Knowledge and Communication
(*A tutorial*)

Yoram Moses
The Weizmann Institute of Science
Rehovot, 76100 Israel

Abstract

The subtle interaction between knowledge, action, and communication is an important theme underlying much of the literature on reasoning about knowledge in a variety of disciplines. The purpose of this paper is to review work on the relationship between knowledge and communication in distributed systems. Communication is the basic means by which knowledge is obtained and transferred in a distributed system. As a result, properties of the communication medium play a central role in determining what states of knowledge can result from communication. Most of this paper is a review of theorems that illustrate this connection and a discussion of their implications. In the latter part of this paper, we consider *knowledge-oriented programming*, a novel approach to describing agents' behavior in a distributed system. Knowledge-oriented programs are an extension of the knowledge-based protocols of Halpern and Fagin, in which communication is abstracted away completely from the description of agents' behavior. In its stead, agents can perform high level actions that are defined in terms of changing the state of knowledge of other agents. Such actions are called *knowledge-oriented* actions. Communication then enters when we come to implement the knowledge-oriented actions in a given context. Knowledge-oriented programming makes explicit use of the fact that the role of communication is to change the state of knowledge of agents in the system.

1 Introduction

The subtle interaction and interdependence between knowledge, action, and communication is an important theme underlying much of the literature on reasoning about knowledge in a variety of disciplines. In some disciplines, much of the interaction and information transfer between agents (or players) is caused by the actions agents take. This is the case in many instances of negotiations, where in parallel to any explicit communication between the parties, actions taken in the field can signal intentions, threats, etc. The same happens often in economic settings, where actions such as price setting, bidding and the like have an important role in conveying information. Related phenomena can take place in the context of a distributed AI setting in which a set of agents, perhaps robots, are in a shared environment, attempting to coexist fruitfully and pursue their individual and perhaps also joint goals. In distributed systems, however, the vast majority of the interaction between agents takes place via explicit messages sent among them. This being the case, the properties of the communication medium affect the evolution of the system's state of knowledge in a direct manner.

This work was supported in part by grants from the Israel Academy of Sciences and from the Minerva Foundation.

The purpose of this paper is to review work on the relationship between knowledge and communication in distributed systems. The crucial role of communication in the study of knowledge in distributed systems stems from the fact that communication is the means by which knowledge is obtained and transferred in such systems. As a result, properties of the communication medium play a central role in determining what states of knowledge can result from communication. We present two complementing approaches to studying this relationship. We first review a number of theorems that relate communication and knowledge, and discuss their implications. We then turn to a brief discussion of *knowledge-oriented programming*, which is an extension of the notion of knowledge-based protocols due to Halpern and Fagin [HF85]. In a knowledge-oriented program, agents can perform high-level actions that are defined in terms of changing the state of knowledge of other agents. This allows for a new kind of high-level description for a variety of distributed protocols.

Presenting a detailed model for knowledge in distributed systems is beyond the scope of this tutorial. For a detailed exposition of such a model and the choices involved in its design see [HM90] and [FHMV92]. We now make a few comments about the model and the framework, which will serve the discussion later on. Roughly and briefly speaking, we identify a distributed system with a set of the possible runs, or histories, of the system. Formulas are considered true or false with respect to *points* in time, which correspond to a run r of the system and a time t during that run. At any given point, an agent is assumed to have a certain view of the world, or *local state*, which is a function of its history up to that point. Knowledge of agents is defined using a possible-worlds approach [Hin62,Kri63], where points play the role of the possible worlds, and an agents can distinguish between two points if it has different local states at these points. This definition yields a notion of knowledge that satisfies the properties of the modal system S5. We denote by $K_i\varphi$ the fact that agent i knows φ, and by $C\varphi$ the fact that φ is common knowledge. We refer to the individual "knowers" by the term *agents*, or *sites of the system.* While in most technical applications of the results given here these agents will be computers connected by some communication network, our discussion here should also make sense when thinking of the agents as people, players in a game, or robots. At this point we'd like to add a disclaimer: This presentation is a biased collection of results on the subject of knowledge and communication. It is not a survey of the work on knowledge in distributed systems, and there is a considerable body of this work we do not touch upon here. For a survey of work on knowledge in distributed systems, see [Hal87].

This paper is structured as follows. In Section 2 we consider the impact of the possibility of arbitrary communication failures loss on the ability to attain common knowledge and discuss the implications this has on the ability to implement various types of coordinated actions. Section 3 deals with message chain theorems, which capture the fact that certain knowledge in particular types of systems can only be obtained using explicit communication. Section 4 discusses the extent to which the knowing that communication is reliable can be used to obtain knowledge in the absence of explicit communication. In Section 5 we discuss the notion of knowledge-based protocols, and a novel extension to it called *knowledge-oriented programming.* Finally, Section 6 presents some closing remarks, and considers some related issues that arise in other disciplines, such as artificial intelligence, economics, and philosophy.

2 Knowledge and unreliable communication

One of the interesting aspects of the interaction between knowledge and communication is that the state of knowledge resulting from a communication act depends, sometimes in a critical way,

on the properties of the communication medium. As an example, let us consider the effects of communication failures on the agents' state of knowledge. To be concrete, we shall consider systems that display *arbitrary message loss* (AML), by which we mean that at any point it is possible for no further messages to be delivered, without anyone but the receiver of the last delivered message ever being able to know whether this message was delivered. Clearly, the communication medium in such a system does not guarantee a high degree of reliability. However, when designing communication protocols it is generally desirable to make as weak an assumption about the communication medium as possible. Moreover, we often need to handle communication failures and are sometimes unable to make considerably stronger assumptions. In systems satisfying AML, it turns out, it is impossible to attain common knowledge that any successful communication has taken place. More formally, let *delivered* represent the fact that "*at least one message has been delivered*". Recall that we use the formula $C\varphi$ to denote the fact that φ is common knowledge. By techniques of [HM90] we then have:

Proposition 2.1: *In a system that displays arbitrary message loss, if there are at least two agents, then* $\neg C(\textit{delivered})$ *is valid.*

The proof of Proposition 2.1 is by induction on the number d of messages delivered up to a given point in question. When $d = 0$, the formula *delivered* doesn't hold, and hence $\neg C(\textit{delivered})$ trivially holds. For $d = k+1$, the property AML implies that only the receiver R of the last delivered message knows it was delivered. Since there are at least two agents, there is an agent $j \neq R$ whose view is consistent with a world in which $d = k$. By the inductive assumption, $\neg C(\textit{delivered})$ holds in that world. It follows then that $\neg K_j C(\textit{delivered})$ holds in the current world. However, the formula $C\varphi \Rightarrow K_i C\varphi$ is valid for all agents i and formulas φ. Thus, $\neg K_j C(\textit{delivered})$ implies that $\neg C(\textit{delivered})$ holds and this completes the inductive proof.

To illustrate the implications of Proposition 2.1, let us consider the *coordinated attack* problem (which we will denote by CA). This problem is due to Gray in [Gra78]; it was first analyzed explicitly in terms of knowledge in [HM90].

> Two divisions of an army are camped on two hilltops overlooking a common valley. In the valley awaits the enemy. It is clear that if both divisions attack the enemy simultaneously they will win the battle, whereas if only one division attacks it will be defeated. The divisions do not initially have plans for launching an attack on the enemy, and the commanding general of the first division wishes to coordinate a simultaneous attack (at some time the next day). Neither general will decide to attack unless he is sure that the other will attack with him. The generals can communicate only by means of a messenger. Normally, it takes the messenger one hour to get from one encampment to the other. However, it is possible that he will get lost in the dark or, worse yet, be captured by the enemy. Fortunately, on this particular night, everything goes smoothly. How long will it take them to coordinate an attack?

We say that a protocol (or set of strategies, one for each general) *meets the specifications of* CA if it guarantees that under no circumstances will a general attack alone. It is possible to prove that if the generals follow a protocol that meets the specifications of CA, then they can never attack. In particular, even on a night in which all communication succeeds and everything goes smoothly, no attack can be coordinated. The deeper reasons for this can be cast in terms of knowledge and communication as follows. Let *attack* represent the fact that "*both generals are attacking*". The

description of the coordinated attack problem implies that the following formulas are valid for every protocol that meets the specifications of CA:

(a) $attack \Rightarrow delivered$; and

(b) $attack \Rightarrow C(attack)$.

Formula (a) states that no attack can take place in the absence of successful communication. This corresponds to the generals' not having a predetermined plan to attack. Formula (b), on the other hand, states that when the generals attack they have *common knowledge* that they are attacking. Proving this requires an appropriate model for representing a general's knowledge and how it evolves as the general receives messages from others. Roughly speaking, however, the proof of (b) depends only on the assumption that at any given point a general knows whether or not he is currently attacking. Formula (b) then follows from the fact that when the generals attack, he knows that they are both attacking, and that both know they are both attacking, etc. (for more details, see [HM90]). An immediate corollary of (a) and (b) is:

(c) $attack \Rightarrow C(delivered)$.

Thus, in order to attack, the generals must attain common knowledge of the fact that at least one message has been delivered. In the context of CA, we view the messengers as constituting the communication medium. The guarantees about the reliability of the communication medium in the coordinated attack problem are rather weak. On a bad night, any of the messengers fail to deliver a message. Hence, the resulting system displays AML. As an immediate corollary of formula (c) and Proposition 2.1 we thus obtain:

Corollary 2.2: *Generals that follow a protocol satisfying the specification of CA can* never *attack.*

The coordinated attack problem is an example of how the properties of a communication system can affect the quality and usefulness of the information it can convey. Notice that on the particular night being discussed in the above description of the coordinated attack problem, the messenger in fact reliably delivers messages between the two generals. On that night the generals happen to be communicating over a reliable channel, although that they do not know it is reliable. If, a priori, the generals had common knowledge that the messenger is going to succeed in delivering their messages that night, then they would have no problem coordinating the attack. The inability to coordinate an attack is not the result of communication failures that actually happen, but rather from the uncertainty regarding whether failures may happen.

More generally, Proposition 2.1 implies essentially that communication cannot be used to cause facts to become common knowledge in a system that displays arbitrary message loss. The reason for this is that common knowledge is a prerequisite for simultaneous coordination of practically any kind. Intuitively, in most instances of simultaneous coordination, the fact $action$ = "*the coordinated action is being performed*" is known to an agent when he or she performs this action. We thus obtain that $action \Rightarrow E(action)$ is valid, and hence that $action \Rightarrow C(action)$ is also valid. Proposition 2.1 implies the impossibility of using communication to coordinate any such acts of coordination in systems that display AML.

In the coordinated attack problem, the requirement that the attack be simultaneous was essential for deriving the validity of formulas (b) and (c). Coordinated actions are often not required to be performed simultaneously. For example, it makes sense to modify the specification of coordinated attack to require that if one general attacks, the other one will *eventually* attack. Such

eventual coordination is closely related to a variant of common knowledge, called *eventual common knowledge*. The definition of eventual common knowledge is in terms of a fixed point. Intuitively, a formula φ is eventual common knowledge, which we denote by $C^\diamond\varphi$, if the following holds:

(1) Everyone will eventually know φ and (1).

The original definition of common knowledge in [Lew69] was given in terms of a similar fixed point. For a more formal and technically complete definition of eventual common knowledge, see [HM90] and [FHMV92].

It is possible to show that eventual common knowledge is related to eventual coordination just as common knowledge is related to simultaneous coordination. In particular, one can show that for any protocol guaranteeing eventually coordinated attack, the following is valid:

(c′) $\quad attack \Rightarrow C^\diamond(delivered)$.

Moreover, a slightly more delicate proof than that of Proposition 2.1 shows:

Proposition 2.3: *In a system that displays arbitrary message loss, if there are at least two agents, then* $\neg C^\diamond(delivered)$ *is valid.*

Intuitively, Proposition 2.3 implies that in a system displaying AML, communication can not be used to coordinate eventually coordinated actions. In particular, as in the case of Corollary 2.2, the validity of (c′) together with Proposition 2.3 imply that even coordinating an eventually coordinated attack in a system displaying AML is impossible.

Given that designing useful protocols for systems displaying AML is so difficult, it is common practice to make stronger assumptions about the communication medium. One popular assumption is that while arbitrarily many messages may be lost, any message sent repeatedly will eventually be delivered. The communication medium is thus assumed to display a limited degree of *fairness* to the senders of messages. We thus say that systems satisfying this property display *fair message loss* (FML). One natural interpretation of the FML property is that all communication failures are transient, as opposed to the possibility of permanent failures in the case of AML. It is not hard to show that the analogue of Proposition 2.3 does not hold for the case of systems displaying fair message loss. Indeed, in systems displaying FML it is possible to use communication to achieve guaranteed coordination of various activities. In practice, most communication protocols are designed for systems displaying FML.

One desirable property to require from a distributed protocol is that after the events for which the protocol was invoked are done, all of the participants will reach a *terminated* state. By this we mean that they will know they need not wait for any further interaction regarding this invocation of the protocol. A protocol is said to be *terminating* if it guarantees that every participant eventually reaches a terminated state. An interesting result of Koo and Toueg shows that, while it is possible to design useful protocols for systems displaying FML, such protocols cannot be terminating. Namely, they must have runs in which some of the participants do not terminate. More formally, let *eventually_delivered* represent the fact that "*at least one message either has been or will eventually be delivered*". Notice that $delivered \Rightarrow eventually\_delivered$. Let us call a protocol *nontrivial* if its specification requires that at least one message be delivered in each run of the protocol. Practically every useful protocol is nontrivial in this sense. It is not hard to show that in a terminating nontrivial protocol, when an agent terminates it knows $C^\diamond eventually\_delivered$. However, Koo and Toueg's theorem in [KT88] implies the following:

Proposition 2.4: *For any terminating protocol involving two agents or more in a system that displays fair message loss,* $\neg C^\diamond(eventually\_delivered)$ *is valid.*

3 Message chain theorems

It is fairly intuitive for an agent to gain knowledge about other agents by communicating with them. While this is usually the case, receiving information in the form of explicit messages is not always the only way in which an agent can learn about the state of others. For example, suppose a reliable agent i promises to send another agent j a message by a certain time t unless an event e of interest takes place. Moreover, messages take at most ϵ time units to be delivered. Then j can know that e occurred if it receives no message from i by time $t + \epsilon$. We discuss this type of knowledge gain in Section 4. One case where it is possible to demonstrate that the only way an agent can gain knowledge about the state of others is by explicit communication, is in the context of *asynchronous systems*. These are systems in which messages can take arbitrarily long to be delivered, and agents do not have local clocks. Moreover, each of the different agents operates at an arbitrary pace, independent of the others'. An agent's state of knowledge in an asynchronous system can change only as a result of its either receiving a message, sending a message, or performing an internal action. For a definition of such systems, see [CM86].

As we shall soon see, in asynchronous systems knowledge is propagated among the different sites only by means of message chains, which we now formally define. Given two agents i and j, we say that $\langle i, j \rangle$ is a *message chain* in the time interval (t_1, t_2) if $i \neq j$, $t_1 < t_2$ and either

1. Some message m is sent by i at or after time t_1 and is received by j before time t_2, or
2. there exist a third agent i' and time t' such that $\langle i, i' \rangle$ is a message chain in (t_1, t') and $\langle i', j \rangle$ is a message chain in (t', t_2).

Thus, roughly speaking, $\langle i, j \rangle$ is a message chain in (t_1, t_2) if there is a finite sequence of messages sent and received in that interval, starting with a message sent by i, and ending with a message received by j. The intuition behind the notion of a message chain is that $\langle i, j \rangle$ is a message chain if some message sent by i could have affected the contents of a message received by j. A sequence $\langle i_1, i_2, \ldots, i_\ell \rangle$ with $\ell > 2$ is said to be a message chain in the interval (t, t') if there are times $t_1, t_2, \ldots, t_\ell$ such that $t = t_1$, $t_\ell = t'$, and $\langle i_j, i_{j+1} \rangle$ is a message chain in the interval (t_j, t_{j+1}) for $j = 1 \ldots \ell - 1$. Notice that if $\langle i_1, i_2, \ldots, i_\ell \rangle$ is a message chain, then it does not contain consecutive repetitions (i.e., , $i_j \neq i_{j+1}$ for all $j < \ell$). It may, however, contain nonconsecutive repetitions. The following theorem captures the fact that the only way agents in an asynchronous system gain knowledge about other agents is via message chains:

Theorem 3.1: *In a run of an asynchronous system, if $\neg K_{i_\ell} \varphi$ holds at time t, and $K_{i_1} \cdots K_{i_\ell} \varphi$ holds at a later time $t + d$, then $\langle i_\ell, \ldots, i_1 \rangle$ is a message chain in the interval $(t, t + d)$.*

This theorem is a slightly simplified version of the *Knowledge Gain* theorem of Chandy and Misra in [CM86]. Such theorems were termed *message chain theorems* by Mazer in [Maz90]. Intuitively, the reason that learning about i_ℓ's coming to know φ requires an explicit message chain is that as far as the rest of the world is concerned, i_ℓ may not receive information regarding φ for an arbitrarily long time, regardless of the messages sent to i_ℓ or of any other events taking place at the other sites. Only explicit evidence that this state of knowledge has changed, where the evidence requires an actual message chain originating at i_ℓ, can cause another agent to learn that $K_{i_\ell} \varphi$ holds. Theorem 3.1 is useful for proving lower bounds on the numbers of messages required to carry out certain tasks in an asynchronous environment. We remark that Theorem 3.1 implies that, in asynchronous systems, a fact that is not common knowledge cannot later become common

knowledge. However, in contrast to the case of systems displaying AML (Proposition 2.3), in an asynchronous system in which communication is reliable, $C^\diamond(\textit{delivered})$ is easily attainable.

Theorem 3.1 provides a very strong statement regarding the need for message chains to attain knowledge in asynchronous systems. In other types of systems it is unlikely that such a strong statement can be obtained. For example, consider a system where every site has a local clock, and these clocks all show the same time and advance at the same rate. In such a setting an agent can learn quite a bit about other agents just by observing its own clock. In particular, using an appropriate model of the agents' knowledge, the current time is common knowledge. Moreover, an agent that knows that another will perform a certain action at a given time, may know that the action has been performed once this time has passed. We thus obtain various types of knowledge about others without the explicit need for message chains. Nevertheless, if we appropriately restrict the formulas φ in the statement of the theorem, message chain theorems can be proven for a variety of systems. To illustrate this point, let us consider a particular scenario. Imagine that an agent j has a local variable x_j in its memory, and that x_j can start out with either a value of 0 or a value of 1, independent of the rest of the system. Moreover, suppose that the communication medium displays arbitrary message loss (AML). Let ψ represent the fact that $x_j = 1$. If $j \notin \{i_1, \ldots, i_\ell\}$, then it can be shown that if $K_{i_1} \cdots K_{i_\ell}\psi$ holds at any point in time t, then $\langle j, i_\ell, \ldots, i_1\rangle$ is a message chain in the interval $(0, t)$. Notice that this message chain is needed even if all sites have identical clocks as described above, and even if they all operate at the same pace. Proofs of message chain theorems with respect to particular relevant classes of formulas are given for a number of contexts by Hadzilacos and by Mazer in [Had87,Maz90].

Proving message chain theorems in a given context is useful for a number of reasons. First, they provide insight into the way knowledge can evolve, and suggest guidelines for the design of protocols to solve various problems in that context. In addition, notice that by definition a message chain of length ℓ in an interval (t, t') requires at least $\ell - 1$ messages to be sent and received in this interval. As a result, message chain theorems yield certain lower bounds on the communication necessary for the solution of various problems. Finally, as we mentioned for the case of asynchronous systems, a message chain theorem, even with respect to a subclass of formulas φ, implies restrictions on what facts can become common knowledge in the course of an interaction between agents.

4 Reliable communication and timeouts

Message chain theorems apply in contexts where information can be transmitted only by explicit communication. As mentioned in Section 3, when communication is reliable and message transmission times are bounded, it is often possible to transmit information by refraining to send a message. We now consider the extent to which results in the spirit of message chain theorems can be obtained in systems with reliable communication.

Consider the following problem, which is a variation on a problem due to Mike Fischer [Fis90]: Again, an agent j has a local variable x_j, and x_j can start out with either a value of 0 or a value of 1, independent of the rest of the system. This time, however, message transmission is reliable and all messages are guaranteed to be delivered within, say, ϵ time units. Let us consider the problem of designing a protocol in this setting that will guarantee that every agent in the system will come to know the value of x_j. We call this the *binary broadcast problem.* One simple solution to this problem is to have j send each one of the other agents a message reporting what the value of x_j is. If the total number of agents is n, this requires at least $n - 1$ messages to be sent and received in every run of the protocol. However, if all of the sites have accurate local clocks and everyone

starts executing the protocol at time 0, the reliability of communication makes it possible to design more efficient protocols. For example, the protocol could have j send a message to everyone in case $x_j = 1$, and send no message if $x_j = 0$. (Notice that no messages whatsoever are sent by this protocol if $x_j = 0$.) It is easy to verify that this protocol guarantees that if j can send messages directly to every other agent, then everyone will know the value of x_j at time ϵ in either case. Let n denote the number of agents in the system. It is possible to show

Lemma 4.1: *Let P be an n-agent protocol for the binary broadcast problem, and let r_0 (resp. r_1) be a run of P in which initially $x_j = 0$ (resp. $x_j = 1$). Then the sum of messages sent in both r_0 and r_1 is at least $n - 1$.*

Lemma 4.1 holds regardless of the topology of the communication network. It applies for example also if the agents are organized in a ring and each can communicate directly only with its neighbors.

We use the term *knowledge gain by timeout* to refer to the gain of knowledge as a result of not receiving a message by a certain time. The binary broadcast example above shows that there are cases where using timeouts can save up to half of the messages needed to solve a problem. One wonders whether timeouts can ever save more than half of the messages needed to solve a problem. Intuitively, there seems to be a good reason why the answer should be NO, although a precise statement and proof have not been provided. Some more insight into knowledge gain by timeouts can be obtained by considering the generalization of the binary broadcast problem to a k-ary broadcast problem: Here the variable x_j may start out with any of the values $0, \ldots, k-1$. Let $S = \{r_0, \ldots, r_{k-1}\}$ be a set of k runs of a protocol for k-ary broadcast, where initially $x_j = i$ in run r_i for $0 \leq i \leq k-1$. It is not hard to show that the sum of messages sent in any such set S must be at least $(k-1)(n-1)$. It follows that timeouts can save only a fraction of $\frac{1}{k}$ over the trivial solution in k-ary boradcast. We remark that while the trivial protocol makes use of k different types of messages in different runs, it is possible to design communication-efficient protocols for k-ary broadcast that use as few as one type of (explicit) message. There is a tradeoff, however, between the following three parameters: (i) The number of messages used by the protocol; (ii) the number of message types used; and (iii) the time required by the protocol to attain its goal.

In systems where knowledge gain by timeout is possible it seems natural to think of not sending a message, as a form of implicitly sending a *silent* message. It is then possible to extend the notion of message chains by adding silent messages, and may be possible to prove extended-message-chain theorems. To the best of our knowledge, this approach has not been pursued, and is open for future work. Notice that timeouts require using some type of explicit or implicit local clock. The way in which knowledge evolves in a system with such clocks depends not only on communication, but also on the known properties of the clocks and their relative behavior. We now turn to consider an example illustrating how the question of whether clocks are synchronized in a system where communication is reliable and message transmission times are bounded can affect the problem of attaining common knowledge. This example will also hint at some of the subtleties involved in trying to come up with extended-message-chain theorems in systems with reliable communication. The example is taken from [FHMV92].

Assume that two agents, Alice and Bob, communicate over a channel in which (it is common knowledge that) message delivery is guaranteed. Moreover, suppose that there is only slight uncertainty concerning message delivery times. It is commonly known that any message sent from Alice to Bob reaches Bob either immediately or after exactly ϵ time units. Finally, Alice and Bob use the same (global) clock, which advances at the rate of real time. Suppose that Alice sends Bob the

following message m':

"This message is being sent at time t; m."

When Bob receives m' he knows that m' was sent at time t. Moreover, Bob's receipt of m' is guaranteed to happen no later than time $t + \epsilon$. Since Alice and Bob use the same clock, it is common knowledge at time $t + \epsilon$ that the time is $t + \epsilon$. It is also common knowledge that any message sent at time t is received by time $t + \epsilon$. We conclude that the fact that Alice sent m' to Bob is common knowledge at time $t + \epsilon$.

Now suppose that at some point Alice sends Bob a message m that does not specify the sending time in any way. Let t_S and t_D be variables denoting the time m is sent and the time it is delivered respectively. Thus, having sent m, Alice knows the value of t_S, while once m is delivered, Bob knows the value of t_D. Given our assumption about message delivery, it is common knowledge in the system that $(t_D = t_S + \epsilon) \vee (t_D = t_S)$. Since the events $t_D = t_S + \epsilon$ and $t_D = t_S$ are mutually exclusive, they partition the runs consistent with this example into two classes, depicted by scenario (a) and scenario (b) of Figure 1.

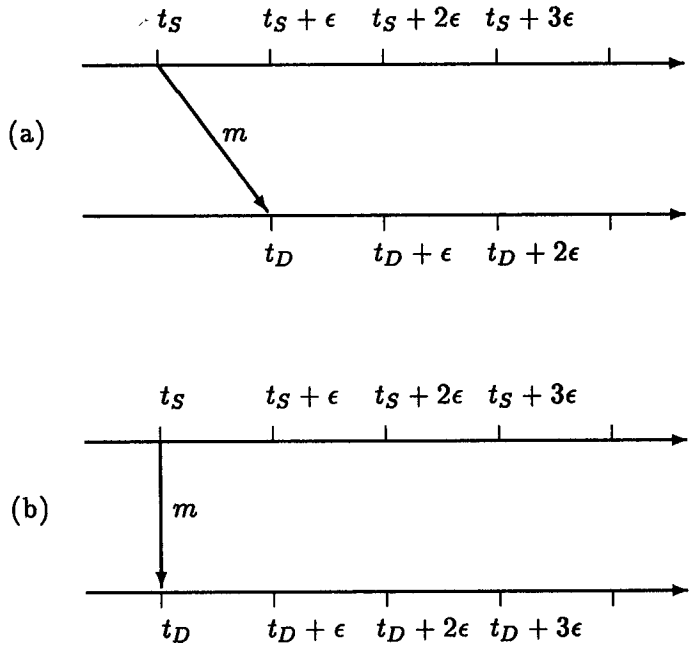

Figure 1: The two basic scenarios

Notice that $t_D = t_S + \epsilon$ in scenario (a), while $t_D = t_S$ in scenario (b). Let *sent* be the fact "*the message m has been sent*". Bob does not know *sent* initially. How does the state of knowledge of *sent* by the group consisting of Alice and Bob change with time?

At time t_D, Bob receives m and hence knows *sent*. However, because it may take ϵ time units for m to be delivered, as in scenario (a), Alice cannot be sure that Bob knows *sent* before $t_S + \epsilon$. Thus, $K_A K_B sent$ holds at time $t_S + \epsilon$ and does not hold beforehand. Bob, on the other hand, knows that Alice will not know that Bob knows *sent* before $t_S + \epsilon$. Since, for all Bob knows, m may have been delivered immediately (scenario (b)), Bob does not know that $t_S + \epsilon$ has taken place before time $t_D + \epsilon$. Bob therefore does not know $K_A K_B sent$ before time $t_D + \epsilon$. Alice, again, must consider the possibility of scenario (a) being the actual state of affairs, and hence must wait

until $t_S + 2\epsilon$ before she knows that $t_D + \epsilon$ has taken place. Thus, $K_A K_B K_A K_B sent$ holds at time $t_S + 2\epsilon$ but no earlier. This line of reasoning can be continued indefinitely, and an easy proof by induction shows that before time $t_S + k\epsilon$, the formula $(K_A K_B)^k sent$ does not hold, while at $t_S + k\epsilon$ it does. Thus, every ϵ time units Alice and Bob acquire an additional level of "Alice knows that Bob knows", and each such level actually requires these ϵ units of time. However, since $C(sent)$ implies $(K_A K_B)^k sent$ for every k, it follows that if $\epsilon > 0$ then $C(sent)$ is never attained. This may not seem too striking when we consider values of ϵ that are relatively large, say a day or an hour. However, the argument is independent of the magnitude of ϵ, and remains true even for small values of ϵ. Thus, even if Alice and Bob are guaranteed that the message m either arrives instantaneously or within one nanosecond, they still never attain common knowledge that m was sent!

As this example shows, uncertainty about clocks can play an important role in determining the states of knowledge that are attained in a system where communication is reliable and message transmission times are bounded. In fact, it can be shown that common knowledge is closely related to the ability to perform simultaneous actions. When simultaneity is not attainable, neither is common knowledge. For a proof of this and a discussion of its implications see [HM90]. This example also shows that an agent may be able to obtain a significant amount of knowledge from the passage of time in a context with reliable communication, even without needing to timeout on a message it is expecting.

5 Knowledge-oriented programming

The results presented in the previous sections show the crucial role communication plays in the design of protocols in distributed systems. However, communication is never an explicit goal of agents in a distributed system. Agents act locally, and need to think globally. They often need to know things about the global state of the system in order to ensure their local actions are suitably coordinated with what is going on at the other sites of the system. Thus, for example, an automated teller machine (ATM) on Main Street should not dispense money to a customer whose credit at the local branch shows a big deficit. Communication is the means used in order to inform the ATM whether the customers' account allows dispensing the money, as well as to update the branch about the money being dispensed. In this section we shall describe a framework for the design of distributed protocols called *knowledge-oriented programming*[1] that abstracts communication away completely. This exposition is somewhat preliminary, since the ideas reported on here are part of work in progress [Mos92].

Before presenting knowledge-oriented programming, let us briefly review the notion of *knowledge-based protocols*, due to Halpern and Fagin [HF85]. In standard protocols, an agent's actions at any given point are a function of the agent's local state at that point. Therefore, describing a standard protocol often requires going into rather low-level details of the agents' local states. The intuition behind the design of the protocol is sometimes hard to obtain from the formal description of the protocol. Moreover, protocols that implement the same idea in slightly different models can end up looking quite different. The idea behind knowledge-based protocols is that an agent's choice of which actions to perform can be thought of as being a function of its knowledge. Thus, the behavior of the ATM at a critical point in its interaction with the customer can be described by the following rule:

[1] The name was suggested by Moshe Vardi.

```
IF K(customer has sufficient funds)
THEN dispense money and send message to branch
ELSE display refusal message on screen
```

The idea, then, is that we need not be concerned about the details of the agent's local state. Describing an agent's behavior as a function of its knowledge yields a more intuitive result. Moreover, knowledge-based protocols can be used to translate a protocol from one context to another: Find out what the knowledge-based protocol underlying the given standard protocol is, and implement this knowledge-based protocol in the new context. In fact, in [HZ87], Halpern and Zuck show that a number of protocols designed to solve the same problem in different contexts are all implementations of the same knowledge-based protocol. Knowledge-based protocols have also been used in the design of efficient protocols for various problems [DM90,HMW90,MT88,NB]. The formal definition of a knowledge-based protocol is as a function from local states and systems to actions. The role of the system here is to determine what every agent knows in each local state. An important feature of knowledge-based protocols is that once the system is given, the knowledge-based protocol reduces to a standard protocol.

In standard models of distributed systems, an agent can perform two types of actions: local actions and communication actions. The local actions may affect its immediate environment (e.g., the ATM dispensing money to the customer) or they can modify its internal state (e.g., changing the value of an internal variable). Communication actions consist simply of sending messages to other agents. (The delivery of messages is controlled by the communication medium, and reading a message once it has arrived can be considered a local action). In knowledge-based protocols, the way an agent decides what actions to perform is described based on the agent's knowledge, while the actions performed are the same as the actions performed in standard protocols. The idea behind knowledge-oriented programming is to extend knowledge-based protocols so that agents no longer perform communication actions. Instead, they can perform high level actions that are defined explicitly in terms of changing the state of knowledge of other agents. We call such actions *knowledge-oriented actions.* Communication comes in only when we come to *implement* the knowledge-oriented actions.

An example of a class of knowledge-oriented actions are the so-called *notify* actions. The meaning of agent i performing the action $\texttt{notify}(j, \varphi)$ is that i must ensure that j will know φ (i.e., that $K_j\varphi$ will hold). It is important to notice that there are many ways in which this action may be implemented. Of course, it can be implemented by i sending j an explicit message about φ. However, an optimized implementation might allow i to do nothing if it already knows that j knows φ, or once it knows that j will know φ by other means. Moreover, in different contexts the notify action needs to be implemented in different ways. For example, in a system with reliable communication, sending a single message may suffice, while in a system in which communication displays fair message loss messages may need to be sent repeatedly until an acknowledgment is received. In a system displaying AML, for example, a notify action (which is guaranteed to succeed) cannot be implemented. There are a number of subtleties involved in providing a precise formal definition of notify. For example, when does an agent complete the notify action? (Notice that if i performs $\texttt{notify}(j, \varphi)$, we don't require that $K_iK_j\varphi$ ever hold.) Also, in some contexts it is possible to guarantee that the implementation of the notify action involve only actions performed by the notifier, while in others this is impossible. These and other considerations make it desirable to define a class of notify actions, each of which provides slightly different guarantees. In any given context, the subset of the notify actions that are implementable will be determined by the properties of communication, clocks, and the types of failures that can occur. For a discussion of

different variants of notify actions, their definitions and their use, see [Kis90].

Many distributed protocols can be given simple and fairly intuitive descriptions in terms of knowledge-oriented programs whose only high-level action is notify. Their general structure consists of repeatedly performing blocks of the following form:

```
Wait until Kφ (for an appropriate φ)
Perform local actions (the choice of actions is knowledge based)
Perform notify actions (the choice is again knowledge based)
```

In addition to variants of the notify action, one could consider more sophisticated knowledge-oriented actions. For example, an action that would cause a formula φ to become common knowledge would be quite useful for describing certain distributed protocols. We can similarly define knowledge-oriented actions whose purpose is to bring about various other states of knowledge. We believe that some of these actions are quite useful in describing protocols. Initial experience with knowledge-oriented programs shows that they often allow fairly straightforward proofs of correctness of protocols. These proofs break into two independent parts: One involves proving that a given knowledge-oriented program solves the problem of interest, and the other involves proving that implementations of the knowledge-oriented actions used in this program are correct. The independence of these two parts provides modularity that saves a considerable amount of effort. In a precise sense, knowledge-oriented programming makes explicit the idea that communication is merely the means by which the state of knowledge of the system is modified. The extent to which knowledge-oriented programs provide a superior abstraction of distributed protocols is for the future to prove. We feel that they present a natural abstraction and believe they will serve a useful role in the study of distributed systems.

6 Conclusions

In this paper, we considered two approaches to the relationship between knowledge and communication in a distributed system. In the first, we reviewed results on how the properties of the communication medium affect what states of knowledge are attainable in the system, and how knowledge can be obtained. The results reviewed provide considerable insight into what problems can be solved in various contexts, and what structure a protocol solving these problems should have. The second approach we considered suggested that we abstract away communication in the study of distributed protocols. We should consider them only in terms of knowledge and actions, where high level knowledge-oriented actions serve an agent in order to affect the state of knowledge of other agents. In this approach, the close connection between knowledge and communication plays an important role in the implementation of knowledge-oriented actions: The properties of the communication medium affect what actions are available.

Distributed systems are not the only domain in which the relationship between knowledge and communication plays an important role. See [Dre81] for a discussion of this relationship in the context of a philosophical setting. The relationship between the properties (mainly noisiness) of the communication medium and the amount of information it can transmit is a central topic in *Information Theory* [Sha49]. In the work of Rosenschein and Kaelbling in situated automata [RK86], knowledge-based specifications of a robot's desired behavior are compiled into electronic circuits implementing this behavior. Positive current on a given wire corresponds directly to knowledge of certain facts about the world. Finally, in his invited talk at TARK II [Mye88], Myerson surveyed work in game theory where situations arise in which an unreliable communication link is more

useful than a reliable one. Namely, there are situations where the fact that a communication link fails to deliver a certain fraction of the messages sent on it can make it advantageous for a player to send messages describing its true preferences and intentions on the link. Moreover, on a perfectly reliable link it is not always to the player's advantage to tell the truth in that situation. As a result, a message received on the unreliable link is believable, while one received on a reliable link under the same conditions may not be believed. Finally, we started the introduction claiming that in many disciplines a great deal of knowledge is transmitted via the actions that agents perform. One could imagine an analysis of actions that would separate the signal involved in an action from the other effects of the action. Once we do this, the discussion in this paper should be of useful in developing a comprehensive theory of teh interplay between knowledge, action and communication.

Acknowledgements

I'd like to thank Mark Tuttle and Juan Garay for commenting on an early draft of this paper and improving the presentation.

References

[CM86] K. M. Chandy and J. Misra. How processes learn. *Distributed Computing*, 1(1):40–52, 1986.

[DM90] C. Dwork and Y. Moses. Knowledge and common knowledge in a Byzantine environment: crash failures. *Information and Computation*, 88(2):156–186, 1990.

[Dre81] F. I. Dretske. *Knowledge and the flow of information.* MIT Press, 1981.

[FHMV92] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge.* 1992. To appear.

[Fis90] M. Fischer. Personal communication, reported by G. Taubenfeld, 1990.

[Gra78] J. Gray. Notes on database operating systems. In R. Bayer, R. M. Graham, and G. Seegmuller, editors, *Operating Systems: An Advanced Course*, Lecture Notes in Computer Science, Vol. 66. Springer-Verlag, 1978. Also appears as *IBM Research Report RJ 2188*, 1978.

[Had87] V. Hadzilacos. A knowledge-theoretic analysis of atomic commitment protocols. In *Proc. 6th ACM Symp. on Principles of Database Systems*, pages 129–134, 1987. A revised version has been submitted for publication.

[Hal87] J. Y. Halpern. Using reasoning about knowledge to analyze distributed systems. In J. Traub et al., editors, *Annual Review of Computer Science, Vol. 2*, pages 37–68. Annual Reviews Inc., 1987.

[HF85] J. Y. Halpern and R. Fagin. A formal model of knowledge, action, and communication in distributed systems: preliminary report. In *Proc. 4th ACM Symp. on Principles of Distributed Computing*, pages 224–236, 1985.

[Hin62] J. Hintikka. *Knowledge and Belief.* Cornell University Press, 1962.

[HM90] J. Y. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37(3):549–587, 1990. An early version appeared in *Proceedings of the 3rd ACM Symposium on Principles of Distributed Computing*, 1984.

[HMW90] J. Y. Halpern, Y. Moses, and O. Waarts. A characterization of eventual Byzantine agreement. In *Proc. 9th ACM Symp. on Principles of Distributed Computing*, pages 333–346, 1990.

[HZ87] J. Y. Halpern and L. D. Zuck. A little knowledge goes a long way: Simple knowledge-based derivations and correctness proofs for a family of protocols. In *Proc. 6th ACM Symp. on Principles of Distributed Computing*, pages 269–280, 1987. A revised and expanded version appears as IBM Research Report RJ 5857, 1987 and will appear in *Journal of the ACM.*

[Kis90] O. Kislev. Notify: A high-level, knowledge based action. Master's thesis, The Weizmann Institute, 1990.

[Kri63] S. Kripke. A semantical analysis of modal logic I: normal modal propositional calculi. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 9:67–96, 1963. Announced in *Journal of Symbolic Logic*, **24**, 1959, p. 323.

[KT88] R. Koo and S. Toueg. Effects of message loss on the termination of distributed protocols. *Information Processing Letters*, 27:181–188, 1988.

[Lew69] D. Lewis. *Convention, A Philosophical Study.* Harvard University Press, 1969.

[Maz90] M. S. Mazer. A link between knowledge and communication in faulty distributed systems. In R. Parikh, editor, *Proceedings of the Third Conference on Theoretical Aspects of Reasoning about Knowledge*, pages 289–304. Morgan Kaufmann, 1990.

[Mos92] Y. Moses. Knowledge-oriented programming. In prepapration, 1992.

[MT88] Y. Moses and M. R. Tuttle. Programming simultaneous actions using common knowledge. *Algorithmica*, 3:121–169, 1988.

[Mye88] R. B. Myerson. Incentice constraints and optimal communication systems. In M. Y. Vardi, editor, *Proceedings of the Second Conference on Theoretical Aspects of Reasoning about Knowledge*, pages 179–193. Morgan Kaufmann, 1988.

[NB] G. Neiger and R. Bazzi. Using knowledge to optimally achieve coordination in distributed systems. This volume.

[RK86] S. J. Rosenschein and L. P. Kaelbling. The synthesis of digital machines with provable epistemic properties. In J. Y. Halpern, editor, *Theoretical Aspects of Reasoning about Knowledge: Proceedings of the 1986 Conference*, pages 83–97. Morgan Kaufmann, 1986.

[Sha49] C. Shannon. *The mathematical theory of information.* University of Illinois Press, 1949.