Chapter 1

Multiagent Systems

Multiagent systems are distributed computing systems. Like all distributed systems, they are composed of a number of interacting computational entities. However, unlike classical distributed systems they, and their constituent entities, are intelligent. I shall adopt as my operational definition of intelligent systems one that is inspired by the works of McCarthy [1979], Newell [1982, p. 115], and Dennett [1987, pp. 13–35]. This definition sidesteps most slippery philosophical debate concerning the nature of intelligence. In essence, it states that intelligent systems are those for which it is helpful to take the *intentional stance*, to use McCarthy and Dennett's term, or those that can be said to have a distinct *knowledge level*, to use Newell's term.

In other words, a system is intelligent if you need, for intuitive or scientific purposes, to attribute cognitive concepts such as intentions and beliefs to it in order to characterize, understand, analyze, or predict its behavior. Thus one might say of an automatic teller machine that it "knows who I am," "does not want to give me more money than my account balance," "cannot give more than 100 dollars," "does not know how to approve an overdraft," and so on. Further such examples are discussed in the works cited above. The term *intentional* as used in this manner pertains not only to intentions, but also to other mental terms such as knowledge, beliefs, and desires.

Although some philosophers may dispute the validity of the above definition, and indeed it pretends to no philosophical sophistication, it proves particularly useful from the standpoint of computer science. It tells us that our intuitive conception of intelligence yields some nice abstractions for dealing with systems that behave intelligently. I shall return to this point later in this chapter, but suffice it to say here that this monograph is an exercise in providing a rigorous foundation to some of the abstractions that result from seriously taking the intentional stance towards computational systems.

The trend towards the development of increasingly intelligent systems is matched only by the trend towards the distribution of computing. The science of multiagent systems lies at the intersection of these trends. Multiagent systems are of great significance in a number of current and future applications of computer science. For example, they arise in systems for electronic data interchange, air traffic control, manufacturing automation, computer supported cooperative work, and electronic banking, as well as in robotics and heterogeneous information systems.

Besides the well-known reasons for the usefulness of distributed systems in general, continued progress in the study of multiagent systems is attractive for the following additional reasons. Advances in this study will permit intelligent systems to be developed independently of each other and to be reused as components of new systems. These components can be thought of as member *agents* in new multiagent systems. This modularization is also useful when designing systems for applications, such as medical diagnosis, in which expertise is naturally distributed over agents who specialize in different domains. A system designed as a multiagent system can also be more robust than otherwise, since the acquisition and validation of design requirements is simpler for such a system. Moreover, such a system can be simpler to design for many applications, including manufacturing planning and air-traffic control, by allowing an intelligent agent to be located at the site where the data are available and where the necessary decisions have to be taken.

Multiagent systems are thus of great practical importance in computer science. Unfortunately, no general framework is available at present that we may use to analyze, specify, design, or implement multiagent systems. In the next section, I briefly describe what the main components of the desired framework might be. These components are the high-level abstractions for multiagent systems that we must define and formalize.

Multiagent systems have usually been studied as a part of artificial intelligence (AI). This has been largely because of the experimental nature of most such systems. Also, their claims to intelligence often rest in languages and approaches, such as Lisp, rule-based expert-system shells, and blackboard architectures, which are traditionally associated with AI. It is hoped that the framework developed here will permit the expansion of multiagent systems into novel domains, partially by abstracting out the details of implementation and partially by developing a semantics that is closely related to the semantics for classical systems. Doing so would facilitate the implementation of multiagent systems on standard architectures and platforms.

1.1 Intentions, Know-How, and Communications

The term *agent* is widely used in computer science. It is variously applied to actors, to instantiated expert-system shells, to processes in general, to finite-state machines that monitor such processes, to physical robots, and to intelligent entities that react to their environment. The definition I adopt captures the most basic connotations of the term *agent.* According to this definition, agents are intelligent systems, towards which we need to take the intentional stance. In other words, agents are the basic units of intelligence that we consider. The examples from the literature listed above are all interesting realizations of agents as construed here.

The intentional stance makes available such abstractions as the intentions and know-how of agents, and the communications that take place among them. These turn out to be important scientific abstractions for multiagent systems. These abstractions no doubt have much conceptual appeal. Furthermore, there are simple pragmatic and technical reasons for considering them seriously. They

- are natural to humans, who are not only the designers and analyzers of multiagent systems, but also the end users and requirements specifiers;
- provide succinct descriptions of, and help understand and explain, the behavior of complex systems;
- make available certain regularities and patterns of action that are independent of the exact physical implementation of the agents in the system; and
- may be used by the agents themselves in reasoning about each other.

Consequently, these abstractions can be profitably applied in systems that may have unknown or evolving implementations. Their utility grows as we consider increasingly complex systems. The intentional stance gives us a way of proceeding with what we, and our agents, know or can find out about a given system. In this way, it addresses the issue of the partiality of the information we have about complex systems.

For any concept to be effectively used in science and engineering, it must have a rigorous foundation in theory. In particular, for the above abstractions to be useful in the science of multiagent systems, they must be

given an objective grounding in terms of the architectures that different kinds of systems have, and the actions they perform.

I consider multiagent systems from without, i.e., from the perspective of a designer or analyzer. I do *not* directly take the point of view of the different agents who compose the system. I adopt an external perspective from which to attribute beliefs and intentions to agents, and describe their communications. Thus, issues of how they might actually be represented in the agents need not be considered here. This leaves the exact design of the agents an open issue to be settled later in the design process, provided certain minimal requirements are met. These requirements would be stated in terms of intentions, beliefs, and know-how.

The intentional stance is closely related to what Newell has called the *knowledge level.* Indeed, both Dennett and Newell agree that they agree: for instance, see [Dennett, 1987, p. 256] and [Newell, 1982, pp. 122–123]. The main differences are that Dennett defines the intentional stance as the choice of an observer, whereas Newell defines the knowledge level as a distinct level of computer architecture. In fact, even Dennett allows that the intentional stance may present objective patterns of behavior that are not visible otherwise (p. 25). Also, Dennett applies the stance to all systems, e.g., apple trees, not just computational systems (p. 22). But these distinctions are not relevant for our purposes and I shall freely use the insights of Dennett, McCarthy, and Newell.

In fact, individual agents are often described in terms of abstractions, such as knowledge, intentions, and desires. However, the abstractions are usually chosen in an *ad hoc* manner and not formalized in a uniform framework to the detail necessary. Of these abstractions, I pick knowledge, intentions, and know-how and argue that they are the most useful for the purposes of designing and understanding multiagent systems.

However, even after we formalize intentions and know-how in multiagent systems, we would not have completely established the conceptual foundations necessary for a science of multiagent systems. This is because one important ingredient, namely, communication, would still be missing. A major bottleneck in the design of multiagent systems is the design of the protocols of interaction among their member agents. Unfortunately, while individual agents are usually described in terms of their knowledge, intentions, and know-how, extant approaches to understanding the interactions among them are not able to make full use of those abstractions. Even fairly recent research, which provides primitives for communication among agents, has tended to be concerned with the workings of the TCP/IP and similar protocols. It has not been possible to ignore aspects of communication roughly at or below the so-called

Transport Layer of the classical ISO/OSI standard. And, more to the point, current theories do not provide any kind of a formal semantics for the messages exchanged in a multiagent system.

This lack of a general theory of the interactions among agents forces the system designer to think in terms of what are, from the point of view of multiagent systems, merely details of the underlying architecture. These details are important, but are simply out of place at the level at which we wish to study such systems. The concomitant mixing up of concerns often makes the behavior of the designed system depend crucially on details of the operating system and the network hardware. At the same time, the behavior of the individual agents is based on the knowledge they have at different stages. Thus there is no principled way to relate the interactions *among* the agents to the available abstract descriptions of what is *within* each of them. The designer must design some acceptable modes of interaction in an *ad hoc* fashion and relate them as effectively as possible to the agents' states. Not only is this tedious task error-prone, it also has to be redone if the system is ever reimplemented. Further, the designer is accorded no assistance when systems implemented in different ways are to be integrated. In short, the extant technology suffers from the following limitations:

1. It requires that the interactions among agents be designed from scratch each time.

2. The semantics of these interactions is embedded in different procedures, some of which involve network and operating system code. This makes the nontrivial task of validating and modifying multiagent systems even more difficult.

3. Systems designed independently cannot be easily integrated.

4. It is virtually impossible to gracefully update or redesign a system: one cannot easily replace an existing agent with a new one.

Taken together, these limitations subvert many of the main original motivations for developing distributed intelligent systems. I seek to present a theory of the interaction among agents and a formal semantics for their interactions that will form the basis for a framework for designing multiagent systems.

Consider the following example of a simple, but in some ways quite typical, multiagent system. This system comprises three agents: two air-traffic controllers and a pilot. These agents may or may not be human, but one can initially think of them as if they were. Figure 1.1 shows an example execution

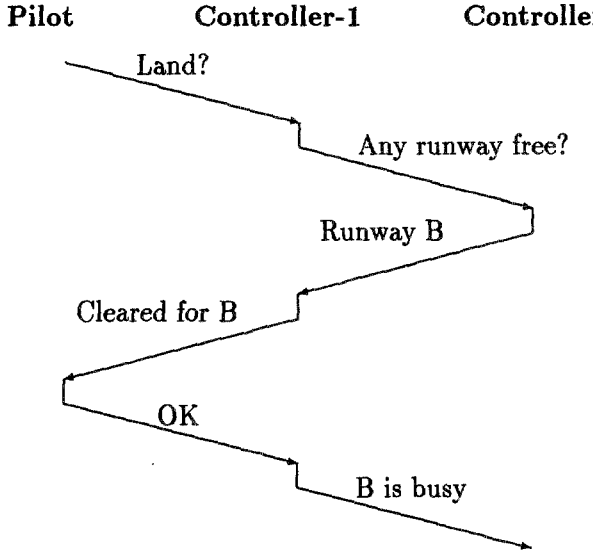

Figure 1.1: Example: Air-Traffic Control

of this system. It begins with the pilot requesting permission to land from one of the controllers. The controller does not know of any available lanes, so he asks the other controller if she knows of any vacant lanes. She informs him that Runway B is free. He offers the pilot permission to land there, if the pilot would agree to pay all applicable charges. The pilot accepts the offer and agrees to pay the charges. The first controller then prohibits the second controller from assigning Runway B to any other plane.

This brief example contains many of the interesting features of multiagent systems. The interactions among the different agents can be characterized by means of the messages they exchange. The messages fall into a handful of major logical categories: requests, queries, assertions, permissions, promises, and prohibitions. Of course, a single physical message may have the effect of two or more logical ones. Similarly, multiple physical messages may be required to achieve the effect of one logical message. However, details of the underlying communication mechanism are not our main concern here. The member agents of the system participate in different protocols for interaction: thus these protocols define the interfaces among the agents. The agents' internal structures can be modified as long as their participation in the relevant protocols is unaffected. Indeed, we would like to be able to reinstantiate the above multiagent system with different pilots and to reimplement the controllers, if we can preserve the correctness of their behavior. A clean specification of the interfaces

would allow us to design the component agents independently of one another and upgrade the system incrementally.

The requirement of correct behavior imposes some constraints on how the member agents of a multiagent system should act, and the intentions and know-how they must have, at different stages in the applicable protocols. In the above example, controllers must accede to pilots' requests when possible and must respect each others' prohibitions. A pilot should intend to use a runway if he requests one. Other, more detailed, conditions can also be stated.

In order to capture correctness requirements abstractly and to allow the agents to evolve, a framework for multiagent systems must include at least the following two components: (a) a semantics for abstractions such as intentions and know-how with which the individual agents can be specified, and (b) a semantics of communications that goes beyond message transmission and considers the contents of messages in terms of the participants' intentions, know-how, and actions. A framework that includes these components is developed here.

1.2 The State of the Art

In this section, I briefly survey the state of the art in the areas of computer science that touch upon the topics of interest here. Some of these issues are discussed in greater detail in appropriate chapters. In giving this survey, I hope to identify the parts of my work that have been anticipated by others and the parts that I believe are new.

I borrow many of the underlying motivations and intuitions from the field of distributed artificial intelligence, a small cross-section of which is reported in collected volumes, such as [Huhns, 1987], [Gasser & Huhns, 1989], and [Demazeau & Müller, 1991]. A number of multiagent systems have been implemented. Examples include the GRATE system applied in the ARCHON project for controlling electrical power grids [Jennings, 1992], the Distributed Vehicle Monitoring Testbed (DVMT) for distributed sensing [Durfee *et al.*, 1987], and the MINDS system for information retrieval [Huhns *et al.*, 1987]. These implementations are usually designed to demonstrate the feasibility of some proposed architectures and sometimes to solve particular problems in specific application domains. However, despite such limitations, this kind of work is important because it builds experience and expertise for more general and deployable implementations of multiagent systems.

More immediately relevant is the increasing body of work pertaining

to algorithms and mechanisms for cooperation and negotiation in multiagent systems. This work involves higher-level primitives such as the beliefs and intentions of agents and imposes various kinds of logical structures on the communications among agents. Notable work in this area includes [Bussman & Müller, 1993], [Burmeister *et al.*, 1993], [Chang, 1991], [Berthet *et al.*, 1992], and [Müller, 1993]. The present monograph can be thought as defining primitives that will capture the intuitions that emerge from the above works. This work now tends to take a moderately formal view of these mechanisms: it provides a formal language of one sort or another, but is not rigorous in terms of formal models and semantics like the present work.

Hence, whereas overall a large amount of good research has been conducted in distributed artificial intelligence, there are some limitations that it tends to suffer from. In particular, despite several implementation efforts, no principles for the systematic design of multiagent systems are available. The procedural characterizations that are usually given cannot easily be adapted to new applications. Further, typically, no formal theory of any kind is available that corresponds to these implementations. But these are precisely the issues addressed by the present work.

There has been considerable work in AI on planning. In the early days, this work was almost exclusively procedural and not quite formalized [Fikes & Nilsson, 1971; Sacerdoti, 1977]. Recent work is more rigorous and is based on formal theories of action. One of the most significant theoretical contributions to the study of actions in AI is the situation calculus, which was developed by McCarthy & Hayes [1969]. Several theories of action have been proposed in the last decade or so. However, many of these theories make assumptions that can prove quite restrictive in practice. These assumptions commonly include the following:

- only one event happens at a time, which entails that only one agent acts at a time;
- events have precisely determined effects; and
- events are necessarily associated with a state change.

Recently, Lifschitz and his coworkers have shown how the above assumptions, which are usually associated with models of the situation calculus, can be relaxed [Gelfond *et al.*, 1991]. However, this involves defining functions assigning real times to situations and real time durations to actions. Traditionally, this was not done. But, augmented in this manner, the situation calculus can be thought of as a possible metalanguage for the framework developed here.

However, since my main concern is to formalize some useful concepts, rather than to express everything in a minimal language, I shall be content with the framework described below.

Theories of action have been applied in planning. Not many of the abovementioned assumptions have yet been relaxed in this research, although events with context-dependent effects are now considered, which could not be handled by the original STRIPS approach [Pednault, 1988]. More recently, other researchers have also allowed multiple events to take place simultaneously [Allen, 1991]. Although the problem of generating plans is not addressed here, the development of a general framework of actions, time, know-how, and intentions would prove beneficial there. Indeed, classical temporal logic, which I seek to extend here, has been applied to planning [Lansky, 1989].

Besides AI, there is another body of work that I am indebted to. This is the work on logics of programs, both for sequential and concurrent systems. Two of the main strands of research in this area are on temporal logics [Emerson, 1990] and dynamic logics [Kozen & Tiurzyn, 1990]. Temporal logics have operators to deal with time, but do not consider actions explicitly. Dynamic logics provide a rich syntax for actions, but do not explicitly consider time. In the most popular variant, which is the one I use, actions are given a regular language syntax. I explicitly relate actions with time. The relationship is not complicated, but it needs to be exhibited so that we can proceed with a general logic of actions and time to use as a basis for further development.

Usually, the models considered for both temporal and dynamic logics are discrete, with only one action happening at a time. I consider nondiscrete models in my definitions. I also allow multiple actions and events to happen simultaneously and out of synchronization with each other. Even though the basic models may be nondiscrete, we can induce a discrete structure on them for the purposes of computing. The details of this are not explored here. However, having multiple actions has some ramifications on the definition of know-how, since it enables us to consider games in which the players do not take turns. Thus it enables us to consider more general cases of games than are usually considered. It is not clear that these cases can be captured in a strict interleaving framework.

Traditionally, theories of action in AI are designed for the reasoning of an intelligent agent. In other words, one expects to design an agent as a formal reasoner or theorem prover that operates by explicitly using the given theory. By contrast, theories of actions and time in classical computer science are meant to characterize models and computations from without. In other words, someone may characterize a distributed system using temporal or dynamic logic *only* to prove some results about the behavior of the system. Such results

would state that the given system is correct in some sense, for instance, that it does not violate any *safety* condition and exhibits *liveness* by never entering a state of deadlock. But, no matter what the theorems exactly are, they are always proved by the designer or analyzer, and not by the system itself. In some cases, two processes that interact, for instance by sharing some storage, might be unaware of each other. Yet theorems about the system they constitute may involve joint conditions on them.

Whether the designer's or the agent's perspective is taken has major technical ramifications on the nature of a semantical framework, such as the one presented here. If an agent is to prove theorems, then his knowledge and ignorance about the relevant conditions has great significance. Given the finiteness of agents' knowledge, it is virtually essential, in this case, to adopt some kind of a defeasible or nonmonotonic formalism. Unfortunately, nonmonotonic reasoning can often be quite intractable. On the other hand, if a designer has to prove theorems, then he has to ensure that all the relevant conditions have been included in the model. Reasoning can then proceed in a monotonic framework.

An intuitively appealing way to think of this dichotomy, due to Benjamin Kuipers [Kuipers, 1986; Crawford *et al.*, 1990], is the following. Although a designer explicitly constructs a model *before* reasoning, an agent engaging in nonmonotonic reasoning effectively constructs or instantiates a model *while* reasoning. Kuipers has argued that the task of building a model can be separated from the task of reasoning in it, even when agents have to do both of these tasks. I accept this view. What I seek to describe is a formal model and language that captures many of the required properties of actions and time. I also aim to use it to give formal definitions to a number of useful concepts. The definitions lead to a clear way of computing with these concepts. However, I do not address the problem of generating good models automatically. That I leave to future work. As a result, the framework proposed here is sufficiently developed only to apply from the perspective of a designer of multiagent systems.

Interestingly, the above remarks help relate this research to Rosenschein's efforts, who too takes the designer's point of view [Rosenschein, 1985]. His goal is to convert specifications of agents into their implementations in software or hardware. Thus, in a broad sense, our respective approaches are in agreement. However, he considers only the concept of knowledge, i.e., know-that, in his theory. Therefore, I believe that my approach is more sophisticated than his. Not only have I considered time and actions explicitly, I have also incorporated concepts such as intentions, know-how, and communications.

The concepts of intentions and knowledge have been used to great

advantage in user modeling and in attacking the problems of planning and understanding of speech acts and discourses. Following Grice, there is a long tradition in natural language processing (NLP) that to understand a sentence is to understand what its speaker meant by it [Grice, 1969]. It is customary to derive what is meant by an utterance from what a speaker may intend to achieve by making it. In fact, it is widely accepted that the basis for a speech act is the corresponding *communicative intention* of its speaker [Kuroda, 1989; Searle, 1969]. Communicative intentions are a special kind of the intentions studied here. Speech acts theory, which underlies much work in NLP, is based on the view that communication is a variety of action [Austin, 1962]. This observation motivates the application of planning techniques to communication [Appelt, 1986]. Theories of action apply similarly.

User modeling focuses on how a user interface may present information to, and process information from, a human being. Both these tasks depend greatly on what the interface system expects the user to know, intend, and know how to achieve. In this way, user modeling is required for effective natural language and other, e.g., graphical, interactions with human beings.

I propose a semantics for communications in Chapter 6. Although I consider communications in multiagent systems as speech acts, I do not focus on the natural language aspects of the problem. In other words, I do not provide a theory of what a given natural language utterance may be interpreted as. But I do relate speech acts to the actions, know-how, and intentions of agents. This connection, I submit, is necessary for a theory of communications to fit as an integral part of the larger theory of multiagent systems. My proposal on communication and traditional work in NLP and user modeling are complementary in one respect. The former is concerned with the *content* that different communications must have; the latter is concerned with the *form* they must take to accurately correspond to that content.

Several formal theories of knowledge and belief have been proposed in the literature. These theories are of great value to the study of multiagent systems. However, as I argue in Chapter 4, the conception of knowledge corresponding to *know-how* is by itself of great importance in the study of multiagent systems. Traditional theories of knowledge are usually about *know-that.* It is implicitly assumed that know-how poses no special challenges. A notable exception is Ryle [1949], whose views I discuss in Chapter 4. Even when this assumption is reasonable, and it is not always so, it has the effect of burying our intuitions about know-how inside the technical properties of know-that. We must tightly relate the abstractions we define for multiagent systems to the agents' possible actions. This relationship is captured more naturally when know-how is considered as an independent abstraction.

Some formal theories of intentions have also been proposed in the literature. The most well-known of these is due to Cohen & Levesque [1990]. Unfortunately, this theory is terribly complicated. Moreover, it has certain conceptual and technical shortcomings. At the conceptual level, this theory allows an agent to succeed with an intention merely through persistence. The agent simply has to be able to correctly identify the intended condition; he does not need to know how to achieve it. Clearly, this requirement is not sufficient: one can come up with several natural conditions that an agent may be able to identify, but would not be able to achieve. An example of a technical shortcoming is that the authors state certain properties as "easy to see," but it is possible to construct counterexamples to these properties in the theory itself. I have developed these arguments in greater detail elsewhere [Singh, 1992a].

Other theories of intentions include the ones of [Rao & Georgeff, 1991a] and [Singh & Asher, 1993]. The former is similar, in some respects, to the theory developed here. It is considered in detail in Chapter 3. The latter is based on Kamp's Discourse Representation Theory [Kamp, 1984] and seeks to be cognitively more accurate than the theory presented here. For example, that theory rightly invalidates the inference that an agent's intentions must be closed under logical equivalence. I discuss this inference too in Chapter 3.

Another related category of research pertains to intentions of multiagent systems. This involves considering sets or groups of agents as having joint intentions. In effect, it attempts to define and formalize the intentions of a multiagent system itself, rather than of its component agents. The relevant literature in this area includes [Grosz & Sidner, 1988], [Cohen & Levesque, 1988a], [Singh, 1991c], [Tuomela & Miller, 1988], [Tuomela, 1991], and [Jennings, 1992]. This research is important, but from a distributed computing perspective, success in it presupposes a good understanding of the component agents themselves. Work on the social aspects of agents will eventually prove essential. However, to be useful in computer science, these social aspects must be studied and formalized to the same technical standards as classical distributed computing. The present work seeks to give a rigorous treatment of the mental and communicative aspects of agents, the need for which is more pressing given the state of the art. However, this work will facilitate the development of formalizations of social concepts as well.

Recently, some work has been done on the design of communication protocols based on a notion of knowledge [Fischer & Immerman, 1986; Halpern & Moses, 1987]. However, the knowledge considered therein is of the process of communication itself. Thus, in these approaches, the delivery of messages is significant, while their content is ignored. By contrast, my aim is to emphasize and study the semantics of the messages exchanged, not the

process of exchanging them. Also, the classical work on knowledge is about protocols for lower-level data transmission, which are assumed as an available primitive here.

1.3 Major Contributions

The present work proposes a theory of intentions and know-how in a general framework of action and time. It uses these concepts to define a semantics of communications in multiagent systems. The major contributions include

- A rigorous technical framework that
  - Allows concurrent actions by multiple agents
  - Allows actions to be of varying durations
  - Allows nondiscrete models of time, which underlies actions
  - Carefully relates actions to the temporal aspects of the framework
  - Admits a notion of weak determinism in which different possible choices of agents can be simultaneously captured
  - Defines abstract actions or strategies such that they can coexist with basic or primitive actions
- A formalization of intentions and know-how, in which
  - Intentions are independent of beliefs and know-how
  - Know-How is independent of intentions
  - Constraints relating intentions, beliefs, and know-how can be stated
  - Constraints on how agents with certain intentions and know-how may act can be stated
  - Conclusions about what will transpire given the agents' intentions and know-how can be drawn
- A formalization of communications, which
  - Builds on top of speech act theory
  - Gives a semantics for communications based on their conditions of *whole-hearted* satisfaction, which in turn are determined by the content of the communication.

As discussed above, there has been much research on knowledge and belief and some on intentions. However, such concepts cannot be properly understood, except in the presence of know-how. For it is know-how that relates beliefs and intentions closely to actions, and it is actions that make ascriptions of belief and intentions nongratuitous. For example, a missionary who intends to cross a river may not be able to do so, even if he persists with his intention, and performs actions in attempts to achieve his intention. However, if he has the requisite know-how and applies it, he would succeed eventually, provided he keeps his intention to cross the river.

I propose a formalization of intentions and know-how in a general model of actions and time. This semantics captures many of the properties of intentions and know-how that are relevant from the standpoint of multiagent systems. Using this semantics, I also seek to provide a new semantics for the different modes of communication, such as promises and prohibitions. The proposed framework involves the programs that agents can, and do, execute. As a result, we can use intentions, know-how, and communications as more than just conceptual descriptions. The proposed semantics helps us compare implementations and guides the creation of design tools. It also helps assign meaning to different constraints on system behavior that are natural in a given domain.

Formal theories and formal semantics are useful not only because their properties can be precisely specified, but also because they can be used as a rigorous backdrop for various design rules. This holds even if, in one's chosen approach, these rules are applied only informally. However, the approach I prefer involves the use of a formal language and its semantics for specifying and verifying multiagent systems with mechanical tools. These tools, which include automatic theorem provers and model checkers, have been used to great advantage for classical systems [Boyer & Moore, 1979; Emerson & Clarke, 1982; Burch *et al.*, 1990]. Their development for the proposed framework would advance the state of the art in multiagent systems considerably. Formal theories can also be used to motivate and design concise and clean representations that agents may use in interacting with one another. The proposed abstractions are, in fact, often used informally. The absence of general formal theories is thus a major weakness in the science and engineering of multiagent systems.