output and input on a given channel), hence a handshake synchronizes the processes. Handshakes are remarkably like speech acts in face-to-face communication. The symmetry of a handshake distributes control between the participating processes hence respects their autonomy. Since their are handshakes with the environment, the reactive nature of agents can be accommodated. A process-algebraic framework does not so directly capture aspects of the agent abstraction that are not directly related to communication. The pro-active aspect involves (internal) planning, and negotiation