March 1986

Report No. STAN-CS-86-1120

1

[redacted]

86- 0334

OK DTIC

Knowledge in a Distributed Environment

by

Yoram Ofer Moses

Department of Computer Science

Stanford University
Stanford, CA 94305

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) KNOWLEDGE IN A DISTRIBUTED ENVIRONMENT		5. TYPE OF REPORT & PERIOD COVERED technical
		6. PERFORMING ORG. REPORT NUMBER STAN-CS-86-1120
7. AUTHOR(s) Yoram Moses		8. CONTRACT OR GRANT NUMBER(s) ARPA N00039-82-C-0250
9. PERFORMING ORGANIZATION NAME AND ADDRESS Computer Science Department Stanford University		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE March 1986
		13. NUMBER OF PAGES 104
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Defense Advanced Research Project Agency 1400 Wilson Blvd Arlington, VA 22209		15. SECURITY CLASS. (of this report)
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release: distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

see next page.

19 KEY WORDS (Continued)

20 ABSTRACT (Continued)

The distributed nature of information in a distributed system is one of the major issues that protocols for cooperation and coordination between individual components in a such systems must handle. Individual sites customarily have only partial knowledge about the general state of the system. Moreover, different information is available at different sites of the system. Consequently, a central role of communication in such protocols is to inform particular sites about events that take place at other sites, and to transform the system's state of knowledge in a way that will guarantee the successful achievement of the goals of the protocol.

This thesis takes a few initial steps towards the study of the role of knowledge in distributed systems. We present a general framework for defining knowledge in a distributed system, and identify a variety of *states of knowledge of sets of processors*, which seem to capture some basic aspects of coordinated actions in a distributed environment. This machinery is applied to the analysis of a number of problems: we generalize and extend the well-known ***coordinated attack*** problem, which deals with the effects of unreliable communication on coordination in a distributed system; we analyze a generalized version of the ***cheating wives*** puzzle, obtaining insight into the subtle differences between broadcasting messages via different communication channels, and into the the subtle interaction between knowledge, communication and action. Finally, we apply this machinery to the study of fault-tolerance in systems of unreliable processors, providing considerable insight into the ***Byzantine agreement*** problem, and obtaining improved protocols for Byzantine agreement and many related problems.

Knowledge in a Distributed Environment

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

by
Yoram Ofer Moses
March 1986

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Joe Halpern

Joseph Y. Halpern
(Principal Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Nils Nilsson
(Co-Principal Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

John McCarthy

John McCarthy

Approved for the University Committee on Graduate Studies:

Elizabeth Closs Traugott

Dean of Graduate Studies

Abstract

The distributed nature of information in a distributed system is one of the major issues that protocols for cooperation and coordination between individual components in a such systems must handle. Individual sites customarily have only partial knowledge about the general state of the system. Moreover, different information is available at different sites of the system. Consequently, a central role of communication in such protocols is to inform particular sites about events that take place at other sites, and to transform the system's state of knowledge in a way that will guarantee the successful achievement of the goals of the protocol.

This thesis takes a few initial steps towards the study of the role of knowledge in distributed systems. We present a general framework for defining knowledge in a distributed system, and identify a variety of states of knowledge of sets of processors, which seem to capture some basic aspects of coordinated actions in a distributed environment. This machinery is applied to the analysis of a number of problems: we generalize and extend the well-known ***coordinated attack*** problem, which deals with the effects of unreliable communication on coordination in a distributed system; we analyze a generalized version of the ***cheating wives*** puzzle, obtaining insight into the subtle differences between broadcasting messages via different communication channels, and into the the subtle interaction between knowledge, communication and action. Finally, we apply this machinery to the study of fault-tolerance in systems of unreliable processors, providing considerable insight into the *Byzantine agreement* problem, and obtaining improved protocols for Byzantine agreement and many related problems. (KR)

ACKNOWLEDGEMENTS

I have been most fortunate in having the opportunity to work on this thesis in a most stimulating and supportive environment. Many people have contributed to this work in a multitude of ways. I thank them all, and apologize in advance to those I forget to mention explicitly below.

Above all I am grateful to Joe Halpern, my thesis advisor. His guidance, enthusiasm, and research style have had a profound influence on this thesis and on my own research standards. His friendship and keen insight made him an exceptional advisor and collaborator.

I would like to thank John McCarthy for his continued support and challenging discussions throughout my program at Stanford. I thank Nils Nilsson, who officially co-advised this thesis, for his support and encouragement.

Stimulating discussions with Danny Dolev provided the original motivation for much of the work in this thesis. I would like to thank Cynthia Dwork for a most enjoyable and inspiring collaboration on the material of Chapter 6. Technical discussions with Ron Fagin and Moshe Vardi on various aspects of this work were a great help in determining the direction of this work. I would also like to thank Martin Abadi, Yisrael Aumann, Dave Chelberg, Herb Clark, Brian Coan, Steve Deering, Raul Duran, Carolyn Foss, Martin Gardner, Bengt Jonsson, Don Knuth, Danny Lehmann, Nancy Lynch, Yoni Malachi, Tim Mann, Stan Rosenschein, Ray Strong, Ashok Subramanian, Mark Tuttle, and Joe Weening for insightful comments and stimulating discussions.

I thank my parents and brother, as well as my great uncle George Rothstein and his late wife Erna, for their love and support. Special thanks to Dorothy Lush for being a source of strength and true friendship in difficult times.

This work was supported by DARPA contract N00039-82-C-0250, by an IBM Research Student Associateship, and by an IBM fellowship.

Credits

The work presented in this thesis is based on three papers, all of which are the result of joint work. Chapters 1–4 are the result of joint work with Joe Halpern, and are based on [HM2]. Chapter 5 represents an answer to a set of questions posed by Danny Dolev, and is joint work with him and Joe Halpern on [MDH]. Chapter 6 is the result of joint work with Cynthia Dwork on [DM]. All three papers have been submitted to journals: [HM2] to the Journal of the ACM, [MDH] to Distributed Computing, and [DM] to Information and Control. All of the writing in both the papers and the thesis is mine, and none of the theorems is due solely to my collaborators.

To My parents, Ala and Shimon Moses

Table of Contents

CHAPTER 1

INTRODUCTION

Distributed systems of computers are rapidly gaining popularity in a wide variety of applications. The cooperation and coordination between computers at distinct locations made possible in a distributed system greatly enhances the usefulness of the individual computers in the system. Such cooperation and coordination is carried out by the execution of ***distributed protocols*** (sometimes also called ***distributed programs*** or ***plans***) at the different sites of the system. However, the distributed nature of control and information in such systems makes the design and analysis of distributed protocols a complex task. Unfortunately, at the current time the design of distributed protocols is more an art than a science. Basic foundations, general techniques, and a clear methodology are urgently needed to improve our understanding and ability to deal effectively with distributed systems.

Whereas the tasks that distributed protocols are required to perform are normally stated in terms of the overall behavior of the system, the actions of an individual processor in a distributed system can only depend on its local information. This local information varies from site to site, and generally provides only a partial view of the state of the system. In determining the interaction between the individual processors, a distributed protocol must ensure that the states of knowledge attained by the system during an excution of the protocol allow the achievement of the protocol's goals. Thus, reasoning about the system's state of knowledge seems to be an important part of the design of distributed protocols. Indeed, designers of such protocols frequently find it useful to reason intuitively about processors' states of knowledge at various points in the execution of a protocol. However, formal descriptions of distributed protocols, as well as actual proofs of their correctness or impossibility, have traditionally avoided any explicit treatment of knowledge. Rather, the intuitive arguments about the state of knowledge of components of the system are customarily buried in combinatorial proofs that are often unintuitive and hard to follow. Consequently, essentially the same proof is often repeated with slight variations for closely related models of distributed systems.

This thesis attempts to take a few initial steps towards making reasoning about knowledge in a distributed environment more explicit, and towards understanding the relationship between knowledge, communication, and action in a distributed system. Explicitly reasoning about the states of knowledge of processors in a distributed system, it is hoped, will provide a more general and uniform setting that will offer genuine insight into the basic structure and limitations of protocols in such systems. Clear semantics for knowledge in a distributed system should reveal subtleties that may not be otherwise apparent, sharpen our understanding of basic issues, and improve even the high level intuitive reasoning about knowledge so intimately involved in the design of distributed protocols and plans. In the long run, we hope that a theory of knowledge, communication, and action will prove rich enough to provide general foundations for a unified theoretical treatment of distributed systems.

The general concept of knowledge has received considerable attention in a variety of fields, ranging from Philosophy [Hi] and Artificial Intelligence [Mc], to Game Theory [A] and Psychology [ClM]. A study of knowledge in distributed systems can greatly benefit from work done in those fields and the paradigms presented by them. Furthermore, given that it is somewhat easier to formally model the "knowers" and their knowledge in a distributed system, work on knowledge in distributed systems promises to shed light on aspects of knowledge that are relevant to related fields.

In the next section we look at the "muddy children" puzzle, which illustrates some of the subtleties involved in reasoning about knowledge in the presence of many "knowers" or "agents". In Section 1.2 we introduce a hierarchy of states of knowledge that a group may be in. Section 1.3 focuses on the relationship between knowledge and communication by looking at the *coordinated attack* problem. Section 1.4 contains an outline of the thesis.

1.1 The "muddy children" puzzle

A crucial aspect of distributed protocols is the fact that a number of different processors cooperate in order to achieve a particular goal. Thus, since more than one individual is present, an individual may have knowledge about other individuals' knowledge in addition to his knowledge about the physical world. This often requires care in distinguishing subtle differences between seemingly similar states of knowledge. A classical example of this phenomenon is the "muddy children" puzzle — a variant of the well known "wise men" or "cheating wives" puzzles. The version given here is taken from [B]:

> Imagine n children playing together. The mother of these children has told them that if they get dirty there will be severe consequences. So, of course, each child wants to keep clean, but each would love to see the others get dirty. Now

> it happens during their play that some of the children, say k of them, get mud on their foreheads. Each can see the mud on others but not on his own forehead. So, of course, no one says a thing. Along comes the father, who says, "At least one of you has mud on your head," thus expressing a fact known to each of them before he spoke (if $k > 1$). The father then asks the following question, over and over: "Can any of you prove you have mud on your head?" Assuming that all the children are perceptive, intelligent, truthful, and that they answer simultaneously, what will happen?
>
> There is a "proof" that the first $k - 1$ times he asks the question, they will all say "no" but then the kth time the dirty children will answer "yes."
>
> The "proof" is by induction on k. For $k = 1$ the result is obvious: the dirty child sees that no one else is muddy, so he must be the muddy one. Let us do $k = 2$. So there are just two dirty children, a and b. Each answers "no" the first time, because of the mud on the other. But, when b says "no," a realizes that he must be muddy, for otherwise b would have known the mud was on his head and answered "yes" the first time. Thus a answers "yes" the second time. But b goes through the same reasoning. Now suppose $k = 3$; so there are three dirty children, a, b, c. Child a argues as follows. Assume I don't have mud on my head. Then, by the $k = 2$ case, both b and c will answer "yes" the second time. When they don't, he realizes that the assumption was false, that he is muddy, and so will answer "yes" on the third question. Similarly for b and c.

Let us denote the fact "At least one child has a muddy forehead" by $\mathbf{m}$. Notice that if $k > 1$, i.e., more than one child has a muddy forehead, then every child can see at least one muddy forehead, and the children initially all know $\mathbf{m}$. Thus, it would seem, the father does not need to tell the children that $\mathbf{m}$ holds when $k > 1$. But this is false! In fact, had the father not announced $\mathbf{m}$, the muddy children would never have been able to conclude that their foreheads are muddy. We now sketch a proof of this fact.

First of all, given that the children are intelligent and truthful, a child with a clean forehead will never answer "yes" to any of the father's questions. Thus, if $k = 0$, all of the children answer all of the father's questions "no". Assume inductively that if there are exactly k muddy children and the father does not announce $\mathbf{m}$, then all children answer "no" to all of the father's questions. Note that, in particular, when there are exactly k muddy foreheads, a child with a clean forehead initially sees k muddy foreheads and hears all of the father's questions answered "no". Now assume that there are exactly $k + 1$ muddy children. We prove by induction on the number of questions asked that all of the children answer "no" to all of the father's questions. Assume inductively that all of the children have answered "no" the father's first n questions (for $n = 0$ this condition is vacuously true). Recall that a clean child will necessarily answer "no" to the father's $n + 1^{st}$ question. Next observe that before answering the father's $n+1^{st}$ question, a muddy child has exactly the same information as a clean child has at the corresponding point in the case of k muddy foreheads. It follows that the muddy children must all answer "no" to the father's $n + 1^{st}$ question, and we are done. (Note that a very similar proof shows that if there are k muddy children and the father does announce **m**, his first $k - 1$ questions are answered "no".)

So, by announcing something that the children all know, the father somehow manages to give the children useful information! How can this be? Exactly what *was* the role of the father's statement? In order to answer this question, we need to take a closer look at knowledge in the presence of more than one knower; this is the subject of the next section.

1.2 A hierarchy of states of knowledge

Although in different contexts knowledge may be assumed to mean different things, one property generally required of knowledge is that only true things be known, or, more formally, that knowledge satisfy the axiom

$$K_i\varphi \supset \varphi;$$

i.e., if an individual i knows φ, then φ is true.[1] In Chapter 2 we will discuss specific interpretations for knowledge that seem to be particularly useful in the context of distributed systems. For the purposes of our discussion in the next few sections, the only properties we require of an individual's knowledge is that it be a function of the individual's view of the past and that it satisfy the above axiom (we make this precise in Section 2.2).

Given a reasonable interpretation for individuals' knowledge, how does the notion of knowledge generalize from an individual to a group? In other words, what does it mean to say that a group G of individuals knows a fact φ? More than one possibility is reasonable, with the appropriate choice depending on the application:

- $I_G\varphi$ (read "the group G has *Implicit Knowledge* of φ"): We say that G has implicit knowledge of φ iff someone who would have complete knowledge of what

[1] Notions that do not satisfy the $K_i\varphi \supset \varphi$ axiom are customarily thought of as *belief*.

each member of G knows would know φ. Thus, roughly speaking, φ is implicit knowledge in G if the knowledge about φ is distributed among the members of G. For instance, if one member of G knows ψ and another knows that $\psi \supset \varphi$, the group G can be said to have implicit knowledge of φ.

- $S_G\varphi$ (read "someone in G knows φ"): We say that $S_G\varphi$ holds iff some member of G knows φ. More formally,

$$S_G\varphi \equiv \bigvee_{i \epsilon G} K_i\varphi.$$

- $E_G\varphi$ (read "everyone in G knows φ"): We say that $E_G\varphi$ holds iff all members of G know φ. More formally,

$$E_G\varphi \equiv \bigwedge_{i \epsilon G} K_i\varphi.$$

- $E^k_G\varphi$, $k \geq 2$ (read "φ is E^k-knowledge in G"): $E^k_G\varphi$ is defined by

$$E^1_G\varphi = E_G\varphi,$$

$$E^{k+1}_G\varphi = E_G E^k_G\varphi, \text{ for } k \geq 1.$$

φ is said to be E^k-knowledge in G if "everyone in G knows that everyone in G knows that ... that everyone in G knows that φ is true" holds, where the phrase "everyone in G knows that" appears in the the sentence k times. Equivalently,

$$E^k_G\varphi \equiv \bigwedge_{i_j \epsilon G,\, 1 \leq j \leq k} K_{i_1}K_{i_2}\cdots K_{i_k}\varphi.$$

- $C_G\varphi$ (read "φ is *Common Knowledge* in G")· Roughly speaking, φ is said to be common knowledge in G if φ is true, and is E^k_G-knowledge for all $k \geq 1$. In other words,

$$C_G\varphi \equiv \varphi \wedge E_G\varphi \wedge E^2_G\varphi \wedge \cdots \wedge E^m_G\varphi \wedge \cdots$$

In particular, $C_G\varphi$ implies all formulas of the form $K_{i_1}K_{i_2}\cdots K_{i_n}\varphi$, where the i_j are all members of G, for any finite n, and is equivalent to the (infinite) conjunction of all such formulas.

(The subscript G will be omitted when the group G is understood from context.)

Clearly, the notions of group knowledge introduced above form a hierarchy, with

$$C\varphi \supset \cdots \supset E^{k+1}\varphi \supset \cdots \supset E\varphi \supset S\varphi \supset I\varphi \supset \varphi.$$

However, depending on the circumstances, these notions might not be distinct. For example, consider a model of parallel computation in which a collection of n processors share a common memory. If their knowledge is stored in memory then we arrive at a situation in which $C\varphi \equiv E^k\varphi \equiv E\varphi \equiv S\varphi \equiv I\varphi$. By way of contrast, in a distributed system in which n processors are connected via some communication network and each one of them has its own memory, it is clear that the above hierarchy is strict. Moreover, in such a system, every two levels in the hierarchy can be separated by an actual task, in the sense that there will be an action for which one level in the hierarchy will suffice, but no lower level will. It is quite clear that this is the case with $E\varphi \supset S\varphi \supset I\varphi$, and, as we are about to show, the "muddy children" puzzle is an example of a situation in which $E^{k+1}\varphi$ suffices to perform a required action, but $E^k\varphi$ does not. In the next section we will present the coordinated attack problem, in which $C\varphi$ will suffice to perform a required action, but for *no* k will $E^k\varphi$ suffice.

Returning to the muddy children puzzle, let us observe the state of the children's knowledge of **m**: "At least one forehead is muddy". Before the father speaks, E^{k-1}**m** holds, and E^k**m** doesn't. To see this, consider the case $k = 2$ and suppose that Alice and Bob are the only muddy children. Clearly everyone sees at least one one muddy child, so E**m** holds. But the only muddy child that Alice sees is Bob, and, not knowing whether she is muddy, Alice considers it possible that Bob is the only muddy child. Alice therefore considers it possible that Bob sees no muddy child. Thus, although both Alice and Bob know **m** (i.e., E**m** holds), Alice does not know that Bob knows **m**, and hence E^2**m** does not hold. A similar argument works for the general case. We leave it to the reader to check that when there are k muddy children, E^k**m** suffices to ensure that the muddy children will be able to prove their dirtiness, whereas E^{k-1}**m** does not. (A more detailed analysis of this argument, as well as a more general treatment of variants of the muddy children puzzle more closely related to distributed systems, appears in Chapter 5.)

Thus, the role of the father's statement was to improve the children's' state of knowledge of **m** from E^{k-1}**m** to E^k**m**. In fact, the children have *common knowledge* of **m** after the father announces that **m** holds. Roughly speaking, the father's public announcement of **m** to the children as a group results in all the children knowing **m** and knowing that the father has publicly announced **m**. Assuming that it is common knowledge that all of the children know anything the father announces publicly, it is easy to conclude that **m** is common knowledge once the father announces **m**. Once the father announces **m** all of the children know **m** and know that the father announced **m**. Every child therefore knows that all of the children know **m** and know that the father publicly announced $\mathbf{m}$, and therefore $E^2\mathbf{m}$ holds. It is similarly possible to show that once the father announces $\mathbf{m}$ $E^n\mathbf{m}$ holds for all n, so $C\mathbf{m}$ holds (see Section 3.1 for further discussion). Since, in particular, $E^k\mathbf{m}$ holds, the muddy children can succeed in proving their dirtiness.

A large part of the communication in a distributed system can also be viewed as the act of improving the state of knowledge (in the sense of "climbing up a hierarchy") of certain facts. This is an elaboration of the view of communication in a network as the act of "sharing knowledge". Taking this view, two notions come to mind. One is *fact discovery* – the act of changing the state of knowledge of a fact φ from being implicit knowledge to levels of explicit knowledge (usually S-, E-, or C-knowledge), and the other is *fact publication* – the act of changing the state of knowledge of a fact that is not common knowledge to common knowledge. An example of fact discovery is the detection of global properties of a system, such as deadlock. The system initially has implicit knowledge of the deadlock, and the detection algorithm improves this state to S-knowledge (see [CL] for work related to fact discovery). An example of fact publication is the introduction of a new communication convention in a computer network. Here the initiator(s) of the convention wish to make the new convention common knowledge.

In the following chapters we will devote a considerable amount of attention to fact publication and common knowledge. As we shall show, common knowledge is inherent in a variety of notions such as agreement, conventions, and coordinated action. Furthermore, having common knowledge of a large number of facts allows for better and shorter communication. Since these are goals frequently sought in distributed computing, the problem of fact publication — how to attain common knowledge — becomes crucial. Common knowledge is also a basic notion in everyday communication between people. For example, shaking hands to seal an agreement signifies that the handshakers have common knowledge of the agreement. Also, it can be argued that when we use a definite reference such as "the president" in a sentence, we assume common knowledge of who is being referred to (cf. [ClM]).

In [ClM], Clark and Marshall present two basic ways in which a group can come to have common knowledge of a fact. One is by membership in a community, e.g., the meaning of a red traffic light is common knowledge to the community of licensed drivers. The other is by being copresent with the occurrence of the fact, e.g., the father's gathering the children and publicly announcing the existence of muddy foreheads made that fact common knowledge. Notice that if, instead, the father had taken each child aside (without the other children noticing) and told her or him about it privately, this information would have been of no help at all. Indeed, the children would probably think it was rather strange of him to tell them such an obvious fact.

In the context of distributed systems, community membership corresponds to information that the processors are guaranteed to have by virtue of their presence in the system (e.g., information that is "inserted into" the processors before they enter the system). However, it is not obvious how to simulate copresence using message passing in a distributed system. What is the analogue of making "public" announcements in a distributed system? As we shall see, there are serious problems in attempting to do so.

1.3 The coordinated attack problem

To get a flavor of the issues involved in attaining common knowledge by simulating copresence in a distributed system, consider the coordinated attack problem, taken from the operating systems folklore (cf. [Gal], [Gr], [YC]):

> 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 only communicate 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 now show that despite the fact that everything goes smoothly, no agreement can be reached and no general can decide to attack. (This is, in a way, a folk theorem of operating systems theory; cf. [Gal], [Gr], [YC].) Suppose the messenger starts out in camp A carrying the message "Let's attack at dawn", and delivers it to camp B an hour later. General A does not immediately know whether the messenger succeeded in delivering the message. And because general B would not attack at dawn if the messenger is captured and fails to deliver the message, general A will not attack unless he knows that the message was successfully delivered. Consequently, general B sends the messenger back to general A with an acknowledgement. Suppose the messenger delivers the acknowledgement to general A an hour later. Since general B knows that general A will not attack without knowing that B received the original message, he knows that A will not attack unless the acknowledgement is successfully delivered. Thus, general B will not attack unless he knows that the acknowledgement has been successfully delivered. However, for general B to know that the acknowledgement has been successfully delivered, general A must send the messenger back with an acknowledgement to the acknowledgement .... Similar arguments can be used to show that no fixed finite number of acknowledgements, acknowledgements to acknowledgements etc. suffices for the generals to attack. Note that in the discussion above the generals are essentially running a *handshake* protocol (cf. [Gr]). The above discussion shows that for no k does a k-round handshake protocol guarantee that the generals be able to coordinate an attack.

In fact, we can use this intuition to actually prove that the generals can never attack and be guaranteed that they are attacking simultaneously. We argue by induction on n — the number of messages delivered by the time of the attack — that n messages do not suffice. Clearly, if no message is delivered, then general B will not know of the intended attack, and a simultaneous attack is impossible. For the inductive step, assume that k messages do not suffice. If $k+1$ messages suffice, then the sender of the $k+1^{st}$ message attacks without knowing whether his last message arrived. Since whenever one general attacks they both do, the intended receiver of the $k+1^{st}$ message must attack regardless of whether the $k+1^{st}$ message is delivered. Thus, the $k+1^{st}$ message is irrelevant, and k messages suffice, contradicting the inductive hypothesis.

After presenting a detailed proof of the fact that no protocol the generals can use will satisfy their requirements and allow them to coordinate an attack, Yemini and Cohen in [YC] make the following remark:

> ... Furthermore, proving protocols correct (or impossible) is a difficult and cumbersome art in the absence of proper formal tools to reason about protocols. Such backward-induction argument as the one used in the impossibility proof should require less space and become more convincing with a proper set of tools.

Yemini and Cohen's proof does not explicitly reason about knowledge, but it uses a many-scenarios argument to show that if the generals safely attack in one scenario, then there is another scenario in which one general will attack and the other will not. We feel that understanding the role knowledge plays in problems such as coordinated attack is a first step towards simplifying the task of designing and proving the correctness of protocols.

A protocol for the coordinated attack problem, if one did exist, would ensure that when the generals attack, they are guaranteed to be attacking simultaneously. Thus, in a sense an attacking general (say A) would know that the other general (say B) is also attacking. Furthermore, A would know that B similarly knows that A is attacking. It is easy to extend this reasoning to show that when the generals attack they in some sense have common knowledge of the attack. However, each message that the messenger delivers can add at most one level of knowledge about the desired attack, and no more. For example, when the messenger first arrives at camp B, general B knows about A's desire to coordinate an attack, but A does not know whether the message was delivered, and therefore A does not know that B knows about the intended attack. And when the messenger returns to camp A with an acknowledgement, A knows that B knows about the intended attack, but, not knowing whether the messenger delivered the acknowledgement, B does not know that A knows (that B knows of the intended attack). This in some sense explains why the generals cannot reach an agreement to attack using a finite number of messages. We are about to formalize this intuition. Indeed, we will prove a more general result from which the inability to achieve a guaranteed coordinated attack will follow as a corollary. Namely, we will prove that communication cannot be used to attain common knowledge in a system in which communication is not guaranteed, and formally relate a guaranteed coordinated attack to attaining common knowledge. Before we do so, we need to define some of the terms that we use more precisely.

1.4 Outline of the thesis

In this chapter we have discussed some of the motivation for studying knowledge in a distributed environment, considered two puzzles – the muddy children puzzle and the coordinated attack problem – which illustrated some of the subtleties involved in reasoning about knowledge in a distributed environment, and introduced a variety of states of knowledge that correspond to the knowledge of a group of individuals. In the next chapter we give a brief formal definition of a distributed system and present a general framework for ascribing knowledge (and belief) to processors in such a system. Chapter 3 deals with the general problem of fact publication – attaining common knowledge of new facts, resulting among other things in a formal proof of a generalization of the coordinated attack problem, and establishing a close relationship between common knowledge and simultaneous actions. Chapter 4 introduces states of knowledge that are related to common knowledge that correspond to various types of coordinated actions, and states of knowledge relative to a relativistic notion of time. Results about the inability to attain some of these weaker states of knowledge when communiaction is unreliable further generalizes the coordinated attack problem. Chapter 5 is a case study of the relationship between knowledge, action, and communication. An analysis of variants of the cheating husbands puzzle (essentially the muddy children puzzle) that include broadcasts of a message over a variety of communication mediums is performed, within the context of a fictional story. Chapter 6 applies the formalism developed in Chapter 2 to the study of systems of unreliable processors and the Byzantine agreement problem. It is shown that an analysis of when facts that are implicit knowledge become common knowledge in such a system provides considerable insight into the fundamental structure of fault-tolerant protocols, resulting in improved protocols for Byzantine agreement problem and many related proolems. Chapter 7 includes some concluding remarks.

Chapter 2

Modeling Knowledge in Distributed Systems

In order to be able to explicitly reason about knowledge in a distributed system, one needs to have the means to make precise statements about the state of knowledge of the system at any given point. We need a way of determining both what individual processors know and what states of knowledge groups of processors have. Given that for different applications we are interested in different interpretations of knowledge, this chapter will present a very general framework for ascribing knowledge to processors in a distributed system. We start by presenting a model of a distributed system in Section 2.1. In Section 2.2 we discuss how knowledge (and beliefs) can be ascribed to processors in such a system. Section 2.3 introduces a special class of interpretations of knowledge in a distributed system that will prove to be very useful later on. Finally, in Section 2.4 we discuss some of the implications of our definitions.

2.1 A general model of a distributed system

We view a distributed system as a finite collection $\{p_1, p_2, \ldots, p_n\}$ of two or more processors that are connected by a communication network. We assume an external source of "real time" that in general is not directly observable by the processors. The processors are state machines that possibly have *clocks*, where a clock is a monotone nondecreasing function of real time. The processors communicate with each other by sending messages along the links in the network. At a given real time, a processor's *message history* is the sequence of messages it has sent and received (in the order they were sent/received) up to (but not including) that time. If the processor has a clock, then the messages are also marked by the time on the processor's clock at which they were sent or received. Every processor's message history is always finite. (In particular this implies that only a finite number of messages can be delivered in the system in a finite amount of time.)

A processor is assumed to be in a distinguished "sleeping" state until it "wakes up" or joins the system at some point in real time. The real time at which the processor wakes up is called the processor's *initial time*. The processor's internal state when it wakes up is called its *initial state*. Before waking up, a processor sends and receives no messages. Thus, a processor's message history when it wakes up is empty. After it wakes up, we assume that a processor follows (or *executes*) some deterministic protocol.[1] We give a formal definition of protocols shortly.

A *protocol* is a function specifying what actions a processor takes (which in our case amounts to what messages it sends) at any given point as a function of the processor's initial state, message history, and the range of values that its clock has read since the processor "woke up". (We restrict our attention to deterministic protocols for notational and conceptual clarity. A nondeterministic protocol can be thought of as a family of deterministic protocols, each corresponding to a particular sequence of nondeterministic choices. Our negative results will immediately carry over to nondeterministic protocols, due to this observation.) A *joint protocol for G* is a tuple consisting of a protocol for every processor in G.

A processor's *message history function* determines the processor's message history as a function of real time. A processor's *clock time function* determines the processor's clock time as a function of real time. A *run r* of a distributed system is a complete history of its behavior, from the beginning of time until the end of time. This includes each processor p_i's initial time $t_0(p_i, r)$, its initial state (its state at time $t_0(p_i, r)$), message history function, clock time function, and the protocol p_i follows. A run is *consistent* iff the actions taken by each processor at all times are precisely those that are specified by its protocol (cf. [HF]). We never consider inconsistent runs. A *point* is a pair (r, t), where r is a run and t is a real number corresponding to the (real) time t.

Corresponding to every distributed system, given an appropriate set of assumptions about the properties of the system and its possible interaction with its environment, there is a natural set S of the possible runs of the system. This set essentially contains all the relevant information about the system. The relative behavior of clocks, the properties of communication in the system, and many other properties of the system, are directly reflected in the properties of this set of runs. We will identify a distributed system with such a set S of its possible runs. As we shall see in the sequel, identifying a distributed system with a set of runs S allows us to define properties of a system formally in a rather clean way. A (possibly joint) protocol is said to be *executed in S* if there is a run of S in which it is executed.

[1] For the purposes of our discussion through Chapter 5 we are essentially assuming that processors are reliable, i.e., they are guaranteed to follow their protocols in good faith. Modeling systems in which processors may be faulty, and furthermore modeling knowledge in such systems, adds another layer of complexity, as we will see in Chapter 6.

Intuitively, a processor's *view* at a given point describes everything that the processor may have observed by that point in the run. More formally, we define a processor p_j's view at a point (r, t), denoted $v(p_j, r, t)$, to be a distinguished "inactive" view if $t < t_0(p_j, r)$, and otherwise to consist of p_j's initial state, message history, and the range of values its clock has read since it woke up, together with the protocol p_j is following in r. We include the protocol as part of the view because a processor may often have access to the protocol it is following.

2.2 Ascribing knowledge to processors

What does it mean for a processor to know a fact φ? In our opinion, there is no unique "correct" answer to this question. Different interpretations of knowledge in a distributed system are appropriate for different applications. For example, an interpretation by which a processor is said to know φ only if φ appears explicitly in a designated part of the processor's storage (its "database") seems interesting for certain applications. In other contexts we may be interested in saying that a processor knows φ if the processor could deduce φ from the information available to it (say by using a logical system such as the axioms of [Sa],[Hi],[Le], or [HM], possibly with a specified limitation on computational resources; cf. [Kon]). We will shortly define yet another interpretation of knowledge in a distributed system in which, roughly speaking, a processor will be said to know φ if the processor's state implies that φ holds.

Although the notion of knowledge may have a number of interesting interpretations, there are two properties that we require any notion of knowledge in a distributed system to satisfy. First of all, a processor's knowledge at a given point must be a function of its view of the run by that point. At two points in which the processor has the same view, the processor should know exactly the same facts. Secondly, under no circumstances should a fact φ be false and be known to be true at the same time (i.e., only true facts can be known; this is exactly the content of the requirement that $K_i\varphi \supset \varphi$ hold). In order to make these intuitions precise, we proceed as follows.

We assume the existence of an underlying logical language of formulas for representing *ground* facts about the system. A ground fact is a fact about the state of the system that does not explicitly involve processors' knowledge. Formally, a ground fact φ will be identified with a set of points $\tau(\varphi) \subseteq S \times (-\infty, \infty)$. Given a run $r \in S$ of the system and a time t, we will say that φ holds at (r, t), denoted $(r, t) \models \varphi$, iff $(r, t) \in \tau(\varphi)$.

We extend the original language of ground formulas to a language that is closed under knowledge operators (for every formula φ and processor p_i, $K_i\varphi$ is a formula), common knowledge operators (for every formula φ and subset G of the processors, $C_G\varphi$ is a formula), and boolean connectives.[2] (In Chapter 4 we will also add temporal operators.)

Intuitively, an *epistemic interpretation* for the system is a specification of what every processor knows (or, more precisely, believes) at any given point as a function of the processor's view of its history at that point. More precisely, an epistemic interpretation $\mathcal{I}$ is a function assigning to every processor p_j at any given point (r,t), a set $\mathcal{K}_j^{\mathcal{I}}(r,t)$ of facts in the extended language that p_j is said to "know". $\mathcal{K}_j^{\mathcal{I}}(r,t)$ is required to be a function of p_j's view at (r,t). Thus if $v(p_j,r,t) = v(p_j,r',t')$ for two points (r,t) and (r',t'), then $\mathcal{K}_j^{\mathcal{I}}(r,t) = \mathcal{K}_j^{\mathcal{I}}(r',t')$.

Given an epistemic interpretation $\mathcal{I}$, we now specify when a formula φ of the extended language holds at a point (r,t) (denoted $(\mathcal{I},r,t) \models \varphi$). If φ is a formula of the original language (a "ground" formula), then φ holds at (r,t) iff $(r,t) \in \tau(\varphi)$ (i.e., iff φ holds at (r,t) according to the original semantics). If φ is a conjunction or a negation, then its truth is defined based on the truth of its subformulas in the obvious way. If φ is of the form $K_j\psi$, then φ holds (i.e., $(\mathcal{I},r,t) \models K_j\psi$) iff $\psi \in \mathcal{K}_j^{\mathcal{I}}(r,t)$. If φ is of the form $C_G\psi$, then $(\mathcal{I},r,t) \models C_G\psi$ iff for all sequences $p_{i_1}, p_{i_2}, \ldots, p_{i_n}$ of members of G, it is the case that $(\mathcal{I},r,t) \models K_{i_1}K_{i_2}\cdots K_{i_n}\psi$.

When talking about a processor's *knowledge*, we are not interested in epistemic interpretations that state that a processor knows a fact φ, when φ is in fact false! (Otherwise we are dealing with *belief* or some related notion, but not strictly with knowledge; cf. [HM].) Given an epistemic interpretation $\mathcal{I}$ and a set of runs S, we say that $\mathcal{I}$ is a *knowledge interpretation for* S if for all processors p_j, times t, runs $r \in S$ and formulas φ in the extended language, it is the case that $(\mathcal{I},r,t) \models K_j\varphi$ implies that $(\mathcal{I},r,t) \models \varphi$, *I.e.,* a knowledge interpretation for S is an epistemic interpretation that for the runs of S satisfies the axiom $K_i\varphi \supset \varphi$.

We say that a processor p_i *supports* $C_G\varphi$ at $(\mathcal{I},r,t)$ if p_i knows all the forumlas of the form $K_{i_1}K_{i_2}\cdots K_{i_n}\varphi$ that constitute $C_G\varphi$ (that is, for all sequences $p_{i_1}, p_{i_2}, \ldots, p_{i_n}$ of members of the group G, $(\mathcal{I},r,t) \models K_iK_{i_1}K_{i_2}\cdots K_{i_n}\varphi$ holds).

[2] When appropriate, operators for implicit knowledge and other states of knowledge should also be added, although for simplicity we do not add them here. We will comment on implicit knowledge later in the next section, and make strong use of it in Chapter 6. For a formal treatment of implicit knowledge, also see [HM].

Lemma 2.1: Let $\mathcal{I}$ be a knowledge interpretation for S and let $r \in S$. For all processors $p_i \in G$ at all points (r,t), it is the case that p_i supports $C_G\varphi$ at $(\mathcal{I},r,t)$ iff $(\mathcal{I},r,t) \models C_G\varphi$.

Proof: The 'if' direction follows directly from the definition. For the other direction, assume to the contrary that $p_i \in G$ supports $C_G\varphi$ and that $(\mathcal{I},r,t) \not\models C_G\varphi$. Since $(\mathcal{I},r,t) \not\models C_G\varphi$, there must be some formula ψ of the form $K_{i_1}K_{i_2}\cdots K_{i_n}\varphi$ such that $(\mathcal{I},r,t) \not\models \psi$. But since p_i supports $C_G\psi$ at $(\mathcal{I},r,t)$, we must have $(\mathcal{I},r,t) \models K_i\psi$. It follows that the interpretation $(\mathcal{I},r,t)$ does not satisfy $K_i\psi \supset \psi$ and thus $\mathcal{I}$ is not a knowledge interpretation for S, contradicting our original assumption. ⋈

2.3 State-based knowledge interpretations

There are many possible knowledge interpretations for a given set of runs S. A very important class of interpretations that will form the basis of our discussion in a large part of this thesis is the class of ***state-based interpretations relative to S***. In a state-based interpretation $\mathcal{I}_\sigma$, a state $\sigma(p_j,r,t)$ is associated with every processor p_j at any given point (r,t). This state is required to be a function of p_j's view at (r,t). Thus, if $v(p_j,r,t) = v(p_j,r',t')$ then necessarily $\sigma(p_j,r,t) = \sigma(p_j,r',t')$. Roughly speaking, under this interpretation a processor in state σ_0 knows φ iff φ holds whenever the processor is in state σ_0 in a run of S. More formally, a state-based interpretation $\mathcal{I}_\sigma$ is the unique interpretation satisfying: $\varphi \in \mathcal{K}_j^{\mathcal{I}_\sigma}(r,t)$ iff $(\mathcal{I}_\sigma,r',t') \models \varphi$ for all points (r',t') with $r' \in S$, satisfying $\sigma(p_j,r,t) = \sigma(p_j,r',t')$. Under a state-based knowledge interpretation, a processor does not know φ exactly if it is possible for the processor to be in the same state, and at the same time for φ not to hold.[3] Thus, under a state-based interpretation $\mathcal{I}_\sigma$ we have:

$$(\mathcal{I}_\sigma,r,t) \models K_i\varphi \text{ iff } (\mathcal{I}_\sigma,r',t') \models \varphi \text{ for all } (r',t') \text{ satisfying } \sigma(p_i,r,t) = \sigma(p_i,r',t').$$

Given the definition of $E_G\varphi$ from $K_i\varphi$, we now also have that $(\mathcal{I}_\sigma,r,t) \models E_G\varphi$ iff $(\mathcal{I}_\sigma,r',t') \models \varphi$ for all (r',t') satisfying $\sigma(p_j,r,t) = \sigma(p_j,r',t')$ for some $p_j \in G$

Given S, G, and σ, we say that the point (r',t') is *reachable from* (r,t) if there exist points $(r_1,t_1),\ldots,(r_m,t_m)$ with $r_i \in S$, and processors $p_{j_1},\ldots,p_{j_{m-1}} \in G$, such that $(r,t) = (r_1,t_1)$, $(r_m,t_m) = (r',t')$, and $\sigma(p_{j_i},r_i,t_i) = \sigma(p_{j_i},r_{i+1},t_{i+1})$ for $1 \leq i < m$. (Notice that reachability is an equivalence relation, it is reflexive,

[3] Particular state-based interpretations were first suggested to us independently by Cynthia Dwork and by Stan Rosenschein.

symmetric, and transitive.) It is now easy to check from the definition of $E_G\varphi$ that $E_G^m\varphi$ holds for all m at the point (r,t) iff φ holds at all points reachable from (r,t). We thus have:

$$(\mathcal{I}_\sigma, r, t) \models C_G\varphi \text{ iff } (\mathcal{I}_\sigma, r', t') \models \varphi \text{ for all } (r', t') \text{ reachable from } (r,t).$$

State-based interpretations are used by many researchers in the field (cf. [ChM], [HF], [R], [PR]). It is interesting to note that, for the runs of S, a state-based knowledge interpretation for S satisfies all the axioms of S5. In particular this means that a processor's knowledge is closed under deduction: If the processor knows φ and knows that $\varphi \supset \psi$ then it knows ψ. Furthermore, processors are fully introspective: each processor knows what facts it knows and what facts it doesn't know. (See Appendix A for the axioms of S5, and see [Hi], [HM], [Sa] and [FHV] for general models for theories of knowledge satisfying S5.)

A state-based interpretation ascribes knowledge to a processor without the processor necessarily being "aware" of this knowledge, and without the processor needing to perform any particular action in order to obtain such knowledge. In particular, *if our assignment of states* does not distinguish between possibilities at all, i.e., there is a single state Λ such that for all p_j, r, and t, $\sigma(p_j, r, t) = \Lambda$, the processors are still ascribed quite a bit of knowledge: Every fact that is true in all runs at all times is common knowledge to all the processors under this interpretation. It is interesting to note that the hierarchy of Section 1.2 collapses under this interpretation, and $I\varphi \equiv E\varphi \equiv C\varphi$. On the other hand, an interpretation which we will find useful in the sequel is the ***total view*** interpretation, which makes the finest possible distinction among views. In the total view interpretation the state $\sigma(p_j, r, t)$ is defined to be $v(p_j, r, t)$ — p_j's view at (r,t).

Another popular state-based interpretation is one in which $\sigma(p_j, r, t)$ is defined to be p_j's internal state at (r,t) (recall that processors are state machines). Under this interpretation a processor might "forget" facts that it knows. In particular, if a processor can arrive at a given state by two different message histories, then, once in that state, the processor's knowledge cannot distinguish between these two "possible pasts". However, in the total view interpretation, a processor's state encodes all of the processor's previous states, and therefore processors do not forget what they know; if a processor knows φ at a point (r,t), then at all points (r,t') with $t' > t$ the processor will know that it once knew φ. Thus, while there may be temporary facts such as "it is 3 on my clock" which a processor will not know at 4 o'clock, it will know at 4 o'clock that it previously knew that it was 3 o'clock.

Once we have an assignment of states to processors, we can extend this assignment to groups G of processors by taking the state of G at a given point to consist of a description of the states of the members of G at that point. More formally, we define:

$$\sigma(G,r,t) \quad = \quad \{\langle p_i, \sigma(p_i,r,t)\rangle \ : \ p_i \in G\}.$$

It is natural to consider what happens if we ascribe "state-based" knowledge to G. The group G will be said to know all of the facts known to all of its members, as well as all of their consequences. Thus, the group's knowledge in this case corresponds to what we have called ***implicit knowledge*** in Section 1.2. Recall that we denote this knowledge by $I_G\varphi$, read *the group G has implicit knowledge of φ*. More formally, we say that $(\mathcal{I}_\sigma,r,t) \models I_G\varphi$ iff $(\mathcal{I}_\sigma,r',t') \models \varphi$ for all points (r',t') such that $\sigma(G,r,t) = \sigma(G,r',t')$. Thus, in the case of state-based interpretations of knowledge, implicit knowledge provides a modular way of defining the knowledge of elements of a system in terms of their components' knowledge.[4]

2.4 Discussion

This chapter has presented a general framework for modeling knowledge. An interesting property of our framework is that it uses a model of a distributed system as the (semantic) object relative to which knowledge is defined. Given that in the process of designing protocols for a distributed system, one usually starts out from a model of the system, it seems very natural to define knowledge in this way, rather than starting out from some abstract model theory for knowledge (e.g., [Hi],[Sa],[FHV],[HM]) and then having the burden of relating this model to the system. In fact, this idea is by now quite popular in the field (cf. [ChM],[FI],[HF],[PR]). However, work on abstract models of knowledge may often prove relevant to our interests. For example, it can be shown that there is a direct correspondence between state-based knowledge interpretations and a particular well-behaved kind of Kripke structures (cf. [HM]) or knowledge structures (cf. [FHV]). Epistemic interpretations and knowledge interpretations were deliberately defined in a very general way. In any particular application, a particular knowledge interpretation should be chosen. Indeed, the models for knowledge used in [ChM],[HF], and [PR], among others, immediately fit into our framework. We would argue that this framework is sufficiently general to accomodate any reasonable notion of knowledge.

[4] The knowledge ascribed to a set of processes by Chandy and Misra in [ChM] essentially corresponds to the implicit knowledge of its members, as defined here. See also [R].

Given a particular interpretation for knowledge in a particular system, it becomes possible to determine what the system's state of knowledge at any given point is. It is then possible, for instance, to specify attaining a specific state of knowledge as the goal of a particular communication protocol. For example, the goal of standard handshake protocols (cf. [Gr]) seems to be to guarantee that the sender of a piece of data will repeatedly attempt to send the data (and not discard the data) until it *knows* that the data was delivered to its destination. Indeed, Papageorgiou (cf. [P]) makes essential use of such specifications, and it is not clear how one would otherwise state the goals of the communication protocols he designs. Thus, it appears that formalisms based on knowledge may prove to be a powerful tool for specifying and verifying protocols. Furthermore, it seems quite reasonable that such a formalism will readily applicable to the synthesis of protocols and plans. (Temporal logic has already proved somewhat successful in this regard; cf. [CE], [MW].) Halpern and Fagin in [HF] take this idea one step further. They suggest a notion of of a *knowledge-based protocol* in which the actions performed by a processor are a function of the facts *known* to the processor. Since a processor's knowledge is determined by the processor's view, and since in normal protocols a processor's actions are a function of the processor's view, it turns out that in any fixed system an implementable knowledge-based protocol is equivalent to a normal protocol. However, a knowledge-based description of a protocol seems to communicate the protocol at a much higher level. In some cases it seems to focus attention to the basic structure and essential ingredients of the protocol, making it easier to communicate and to port from one system to another. We will find knowledge-based protocols useful for our analysis in chapters 5 and 6.

Chapter 3

Attaining Common Knowledge

As we have seen in the introduction, common knowledge seems to play an important role in certain situations, e.g., in everyday agreements between people and in coordinating actions. In this chapter we address the problem of attaining common knowledge in a distributed environment. Sections 3.1 shows that when communication is not guaranteed no amount of successful communication can bring about common knowledge. Section 3.2 goes on to show that, formally speaking, common knowledge cannot be attained at all in practical distributed systems. Section 3.3 discusses the implications of the results of Section 3.2, and Section 3.4 presents a sense in which something very similar to common knowledge can in fact be attained in practical systems.

3.1 Unreliable communication

Following the coordinated attack example, we first consider systems in which communication is not guaranteed. Intuitively, communication is not guaranteed in a system S if messages sent in S might fail to be delivered in an arbitrary fashion, independent of any other event in the system. Making this intuition precise is somewhat cumbersome (cf. [HF]), and we will not attempt to do so here. For our purposes, a weaker condition, which must be satisfied by any reasonable definition of the notion "communication is not guaranteed", will suffice. Roughly speaking, if communication in the system is not guaranteed then it must always be possible that all messages past a certain point will not be delivered.

A run r' is said to *extend* a point (r,t) if the histories of (r,t) and of (r',t) are identical. More formally, r' extends (r,t) if all processors have the same initial times, initial states in both r and r', and all processors' message history functions and clock time functions up to (but not including) time t are identical in r and in r'. Given a system S, we say that *communication in S is not guaranteed* if the following condition holds:

(*) For every run $r \in S$, real time t, processor p_i, and set M of messages, there is a run $r' \in S$ that extends (r,t), such that at (r',t) processor p_i does not receive any message from the set M and p_i does receive all the messages $m \notin M$ that it receives at (r,t), all processors $p_j \neq p_i$ receive exactly the same messages at (r',t) as they do at (r,t), and no messages are delivered in r' after time t.

Thus, (r,t') and (r',t') are identical for all $t' < t$, and at time t all processors $p_j \neq p_i$ receive the same messages in r and r', while processor p_i might miss some in r' that it receives in r (namely, those in M). The coordinated attack problem suggests that when communication is not guaranteed, common knowledge is not attainable. In fact, the following lemma shows that in such a system, communicated messages play no role in determining what facts are common knowledge and when facts become common knowledge.

Lemma 3.1: Let S be a system in which communication is not guaranteed, and let $\mathcal{I}$ be a knowledge interpretation for S. Let $r_1, r \in S$ be runs such that r extends (r_1,t_1), and let $t \geq t_1$. Then for all formulas φ it is the case that $(\mathcal{I},r_1,t) \models C\varphi$ iff $(\mathcal{I},r,t) \models C\varphi$.

Proof: Fix φ. Denote by r^- the run extending (r_1,t_1) in which no messages are delivered after time t_1. Notice that it follows from (*) that $r^- \in S$. We will now show that for all runs r extending (r_1,t_1), it is the case that $(\mathcal{I},r,t) \models C\varphi$ iff $(\mathcal{I},r^-,t) \models C\varphi$. The proof is by induction on the number $n(r)$ of messages delivered in r in the interval $[t_1,t)$. The case $n(r) = 0$ is trivial, because all processors have identical views at (r,t) and at (r^-,t). Assume inductively that the claim holds for all runs $r' \in S$ extending (r_1,t_1) with $n(r') \leq k$, and assume that $n(r) = k+1$. Let $t' < t$ be the latest time in which a message is delivered in r before time t. Let p_i and m be such that m is delivered to processor p_i at time t' in r. Since communication in the system is not guaranteed, there is a run $r' \in S$ extending (r,t') such that all processors $p_j \neq p_i$ receive the same messages at (r',t') and at (r,t'), processor p_i does not reccive m at (r',t'), and no messages are delivered in r' after t'. Since only k messages are delivered in r' between t_1 and t', by assumption $(\mathcal{I},r^-,t) \models C\varphi$ iff $(\mathcal{I},r',t) \models C\varphi$. Let $p_j \neq p_i$ be another processor. By Lemma 2.1, p_j supports $C\varphi$ at (r',t) iff $(\mathcal{I},r',t) \models C\varphi$. However, p_j's view at (r',t) is identical to its view at (r,t). Therefore, p_j supports $C\varphi$ at (r,t) iff p_j supports $C\varphi$ at (r',t). Again by Lemma 2.1 it follows that $(\mathcal{I},r^-,t) \models C\varphi$ iff $(\mathcal{I},r,t) \models C\varphi$, which concludes the proof of the inductive step. The claim follows by induction. ⋈

We say that a formula ψ is *undetermined* in a system S at a given point (r,t) if it is possible at that point that ψ will never hold. More formally, ψ is undetermined in S at (r,t) if for some run $r' \in S$ extending (r,t), it is the case that for no $t' \geq t$ does ψ hold at (r',t'). A formula φ is said to be *determined* in S at (r,t) if it is not undetermined. As an easy corollary to Lemma 3.1, we have:

Theorem 3.2: Let S be a system in which communication is not guaranteed such that $C\varphi$ is undetermined in S at (r_1, t_1). If $r \in S$ extends (r_1, t_1) and $t \geq t_1$, then $C\varphi$ does not hold at (r, t).

Proof: From the fact that $C\varphi$ is undetermined in S at (r_1, t_1) it follows that there is a run $r' \in S$ extending (r_1, t_1) such that $C\varphi$ does not hold at (r', t') for any $t' \geq t_1$. Let $r \in S$ be a run that extends (r_1, t_1), and fix $t \geq t_1$. By Lemma 3.1, $C\varphi$ holds at (r', t) iff it holds at (r_1, t) iff it holds at (r, t). Thus, $C\varphi$ does not hold at (r, t). Since r and $t \geq t_1$ were chosen arbitrarily, the theorem follows. ⋈

The proofs of Lemma 3.1 and Theorem 3.2 apply to weaker conditions than communication not being guaranteed. A system S is said to be a system with *unbounded message delivery times* if the following condition holds:

(**) For all runs $r \in S$, real times t and t', processors p_i, and sets M of messages, there is a run $r' \in S$ that extends (r, t) such that at (r', t) processor p_i receives all and only the messages $m \notin M$ that it receives at (r, t), and all processors $p_j \neq p_i$ receive exactly the same messages at (r', t) as they do at (r, t), and no messages are delivered in r' after time t and before time t'.

Asynchronous systems are often defined to be systems with unbounded message delivery times. Intuitively, condition (**) says that it is always possible for no messages to be delivered for arbitrarily long periods of time, whereas (*) says that it is always possible for no message to be delivered at all from some time on. In some sense, we can view (*) as a limit case of (**). Not surprisingly, it is easy to check that we can replace (*) by (**) in the proofs of Lemma 3.1 and Theorem 3.2, so we also get:

Corollary 3.3: If S is a system with unbounded message delivery times and $C\varphi$ is undetermined in S at (r_1, t_1), then in no run $r \in S$ extending (r_1, t_1) does $C\varphi$ ever hold at a time $t \geq t_1$. ⋈

Returning to the coordinated attack problem, we are now in a position to relate the generals' problem to the problem of attaining common knowledge, and present a simple formal proof of the impossibility of their agreeing to attack. We do this as follows: The description of the coordinated attack problem in Section 1.3 describes a specific state of affairs. Without loss of generality, we denote the real time in which the generals are in this situation by t_1. Formally, we consider the generals as processors and their messengers as communication links between them. The generals are assumed to each behave according to some predetermined deterministic protocol; i.e., a general's actions (what messages it sends and whether it attacks) at a given point are a deterministic function of his message history and the time on his clock. In particular, we assume that the generals are following a joint protocol (P, P'), where general A follows P and the general B follows P'. Thus, we identify the generals with a distributed system S. The runs of S are simply all possible runs of (P, P') from t_1 on.

Proposition 3.4: In the coordinated attack problem, any protocol that guarantees that if either party attacks then they both attack simultaneously, is a protocol in which necessarily neither party attacks (!).

Proof: Let (P, P') be a joint protocol for the generals, and assume that it guarantees that no general will attack alone. Since the generals are said not to initially have plans to attack the enemy, we can also assume that (P, P') is such that no general will attack in the absence of any successful communication. Let t_1 and S be defined as above. For $p_i \in \{A, B\}$, $r \in S$, and $t \geq t_1$, define $\sigma(p_i, r, t)$ by:

$$\sigma(p_i, r, t) = \begin{cases} \mathit{attacking} & \text{if } p_i \text{ has started attacking by } (r,t); \\ \mathit{not\ attacking} & \text{otherwise.} \end{cases}$$

σ is clearly defined as a function of a general's view. Let $\mathcal{I}_\sigma$ be the state-based knowledge interpretation relative to S corresponding to σ. The system S is clearly one in which communication is not guaranteed, since it is always possible that no messenger will succeed in delivering any message from some point on. In the run $r^- \in S$ in which no messages are successfully delivered, no general will attack. Consider the fact $\psi =$"both generals are attacking". Since for all $r \in S$ it the case that r^- extends (r, t_1), it follows that ψ is undetermined at (r, t_1) for all $r \in S$. Because $C\psi \supset \psi$, the fact that ψ is undetermined at (r, t_1) for all $r \in S$ implies that $C\psi$ is undetermined at (r, t_1) for all $r \in S$. Since (P, P') guarantees that whenever one division attacks both attack simultaneously, it follows that at all points $(r, t) \in S \times [t_1, \infty)$ it is the case that both generals are ascribed the same state, i.e., $\sigma(A, r, t) = \sigma(B, r, t)$. Consequently, all of the points reachable from a point (r, t) in which the generals are in an *attacking* state have the property that both generals are in an *attacking* state. From Section 2.3 we thus have that $C\psi$ holds when the generals attack. Because communication is not guaranteed in S and $C\psi$ is undetermined at (r, t_1), Theorem 3.2 implies that $C\psi$ does not hold at (r, t). It follows that the generals can never attack! ⋈

The requirement of a simultaneous attack in the coordinated attack problem is a very strong one. It seems that real life generals do not need a protocol that guarantees such a strong condition, and can probably make do with one that guarantees a non-simultaneous attack. Theorem 3.2 does not imply that a protocol for achieving this does not exist. However, in Section 4.2 we will use a variant of this argument to show that no protocol can even guarantee that if one party attacks then the other will *eventually* attack! We might also consider weakening the condition to one where if one party attacks, then with high likelihood the other will attack. This in fact is achievable. We discuss it in further detail in Section 4.4.

3.2 Reliable communication

The previous results show that, in a strong sense, common knowledge is not attainable in a system in which communication is not guaranteed. However, even when communication *is* guaranteed, common knowledge can be elusive. To see this, consider a system consisting of two processors, R2 and D2, connected by a communication link. R2 and D2 use a common (global) clock, communication (delivery) is guaranteed, and furthermore, it is (say commonly) known that any message sent from R2 to D2 reaches D2 either immediately or after exactly ϵ seconds. At time t_S, R2 sends D2 a message m that does not contain a time stamp, i.e., does not mention t_S in any way. The message m is delivered to D2 at time t_D. Let $sent(m)$ be the fact "the message m has been sent". D2 doesn't know $sent(m)$ initially. How does $\{R2, D2\}$'s state of knowledge of $sent(m)$ change with time?

At time t_D, D2 knows $sent(m)$. Because it might have taken ϵ time units for m to be delivered, R2 cannot be sure that D2 knows $sent(m)$ before $t_S + \epsilon$. D2 knows that R2 will not know that D2 knows $sent(m)$ before $t_S + \epsilon$, and because for all D2 knows m may have been delivered immediately (in which case $t_S = t_D$), D2 does not know that R2 knows that D2 knows $sent(m)$ before $t_D + \epsilon$. Now, R2 must wait until $t_S + 2\epsilon$ before he knows that $t_D + \epsilon$ has passed. This line of reasoning can be continued indefinitely, and an easy proof by induction shows that before time $t_S + n\epsilon$, the formula $(K_R K_D)^n sent(m)$ does not hold, while at $t_S + n\epsilon$ it does hold. Thus, it "costs" ϵ time units to acquire every level of "R2 knows that D2 knows". Recall that $C(sent(m))$ implies $(K_R K_D)^n sent(m)$ for every n. It follows that $C(sent(m))$ will never be attained!

Now consider what would happen if R2 sends D2 the following message m':

> "This message is being sent at time t_S (and will reach D2 by $t_S + \epsilon$ at the latest); m."

Since they are using a common clock, the fact that R2 sent m' to D2 would become common knowledge at time $t_S + \epsilon$!

What is the essential difference between these two situations? It seems that what made achieving common knowledge easy in the latter case was the possibility of simultaneously making the transition from not having common knowledge to having common knowledge. The impossibility of doing so in the former case was the driving force behind the extra cost in time incurred in attaining extra levels of knowledge. In fact, Lemma 2.1 already implies that when $C\varphi$ first holds all processors must come to support $C\varphi$ simultaneously. In particular, this means that all of the processors' views must change simultaneously. However, there is a sense in which practical systems cannot guarantee such simultaneity. Intuitively, the uncertainties regarding relative readings of clocks and message transmission times in practical systems imply that there will always be two processors p_i and p_j and two runs such that p_i's behavior in both runs is identical, and in one of the runs p_j performs all actions δ time units later than in the first, for some $\delta \neq 0$. (This δ may, in some cases, be very small.) We now make this claim precise.

Given a system S and a knowledge interpretation $\mathcal{I}$, we say that *common knowledge is attainable* in S w.r.t. $\mathcal{I}$ if there is a (joint) protocol $\bar{P} = (P_1, \ldots, P_n)$ executed in S and a fact φ such that in all the runs of S in which the processors follow $\bar{P}$ it is the case that φ is not common knowledge before any processor "wakes up" (i.e., $(\mathcal{I}, r, t) \models \neg C\varphi$ for all times $t < \min_j t_0(p_j, r)$), and $C\varphi$ holds at some later point in the run. Thus, roughly speaking, $\bar{P}$ "makes φ common knowledge".

A system S is said to have *essential temporal imprecision* if for all (joint) protocols $\bar{P} = (P_1, \ldots, P_n)$ executed in S, there exist runs $r_1, r_2 \in S$ in which $\bar{P}$ is executed, processors p_i,p_j, and a real number $\delta \neq 0$ such that for all t it is the case that $v(p_i, r_1, t) = v(p_i, r_2, t)$ and $v(p_j, r_1, t) = v(p_j, r_2, t + \delta)$.

Dolev, Halpern, and Strong show in [DHS] that a system in which (i) there is an uncertainty regarding the relative "initial times" in which the processors start running, and (ii) there are upper and lower bounds on message transmission times along the links in the system, with the upper bounds strictly larger than the lower bounds, is a system with inherent temporal imprecision (even if the processors' clocks are guaranteed to run at the same rate!). It can thus be argued that all practical distributed systems have essential temporal imprecision.

As an easy consequence of the definitions, we now have:

Theorem 3.5: If S is a system with essential temporal imprecision, then common knowledge is not attainable in S.

Proof: Assume the contrary, and let $\bar{P} = (P_1, \ldots, P_n)$ be a protocol executed in S and let φ be a fact such that before $C\varphi$ does not initially hold in the runs of $\bar{P}$ and such that $C\varphi$ does hold at some point in all runs $r \in S$ in which the processors execute $\bar{P}$. Given that there is essential temporal imprecision in S, let p_i, p_j, δ,

r_1 and r_2 be as in the definition above. In particular, the processors follow $\bar{P}$ in r_1 and in r_2. Lemma 2.1 implies that in all runs of $\bar{P}$, processors p_i and p_j start supporting $C\varphi$ simultaneously (i.e., at the same real time). Let t' be the (real) time in which p_i and p_j first support $C\varphi$ in r_1. In particular, p_j does not support $C\varphi$ at any earlier time in r_1. Since $v(p_i, r_1, t') = v(p_i, r_2, t')$ we have that p_i first starts supporting $C\varphi$ at time t' in r_2 as well. However, since $v(p_j, r_1, t') = v(p_j, r_2, t' + \delta)$, processor p_j does not support $C\varphi$ before $(r_2, t' + \delta)$. It follows that p_i and p_j do not start supporting $C\varphi$ at the same time in r_2, contradicting Lemma 2.1. $\bowtie$

In fact, a stronger notion of imprecision holds in practice: for *all* runs of the system there is another run in which p_i's behavior is identical to the current run, and p_j's behavior is shifted by δ. More formally, we say that ***a system S has temporal imprecision in all runs*** if for all runs $r \in S$ there exists a run $r' \in S$, processors p_i and p_j and real number $\delta \neq 0$ such that for all times t it is the case that $v(p_i, r, t) = v(p_i, r', t)$ and $v(p_j, r, t) = v(p_j, r', t + \delta)$. (Although, again, the δ's involved may be very small.) A proof similar to that of Theorem 3.5 now shows:

Theorem 3.6: If S is a system with temporal imprecision in all runs, $\mathcal{I}$ is a knowledge interpretation for S and $(\mathcal{I}, r, t) \models \neg C\varphi$, then $(\mathcal{I}, r, t') \models \neg C\varphi$ for all $t' \geq t$. $\bowtie$

Theorems 3.5 and 3.6 imply that, strictly speaking, common knowledge cannot be attained in practical distributed systems! In such systems, we have the following situation: a fact φ can be known to a processor without being common knowledge, or it can be common knowledge (in which case that processor also knows φ), but due to (possibly negligible) imperfections in the system's state of synchronization and its communication medium, there is no way of getting from the first situation to the second!

Observe that we can now show that, formally speaking, even people cannot attain common knowledge of any new fact! Consider the father publicly announcing **m** to the children in the muddy children puzzle. Even if we assume that it is common knowledge that the children all hear whatever the father says and understand it, there remains some uncertainty as to exactly when the children each come to know (or comprehend) the father's statement. Thus, it is easy to see that the children do not immediately have common knowledge of the father's announcement. Furthermore, for similar reasons the father's statement can never become common knowledge.

3.3 A paradox?

There is a close correspondence between agreements, coordinated actions, and common knowledge. We have shown that in a precise sense, reaching agreements and coordinating actions in a distributed system amount to attaining common knowledge of certain facts. We also proved that common knowledge cannot be attained in practical distributed systems! However, it is well known that operations such as reaching agreement and coordinating actions are routinely performed in many actual distributed systems. And it might seem as if the designers of such systems do not find it necessary to worry about common knowledge.

Where is the catch? How can we explain this apparent discrepancy between our formal treatment and practical experience? It seems that our insistence on defining knowledge in such a way that facts that are known must be true at the same instant in ***absolute time*** is at the root of the problem. After all, absolute time often does not seem to be the relevant notion of time in many distributed applications. But does it make any sense to define knowledge in a distributed system in any other fashion? We consider this a major open question. We will touch upon it in the next chapter when we discuss knowledge of facts relative to a relativistic notion of time. Another weakness of the impossibility result is the fact that it relies on a very fine-grained view of practical systems, which forces us to conclude that simultaneity is not attainable. This is similar to the claim that actual bits in a computer do not exclusively contain the values 0 or 1, but can sometimes be in an undefined or incoherent intermediate state. While true, this fact does not seem to have an overwhelming effect on many aspects of computing. It can be taken care of on the hardware level, and for all practical purposes software designers can successfully use a model of the machine in which bits do in fact attain only the values 0 and 1. Similarly, slight abstractions of practical systems that do guarantee simultaneity in many cases model the actual system very well, in which case we may identify the system with its abstraction for all practical purposes. The next section presents a formal argument with a similar flavor.

3.4 Internal knowledge consistency

Strictly speaking, common knowledge is not attainable in practical systems because such systems cannot guarantee to perform actions at different sites absolutely simultaneously. However, the operations performed in many practical distributed systems do not require events at different sites to be simultaneous. Indeed, it is often the case that from within the system it is not possible to determine the exact relative (real time) difference between events that occur at different sites. We say that a run of the system is *δ-insensitive* if given the information available to the processors, the (real) time difference between any two events that happen at different sites of the system can be determined with precision no better than δ (cf. [HMM]). A system S is said to be δ-insensitive if all of its runs are. In a δ-insensitive system, it is not possible to distinguish from within the system between the clocks being perfectly synchronized and the clocks being synchronized to within δ. Thus, it seems that when the clocks in such a system are sufficiently synchronized, treating them as if they were perfectly synchronized should have no noticeable negative effect on the system. The processors' views of the system would be perfectly consistent.

We define an epistemic interpretation $\mathcal{I}$ to be a ***pseudo-knowledge*** interpretation for S if for all runs $r \in S$ there is a run $r' \in S$ such that all processors' views in r are identical to their views in r', and $\mathcal{I}$ is knowledge consistent with r'. (More formally, for all $r \in S$ there is a run $r' \in S$ such that for all processors p_j and times $t \geq t_0(p_j, r)$ there is a time $t' \geq t_0(p_j, r')$ such that $v(p_j, r, t) = v(p_j, r', t')$.) Notice that a knowledge interpretation for S is in particular also a pseudo-knowledge interpretation for S. The converse is not true. However, from within the system it is impossible to determine whether or not a pseudo knowledge interpretation is in fact a knowledge interpretation. Therefore, basing actions on "pseudo-knowledge" is as good as basing them on true knowledge.

Our discussion above can be formally stated as:

Proposition 3.7: Let S be a set of runs such that in every run $r \in S$ clocks are synchronized to within the insensitivity of r (i.e., for some δ, r is δ-insensitive and clocks are synchronized to within δ throughout r). Let $\mathcal{I}$ be an epistemic interpretation for S. Let $S_0 \subseteq S$ be the set of runs of S in which all clocks are perfectly synchronized. If $\mathcal{I}$ is a knowledge interpretation for S_0, then $\mathcal{I}$ is a pseudo-knowledge interpretation for S. $\bowtie$

Since pseudo-knowledge cannot be distinguished from "real" knowledge, Proposition 3.7 shows us that if clocks are synchronized to within the insensitivity of the system, then we might as well work under the assumption that all clocks are perfectly synchronized. Thus, the reasoning is essentially carried out in a simpler abstract model of the system, in which the processors have a global clock. The Proposition shows that such reasoning will be ***internally knowledge consistent***: every fact that a processor "knows" will be consistent with the processor's view of the system.

Consider the children hearing their father announce $\mathbf{m}$. As we have argued in Section 3.2, they do not truly attain common knowledge of $\mathbf{m}$. However, since the uncertainty regarding when they comprehend the father's statement is very small, to the extent that they practically cannot tell whether or not they in fact came to know $\mathbf{m}$ simultaneously, they can be said to attain pseudo-common knowledge of $\mathbf{m}$. Working under the assumption that they have common knowledge of this fact will never fail them. This explains how people can routinely rely on the assumption that they attain common knowledge of a multitude of facts without suffering any negative consequences.

Chapter 4

Related states of knowledge

Lemma 2.1 and the proof of Proposition 3.4 show that the state of common knowledge is closely related to events that are guaranteed to occur simultaneously at different sites of the system. Analyzing the attainability of common knowledge in various systems can be used to study the ability to perform various simultaneous coordinated actions. In fact, Chapter 6 uses such an analysis to study the design of fault-tolerant protocols for simultaneous actions in systems of unreliable processors. However, most coordinated actions that are carried out in a distributed system are not required to take place simultaneously. This chapter considers states of knowledge related to common knowledge that similarly correspond to other levels of coordination of actions in a distributed system. These are states of knowledge that are much more easily attainable than common knowledge in many systems of interest. Section 4.1 starts out by reconsidering the state of common knowledge under state-based interpretation, providing us with the necessary machinery for defining many related states of knowledge. Section 4.2 studies the states of knowledge that arise from broadcasting a message using synchronous and asynchronous channels, and relates them to coordinated actions that are not guaranteed to be performed at all sites simultaneously. Section 4.3 considers the effect of defining knowledge relative to a relativistic notion of time, and compares the states of knowledge arising there with those that are defined relative to absolute time. Finally, Section 4.4 looks at the states of knowledge arising when actions are only likely to be coordinated, capturing among other things the state of knowledge that actually arises in the coordinated attack problem.

4.1 Common knowledge revisited

Throughout this chapter we will restrict our attention to state-based interpretations of knowledge, since they seem to be the most appropriate for the kinds of applications we will be interested in. It is useful to start by reconsidering the notion of common knowledge under such interpretations, from a slightly different point of view. Recall the children's state of knowledge of the fact $\mathbf{m}$ in the dirty children puzzle. If we assume that it is common knowledge that all children comprehend $\mathbf{m}$ simultaneously, then after the father announces $\mathbf{m}$, the children attain $C\mathbf{m}$. However, when they attain $C\mathbf{m}$ it is not the case that the children learn an infinite collection of facts of the form $E^n\mathbf{m}$ separately. Rather, after the father speaks, the children are in a state of knowledge Ψ characterized by the fact that Ψ implies that $\mathbf{m}$ holds and that every child knows that Ψ holds. Thus, Ψ satisfies the equation

$$\Psi \equiv \mathbf{m} \wedge E\Psi.$$

The following Theorem shows that this phenomenon arises in quite the same way in state-based knowledge interpretations:

Theorem 4.1: Let $\mathcal{I}_\sigma$ be a state-based interpretation for S. Then for all runs $r \in S$ and times t it is the case that

$$(\mathcal{I}_\sigma, r, t) \models \; C_G\varphi \equiv \varphi \wedge E_G C_G\varphi.$$

Proof: The formulas $C_G\varphi \supset \varphi$ and $EC_G\varphi \supset C_G\varphi$ are clearly valid for all knowledge interpretations. It remains to show that $C_G\varphi \supset E_G C_G\varphi$ is valid. Since $\mathcal{I}_\sigma$ is a state-based knowledge interpretation, we have from Section 2.3 that φ is common knowledge at a given point (r,t) iff φ holds in all points that are reachable from (r,t). If $\sigma(i,r,t) = \sigma(i,r',t')$ for some $p_i \in G$ then (r',t') is clearly reachable from (r,t). Furthermore, because reachability is an equivalence relation in our case, any point reachable from (r,t) is clearly also reachable from (r',t'). Since we have from Section 2.3 that $(\mathcal{I}_\sigma, r, t) \models E_G\psi$ iff $(\mathcal{I}_\sigma, r', t') \models \psi$ for all (r',t') such that $\sigma(i,r,t) = \sigma(i,r',t')$ for some $p_i \in G$, the result directly follows. ⋈

Thus, under a state-based interpretation, $C_G\varphi$ is a "fixpoint" of the E_G operator, in which φ holds. An equivalent definition for $C_G\varphi$ in the case of a state-based interpretation $\mathcal{I}_\sigma$ is as the *weakest* solution for X in the equation

$$X \equiv \varphi \wedge E_G X,$$

by which we mean that any solution for X in this equation implies $C_G\varphi$. (Notice that the above equation may have many solutions. For example, both **false** and $C_G(\varphi \wedge \psi)$ solve it.) $C_G\varphi$ is what is called the *greatest fixpoint* of this equation. As our discussion of common knowledge in the case of the muddy children puzzle suggests, expressing common knowledge as a greatest fixpoint of such an equation seems to correspond more closely to the way it actually arises. While it is beyond the scope of this thesis to give a detailed formal semantic definition of a logic with fixpoint definitions, we sketch a semantics for a propositional state-based logic of knowledge with fixpoints in Appendix B.

In the case of state-based knowledge interpretations, the following axioms are valid for common knowledge (for logical systems with these and other axioms, see [Mi], [Le], [HM], and the analogous axioms for the PDL "*" operator in [KP]):

(1) The "fixpoint" axiom:

$$C_G\varphi \supset \varphi \wedge E_G C_G\varphi.$$

(2) The "induction" axiom:

$$C_G(\varphi \supset E_G\varphi) \supset (\varphi \supset C_G\varphi).$$

(3) The "consequence closure" axiom:

$$(C_G\varphi \wedge C_G(\varphi \supset \psi)) \supset C_G\psi.$$

The induction axiom states that if it is common knowledge that whenever φ holds everybody knows φ, then when φ holds, φ is common knowledge. It is called an induction axiom because from the antecedent $C(\varphi \supset E\varphi)$ we can prove by induction that $\varphi \supset E^n\varphi$ holds for all n. Roughly speaking, it traces our line of reasoning when we argued that the children in the muddy children puzzle attain common knowledge of the father's statement.

Another interesting property of common knowledge under state-based interpretations is:

$$\neg C_G\varphi \supset C_G\neg C_G\varphi.$$

Notice that it is possible to deduce this fact from Lemma 2.1 using the induction axiom. Axiomɛ (1)–(3) completely characterize common knowledge in the logical systems of [Le], [HM].

4.2 ϵ-common knowledge and $\Diamond$-common knowledge

Since attaining common knowledge in practical distributed systems is problematic, it is natural to ask what states of knowledge *can* be obtained by the communication process. In this section we consider what states of knowledge are attained in systems in which communication delivery is guaranteed but message transmission times are uncertain. However, before we can do so, we need to extend our language to allow reasoning about time. We introduce temporal operators to the language by adding the following clause to the inductive definition of formulas in the language (cf. Section 2.2): If φ is a formula and δ is a real number, then $\Diamond\varphi$ and $\bigcirc^{\delta}\varphi$ are formulas. Roughly speaking, $\Diamond\varphi$ stands for "eventually φ", while $\bigcirc^{\delta}\varphi$ stands for "δ time units from now φ". We then extend our semantic definitions so that $(\mathcal{I}, r, t) \models \Diamond\varphi$ iff for some $t' \geq t$ it is the case that $(\mathcal{I}, r, t') \models \varphi$; and $(\mathcal{I}, r, t) \models \bigcirc^{\delta}\varphi$ iff $(\mathcal{I}, r, t+\delta) \models \varphi$. (This definition corresponds to *linear time* semantics for temporal logic; cf. [MP].) It is customary to define $\Box\varphi$ (read "always φ") as the dual of $\Diamond\varphi$, i.e., $\Box p \overset{\text{def}}{=} \neg\Diamond\neg\varphi$.

In dealing with distributed systems in which communication is not instantaneous, we are not so interested in facts whose truth value might change between the time a message regarding them is sent and when it is received. A fact φ is called ***stable*** if it has the property that once true it remains true forever. More formally, φ is stable if $\varphi \supset \Box\varphi$ is valid. Notice that given any fact ψ, the following facts are stable: "ψ held at some point in the past", "ψ holds and will hold throughout the future", "ψ holds at time t on p_i's clock", and "ψ holds throughout time interval Δ". (For the relevance of stable facts to distributed systems, see also [CL].) Moreover, if ψ is stable then so are $\Diamond\psi$ and $\bigcirc^{\delta}\psi$. However, it is not in general the case that knowledge of stable facts is stable. For example, given a general state-based knowledge interpretation $\mathcal{I}_\sigma$ and a stable fact ψ, it is possible $(\mathcal{I}, r, t) \models K_i\psi$ and $(\mathcal{I}, r, t') \not\models K_i\psi$ for some $t' > t$. This does not happen in the total view interpretation, since in this case a fact is stable iff all processors know that it is stable, and processors do not forget what facts they knew. Thus, in particular, they don't forget stable facts. Under the total view interpretation, if φ is stable, then so are $K_i\varphi$, $E\varphi$, $C\varphi$, etc. For the remainder of this section we will restrict our attention to the total view interpretation.

We begin by considering ***synchronous broadcast*** channels of communication: every message sent is delivered to all processors in the system (including the sender), and processors receive the message up to ϵ units of real time apart. ϵ is called the ***broadcast spread*** of such a channel. Recall that the properties of the system are common knowledge to all the processors in the system. In particular, the properties of the broadcast channel are common knowledge. Let us now consider the state of knowledge of the system (under the total view interpretation) when a processor p_i receives a broadcast message m; p_i knows $sent(m)$, but he knows more: p_i also knows that within ϵ time units everyone will (receive m and) know $sent(m)$. But he knows even more: p_i also knows that within ϵ everyone will know $sent(m)$ and they will all know that within another ϵ everyone will know $sent(m)$. This argument can be continued, and it leads us to the notion of *ϵ-common knowledge*, denoted C^{ϵ}.

$C^{\epsilon}\varphi$ is defined to be the greatest fixpoint of the equation:

$$X \equiv \varphi \wedge \bigcirc^{\epsilon} EX.$$

Again, we refer the reader to Appendix B for a rigorous definition. However, as the above discussion suggests, $C^{\epsilon}\varphi$ can also be equivalently expressed by the following infinite conjunction:

$$C^{\epsilon}\varphi \equiv p \wedge \bigcirc^{\epsilon} Ep \wedge \bigcirc^{\epsilon} E(\bigcirc^{\epsilon} Ep) \wedge \cdots \wedge (\bigcirc^{\epsilon} E)^n p \wedge \cdots.$$

For any message m delivered in an ϵ-spread synchronous broadcast channel, $sent(m)$ becomes ϵ-common knowledge ***as soon as m is delivered to any processor.*** Returning to R2 and D2's communication problem, we can view them as a synchronous broadcast system, and indeed they attain $C^{\epsilon}sent(m)$ immediately when R2 sends the message m. The interested reader is invited to check that R2 and D2 in fact achieve $\epsilon/2$-common knowledge of $sent(m)$ at time $t_S + \epsilon/2$.

In a system in which all clocks run at the rate of real time, if a processor knows that φ will hold "tomorrow", for an arbitrary fact φ, then (under the total view interpretation) "tomorrow" the processor knows that φ holds, i.e., the processors' knowledge satisfies the axiom $K_i \bigcirc \varphi \supset \bigcirc K_i\varphi$. (This is also the case in the logical systems of [Sa] and [Le].) Similar statements hold for $\bigcirc^{\epsilon}$ and E. In this case $(\bigcirc^{\epsilon} E)^k\varphi \supset \bigcirc^{k\epsilon} E^k\varphi$, and it follows that $C^{\epsilon}\varphi \supset \bigcirc^{k\epsilon} E^k\varphi$. So, if $C^{\epsilon}\varphi$ holds in such a system, then $E^k\varphi$ will eventually hold (within $k\epsilon$ time units, to be precise).

It is now interesting to compare common knowledge and ϵ-common knowledge. Whereas $C\varphi$ is a static state of knowledge, which can be true of a point in time irrespective of its past or future, $C^{\epsilon}\varphi$ is a notion that is essentially temporal. Whether or not it holds depends on what processors will know within ϵ, within 2ϵ, etc. In fact, since $C^{\epsilon}\varphi \supset \varphi$ and $C^{\epsilon}\varphi \supset \bigcirc^{\epsilon} C^{\epsilon}\varphi$, if $C^{\epsilon}\varphi$ holds, then φ (as well as $C^{\epsilon}\varphi$) must hold $n\epsilon$ time units from now, for all $n \geq 0$. Note that if φ is a stable fact then $C\varphi \supset C^{\epsilon}\varphi$; as we have shown, there are cases where a stable fact becomes ϵ-common knowledge and cannot become common knowledge (e.g., R2 and D2 attain $C^{\epsilon}sent(m)$, and cannot attain $C(sent(m))$). Thus, for stable facts, common knowledge is strictly stronger than ϵ-common knowledge The axioms (1)–(3) of Section 4.1 remain true of $C^{\epsilon}\varphi$, when we replace E by $\bigcirc^{\epsilon}E$ and C by C^{ϵ}. Furthermore, both C and C^{ϵ} are *conjunctive*, i.e., the formulas $C\varphi \wedge C\psi \supset C(\varphi \wedge \psi)$ and $C^{\epsilon}\varphi \wedge C^{\epsilon}\psi \supset C^{\epsilon}(\varphi \wedge \psi)$ are valid. (Thus, in particular, the uncertainty about information sent in separate messages in a synchronous broadcast channel need not be any worse than the broadcast spread.) However, whereas the formula $\neg C\varphi \supset C\neg C\varphi$ is valid for common knowledge, its analogue, $\neg C^{\epsilon}\varphi \supset C^{\epsilon}\neg C^{\epsilon}\varphi$ is *not* valid.

Just as common knowledge is closely related to simultaneous actions in a distributed system, ϵ-common knowledge is closely related to actions that are guaranteed to be performed within ϵ time units of one another. E.g., in an "early stopping" protocol for Byzantine agreement (cf. [DRS]), all members of G are guaranteed to decide on a common value within ϵ time units of each other. It follows that once the first processor decides, the decision value is ϵ-common knowledge in G.

Our discussion of knowledge in a distributed system is motivated by the fact that we can view processors' actions as being based on their knowledge. Consider an *"eager"* epistemic interpretation $\mathcal{I}$ under which a processor that receives an ϵ-broadcast message m immediately supports $C(sent(m))$. Clearly, $\mathcal{I}$ is not a knowledge interpretation, because it is not knowledge consistent (a processor might be said to "know" that another processor knows $sent(m)$, when in fact the other processor does not!). However, once the last processor receives m, which happens at most ϵ time units after the first processor starts supporting $C(sent(m))$, it is easy to see that $C(sent(m))$ does indeed hold! In a sense, Lemma 2.1 says that attaining common knowledge requires a certain kind of "natural birth"; it is not possible to attain it consistently unless simultaneity is attainable. But if one is willing to give up knowledge consistency (i.e., abandon the $K_i\varphi \supset \varphi$ axiom) for short intervals of time, something very similar to common knowledge can be attained.

The period of up to ϵ time units in which the processors' "knowledge" is inconsistent might have many negative consequences. If the processors need to act based on whether $C(sent(m))$ holds during that interval, they might not act in an appropriately coordinated way. This is a familiar problem in the context of distributed database systems. Committing to a transaction roughly corresponds to joining an agreement that the transaction has taken place in the database. There, it is the case that different sites of the database commit to transactions at different times (although all within a small time interval). When a new transaction is being committed to there is a "window of vulnerability" during which different sites might project inconsistent views of the database. However, once all sites commit to the transaction, the view of the database that the sites project becomes consistent (at least until the next transaction).

Having discussed states of knowledge in synchronous broadcast channels, we now turn our attention to systems in which communication is *asynchronous*: no bound on the delivery times of messages in the system exists. Consider the state of knowledge of *sent*(m) in a system in which m is broadcast over an ***asynchronous channel***: A channel that guarantees that every message broadcast will ***eventually*** reach every processor. Upon receiving m, a processor knows *sent*(m), and knows that eventually every other processor will receive m and know *sent*(m), and that eventually every other processor will receive m and . . . .

This state of knowledge, where it is common knowledge that if m is sent then everyone will eventually know that m has been sent, gives rise to a weak state of group knowledge which we call ***eventual common knowledge***.

Recall that the temporal logic symbol $\Diamond$ stands for "eventually". We denote $\Diamond K_i\varphi$ by $K_i^\diamond\varphi$ and define $E^\diamond\varphi \equiv \bigwedge_i K_i^\diamond\varphi$. (Note that in the total view interpretation, where processors do not forget stable facts, if φ is stable then $E^\diamond p \equiv \Diamond E\varphi$.) ***$\Diamond$-common knowledge*** (read ***eventual common knowledge***), denoted by $C^\diamond$, is defined as (the greatest fixpoint of):

$$C^\diamond\varphi \equiv \varphi \wedge E^\diamond C^\diamond\varphi.$$

The axioms (1)–(3) from section 4.1 are also valid for $C^\diamond\varphi$ (once we replace E by $E^\diamond$ and C by $C^\diamond$). As with C^ϵ, the formula $\neg C^\diamond\varphi \supset C^\diamond\neg C^\diamond\varphi$ is *not* valid. Note that $E\Diamond\varphi$ does not imply $E^\diamond\varphi$, so the fact that $C^\diamond\varphi$ holds does not imply that $E^2\varphi$ will ever hold.

Given our experience with C and C^ϵ, it would be natural to conjecture that $C^\diamond\varphi$ is equivalent to $\varphi \wedge \Diamond E\varphi \wedge \Diamond E\Diamond E\varphi \wedge \cdots$. However, this infinite conjunction is strictly weaker than $C^\diamond\varphi$! The reason for that is that $\Diamond$ and $\bigcirc^\epsilon$ do not interact with infinite conjunctions in the same way. I.e., if each of an infinite number of facts are guaranteed to hold ϵ time units from now, then their conjunction will also hold. However, if an infinite number of facts are each guaranteed to hold *eventually*, then it is not necessarily the case that at any given time in the future they will all hold simultaneously.[1]

An easy consequence of the definition of $C^\Diamond\varphi$ is that if a processor knows $C^\Diamond\varphi$, then $C^\Diamond\varphi$ holds and all the processors eventually know it. As common knowledge corresponds to simultaneous events and ϵ-common knowledge to events that occur within ϵ of each other, $\Diamond$-common knowledge corresponds to events that are guaranteed to happen at all sites eventually. For example, in some of the work on variants of the Byzantine Agreement problem discussed in the literature (cf. [DDS]), the kind of agreement sought is one in which whenever a correct processor decides on a given value, each other correct processor is guaranteed to ***eventually*** decide on the same value. The state of knowledge of the decision value that the processors can be said to attain in such circumstances is precisely $\Diamond$-common knowledge. Also, in asynchronous broadcast channels, *sent*(m) is $\Diamond$-common knowledge at the instant m is sent.

$C^\Diamond$ is the weakest temporal notion of common knowledge that we have introduced. In fact, we now have a hierarchy of the temporal notions of common knowledge. For a stable fact φ and $\epsilon_1 \leq \cdots \leq \epsilon_n \leq \epsilon_{n+1} \leq \cdots$, we have:

$$C\varphi \supset C^{\epsilon_1}\varphi \supset \cdots \supset C^{\epsilon_n}\varphi \supset C^{\epsilon_{n+1}}\varphi \supset \cdots \supset C^\Diamond\varphi.$$

Having defined C^ϵ and $C^\Diamond$, it is interesting to ask how these states of knowledge are affected by communication not being guaranteed. Recall that Lemma 3.1 and Theorem 3.2 imply that if communication is not guaranteed, then common knowledge is independent of the communication process: communication does not affect what facts are common knowledge and when facts become common knowledge. Interestingly, an analogue of Lemma 3.1 does not hold for C^ϵ and $C^\Diamond$. It is not, in general, the case that $C^\epsilon\psi$ is undetermined at (r,t) iff in the absence of any further communication $C^\epsilon\psi$ will never hold (the same applies to $C^\Diamond\psi$). For example, consider a system consisting of R2 and D2 connected by a two-way link. Communication along the link is not guaranteed, R2 and D2's clocks are perfectly synchronized, and each one of them follows the following protocol: *At time 0, send the message "OK". For all $k > 0$, if you have received k "OK" messages by time k on your clock, send an "OK" message at time k; otherwise, send nothing.* Let

[1] However, $C^\Diamond\varphi$ *is* equivalent to —a different infinite conjunction of formulas. Define $\Phi_0 = \varphi$ and $\Phi_{n+1} = \Phi_n \wedge \Diamond E\Phi_n$. Then it is possible to show that $C^\Diamond\varphi \equiv \bigwedge_n \Phi_n$ (cf. the discussion in Appendix B).

$\psi =$"some message was not delivered within one time unit". Fix $\epsilon = 1$. Notice that $C^\epsilon\psi$ is undetermined at time 0, since in the fortunate event that all messages are delivered within one time unit, ψ will never hold, and similarly $C^\epsilon\psi$ will never hold. However, if at any point ψ does hold, then so does $C^\epsilon\psi$. E.g., if R2 fails to receive an "OK" message between time $k-1$ and time k, then $R2$ knows ψ at time k. R2 therefore does not send an "OK" message at time k, and it follows that D2 knows ψ at time $k+1$ at the latest. And because $\psi \supset \bigcirc^\epsilon E\psi$ is guaranteed, it is easy to see that by the induction axiom $\psi \supset C^\epsilon\psi$. (Since $C^\epsilon q \supset C^\diamond\psi$, the same example shows the claim for $C^\diamond\psi$.)

Note that in the above example successful communication in a system with unguaranteed communication helped to prevent $C^\epsilon\psi$ (resp. $C^\diamond\psi$) from holding. But can successful communication in such a system contribute to $C^\epsilon\varphi$ or $C^\diamond\varphi$ coming to hold? A partial analogue to Theorem 3.2 shows that this is not possible:

Theorem 4.2: Let ψ be a stable fact, let S be a system in which communication is not guaranteed, and let $r_1, r^- \in S$ be runs such that r^- extends (r_1, t_1) and no messages are delivered in r^- at or after t_1. If $(r^-, t) \models \neg C^\epsilon\psi$ (resp. $(r^-, t) \models \neg C^\diamond\psi$) for all $t \geq t_1$, then $(r_1, t) \models \neg C^\epsilon\psi$ (resp. $(r_1, t) \models \neg C^\diamond\psi$) for all $t \geq t_1$.

Proof: We sketch the proof for $C^\diamond\psi$. The proof for $C^\epsilon\psi$ is analogous. We prove by induction on n that there is no run $r \in S$ that extends (r_1, t_1) in which $C^\diamond\psi$ holds at a time $t \geq t_1$ and exactly n messages are delivered to their destinations up to (but not including) the time the first processor knows $C^\diamond\psi$. The case $n = 0$ follows from our assumption, since as long as no messages are delivered all processors' knowledge is the same as in r^-, and in r^- no processor ever knows $C^\diamond\psi$. Suppose we have proved the claim for $n \leq k$, and assume that r extends (r_1, t_1) and attains $C^\diamond\psi$ using $k+1$ messages. Let t be the real time at which the first processor (or one of them, in case of a tie), say p_i, comes to know $C^\diamond\psi$ in r. If no message was delivered to p_i before time t in r, then p_i knows $C^\diamond\psi$ at (r, t) iff p_i knows $C^\diamond\psi$ at (r^-, t), a contradiction. Let $t' < t$ be the latest time before t in which p_i receives a message in r, and let m be one of the messages p_i receives at t'. Let r' be a run that extends (r, t') in which no messages are delivered after time t', and all messages delivered at (r, t') are delivered at (r', t') (such a run exists by (*)). Since p_i's view at (r, t) and at (r', t) are the same, p_i knows $C^\diamond\psi$ in (r', t). Thus, in particular, if $p_j \neq p_i$ is another processor, then for some $t'' \geq t$ it is the case that p_j knows $C^\diamond\psi$ at (r', t''). Let r'' be a run that is identical to r' until (and including) time t', except that p_i does not receive the message m at (r'', t'), and in which no message is delivered after t'. Since p_j's view at (r', t'') and at (r'', t'') are identical, p_j knows $C^\diamond\psi$ at

(r'', t''). But at most k messages are delivered in r'' before t'', contradicting the induction hypothesis. $\bowtie$

Theorem 4.2 shows that communication cannot be used in order to attain $C^{\Diamond}\psi$ or $C^{\epsilon}\psi$ when these states of knowledge are not guaranteed to hold in the absence of communication. Thus unreliable communication cannot be used for planning and carrying out coordinated actions in a way that guarantees the participation of all sites. This allows us to prove Corollary 4.3, which characterizes the graveness of the generals' problem in the coordinated attack example:

Corollary 4.3: In the coordinated attack problem, any protocol that guarantees that if either party attacks then the other party will eventually attack, is a protocol in which necessarily neither party attacks.

Proof: The proof is analogous to that of Proposition 3.4. Assume that (P, P') is a joint protocol that guarantees that if either party attacks then they both eventually attack. Let S and t_1 be as in the proof of Proposition 3.4. Let ψ ="general A either has started attacking or will eventually attack, and general B either has started attacking or will eventually attack". By the problem description, $C^{\Diamond}\psi$ is undetermined at (r, t_1), for all $r \in S$. Because of the properties of (P, P'), it is clear that an attacking general knows $C^{\Diamond}\psi$ (under the total view interpretation as well as under the interpretation $\mathcal{I}_{\sigma}$ used in the proof of proposition 3.4). Thus, by Theorem 4.2, the protocol (P, P') guarantees that neither general will ever attack! $\bowtie$

Recall that the proofs that unreliable communication cannot affect what facts are common knowledge carried over to (reliable) asynchronous communication. Our proof in Theorem 4.2 clearly does not carry over. In fact, a message broadcast over a reliable asynchronous channel does become eventual common knowledge. However, it is possible to show that asynchronous channels cannot be used in order to attain ϵ-common knowledge:

Theorem 4.4: Let ψ be a stable fact, let S be a system with unbounded delivery times, let $t \geq t_1$ and let $r_1, r^- \in S$ be runs such that r^- extends (r_1, t_1) and no messages are delivered in r^- in the interval $[t_1, t + \epsilon)$. If $(r^-, t) \not\models C^\epsilon \psi$ then $(r_1, t) \not\models C^\epsilon \psi$.

Sketch of Proof: The proof essentially follows the proof of Theorem 4.2, except that the delivery of messages in the runs r' and r'' constructed in the course of the proof are delayed until after time $t + \epsilon$, rather than not being delivered at all. Details are left to the reader. $\bowtie$

Thus, asynchronous communication channels are of no use for coordinating actions that are guaranteed to be performed at all sites within a predetermined fixed time bound.

4.3 Time stamping: using relativistic time

Real time is not always the appropriate notion of time to consider in a distributed system. Processors in a distributed system often do not have access to a common source of real time, and their clocks do not show identical readings at any given real time. Furthermore, the actions taken by the processors rarely actually depend on real time. Rather, time is often used mainly for correctly sequencing events at the different sites and for maintaining a "consistent" view of the state of the system. In this section we consider states of knowledge relative to *relativistic* notions of time.

Consider the following scenario: R2 knows that R2 and D2's clock differ by at most δ, and that any message R2 sends D2 will arrive within ϵ time units. R2 sends D2 the following message m':

> "This message is being sent at t_S on R2's clock, and will reach D2 by $t_S + \epsilon + \delta$ on both clocks; m."

Let us denote $t_S + \epsilon + \delta$ by T_0. Now, at time T_0 on his clock, R2 would like to claim that *sent*(m') is common knowledge. Is it? Well, we know by now that it is not, but it is interesting to analyze this situation. First, let us introduce a relativistic formalism for knowledge, which we call *time-stamped* knowledge: We denote "at time T on his clock, p_i knows φ" by $K_i^T \varphi$. T is said to be the *time stamp* associated with this knowledge. We then define

$$E_G^T \varphi \equiv \bigwedge_{i \in G} K_i^T \varphi.$$

$E^T \varphi$ corresponds to everyone knowing φ individually at time T on their own clocks. Notice that for T_0 as above, *sent*(m') $\supset$ E^{T_0}*sent*(m'). It is natural to define the corresponding relativistic variant of common knowledge, C^T, which we call *time-stamped* common knowledge:

$$C^T\varphi \equiv \varphi \wedge E^T C^T \varphi.$$

So, once m' is sent, R2 and D2 have time-stamped common knowledge of $sent(m')$ with time stamp T_0. It is easy to check that C^T satisfies axioms (1)–(3) of Section 4.1. Interestingly, the formula $\neg C^T\varphi \supset C^T\neg C^T\varphi$ *is* valid. In this respect, C^T resembles C more than C^ϵ and $C^\diamond$ do.

It is interesting to investigate how the relativistic notion of time-stamped common knowledge relates to the notions of common knowledge, ϵ-common knowledge, and $\diamond$-common knowledge. Not surprisingly, the relative behavior of the clocks in the system plays a crucial role in determining the meaning of C^T.

Theorem 4.5: Under the total view interpretation,

(a) If it is common knowledge that all clocks show identical times, then at T on any processor's clock, $C^T p \equiv C\varphi$.

(b) If φ is a stable fact and it is ϵ-common knowledge that all clocks are within ϵ time units of each other, then at T on any processor's clock, $C^T p \supset C^\epsilon\varphi$.

(c) If φ is a stable fact and it is $\diamond$-common knowledge that all clocks read T at some time, then at time T on any processor's clock, $C^T p \supset C^\diamond\varphi$. ⋈

Theorem 4.5 demonstrates the conditions that allow interchanging the relativistic C^T with C, C^ϵ, and $C^\diamond$. Note that a weak converse of Theorem 4.5 holds as well. Suppose we allowed the processors to set their clocks to a common agreed upon time T when they come to know $C\varphi$ (resp. come to know $C^\epsilon\varphi$, $C^\diamond\varphi$). Then it is easy to see that whenever $C\varphi$ (resp. $C^\epsilon\varphi$, $C^\diamond\varphi$) is attainable, sc is $C^T\varphi$.

In many distributed systems time-stamped common knowledge seems to be a more appropriate notion to reason about than "true" common knowledge. Although common knowledge cannot be attained in practical systems, time-stamped common knowledge is attainable in many cases of interest and seems to correspond closely to the relevant phenomena that protocol designers are confronted with. For example, in distributed protocols that work in phases, we speak of the state of the system at the beginning of phase 2, at the end of phase k, and so on. It is natural to think of the phase number as a "clock" reading, and consider knowledge about what holds in the different phases as "time-stamped" knowledge, with the phase number being the time stamp. In certain protocols for Byzantine agreement, for example, the nonfaulty processors attain common knowledge of the decision value at the end of phase k (see Chapter 6 for details on the relationship between knowledge and the Byzantine agreement problem). In practical systems in which the phases do not end simultaneously at the different sites of the system, the processors actually attain time-stamped common knowledge of the decision value, with the time stamp being "the end of phase k".

4.4 Likely common knowledge and other variants

The properties of the communication channels in a system play an important role in determining the type or quality of the information that can be obtained by the processors in a system. In particular, we have seen that common knowledge cannot be attained in many systems of practical interest. Sections 4.2 and 4.3 dealt mainly with the states of knowledge that arise when there is uncertainty regarding *when* messages are delivered. In systems in which communication is not guaranteed, such as in the case of the coordinated attack problem, it is uncertain *whether* messages are delivered at all. However, it is often the case that successful communication is highly likely in that every message broadcast is highly likely to be delivered to all the processors.

First, let us consider systems in which message delivery, when successful, is immediate. The fact a message broadcast in such a system is likely to be delivered to all processors can be formally captured by the formula $C(sent(m) \supset$ "Likely"$E(sent(m)))$. (The notion of "likely" here is essentially that of [HR].) Theorem 4.2 implies that a message broadcast in such a system never becomes even eventual common knowledge. The notion of common knowledge that the processors in such a system attain when a message is broadcast is called *likely* common knowledge, denoted C^L. If we denote "Likely $E\varphi$" by $E^L\varphi$, then $C^L\varphi$ is defined by:

$$C^L\varphi \equiv \varphi \wedge E^L C^L \varphi.$$

By definition, C^L satisfies the fixpoint axiom (1) of Section 4.1. If we denote "Likely ψ" by $L\psi$, it is not in general the case that likelihood satisfies a consequence closure axiom. Namely, it is not necessarily the case that if both $L\varphi$ and $L(\varphi \supset \psi)$ hold, then $L\psi$ holds (cf. [HR]). As a result, C^L satisfies neither the induction axiom (2) nor the consequence closure axiom (3). However, it does satisfy a *weak* induction axiom:

$$(2') \qquad C(\varphi \supset E^L\varphi) \supset (\varphi \supset C^L\varphi).$$

Here, $\varphi \supset E^L\varphi$ needs to be common knowledge for φ to imply $C^L\varphi$. Note that if it is common knowledge that a message is likely to be delivered to all processors immediately, then the weak induction axiom suffices for a processor that receives a broadcast message m to conclude that $sent(m)$ is likely common knowledge.

Another consequence of the fact that L does not satisfy consequence closure is that although $L(\varphi \wedge \psi) \supset L\varphi \wedge L\psi$, the converse does not hold. Thus, the infinite conjunction $\varphi \wedge E^L\varphi \wedge (E^L)^2\varphi \wedge \cdots$ is strictly weaker than $C^L\varphi$ as defined above. Note that $(E^L)^2\varphi$ ($= LELE\varphi$) does not necessarily imply $L^2E^2\varphi$. Indeed, just as with $C^\diamond\varphi$, $C^L\varphi$ does not necessarily imply $L^2E^2\varphi$. In fact, $E^2\varphi$ does not necessarily hold at any level of likelihood!

In the coordinated attack problem, if (it is common knowledge that) it is likely that a messenger sent from camp A to camp B will deliver the message to camp B within an hour, then any message general A sends general B becomes likely common knowledge an hour after it is sent. Likely common knowledge is a very useful notion. We mentioned earlier that when we say "the president" we assume common knowledge of who the term "the president" refers to. Likely common knowledge actually captures this situation in a better way, since there might be some out-of-touch person who does not know who the president (of the company) is. Thus, when we use a definite reference such as "the president" we are essentially treating a fact that is likely common knowledge as if it were common knowledge. In fact, in practically all the cases in which people believe that they attain common knowledge of a fact, it can be argued that they have only likely common knowledge, where the "likely" qualification in this case corresponds to the (commonly known) likelihood that all the members of the group are aware and clever enough to conclude that the fact is (likely) common knowledge. In some cases, of course, this likelihood may be very close to certainty.

We have defined likely common knowledge for an abstract notion of "likely". It is often the case that the degree of likelihood of message delivery is quantified, e.g., in terms of a given probability π. (Note that identifying a system with the set S of its runs facilitates such definitions. The standard techniques of measure theory and probability theory immediately apply.) Given any specific flavor of likelihood we can clearly define a variant of likely common knowledge that corresponds to it as the greatest fixpoint of an appropriate equation. Thus, we may have variants of common knowledge corresponding to "With probability one", "With probability π", "Unlikely", and so on. The temporal and likelihood aspects of message delivery seem to be orthogonal in nature. In some cases it seems natural to combine them. This is useful, for example, in communication that is characterized by a probability distribution. Consider a communication scheme in which at any given time step each pending (as yet undelivered) message is delivered with probability 1/2. The state of knowledge of a message R2 sends D2 in such a system can be described by notions of common knowledge corresponding to "with probability 1/2, within one time step", "Likely within 100 time steps" (where "likely" here corresponds to "with probability $\geq 1 - \frac{1}{2^{100}}$"), "with probability one, eventually", or "with geometric probability distribution $G(1/2)$ over time". These different notions allow us to characterize different aspects of the system's communication properties, and to work with different levels of abstraction.

4.5 Discussion

This chapter has introduced a variety of states of knowledge, discussed their roles, and analyzed the relationship between them. A pattern that arises from the presentation here is that the properties of a communication channel determine a particular state of knowledge that is the result of broadcasting a message over such a channel. Similarly, the relative manner in which an event is guaranteed to occur in all sites of the system closely corresponds to a related state of knowledge. Many of these states of knowledge seem to be approximations of common knowledge, accounting for things such as an uncertainty in the relative times in which events take place, or an uncertainty as to whether they will take place. (Common knowledge is attained by events that are guaranteed to take place simultaneously at all sites.) Thus, proving statements about the inability to attain such states of knowledge in particular circumstances is a general way of proving the inability to be guaranteed to perform coordinated action of the corresponding kind under the same circumstances. For instance, our results imply that if communication is not guaranteed, then no protocol can guarantee to achieve even *eventual* coordinated attack. Furthermore, if communication is asynchronous, then no protocol can guarantee that the generals sometimes attack, and that whenever they attack they do so within a fixed predetermined amount of time of each other. However, our results are more general than the coordinated attack problem, and they immediately apply to problems such as *transaction commit* in distributed databases (cf. [Gr]).

Chapter 5

The Cheating Husbands Stories

As the previous chapters have illustrated, the success of certain cooperative actions in a distributed environment often depends on the attainment of various states of knowledge by the group of agents involved. In this chapter we perform a case study of a slightly different aspect of the subtle interplay between knowledge, communication and action, by analyzing the effect of broadcasting a particular set of instructions using different communication channels in various settings. In order to enhance readability, the analysis is carried out within the context of a fictional story, and the presentation is to a great extent self-contained.

5.1 Introduction

The "cheating wives" puzzle, a well-known puzzle from the folklore (cf. [GS]), has long been one of the primary examples of the subtle interdependence between knowledge and action. (We have already presented it as the muddy children puzzle in Chapter 1.) We will now reveal the contents of recently discovered scrolls, allegedly written by the great scholar Josephine of the lost continent of Atlantis. These scrolls describe how modernizing the means of communication in Atlantis over the generations affected the resolution of the recurring problem of unfaithful husbands there. They provide some indication of the issues involved in the interaction between knowledge, action and communication. In particular, one of the central issues that are illustrated involves what knowledge an agent who knows something about how other individuals' actions are related to the facts they know, can obtain by observing the other individuals' actions.

The original cheating husbands problem is re-introduced in section 5.2.[1] Section 5.3 describes what happens when an asynchronous communication channel is used to communicate the protocol to be followed. Section 5.4 involves different types of synchronous communication, and includes a discussion of the conditions under which a "cheating husbands"-like protocol can tolerate "faults" (disobedient wives). Section 5.5 deals with ring-based communication. Section 5.6 treats the question

[1] Martin Gardner independently presented this puzzle in terms of "cheating husbands" in the thoroughly amusing [Gar].

of how allowing wives to communicate a small amount of extra information allows a substantially faster solution to the problem. Some conclusions are presented in section 5.7.

5.2 The cheating husbands puzzle

Josephine's account of the history of a major city in Atlantis starts with the following incident:

> The queens of the matriarchal city-state of Mamajorca, on the continent of Atlantis, have a long record of opposing and actively fighting the male infidelity problem. Ever since the technologically-primitive days of queen Henrietta I, women in Mamajorca have been required to be in perfect health and pass an extensive logic and puzzle-solving exam before being allowed to take a husband. The queens of Mamajorca, however, were not required to show such competence.
>
> It has always been common knowledge among the women of Mamajorca that their queens are truthful and that the women are obedient to the queens. It was also common knowledge that all women hear every shot fired in Mamajorca. Queen Henrietta I awoke one morning with a firm resolution to do away with the male infidelity problem in Mamajorca. She summoned all of the women heads-of-households to the town square and read them the following statement:
>
> > *There are (one or more) unfaithful husbands in our community. Although none of you knew before this gathering whether your own husband was faithful, each of you knows which of the other husbands are unfaithful. I forbid you to discuss the matter of your husband's fidelity with anyone. However, should you discover that your husband is unfaithful, you must shoot him on the midnight of the day you find out about it.*
>
> Thirty nine silent nights went by, and on the fortieth night, shots were heard.

Josephine does not explicitly say how many unfaithful husbands were shot, how many unfaithful husbands were in Mamajorca at the time, how some cheated wives learned of their husbands' infidelity after thirty nine nights in which *nothing* happened, or whether any more husbands were shot on later nights. The interested reader should stop at this point and try to answer these questions based on Josephine's account.

Let us consider the questions Josephine leaves unanswered. Since Henrietta I was truthful, there must have been at least one unfaithful husband in Mamajorca. How would events have evolved if there was exactly one unfaithful husband? His wife, upon hearing the queen's statement, would have concluded that her own husband was unfaithful, and would have shot him on the midnight of the first night. Clearly, there must have been more than one unfaithful husband. (Recall that the wives are all perfect logicians.[2]) If there had been exactly two unfaithful husbands, then every cheated wife would have initially known of exactly one unfaithful husband, and would have reasoned as follows: "If the unfaithful husband I know of is the only unfaithful husband, then his wife will shoot him on the first night." Therefore, neither one of the cheated wives would shoot on the first night. On the morning of the second day each cheated wife would realize that the unfaithful husband she knew about was not the only one, and that therefore her own husband must be unfaithful. The unfaithful husbands would thus both be shot on the second night. In fact, similar reasoning is used by the wives in general, and the following theorem, well known in the folklore, resolves our doubts regarding Josephine's presentation of the facts:

Theorem 5.1: If there had been n unfaithful husbands in Mamajorca at the time Henrietta I announced her ruling, they would all have been shot on the midnight of the n^{th} day.

Proof: The discussion above shows the claim for $n = 1$. Assume that the claim holds for $n = k$. Thus, if there were k unfaithful husbands they would be shot on the k^{th} night. We wish to show that if there were $n = k+1$ unfaithful husbands they would have been shot on the $(k+1)^{\text{st}}$ night. Assume therefore that there were $k+1$ unfaithful husbands. Every cheated wife knows of exactly k unfaithful husbands. Because of the wives' logical competence, they know that if there are exactly k unfaithful husbands then those husbands will all be shot on the k^{th} night. Before the k^{th} night, a cheated wife cannot determine that her husband is unfaithful, and therefore no shots are fired in any of the first k nights. Since the k^{th} night is silent, every cheated wife concludes on the morning of the $(k+1)^{\text{st}}$ day that there must be more than k unfaithful husbands, and that her own husband must therefore be

[2] The fact that the wives are perfect reasoners plays a crucial role in all of the cases we treat. The nature of the situation changes substantially if we relax this assumption, since wives must then reason about the logical capabilities of other wives. Some preliminary steps towards dealing with such a situation are presented in [Kon], where Konolige considers a version of the *wise men* puzzle — a well known puzzle that is a special case of the cheating husbands problem — which he calls the *not-so-wise men puzzle*, in which the knowers are not perfect logicians.

unfaithful. The unfaithful husbands are shot on the $(k+1)^{st}$ night. The theorem follows by induction. $\bowtie$

Notice the subtlety of the situation: On the first day, immediately after the queen delivers her statement, a wife who knows of k unfaithful husbands knows that every cheated wife knows of at least $k-1$ unfaithful husbands, and knows that their wives know of at least $k-2$ unfaithful husbands, and that their wives know of at least $k-3$ unfaithful husbands .... It follows that every wife thinks that it is possible that a cheated wife thinks that it is possible that a cheated wife thinks it is possible ... that a cheated wife knows of no unfaithful husbands other than her own. Thus, for all $k > 1$, it is not common knowledge that there are at least k unfaithful husbands. The queen's statement, however, *is* common knowledge. This follows from the fact that the queen announced it publicly, thereby making it common knowledge that all of the wives heard her announcement.[3] It follows that after the queen speaks, it is common knowledge that there is at least one unfaithful husband. Given the wives' famous logical capabilities, it is common knowledge that if there is only one unfaithful husband then he will be shot on the first night. Therefore, once the first night is silent it becomes common knowledge that there are at least two unfaithful husbands. Similarly, after k silent nights (but not earlier!), it is common knowledge that there are at least $k+1$ unfaithful husbands and that every wife knows of at least k unfaithful husbands other than her own. So although a wife that knows of k unfaithful husbands knows that there will be no shots before the k^{th} night, her state of knowledge changes following every silent night, even though there is no "communication" at all!

[3] For a discussion of this point, see Chapter 1.

5.3 Asynchronous communication

Josephine's description of Mamajorca continues with the following account:

> Queen Henrietta I was highly regarded by her subjects for her wisdom in running the monarchy. She ordered her daughters to continue her moral fight against male infidelity.
>
> Her daughter, Henrietta II, succeeded her. In order to facilitate communication with her subjects, Henrietta II installed a mail system from her court to all of the households in Mamajorca. Her first letter to her subjects told them about the properties of the new mail system: every letter she sends her subjects is guaranteed to eventually reach each and every one of them. Thus, she will not need to gather them in the town square for announcements anymore. Eager to fulfill her mother's wish, Henrietta II's second letter to her subjects was an exact copy of her mother's original statement.
>
> Henrietta II suffered great disgrace and died in despair. She ordered her daughters not to repeat her mistake.

Josephine suggests that despite the fact that Henrietta II gave the wives of Mamajorca exactly the same instructions as her mother, her mother was honored, whereas she was disgraced. Again, Josephine refrains from explicitly stating why this happened. Let us consider the possible outcomes of Henrietta II's action. Had there been exactly one unfaithful husband at the time, his wife would have shot him on the first night after receiving the queen's letter, and the queen would have been saved from disgrace. If there had been exactly two unfaithful husbands, however, each of their wives would know about the existence of one unfaithful husband, and that if the husband she knows about is the only unfaithful one, then his wife will shoot him on the day she receives the letter. Because the mail system is asynchronous, with messages only guaranteed to be delivered *eventually*, neither wife would ever know that the other had already received the queen's letter. Thus, neither wife would know that her husband is unfaithful: she would always consider it possible that her own husband is faithful and that the cheated wife she knows about has not shot yet because the queen's letter has yet to reach her. An immediate consequence of the above argument is:

Theorem 5.2: If there is more than one unfaithful husband, and the original instructions are broadcast over an asynchronous channel, then no unfaithful husbands are shot. ⋈

Because the letter is broadcast using an asynchronous channel, the queen's letter becomes *eventual common knowledge:* once the queen sends it, every wife will eventually receive the letter, and when she does she'll know that all wives will eventually receive the letter, and know ... (see Chapter 3). However, at no time does a wife know that all other wives have received the letter. Thus, a wife can never determine whether the silent nights are a result of other wives' reaction to receiving the letter or a result of the fact that they have yet to receive the letter. This property of asynchronous communication comes up in a similar fashion in the analysis of the Byzantine agreement problem in asynchronous networks (cf. [FLP]). There, the asynchronous nature of the system prevents a processor from ever determining whether it has not received messages from another processor because the other processor did not send any (and thus is faulty), or because the messages are still on their way.

Notice that even if all of the wives happened to receive the queen's letter simultaneously, this would not help. The fact that a wife must always consider it *possible* that other wives have not yet received the queen's letter is sufficient to prevent her from being able to figure out whether her own husband is unfaithful.

5.4 Synchronous communication

Josephine proceeds to describe the controversial actions that ensued:

> Henrietta III succeeded her mother, Henrietta II. She decided to upgrade the mail system that her mother had installed in order to avoid her mother's problem. Thus, she improved the mail system so that any letter sent by the queen was guaranteed to reach all of her subjects no later than one day after it was sent.
>
> Henrietta III knew that unless her subjects were aware of the improvement in the mail system, she would repeat her mother's mistake. Thus, Henrietta III's first letter to her subjects announced the new advances in the mail delivery system, and her second one was an exact copy of Henrietta I's statement.
>
> Henrietta III was considered a more effective monarch than her mother, but she will always be remembered for the great injustice she brought upon Mamajorca. If only she had told her subjects to wait a few days before shooting, however, she could have attained her grandmother's fame!

A mail system that guarantees that every letter sent is delivered no more than $b-1$ days after it is sent is called *weakly synchronous with bound b*. If we call the sending day the first day, then such a letter is delivered to all wives no later than on day b. Before we continue, we remark that in Henrietta III's days no calendar had been established in Mamajorca.

Let Ep denote "everyone knows p", and

$$E^{m+1}p \overset{\text{def}}{=} E(E^m p), \qquad \text{for } m > 0.$$

Notice that an easy proof by induction shows that if there are n unfaithful husbands, and E^n("the queen sent the letter") becomes true at some point, then at least one cheated wife will shoot her husband, and the first shot will be fired at most n days after E^n("the queen sent the letter") first holds. In our case, a letter sent by the queen is guaranteed to be delivered to all of the wives in less than b days. Thus, once the letter is sent its contents become *b-common knowledge*: within b days every wife receives the letter and knows that within b days every wife will receive the letter and know that within b days ... every wife will know the contents of the letter (see Chapter 1). Thus, kb days after the queen sends the letter, E^k("the queen sent the letter") holds, so it is certain that at least one unfaithful husband will be eliminated.

Although Henrietta III was probably not familiar with the concept of b-common knowledge, apocryphal records indicate that she was able to prove the following proposition:

Proposition 5.3: In the weakly synchronous case with the bound on delivery being b, a wife that knows of exactly k unfaithful husbands will know that her own husband is unfaithful once kb silent nights pass after the day she receives the queen's letter.

Proof: A wife knowing of $k = 0$ unfaithful husbands requires $kb = 0$ silent nights to conclude that her own husband is unfaithful. By the queen's statement, that wife does not know that her husband is unfaithful any earlier than that. Assume inductively that a wife knowing of k unfaithful husbands requires kb silent nights to conclude that her own husband is unfaithful, and suppose Mary knows of $k + 1$ unfaithful husbands. Mary knows that if her own husband is faithful, then every cheated wife knows of exactly k unfaithful husbands, and, by the induction hypothesis, will shoot her husband on the following night should kb silent nights go by after the cheated wife receives the letter. For all Mary knows, it is initially possible that her husband is faithful, and the letter may reach the first cheated wife to receive it $b - 1$ days after Mary receives it. Thus, she must consider it possible that no shots will be fired before the $(k + 1)b^{\text{th}}$ night after she receives the queen's letter. However, should that night be silent, Mary will know that her husband is unfaithful. The claim follows by induction. ⋈

Thus, Henrietta III was guaranteed not to suffer her mother's disgrace. However, what she didn't realize was that noisy nights might confuse some of the wives. Consider, for example, the following scenario: The queen's letters are guaranteed to arrive in less than 2 days (i.e., $b = 2$), and Susan knows that Mary's husband is unfaithful. Suppose Susan receives the queen's letter on a Monday, and hears Mary shoot her own husband at midnight on Tuesday night. Unfortunately, now Susan will not be able to figure out whether or not her own husband is faithful. Susan does not know whether the queen originally sent the letter on Sunday or on Monday, and thus considers it possible that Mary received the queen's letter on either Sunday, Monday or Tuesday. In particular, Susan considers both of the following scenarios possible:

- Mary received the letter on Tuesday and, knowing that Susan's husband is faithful, shot her own husband on Tuesday night.
- Mary received the letter on Sunday and, knowing that Susan's husband is unfaithful, waited to see if Susan would shoot her husband on Sunday or Monday night. Since Susan did not shoot, on Tuesday Mary concluded that her own husband was unfaithful, and shot him.

Thus, Susan cannot determine whether her own husband is faithful based on Mary's actions. Furthermore, she will never obtain any more information on the subject and will remain in doubt forever.

We call the first day on which the queen's letter is delivered to a cheated wife the first ***significant*** day. Given Proposition 5.3, it is easy to see that cheated wives that receive the queen's letter on the first significant day will be the first to shoot their husbands. Do any other cheated wives shoot their husbands?

Every wife has an interval of $b - 1$ days in which a noisy night would leave her in doubt regarding her husband's fidelity. To see this, recall that a wife knowing of, say, $k > 0$ unfaithful husbands does not initially know whether there are k or $k + 1$ unfaithful husbands in all. Furthermore, for all she knows the first significant day may happen anywhere between $b - 1$ days before she receives queen's letter and $b - 1$ days after she receives it. "If there are k unfaithful husbands," she reasons, "then at least one of them will be shot on the $((k-1)b+1)^{\text{th}}$ night after the day his wife receives the letter, that is, between the $((k-2)b+2)^{\text{nd}}$ and the kb^{th} night after the day I receive the letter. If, however, there are $k + 1$ unfaithful husbands, one of them will be shot between the $((k-1)b+2)^{\text{nd}}$ and the $(kb+1)^{\text{st}}$ night after the day I receive the letter." Thus, if the first shot occurs between the $((k-1)b+2)^{\text{nd}}$ and the kb^{th} night after the day she receives the queen's letter, a wife initially knowing of exactly k unfaithful husbands will be left in doubt regarding her husband's fidelity. Since a cheated wife that receives the queen's letter after the first significant day will hear a shot in her interval of uncertainty, we have:

Theorem 5.4: Using weakly synchronous broadcast, cheated wives that receive the queen's letter on the first significant day shoot their husbands ($(n-1)b$ days after the first significant day, where n is the number of unfaithful husbands). All other cheated wives remain forever in doubt about their husbands' fidelity. ⋈

How could Henrietta III have changed the instructions slightly and avoided the problem? Josephine seems to suggest that this could have been done by requiring a cheated wife to wait a few days after learning of her husband's infidelity, before shooting him. First notice that the wives' reasoning is slowed down considerably if the shooting happens only after a delay:

Proposition 5.5: In a weakly synchronous mail system with bound b, if every wife is required to wait d days from the day she discovers her husband's infidelity before shooting him, then a wife that knows of exactly k unfaithful husbands will know that her own husband is unfaithful once $k(b+d)$ silent nights pass from the day she receives the queen's letter (and, as long as all preceding nights are silent, no earlier!).

Proof: Analogous to the proof of Proposition 5.3. For $k = 0$ the statement is trivially true. Assume inductively that it holds for k and that Mary knows of $k+1$ unfaithful husbands. Mary knows that if there are exactly $k+1$ unfaithful husbands, then every cheated wife knows of k unfaithful husbands. Thus, a cheated wife that receives the queen's letter on the first significant day (i.e., at least as early as any other cheated wife) will know that her husband is unfaithful once $k(b+d)$ silent nights pass from the day she receives the letter. Ordinarily she would wish to shoot on the $(k(b+d)+1)^{st}$ night, but since she must delay d days, she will shoot her husband on the $(k(b+d)+d+1)^{st}$ night after receiving the letter. Since Mary must consider it possible that the first significant day occurs as many as $b-1$ days after she receives the letter, Mary will know that her own husband is unfaithful once $k(b+d)+d+1+b-1 = (k+1)(b+d)$ silent nights pass and no earlier. The lemma follows by induction. ⋈

Josephine's claim is confirmed by the following theorem:

Theorem 5.6: If the delay is sufficiently long, more precisely if $d \geq b-1$, then all cheated wives shoot their husbands and no wife remains in doubt.

Proof: We use Proposition 5.5 in a fashion similar to that in which Theorem 5.4 uses Proposition 5.3. First, some notation is needed. Let F be the first significant day, let D be the day Mary receives the letter, and let S be the day preceding the night of the first shot. Notice that the proof of Proposition 5.5 implies that if $n \geq 1$ is the number of unfaithful husbands, then $S = F + (n-1)(b+d) + d + 1$. Assume that Mary knows of exactly $k \geq 1$ unfaithful husbands. Initially, as far as Mary is concerned, there are two possibilities:

- Mary's own husband is faithful. In this case Mary knows that $D-(b-1) \leq F \leq D+(b-1)$. (Notice that Mary must consider the whole interval possible.) Since the number of unfaithful husbands is k, it follows that $S = F+(k-1)(b+d)+d+1$. Substituting h for $D+(k-1)(b+d)+d+1$, Mary has:
$$h-(b-1) \quad \leq \quad S \quad \leq \quad h+(b-1).$$
- Mary's own husband is unfaithful. In this case Mary knows that $D-(b-1) \leq F \leq D$ (We must have $F \leq D$, since otherwise, the first cheated wife to receive a letter does so after Mary does, contradicting the assumption that Mary's husband is unfaithful.) Also, $S = F + k(b+d) + d + 1$, because there are $k+1$ unfaithful husbands. Substituting h as above, Mary has:
$$h+(d+1) \quad \leq \quad S \quad \leq \quad h+(b+d).$$

Therefore, if $d+1 > b-1$ (i.e., $d \geq b-1$), then Mary can distinguish these possibilities (given that she knows S, h, b, and d), and thus is guaranteed to be able to determine whether her husband is unfaithful. It is easy to present scenarios that show that no smaller delay suffices. One such scenario is the example following Proposition 5.3 above. There $b = 2$ and $d = 0 = b - 2$. $\bowtie$

Josephine remarks:

> ...Of course, the shrewd residents of the Wisegal district of Mamajorca avoided any eventual doubts by bribing the mailperson.

We assume that the social attitude towards bribes in Mamajorca was quite different from the attitude towards infidelity. Consequently, (it was common knowledge that) bribery would be kept a secret between a bribing wife and her mailperson. It is also known that delivering mail was not an acceptable profession for the wives of Mamajorca. Thus, it was common knowledge that no wife knew of a wife that bribed the mailperson. Given these circumstances, the following proposition clarifies Josephine's statement:

Proposition 5.7: In the weakly synchronous case, a wife that bribes the mailperson into telling her when the queen had originally sent the letter, does eventually know whether her own husband is faithful.

Proof: Let the bound on delivery be b. Using Proposition 5.3, it is easy to show by a straightforward induction that if there are k unfaithful husbands then the first shot occurs between the $((k-1)b+1)^{st}$ night and the kb^{th} night after the queen sends the letter. Thus, a wife that knows of k unfaithful husbands and bribes her mailperson, knows that her husband is unfaithful if no shot is heard before the kb^{th} night, and knows that he is faithful otherwise. The crucial point is that a wife that bribes her mailperson knows *which* night is the kb^{th} night, and thus eventually knows whether her husband is faithful. ⋈

Josephine continues with the reign of Henrietta IV:

> Henrietta IV, who succeeded her mother as queen, concluded that the lack of a calendar was the reason behind the injustice of her mother's scheme. She summoned the women of Mamajorca to the town square and announced the initiation of a *calendar* beginning that day. "From this day on," she said, "the mail system will be *strongly synchronous:* every letter sent from the queen will bear the mailing date, and will be guaranteed to be delivered to all of her subjects within less than b days." At a later date, Henrietta IV sent her subjects a letter bearing the mailing date, and containing an exact copy of Henrietta I's original instructions. A thousand silent nights followed, and on the thousand and first day, Henrietta IV decided to send another letter. She had finally realized that as a result of Henrietta III's great injustice, the wives of Mamajorca lost much of their faith in the monarchy and its orders. It was still common knowledge that the queens were truthful, and the vast majority of her subjects were obedient, but it was no longer clear that all wives would obey the queen's orders. Henrietta IV's letter contained one line: "There is at least one obedient wife whose husband is unfaithful."
>
> Henrietta IV's wisdom was greatly appreciated throughout Atlantis, and her success restored her subjects' faith in the monarchy.

Let us see why the obedient wives could not figure out whether their husbands were faithful before receiving Henrietta's second letter:

Proposition 5.8: In the strongly synchronous case, if there is exactly one cheated wife, and she is disobedient, all of the other wives are in danger of shooting their husbands on the second night. ⋈

Clearly, if the other wives had not suspected that the cheated wife might be disobedient, all of the faithful husbands would have been shot, whereas the unfaithful husband would have survived! Notice that once this is a possibility, even if all wives are in fact obedient they cannot shoot. To see this, consider the case in which there are exactly two cheated wives. On the second day each cheated wife cannot determine whether the first night was silent because her own husband is unfaithful or because the other cheated wife was disobedient. Thus, no shots are fired on the second night. Similarly, no shots will be fired on any later nights. It is now easy to show by induction that such is the case if there are k cheated wives, for all $k \geq 1$. So how did the queen's second letter help?

Theorem 5.9: In the strongly synchronous case, if it is common knowledge that there is at least one obedient cheated wife, then all obedient cheated wives will shoot their husbands.

Proof: The argument here is very similar to that of Theorem 5.1, with a slight twist. If there is only one unfaithful husband, then his wife is the only cheated wife. Since there is at least one obedient cheated wife, she must be obedient, and therefore will shoot her husband on the day she receives the second letter. If there are exactly $k = 2$ cheated wives, then each obedient cheated wife reasons as follows: "If my husband is faithful then the cheated wife I know of must be obedient", and therefore will shoot her husband when she receives the letter, at most $b - 1$ days after the queen sent it (on day b at the latest). Thus, if by day $b + 1$ no shots are fired, an obedient cheated wife knows that her own husband is unfaithful, and shoots her husband on that night. Assume inductively that if there are exactly $k \geq 2$ unfaithful husbands then all obedient cheated wives shoot their husbands on the $(b + k - 1)^{st}$ night. If there are exactly $k + 1$ unfaithful husbands, then each obedient cheated wife knows of k unfaithful husbands, and knows that if her own husband is faithful then at least one unfaithful husband will be shot on the $(b + k - 1)^{st}$ night. Thus, once that night is silent, she knows that (even though she might be the only obedient cheated wife) her husband is unfaithful, and shoots him on the $(b + (k + 1) - 1)^{st}$ night. The theorem follows by induction. ⋈

Observe the difference between the bribed dates case, described in Proposition 5.7, and the strongly synchronous case of Theorem 5.9. If all of the wives bribed the mailperson, then all of the unfaithful husbands would be shot, and no wife would remain in doubt regarding her husband's fidelity. However, it takes $(n - 1)b + 1$ days to eliminate $n \geq 2$ cheating husbands. Before the end of the process the wives would not necessarily know that justice would be done, and at the end it would not be known whether any wife remains in doubt regarding her own husband's fidelity. In the strongly synchronous case, it takes $b + n - 1$ nights to eliminate $n \geq 2$ unfaithful husbands, and it is common knowledge that justice is done. The difference between the two cases can be best understood by noting that in the first case every wife knew on what day the queen sent the letter, but no wife knew that others knew, whereas in the strongly synchronous case the day on which the queen sent the letter was common knowledge.

5.5 Ring-based communication

Josephine describes the outcome of a similar approach to the male infidelity problem in the neighboring city-state of Mamaringa, in which the households were arranged in a ring:

> The queens of the neighboring matriarchal city-state of Mamaringa commonly adopted customs and rules from Mamajorca. Thus, Mamaringa was similar to Mamajorca in all respects, except that its households were built in a ring around the great Mt. Rouge. The location of each household in the ring was common knowledge, as was the fact that mail was delivered in clockwise order around the ring.
>
> The queens of Mamaringa tried to eliminate the infidelity problem by sending Henrietta I's letter once around the ring, using the state-of-the-art mail system in every generation. None of the queens of Mamaringa suffered the disgrace of Henrietta II, and none attained the honor of Henrietta IV. They will all be forever remembered as cruel and unjust queens.

It is assumed that the queens of Mamaringa hoped that the extra knowledge of the order in which letters are delivered would be helpful in justly eliminating all unfaithful husbands. However, the asymmetry introduced by this knowledge makes a big difference, as the following theorem shows:

Theorem 5.10:

(a) In asynchronous delivery around a ring, the last cheated wife to receive the letter will shoot her husband. All others will not.

(b) In weakly synchronous dɩlivery around a ring, some cheated wives will shoot their husbands, but some might not.

(c) In strongly synchronous delivery around a ring, some cheated wives will shoot their husbands, but some might not.

Proof: (a) We prove by induction that in the asynchronous case a cheated wife knowing of k cheated wives that are all notified before her, and knowing that no cheated wives will be notified after her, will shoot her husband k nights after she receives the queen's letter (and no earlier). For $k = 0$ the claim is trivial. Assume inductively that the claim holds for k and that Mary knows of k cheated wives in the ring before her, and none after her. Once she receives the letter Mary knows that the last of the k cheated wives she knows about has received the letter no later than the same day Mary did. Thus, if Mary's husband is faithful then the last cheated wife she knows of will shoot her own husband no later than k nights after Mary received the letter. Once that *fails to happen*, Mary *shoots* her own husband on the $(k+1)^{st}$ night after receiving the letter. The claim follows by induction. To see that no other cheated wife shoots her husband, notice that because of the asynchronous nature of delivery, a wife knowing of a cheated wife later in the ring does not know when that cheated wife will receive the letter, and thus cannot deduce from the night on which a later wife shoots that her own husband is unfaithful (although in some cases she will be able to deduce that her own husband *is* faithful).

(b) The proof of Proposition 5.3 can be used to show that some unfaithful husbands will be shot in this case. We need to show that injustice might occur, i.e., that some unfaithful husbands might be spared. Consider the following scenario: the bound on delivery is $b = 2$. Mary knows of only one cheated wife, Susan, who lives farther down the ring than Mary. Mary receives the letter on Sunday and hears Susan shoot her husband on Monday. Mary cannot distinguish between the following possibilities:

- Susan received the letter on Sunday, and knowing that Mary's husband was unfaithful, she waited to hear if Mary would shoot on Sunday night. When she didn't, Susan discovered that her own husband was unfaithful, and shot him Monday night.

- Susan received the letter on Monday, and knowing that Mary's husband was faithful, discovered that her own husband was unfaithful and shot him that night.

Thus, Mary does not know whether her husband is unfaithful in the above scenario, and does not shoot her husband. If her husband is in fact unfaithful, this constitutes a case of injustice.

(c) The proof of Proposition 5.3 again ensures us that some husbands will be shot. To show that a case of injustice can arise with strongly synchronous delivery around a ring, consider the situation described in (b) above, with Sunday being the official sending date of the letter. Mary still considers both of the above scenarios possible, and Mary's husband is spared. Thus, if Mary's husband is unfaithful, a case of injustice occurs. ⋈

Notice that in the asynchronous case knowing the order of delivery does help a cheated wife (in this case only the last cheated wife) discover that her husband is unfaithful. In this case the extra knowledge can be considered "helpful". However, more surprising is the fact that the wives' knowing the order of delivery allows an unjust solution in the strongly synchronous case, where none existed without such knowledge! Thus, by introducing an asymmetry in the wives' reasoning, this extra knowledge has a negative effect on the solution.

5.6 Quick elimination

> Queen Margaret opened a new era in Mamajorca. She made the mail system an *express mail* system: All letters sent from her court were guaranteed to be delivered to all of her subjects on the day they were sent. Her first letter notified her subjects about the great advance in their communication capabilities.
>
> Margaret was an impatient queen. She knew that using her mail system she could successfully execute Henrietta I's instructions. However, knowing that there were many unfaithful husbands in Mamajorca, and not wanting to wait very long for them to be eliminated, she decided to look for a faster way to solve the problem. She did so by giving her subjects instructions that allowed wives to shoot into the air at midnight. Margaret's scheme was very successful; the unfaithful husbands were eliminated from Mamajorca in just a few days.

Notice that in Henrietta I's solution, n unfaithful husbands are eliminated on the n^{th} night following the queen's announcement. Margaret sought a solution that would require writing fewer than $O(n)$ nights. Given that shooting in the air at midnight is allowed, what is the minimal number of nights in which the unfaithful husbands can be eliminated? Margaret's problem can be restated as follows: Given a distributed system in which the processors share a memory consisting of a single toggle bit, each processor has a value, and it is known that the values are at most one apart, how many rounds of communication are needed for the processors with the minimal value to know it? El Gamal and Orlitsky (cf. [EO]) have treated similar questions independently in a more general setting. The following theorem answers this question in Margaret's case:

Theorem 5.11: There is a protocol that allows shooting in the air in which the cheating husbands are all shot by the third night. That is the best possible.

Proof: Let us first show that a protocol in which a wife's actions depend only on the number of unfaithful husbands she initially knows of and the actual run of the protocol must require at least three nights. Such a protocol P can be viewed as a set of protocols $P(k), k \geq 0$, each specifying how a wife initially knowing of exactly k unfaithful husbands should act. If for some $k \geq 1$ both $P(k-1)$ and $P(k+1)$ do not prescribe any shooting on the first night, then clearly $P(k)$ must require at least three nights, since a wife knowing of k unfaithful husbands cannot know whether her own husband is faithful after the first night. If $P(k')$ includes shooting in the air on the first night for some $k' \geq 1$, then $P(k')$ must require at least three nights when there are $k'+1$ unfaithful husbands. A wife knowing of exactly k' unfaithful husbands shoots in the air on the first night, and cannot determine whether her own husband is unfaithful before the second night. Thus, for all $k \geq 1$, one of $P(k)$, $P(k+1)$, or $P(k+2)$ must require at least three nights.

The following protocol solves the problem in three nights:

(a) A wife knowing of k_0 unfaithful husbands, with $k_0 \equiv 0 \pmod 3$, fires her gun at midnight on the first night. If $k_0 = 0$ she shoots her husband, otherwise she shoots in the air.

(b0) If there was no shot on the first night, then a wife knowing of k_1 unfaithful husbands, with $k_1 \equiv 1 \pmod 3$ should shoot her husband on the second night.

(b1) If there was a shot on the first night, then a wife knowing of k_2 unfaithful husbands, with $k_2 \equiv 2 \pmod 3$ should shoot her husband on the second night.

(c00) If both first nights were silent then all wives shoot their husbands on the third night.

(c10) If there was a shot on the first night, and no shots on the second night, then the first night shooters shoot their husbands on the third night (if he is still alive).

Let us briefly check that this protocol is correct; i.e., we now show that if there is at least one unfaithful husband, then all unfaithful husbands are shot, and no faithful husbands are shot. We first consider the case where there is at least one faithful husband. Thus, if $n = k + 1$ is the number of unfaithful husbands, then some wives know of $k + 1$ unfaithful husbands, and some of k. If $k \equiv 2 \pmod 3$, then the wives whose husbands are faithful will shoot in the air on the first night. The cheated wives will shoot their husbands on the second night according to step (b1). If $k = 0$ then the cheated wife will shoot her husband on the first night and no other shooting occurs. If $k \equiv 0 \pmod 3$ and $k > 0$, then the cheated wives shoot in the air on the first night, the other wives are silent on the second night, and the cheated wives shoot their husbands on the third night. If $k \equiv 1 \pmod 3$ then the first night is silent, and the cheated wives shoot their husbands on the second night by (b0). We now need to show that if all wives are cheated then the husbands are shot. This is simple, since in all cases a wife that hears no shots other than on nights she shoots ends up shooting her husband (check!). $\bowtie$

Notice that Margaret could have appended the above protocol to Henrietta I's letter; using it, a cheated wife always shoots her husband on the midnight of the day she discovers his infidelity. In fact, a slightly more elaborate lower bound argument of a similar flavor shows that it is the only protocol Mary could have appended to Henrietta I's letter that is guaranteed to terminate in three nights. We remark that by slightly changing steps (a) and (c10) in the above protocol it is possible to obtain a protocol that works correctly even if there are no unfaithful husbands. (Of course, in the modified protocol a wife knowing of no unfaithful husband will not shoot her husband on the first night, and thus such a protocol cannot be appended to Henrietta I's letter.) Details are left to the reader.

5.7 Discussion

The cheating husbands problem is one in which communication, knowledge, and action interact in subtle ways. We have presented a case analysis of variants of this problem given different communication mediums and different degrees of clock synchronization. This problem demonstrates how sensitive the success of an operation can be to the known properties of the communication medium. It also shows how knowledge can be obtained in indirect ways by observing the actions of elements in the system, once we know something about how their actions are related to the facts they know. In fact, as we see in the bribery of the mailperson in Proposition 5.7, obtaining knowledge about the delivery times of a single letter can in some cases dramatically improve a wife's capability to act.

The queens' instructions in all cases can be viewed as ***knowledge-based protocols*** in the sense of [HF], since the actions that a wife is required to take depend on her knowledge. The basic high-level "knowledge-based" protocol that the wives follow is:

> ***Do not discuss the matter of your husband's fidelity with anyone. However, should you discover that your husband is unfaithful, you must shoot him on the midnight of the day you find out about it.***

Consider a scenario in which the queen's letter reaches all of the wives on the day it is sent. The actual way in which the above protocol will be carried out ("implemented") will depend on the known properties of the mail system. As our analysis shows, the elimination of the $n+1$ unfaithful husbands may take $n+b$ nights, it may take nb nights, and it might never happen at all, depending on whether the mail system is commonly known to be strongly synchronous, weakly synchronous, or asynchronous, and the order of message delivery is unknown. Thus, the execution of the protocol and its success depend not only on what actually happens (in this case, all letters being delivered on the same day); they also crucially depend on the wives' state of knowledge of what happens.

Another interesting point that arises here is that additional knowledge can in some cases be harmful. The results of Theorem 5.10 show that running the same knowledge-based protocol in a situation where the wives initially have strictly *more* knowledge (they know the order of delivery of a message broadcast around a ring) can result in a less desirable outcome. The ignorance present when the order of delivery is unknown gives rise to states of knowledge that allow the wives to all successfully determine whether their husbands are faithful.

CHAPTER 6

SYSTEMS OF UNRELIABLE PROCESSORS

Our analysis in the previous chapters focussed on how uncertainties in behavior of the communication medium and the relative readings of clocks affect the states of knowledge attainable and the actions that can be performed in the system. In this chapter we look at how the uncertainty regarding reliability of processors can affect the actions that can be performed in the system. The systems we consider in this chapter have a particularly simple and reliable structure in terms of clocks and communication mediums, but the processors in the system are unreliable. We will thus need to modify the formalism presented in Chapter 2 somewhat for the purposes of this analysis. Consequently, this chapter is to a large extent self-contained.

6.1 Introduction

The problem of designing effective protocols for distributed systems whose components are unreliable is both important and difficult. In general, a protocol for a distributed system in which all components are liable to fail cannot unconditionally guarantee to achieve non-trivial goals. In particular, if all processors in the system fail at an early stage of an execution of the protocol, then fairly little will be achieved regardless of what actions the protocol intended for the processors to perform. However, such universal failures are not very common in practice, and we are often faced with the problem of seeking protocols that will function correctly so long as the number, type, and pattern of failures during the execution of the protocol are reasonably limited. A requirement that is often made of such protocols is *t-resiliency* — that they be guaranteed to achieve a particular goal so long as no more than t processors fail.

A good example of a desirable goal for a protocol in an unreliable system is called *Simultaneous Byzantine Agreement* (SBA), a variant of the Byzantine agreement problem introduced in [PSL]:

Given are n processors, at most t of which might be faulty. Each processor p_i has an initial value $v_i \in \{0, 1\}$. Required is a protocol with the following properties:

1. Every non-faulty processor p_i irreversibly "decides" on a value $v \in \{0, 1\}$.
2. The non-faulty processors all decide on the same value.
3. The non-faulty processors all decide simultaneously, i.e., in the same round of computation.
4. If all initial values v_i are identical, then all non-faulty processors decide v_i.

Throughout this chapter we will use t to denote an upper bound on the number of faulty processors. We call a distributed system whose processors are unreliable a *Byzantine environment.*

The Byzantine agreement problem embodies some of the fundamental issues involved in the design of effective protocols for unreliable systems, and has been studied extensively in the literature (see [Fis] for a survey). Interestingly, although many researchers have obtained a good intuition for the Byzantine agreement problem, many aspects of this problem still seem to be mysterious in many ways, and the general rules underlying some of the phenomena related to it are still unclear.

what facts can become common knowledge at different points in the execution of a t-resilient protocol. We restrict our attention to systems in which communication is synchronous and reliable, and the only type of processor faults possible are *crash failures*: a faulty processor might crash at some point, after which it sends no messages at all. Despite the fact that crash failures are relatively benign, and dealing with arbitrary possibly malicious failures is often more complicated, work on the Byzantine agreement problem has shown that many of the difficulties of working in a Byzantine environment are already exhibited in this model. In the sequel we will use SBA as our standard example of a desirable simultaneous action.

Our analysis provides new insight into the basic issues involved in performing simultaneous actions in a Byzantine environment. For example, it shows that the pattern in which failures occur completely determines the number of rounds required to attain common knowledge of facts about the initial state of the system. Consequently, we obtain a complete characterization of the patterns of failures that require a t-resilient protocol for SBA to take k rounds, for $2 \leq k \leq t + 1$. This generalizes the well-known fact that SBA requires $t + 1$ rounds in the worst case (cf. [DLM],[DS],[CD],[FL],[H],[LF]). Our proof is a simplification of the well-known lower bound proof for SBA. Interestingly, our analysis immediately suggests a protocol for SBA that is *optimal in all runs*. That is, it halts as early as possible, given the pattern in which failures occur. In many cases this turns out to be much earlier than in any protocol previously known. This is the first protocol for SBA that is optimal in all runs. In fact, it is the first protocol for SBA that *ever* halts before the end of round $t+1$. The $t+1$ round lower bound on the worst case behavior of protocols for SBA has often been misinterpreted to mean that SBA cannot ever be reached in less than $t+1$ rounds.

The analysis presented in this chapter applies to a large class of simultaneous actions, not only to SBA. For example, we present the ***bivalent agreement*** problem, in which clause (4) of SBA is replaced by a requirement that the protocol have at least one run in which the processors decide 0, and at least one run in which they decide 1. We derive a protocol that always reaches bivalent agreement in two rounds. This contradicts a "folk conjecture" in the field that states that performing any non-trivial task simultaneously in a byzantine environment requires $t+1$ rounds in the worst case.

The main contribution of this chapter is to illustrate how a knowledge-based analysis of protocols in a Byzantine environment can provide insight into the fundamental properties of such systems. This insight is very useful in the design of improved t-resilient protocols for Byzantine agreement and many related problems. The analysis also provides some insight into how assumptions about the reliability of the system affect the states of knowledge attainable in the system. We briefly consider some other reliability assumptions and apply our analysis to them.

Section 6.2 contains the basic definitions and some of the fundamental properties of our model of a distributed system and of knowledge in a distributed system. Section 6.3 investigates the states of knowledge attainable in a particular fairly general protocol. Section 6.4 contains an analysis of the lower bounds corresponding to the analysis of Section 6.3, simplifying and generalizing the well-known $t+1$ round worst-case lower bound for reaching SBA. Section 6.5 discusses some applications of our analysis to problems related to SBA, and Section 6.6 includes some concluding remarks.

6.2 Definitions and preliminary results

In this section we present a number of basic definitions that will be used in the rest of the chapter, and discuss some of their implications. Our treatment will generally follow along the lines of Chapter 2, simplified and modified for our purposes.

We consider a synchronous distributed system consisting of a finite collection of $n \geq 2$ processors (automata) $\{p_1, p_2, \ldots, p_n\}$, each pair of which is connected by a two-way communication link. The processors share a discrete global clock that starts out at time 0 and advances by increments of one. Communication in the system proceeds in a sequence of *rounds*, with round k taking place between time $k - 1$ and time k. In each round, every processor first sends the messages it needs to send to other processors, and then it receives the messages that were sent to it by other processors in the same round. The identity of the sender and destination of each message, as well as the round in which it is sent, are assumed to be part of the message. At any given time, a processor's *message history* consists of the set of messages it has sent and received. Every processor p starts out with some *initial state* σ. A processor's *view* at any given time consists of its initial state, message history, and the time on the global clock. We think of the processors as following a *protocol*, which specifies exactly what messages each processor is required to send (and what other actions the processor should take) at each round, as a *deterministic* function of the processor's view. However, a processor might be *faulty*, in which case it might commit a stopping failure at an arbitrary round $k > 0$. If a processor *commits a stopping failure* at round k (or simply *fails* at round k), then it obeys its protocol in all rounds preceding round k, it does not send any messages in the rounds following k, and in round k it sends an arbitrary (not necessarily strict) subset of the messages it is required by its protocol to send. (Since a failed processor sends no further messages, we need not make any assumptions regarding what messages it receives in its failing round and in later rounds.) For technical reasons, we assume that once a processor fails, its view becomes a distinguished *failed* view. The set A of *active* processors at time k consists of all of the processors that did not fail in the first k rounds.

A *run* r of such a system is a complete history of its behavior, from time 0 until the end of time. This includes each processor's initial state, message history, and, if the processor fails, the round in which it fails. An *execution* (sometimes also called a *point*) is a pair (r, k), where r is a run and k is a natural number. We will use (r, k) to refer to the state of r after its first k rounds. Two executions (r, k) and (r', k) will be considered equal if all processors start in the same initial states and display the same behavior in the first k rounds of r and r'. The list of the processors' initial states is called the system's *initial configuration*. We denote processor p's view at (r, k) by $v(p, r, k)$. Furthermore, we will sometimes parameterize the set A of active processors by the particular execution, denoted $A(r, k)$.

We will find it useful to talk about the pattern in which failures occur in a given run. Formally, a *failure pattern* π is a set of triples of the form $\langle p, k(p), Q(p)\rangle$, where p is a processor, $k(p)$ is a round number, and $Q(p)$ is a set of processors. A run r *displays* (or, more precisely, *is consistent with*) *the failure pattern* π if every processor that fails in r is the first element of some triple in π, and for every triple $\langle p, k(p), Q(p)\rangle$ in π it is the case that processor p fails in round $k(p)$ of r, and in round $k(p)$ it sends no messages the processors in $Q(p)$, and does send messages to all processors not in $Q(p)$ to which the protocol prescribes it to send. A protocol $\mathcal{P}$, initial configuration σ, and failure pattern π uniquely determine a run. (However, a run of the protocol may be the result of more than one failure pattern in protocols that don't require all processors to send messages to all other processors in every round.) We denote this run by $\mathcal{P}(\sigma, \pi)$.

Following Chapter 2, we identify a distributed system with the set S of the possible runs of a particular fixed protocol $\mathcal{P} = (P(1), \ldots, P(n))$, where $P(i)$ is the part of the protocol followed by processor p_i. This set essentially encodes all of the relevant information about the execution of the protocol in the system. Given a system S, for $1 \leq i \leq n$ let Σ_i be the set of initial states that processor p_i assumes in the runs of S. The system S is said to be a *t-uniform system for* $\mathcal{P}$ if there is a set of initial configurations $\Gamma \subseteq \Sigma_1 \times \cdots \times \Sigma_n$ such that S is the set of all runs of the protocol $\mathcal{P}$ starting in initial configurations from Γ in which at most t processors fail. t-uniform systems have the property that a processor failure is an event that is independent of the initial configuration and of the time in which other processors fail. A system is said to be *independent* if its set of initial configurations is of the form $\Sigma_1 \times \cdots \times \Sigma_n$. In an independent t-uniform system there is no necessary dependence between the initial states of the different processors. The properties of t-resilient protocols can be studied by analyzing particular t-uniform systems for them. For example, a given protocol is a t-resilient protocol for SBA if all runs of the independent t-uniform system in which the set of possible initial configurations is $\{0, 1\}^n$ satisfy the requirements of SBA.

We assume the existence of an underlying logical language for representing *ground* facts about the system. By ground we mean facts about the state of the system that do not explicitly mention processors' knowledge. Formally, a ground fact φ will be identified with a set of executions $\tau(\varphi) \subseteq S \times N$, where N is the set of natural numbers. Given a run $r \in S$ of the system and a time k, we will say that φ holds at (r, k), denoted $(S, r, k) \models \varphi$, iff $(r, k) \in \tau(\varphi)$. We will define various ground facts as we go along. The set of executions corresponding to these facts will be clear from the context. We close this language under the standard boolean connectives $\wedge$, $\neg$ and $\supset$, interpreted as the standard conjunction, negation and implication.

Given a system S, we now define what facts a processor is said to "know" at any given point (r, k) for $r \in S$. Roughly speaking, p_i is said to know a fact ψ if ψ is guaranteed to hold, given p_i's view of the run. More formally, given a system S, we say that two points (r, k) and (r', k') are *p_i-equivalent* relative to S, denoted $(r, k) \overset{i}{\leftrightarrow} (r', k')$, iff $r, r' \in S$ and $v(p_i, r, k) = v(p_i, r', k')$. (The only case in which $v(p_i, r, k) = v(p_i, r', k')$ is possible for $k' \neq k$ is when $v(p_i, r, k) =$ *failed*.) We say that a processor p_i ***knows*** a fact ψ in S at (r, k), denoted $(S, r, k) \models K_i\psi$, if $(S, r', k') \models \psi$ for all executions $(r', k') \in S \times N$ satisfying $(r, k) \overset{i}{\leftrightarrow} (r', k')$. This definition of knowledge is essentially the ***total view*** interpretation of Chapter 2. We are about to review some of the properties of knowledge under this definition. Other properties will be covered in the sequel (see also Chapter 2 and [HM]).

A formula is said to be ***valid*** if it is true of all executions in all systems. Given a system S, a formula is said to be ***valid in*** S if it true of all executions of S. It follows that a valid fact is valid in S for all systems S. We now show that under our definition of knowledge, K_i satisfies the axioms of the modal system S5. This fact will follow in a straightforward way from the fact that knowledge is determined by the $\overset{i}{\leftrightarrow}$ relations, which in our case are equivalence relations.

Proposition 6.1:

a) If φ is valid in S then $K_i\varphi$ is valid in S.

b) The *consequence closure* axiom is valid:

$$\text{CONSEQUENCE CLOSURE:} \quad (K_i\varphi \wedge K_i(\varphi \supset \psi)) \supset K_i\psi.$$

c) The *knowledge axiom* is valid:

$$\text{KNOWLEDGE AXIOM:} \quad K_i\varphi \supset \varphi.$$

d) The *positive introspection* axiom is valid:

$$\text{POSITIVE INTROSPECTION:} \quad K_i\varphi \supset K_iK_i\varphi.$$

e) The ***negative introspection*** axiom is valid:

$$\text{NEGATIVE INTROSPECTION:} \qquad \neg K_i\varphi \supset K_i\neg K_i\varphi.$$

Proof: For part (a), let (r,k) be an (arbitrarily chosen) execution satisfying $r \in S$, and let φ be a formula that is valid in S. Thus φ is true of all executions $(r',k') \in S \times N$, and, in particular, φ is true of all executions in $S \times N$ that are p_i-equivalent to (r,k). It thus follows that $K_i\varphi$ is true of (r,k), and since (r,k) was an arbitrary execution in $S \times N$, we have that $K_i\varphi$ is valid in S. For (b), let $(S,r,k) \models K_i\varphi \wedge K_i(\varphi \supset \psi)$. Then by the definition of $(S,r,k) \models K_i\varphi$ we have that both φ and $(\varphi \supset \psi)$ hold at all points (r',k) that are p_i-equivalent to (r,k). It thus follows that ψ holds at all such points (r',k), and again by the definition of $(S,r,k) \models \psi$ we are done. Part (c) follows from the fact that $(r,k) \overset{i}{\leftrightarrow} (r,k)$, i.e., p_i-equivalence is reflexive. Now, by definition we have that if $K_i\varphi$ is true of (r,k) then φ is true of all executions that are p_i-equivalent to (r,k), and in particular φ is true of (r,k). For part (d), let $(S,r,k) \models K_i\varphi$. Thus, φ is true of all executions $(r'',k'') \overset{i}{\leftrightarrow} (r,k)$. We wish to show that $(S,r',k') \models K_i\varphi$ for all $(r',k') \overset{i}{\leftrightarrow} (r,k)$. Since $\overset{i}{\leftrightarrow}$ is an equivalence relation, all executions $(r'',k'') \in S \times N$ satisfy that $(r,k) \overset{i}{\leftrightarrow} (r'',k'')$ iff $(r',k') \overset{i}{\leftrightarrow} (r'',k'')$. It thus follows that φ is true of all executions $(r'',k'') \overset{i}{\leftrightarrow} (r',k')$, and we are done. The argument for part (e) is similar. If $(S,r,k) \models \neg\varphi$ then $(S,r,k) \not\models \varphi$, and therefore there must be an execution (r'',k'') that is p_i-equivalent to (r,k) of which φ is not true. Let (r',k') be an execution that is p_i-equivalent to (r,k). Because p_i-equivalence is an equivalence relation, we have that $(r',k') \overset{i}{\leftrightarrow} (r'',k'')$, and hence $(S,r',k') \models \neg K_i\varphi$. It now follows that $(S,r,k) \models K_i\neg K_i\varphi$ and we are done. ⋈

Roughly speaking, clauses (a) through (e) characterize the modal system S5. An operator satisfying clauses (a) through (d) is said to satisfy the modal system S4 (cf. [HM]). An interesting consequence of our choice of having a failed processor's view be a distinguished *failed* view is the fact that a processor always knows whether it is active. Furthermore, the only things that a failed processor knows are the consequences of the fact that the processor has failed and of the formulas that are valid in S. Given that a failed processor is "out of the game" in our model, we will focus our attention on the knowledge of the active processors.

Having defined knowledge for individual processors, we now extend this definition to states of group knowledge. Given a group $G \subseteq \{p_1,\ldots,p_n\}$, we first define G's ***view*** at (r,k), denoted $v(G,r,k)$:

$$v(G,r,k) \overset{\text{def}}{=} \{\langle p, v(p,r,k)\rangle : p \in G\}$$

Thus, roughly speaking, G's view is simply the joint view of its members. Extending our definition for individuals' knowledge, we say that *the group G has implicit knowledge* of φ at (r,k), denoted $(S,r,k) \models I_G\varphi$, if for all runs $r' \in S$ satisfying $v(G,r,k) = v(G,r',k)$ it is the case that $(S,r',k) \models \varphi$. Intuitively, G has implicit knowledge of φ if the joint view of G's members guarantees that φ holds. Notice that if processor p knows φ and processor q knows $\varphi \supset \psi$, then together they have implicit knowledge of ψ, even if neither of them knows ψ individually. An identical proof to that of Proposition 6.1 now shows:

Proposition 6.2: The operator I_G satisfies the modal system S5 (clauses (a) through (e) of Proposition 6.1, substituting I_G for K_i). ⋈

We refer the reader to Chapter 2 and [HM] for a discussion and a formal treatment of I_G. In this chapter we are mainly interested in states of knowledge of the group A of active processors. We say that *the set of active processors implicitly knows* φ, denoted $I\varphi$, exactly if $I_G\varphi$ holds for the set $G = A$. Stated more formally,

$$(S,r,k) \models I\varphi \text{ iff } (S,r,k) \models I_G\varphi \text{ for } G = A(r,k).$$

Although $I\varphi$ is defined in terms of $I_G\varphi$, it is not the case that I and I_G have the same properties. The reason for this is that whereas G is a fixed set, membership in A may vary over time and differs from one run to another. Thus, for example, it is often the case that for $G = A(r,k)$ we have $(S,r,k) \not\models I_G(A = G)$, because there is some run $r' \in S$ such that $v(G,r,k) = v(G,r',k)$ and where G is a strict subset of $A(r',k)$. Consequently, whereas the negative introspection axiom for I_G, i.e., $\neg I_G\varphi \supset I_G\neg I_G\varphi$, is valid, the corresponding formula for I: $\neg I\varphi \supset I\neg I\varphi$, is not valid! (Notice, however, that $I(G \subseteq A)$ holds whenever $G \subseteq A$). For example, it may be the case that processor p_j sends processor p_i a message in round 1 stating p_j's initial state, and fails before sending any other message, and that processor p_i fails in round 1 after sending all of its round 1 messages. Processor p_j's initial state is thus not implicitly known to the set of active processors, but it is consistent with the active processors' joint view that p_i is active, in which case p_j's initial state would be implicitly known. The above discussion can be summarized by:

Proposition 6.3: The implicit knowledge operator I satisfies the modal system S4 (i.e., clauses (a) — (d) of Proposition 6.1). The negative introspection axiom is not valid for I.

⋈

The following lemma describes the relationship between K_i and I:

Lemma 6.4: Let φ be a formula and let $p_i \in A(r,k)$.

a) If $(S,r,k) \models K_i\varphi$ then $(S,r,k) \models I\varphi$.

b) If $(S,r,k) \models K_i\varphi$ then $(S,r,k) \models K_iI\varphi$.

Proof: For part (a), assume that $(S,r,k) \models K_i\varphi$, and let (r',k') be an execution satisfying $v(A(r,k),r',k') = v(A(r,k),r,k)$. In particular, since $p_i \in A(r,k)$ we have that $v(p_i,r',k') = v(p_i,r,k)$, and thus since $K_i\varphi$ holds at (r,k), we have that φ holds at (r',k'). Since this is true for all such executions (r',k'), we are done by the definition of $(S,r,k) \models I\varphi$. For (b), let $(S,r,k) \models K_i\varphi$. By Proposition 6.1(d) we have that $(S,r,k) \models K_iK_i\varphi$. The fact that $p_i \in A(r,k)$ implies that $v(p_i,r,k) \neq$ *failed*. Thus, p_i is an active processor in all executions that are p_i-equivalent to (r,k). Let $(r',k') \overset{i}{\leftrightarrow} (r,k)$. We thus have that $p_i \in A(r',k')$, and that $K_i\varphi$ holds at (r',k'). Part (a) therefore implies that $I\varphi$ holds at (r',k'), and thus $K_iI\varphi$ holds at (r,k). $\bowtie$

We now show that, roughly speaking, in t-uniform systems once a fact about the past is not implicitly known it is lost forever; it will not become implicit knowledge at a later time. We say that a fact ψ is ***about the first k rounds*** if for all runs $r \in S$ it is the case that $(S,r,k) \models \psi$ iff $(S,r,\ell) \models \psi$ for all $\ell \geq k$. In particular, facts about the first 0 rounds are facts about the initial configuration. We now have:

Theorem 6.5: Let S be a t-uniform system, let ψ be a fact about the first k rounds, and let $\ell > k$. If $(S,r,k) \not\models I\psi$ then $(S,r,\ell) \not\models I\psi$.

Proof: Let $\ell > k$, and let r and ψ be such that ψ is about the first k rounds and $(S,r,k) \not\models I\psi$. Let $G = A(r,k)$. It follows that there exists a run $r' \in S$ such that $v(G,r,k) = v(G,r',k)$, and $(S,r',k) \not\models \psi$. Let r'' be a run with the following properties: (i) $(r'',k) = (r',k)$; (ii) all processors in $A(r',k) - G$ fail in round $k+1$ of r'' before sending any messages; and (iii) from round $k+1$ on all processors in G behave in r'' exactly as they do in r. By construction, the number of processors that fail by time k in r'' is no larger than the number in r, and exactly the same processors fail in r and in r'' by any later time. Given that S is a t-uniform system and $r \in S$, no more than t processors fail in r. It follows that $r'' \in S$, since all of the processors follow the same protocol in r'' and in r, and no more than t processors fail in r''. By construction of r'' we also have that $A(r'',\ell) = A(r,\ell)$ and that the active processors have identical views in (r'',ℓ) and in (r,ℓ). It follows that $(S,r'',\ell) \models I\psi$ iff $(S,r,\ell) \models I\psi$. Since ψ is a fact about the first k rounds and $(r'',k) = (r',k)$, we have that $(S,r'',\ell) \not\models \psi$ because $(S,r',k) \not\models \psi$. Thus, in particular, $(S,r'',\ell) \not\models I\psi$ and it follows that $(S,r,\ell) \not\models I\psi$ and we are done. $\bowtie$

Fagin and Vardi perform an interesting analysis of implicit knowledge in reliable systems (cf. [FV]). Among other things, they prove that the set of facts that are implicit knowledge about the initial configuration does not change with time. I.e., in reliable systems the implication in the statement of the Theorem 6.5 becomes an equivalence. However, in t-uniform Byzantine systems it is clearly the case that implicit knowledge can be "lost". For example, if processor p_i may start in initial states σ and σ', and in a particular run of the system p_i starts in state σ and fails in the first round before sending any messages, then whereas $I($"p_i started in state σ"$)$ holds at time 0, it does not hold at any later time.

We now introduce the two other states of group knowledge that are central to our analysis. We define "everyone knows" and "common knowledge" along the lines of Chapter 2. In our case, however, these notions will be defined for the set of active processors. ***Every (active) processor knows*** φ, denoted $E\varphi$, is defined by

$$E\varphi \overset{\text{def}}{=} \bigwedge_{1\leq i\leq n} (p_i \in A \supset K_i\varphi).$$

An immediate corollary of Lemma 6.4 which we will find useful in the sequel is:

Corollary 6.6: $E\varphi \supset E(I\varphi)$ is valid.

⋈

We define $E^1\varphi \overset{\text{def}}{=} E\varphi$, and $E^{m+1}\varphi \overset{\text{def}}{=} E(E^m\varphi)$ for $m \geq 1$. A fact φ is said to be ***common knowledge among the active processors***, denoted $C\varphi$, if $E^m\varphi$ holds for all $m \geq 1$. More formally,

$$C\varphi = \varphi \wedge E\varphi \wedge E^2\varphi \wedge \cdots \wedge E^m\varphi \wedge \cdots .$$

Common knowledge among the active processors, which we will call simply common knowledge, will play a crucial role in the sequel. We now study some of its properties. A useful tool for thinking about $E^m\varphi$ and $C\varphi$ is the labelled undirected graph whose nodes are the executions of a system S, and whose edges are the $\overset{i}{\leftrightarrow}$ relations, restricted so that an edge $e \overset{i}{\leftrightarrow} e'$ exists only if p_i is active in e (and hence also in e'). (This graph is precisely the ***Kripke structure*** modelling the active processors' knowledge in the system; cf. [HM].) The ***distance*** between two executions $e = (r,k)$ and $e' = (r',k)$ in this graph, denoted $\delta(e,e')$, is simply the length of the shortest path in the graph connecting e and e'. If there is no path connecting e to e', then $\delta(e,e')$ is defined to be infinity. Two executions e and e' are said to be ***similar***, denoted $e \sim e'$ if $\delta(e,e')$ is finite (i.e., if e' and e are in the same connected component of the graph). Equivalently, $(r,k) \sim (r',k)$, if for some finite m there are runs $r_1, r_2, \ldots, r_{m-1} \in S$, and processors $p_{i_1}, p_{i_2}, \ldots, p_{i_m}$, satisfying $p_{i_j} \in A(r_j, k)$ for $j \leq m-1$, $p_{i_m} \in A(r', k)$, and

$$(r,k) \overset{i_1}{\leftrightarrow} (r_1,k) \overset{i_2}{\leftrightarrow} \cdots \leftrightarrow (r_{m-1},k) \overset{i_m}{\leftrightarrow} (r',k).$$

(The system S is usually clear from context, and thus we do not add a subscript S to $\sim$.) It is now easy to check that $(S,r,k) \models E\varphi$ iff $(S,r',k) \models \varphi$ for all executions (r',k) of distance ≤ 1 from (r,k). Notice that similarity is an equivalence relation. We can now show:

Proposition 6.7:

a) $(S,r,k) \models C\varphi$ iff $(S,r',k) \models \varphi$ for all $r' \in S$ such that $(r,k) \sim (r',k)$.

b) If φ is valid in S then $C\varphi$ is valid in S.

c) C satisfies the axioms of the modal system S5 (see Proposition 6.1).

d) The ***induction*** axiom is valid:

$$\text{INDUCTION AXIOM:} \qquad C(\varphi \supset E\varphi) \supset (\varphi \supset C\varphi).$$

e) The ***fixpoint*** axiom is valid:

$$\text{FIXPOINT AXIOM:} \qquad C\varphi \supset \varphi \wedge EC\varphi.$$

Proof: (a) follows by a straightforward induction on m showing that $(S,r,k) \models E^m\varphi$ iff $(S,r',k) \models \varphi$ for all (r',k) of distance $\leq m$ from (r,k). Part (b) follows directly from (a). The proof of part (c) is identical to the proof of Proposition 6.1, substituting C for K_i and $\sim$ for $\overset{i}{\leftrightarrow}$. For (d), assume that both φ and $C(\varphi \supset E\varphi)$ hold at $e = (r,k)$. We prove by induction on m that φ holds at all points of distance $\leq m$ from e. The case $m = 0$ follows from our initial assumption. Assume that the claim holds for m, and let e' be a point satisfying $\delta(e,e') = m+1$. It follows that there is a point e'' such that $\delta(e,e'') = m$ and $\delta(e'',e') = 1$. By the inductive hypothesis φ holds at e''. Since $C(\varphi \supset E\varphi)$ holds at e and $e \sim e''$, part (a) implies that $\varphi \supset E\varphi$ holds at e''. It follows that $E\varphi$ holds at e'', and since $\delta(e'',e') = 1$, we have that φ holds at e'. By induction we have that φ holds at all points reachable from (i.e., similar to) e, and by part (a) we have that $C\varphi$ holds at e, and we are done. For part (e), the validity of $C\varphi \supset \varphi$ is immediate. By part (c) we have that C satisfies the positive introspecton axiom, and hence $C\varphi \supset CC\varphi$ is valid. By definition of $C\psi$ we have that $C\psi \supset E\psi$ is valid, and taking $\psi = C\varphi$, we thus have that $C\varphi \supset CC\varphi \supset EC\varphi$ is valid, and we are done. ⋈

Proposition 6.7 is very useful in relating common knowledge and actions that are guaranteed to be performed simultaneously. For example, we can use Proposition 6.7(b) and the induction axiom in order to relate the ability or inability to attain common knowledge of certain facts with the possibility or impossibility of reaching simultaneous Byzantine agreement. We model a processor's "deciding v" by the processor sending the message "the decision value is v" to itself, and have:

Corollary 6.8: Let S be a system in which the processors follow a protocol for SBA. If the active processors decide on a value v at (r, k), then

a) $(S, r, k) \models C($"All processors are deciding v"$)$, and

b) $(S, r, k) \models C($"At least one processor had v as its initial value"$)$.

Proof: Let φ be the fact "all processors are deciding v". Given that the protocol guarantees that SBA is attained in S, it is the case that whenever some processor decides v all active processors do, and thus the formula $\varphi \supset E\varphi$ is valid in S. Thus, by Proposition 6.7(b) $C(\varphi \supset E\varphi)$ is also valid in S. The induction axiom states that $C(\varphi \supset E\varphi) \supset (\varphi \supset C\varphi)$. Combining these two facts we have that $\varphi \supset C\varphi$ is valid in S, and thus if $(S, r, k) \models \varphi$ then $(S, r, k) \models C\varphi$ and we are done with part (a). For (b), let ψ be "at least one processor had v as its initial value", and notice SBA guarantees that $\varphi \supset \psi$ is valid in S. Thus, by Proposition 6.7(b), so is $C(\varphi \supset \psi)$. The consequence closure axiom states that $(C\varphi \wedge C(\varphi \supset \psi)) \supset C\psi$ is valid, and we conclude that $C\varphi \supset C\psi$ is valid in S. By part (a) we have that $(S, r, k) \models \varphi$ implies that $(S, r, k) \models C(\varphi)$, from which we can now conclude that $(S, r, k) \models C\psi$ and we are done. $\bowtie$

The reasoning used in proof of Corollary 6.8 is typical of the way Proposition 6.7(a) and (b) together with the consequence closure and induction axioms are used to prove that certain facts are common knowledge. We will use such reasoning again in later proofs.

6.3 Analysis of a simple protocol

In this section we take a close look at t-uniform systems $S_{\hat{P}}$ in which all processors follow a simple and fairly general protocol $\hat{\mathcal{P}}$:

For $k \geq 0$, in round $k+1$ each processor sends its view at time k (i.e., after k rounds) to all other processors.

This protocol was named the *maximal information protocol* by Hadzilacos (cf. [H]). We are interested in determining what facts about the run become common knowledge at the different stages of the execution of this protocol. Intuitively, the protocol $\hat{\mathcal{P}}$ should provide the processors with "as much knowledge as possible" about the initial configuration and the pattern of failures, and should facilitate the ability of the system to perform actions that depend on the initial configuration. One of the relevant properties of this protocol is that it requires every processor to send messages to all other processors in every round. This ensures that the failure of a processor will be known to all processors at most one round after the round in which the processor fails.

A fact φ is called ***stable*** if once it becomes true it remains true forever (cf. Chapter 2). For example, facts about the first k rounds, and in particular facts about the system's initial configuration, are stable. Since a processor's knowledge is based on the processor's view, and an active processor's view grows monotonically with time, it is the case that if φ is stable then (as long as at least one processor remains active) so are $E\varphi$ and $C\varphi$. As we have seen, $I\varphi$ is not necessarily stable.

A round in which no processor fails is called a ***clean*** round. A round that is not clean is called ***dirty***. If every processor that fails in round k fails to send round k messages only to processors that are not active at time k, then round k is said to be ***seemingly clean***. Notice that a clean round is in particular seemingly clean, and the active processors cannot distinguish between a clean round and a seemngly clean round. The reason we are interested in seemingly clean rounds is that if, for some k, round k of a run in which the processors all follow $\hat{\mathcal{P}}$ is seemingly clean, then every active processor's view at the end of round k includes the view of the active processors at time $k-1$. In particular it follows that any stable fact that is implicit knowledge at time $k-1$ is known to everyone at time k. Consequently, at time k all processors know exactly the same facts about the initial configuration. Furthermore, Theorem 6.5 together with the fact that $E\varphi$ is stable when φ is, imply that at any point after a clean round, all of the processors have identical knowledge about the initial configuration. Therefore, once it is common knowledge that there was a clean round, it is common knowledge that the processors have an identical view of the initial configuration. The above discussion is made precise by the fol!ɔwing theorem:

Theorem 6.9: Assume that $t \leq n-1$.

a) Let φ be a stable fact such that $(S_{\phi}, r, k-1) \models I\varphi$.
If round k of r is seemingly clean then $(S_{\phi}, r, k) \models E\varphi$.

b) Let φ be a fact about the initial configuration.
If $(S_{\phi}, r, \ell) \models C($"a seemingly clean round has occurred"$)$ then $(S_{\phi}, r, \ell) \models I\varphi$ iff $(S_{\phi}, r, \ell) \models C\varphi$.

Proof: By definition, $(S_{\phi}, r, k-1) \models I\varphi$ iff $(S_{\phi}, r, k-1) \models I_G\varphi$ for $G = A(r, k-1)$. If round k is seemingly clean then all processors active at time k receive round k messages from all of the processors in G, and hence the view of each active processor at time k has a copy of $v(G, r, k-1)$, and it follows that every active processor at time k knows φ. For part (b), let φ be a fact about the initial configuration and let ψ be the fact *"a seemingly clean round has occured"*. Let (r', ℓ) be an execution satisfying $(S_{\phi}, r', \ell) \models \psi$. By Theorem 6.5, if $(S_{\phi}, r', \ell) \models I\varphi$ then $(S_{\phi}, r', k) \models I\varphi$ for all $k \leq \ell$. Given that ψ holds at (r', k), let round k of r' be a seemingly clean round, where $k \leq \ell$. Since $(S_{\phi}, r', k-1) \models I\varphi$, by part (a) we have that $(S_{\phi}, r', k) \models E\varphi$. $E\varphi$ is stable because φ is, and therefore $(S_{\phi}, r, \ell) \models E\varphi$. By Corollary 6.6 we have that $(S_{\phi}, r', \ell) \models E(I\varphi)$. We have just shown that $\psi \supset (I\varphi \supset E(I\varphi))$ is valid in S_{ϕ}. Thus, by Proposition 6.7(b) we have that $C(\psi \supset (I\varphi \supset E(I\varphi)))$ is also valid in S_{ϕ}. Now assume that r is a run satisfying $(r, \ell) \models C\psi$. By the consequence closure axiom for C (Proposition 6.7(c)), we have that $(S_{\phi}, r, \ell) \models C(I\varphi \supset E(I\varphi))$. And by the induction axiom we have that $(S_{\phi}, r, \ell) \models I\varphi \supset C(I\varphi)$. Since $C(I\varphi) \supset C\varphi$ is valid, we also have that $(S_{\phi}, r, \ell) \models I\varphi \supset C\varphi$. Finally, since $C\varphi \supset I\varphi$ is valid, we have that $(S_{\phi}, r, \ell) \models I\varphi \equiv C\varphi$, and we are done. $\bowtie$

As a corollary of Theorem 6.9 we can now show:

Corollary 6.10: Let φ be a fact about the initial configuration.

a) $(S_{\phi}, r, t+1) \models I\varphi$ iff $(S_{\phi}, r, t+1) \models C\varphi$.

b) $(S_{\phi}, r, n-1) \models I\varphi$ iff $(S_{\phi}, r, n-1) \models C\varphi$.

Proof: Notice that the "if" direction in both cases is immediate, since $C\psi \supset I\psi$ is valid for all facts ψ. We now show the other direction. All runs of S_{ϕ} have the property that no more than t processors fail during the run. Given that a processor falure occurs in a unique round, we have that one of the first $t+1$ rounds of every run of S_{ϕ} must be clean. Since a clean round is in particular seemingly clean, Proposition 6.7(b) implies that at time $t+1$ it is common knowledge in all runs of $S_{\mathcal{P}}$ that a seemingly clean round has occured. Part (a) now follows from Tbeorem 6.9(b). For the proof of part (b), we need a slightly stronger variant of Theorem 6.9(b), which states that if it is common knowledge that there has either been a clean round or that there is at most one processor then $I\varphi$ holds iff $C\varphi$ does. The proof of this fact is completely analogous to that of Theorem 6.9(b), given that $I\varphi \equiv C\varphi$ is trivially true when there is at most one active processor. ⋈

As a consequence of Theorem 6.9 and Corollary 6.10 we have that any action that depends on the system's initial configuration can be carried out simultaneously in a consistent way by the set of active processors at any time $k \geq \min\{t+1, n-1\}$. This is consistent with the fact that there are well-known t-resilient protocols for SBA that attain SBA in $t+1$ rounds. Interestingly, none of the known protocols for SBA attain SBA in less than $t+1$ rounds in *any* run. It is therefore natural to ask whether a protocol for SBA can ever attain SBA in less than $t+1$ rounds. Clearly, once it is common knowledge that a clean round has occurred, SBA can be attained. And as we shall see, there are cases in which the existence of a clean round becomes common knowledge before time $t+1$. *When* the existence of a clean round becomes common knowledge depends crucially on the pattern of failures, and on the time in which failures become implicitly known to the group of active processors. For example, if a processor p detects t failures in the first round of a run of $\hat{\mathcal{P}}$, then the second round of the run will be clean, and at the end of the second round all active processors will know that p detected t failures in round 1. It follows from the induction axiom and Proposition 6.7(b) that at the end of round 2 it will be *common knowledge* that all processors have an identical view of the initial configuration (check!). Clearly, the processors can then perform any action that depends on the initial configuration (e.g., SBA) in a consistent way. In the remainder of this section we show a class of runs of $S_{\mathcal{P}}$ in which the processors attain common knowledge of an identical view of the initial configuration at time k, for every k between 2 and $t+1$. In the next section, we will prove that this is in fact a precise classification of the runs according to the time in which common knowledge of an identical view of the initial configuration is attained.

Intuitively, if there are more than k failures by the end of round k, then from the point of view of the ability to delay the first clean round, failures have been "wasted". In particular, if for some k it is the case that there are $k+j$ failures by the end of round k, then there must be a clean round before time $t+1-j$ (in fact, between round $k+1$ and round $t+1-j$). This motivates the following definitions: We denote the number of processors that fail by time k in r by $N(r,k)$. We define the *difference at* (r,k), denoted $d(r,k)$, by

$$d(r,k) \overset{\text{def}}{=} N(r,k)-k.$$

We also define the *maximal difference in* (r,ℓ), denoted $D(r,\ell)$, by

$$D(r,\ell) \overset{\text{def}}{=} \max_{k\leq\ell} d(r,k).$$

Observe that $d(r,0)=0$ for all runs r, since $N(r,0)=0$. Furthermore, in a t-uniform system it is always the case that $d(r,k)\leq t-k$, since $N(r,k)\leq t$. Let D be a variable whose value at a point (r,k) is $D(r,k)$, and let $d(k)$ be a variable whose value at any point in r is $d(r,k)$. By Theorem 6.9(b) we have that if at time $t+1-j$ it is common knowledge that $D\geq j$, then it is common knowledge that a clean round has occurred, and that all processors have an identical view of the initial configuration. We are about to show that the protocol $\hat{\mathcal{P}}$ guarantees that if it ever becomes implicit knowledge that $D\geq j$ then at time $t+1-j$ it is common knowledge that $D\geq j$ (and, therfore, that a clean round has occurred). This leads us to the following definition: Given a system S, the *wastefulness* of (r,ℓ) with respect to S, denoted $\mathcal{W}(S,r,\ell)$, is defined by:

$$\mathcal{W}(S,r,\ell) \overset{\text{def}}{=} \max\{j : (S,r,\ell)\models I(D\geq j)\}.$$

In words, the wastefulness of (r,ℓ) is the maximal value that the difference $d(r,\cdot)$ is implicitly known to have assumed by time ℓ. Finally, we define the *wastefulness* of a run r, denoted $\mathcal{W}(S,r)$, by:

$$\mathcal{W}(S,r) \overset{\text{def}}{=} \max_{\ell\geq 0}\mathcal{W}(S,r,\ell).$$

We now formally prove the claims informally stated above. We start with a somewhat technical lemma discussing the properties of wastefulness in the case of $S_{\hat{\mathcal{P}}}$:

Lemma 6.11: Let $t \leq n-1$.

a) If $(S_{\hat{\mathcal{P}}}, r, \ell) \models I(D \geq j)$ then $(S_{\hat{\mathcal{P}}}, r, \ell) \models I(d(k) \geq j)$ for some $k \leq \ell$.

b) If $I(d(k) \geq j)$ holds at time k then at time $k+1$ either $E(d(k) \geq j)$ holds or $I(d(k+1) \geq j)$ does.

c) $\mathcal{W}(S_{\hat{\mathcal{P}}}, r, k+1) \geq \mathcal{W}(S_{\hat{\mathcal{P}}}, r, k)$ for all $k \geq 0$.

Proof: For part (a), let $r \in S_{\hat{\mathcal{P}}}$ satisfy $(S_{\hat{\mathcal{P}}}, r, \ell) \models I(D \geq j)$, and assume that for no k is it the case that $(S_{\hat{\mathcal{P}}}, r, \ell) \models I(d(k) \geq j))$. Let r' be a run of $\hat{\mathcal{P}}$ such that $(r', 0) = (r, 0)$, and in which the only messages not delivered are those that are implicitly known at (r, ℓ) not to have been delivered. It is easy to check that $r' \in S_{\hat{\mathcal{P}}}$, since no more than t processors fail in r'. Because it is not implicit knowledge at (r, ℓ) that $d(k) \geq j$ for any k, it follows that $D(r', \ell) < j$. If we show that the group $G = A(r, \ell)$ has exactly the same view in (r, ℓ) and in (r', ℓ) we will be done, since this will contradict the assumption that $(S_{\hat{\mathcal{P}}}, r, \ell) \models I(D \geq j)$. We now prove that $A(r, \ell)$ has the same view in (r, ℓ) and in (r', ℓ). This is done by showing by induction on k that the set of processors that are implicitly known at (r, ℓ) to have been active at time $k \leq \ell$ have the same views at time k in both r and r'. Define $G(\ell) = A(r, \ell)$. For $k < \ell$, assume inductively that $G(k+1)$ is defined, and for all processors $p_i \in G(k+1)$ let $g(p_i, k)$ be the set of processors from which p_i receives a message in round $k+1$ of r. Define

$$G(k) \stackrel{\text{def}}{=} \bigcup_{p_i \in G(k+1)} g(p_i, k).$$

Let $G'(\ell) = G(\ell)$, and for $k < \ell$ define $g'(p_i, k)$ and $G'(k)$ from $G'(k+1)$ in an analogous fashion (substituting G, g, and r by G', g', and r'). We now show by induction on $\ell - k$ that if $k < \ell$ then for all $p_i \in G(k+1)$ we have that $g(p_i, k) = g'(p_i, k)$ and that $G(k) = G'(k)$. Let $k < \ell$ and assume inductively that $G(k+1) = G'(k+1)$. (Notice that we have defined $G(\ell) = G'(\ell)$.) Let $p_i \in G(k+1)$. The sets $G(k)$ are the sets of processors implicitly known at (r, ℓ) to have been active at time k. The sets $g(p_i, k-1)$ are the sets of processors that send a message to p_i in round k. By requiring messages to contain the sender's complete view, the protocol $\hat{\mathcal{P}}$ guarantees that a processor is implicitly known at (r, ℓ) to have been active at time k iff the processor's view at (r, k) is implicitly known. Thus, the precise identity of $g(p_i, k)$ for $p_i \in G(k+1)$ is implicitly known at (r, ℓ). It follows that processor p_j sends a message to p_i in round $k+1$ of r iff p_j sends p_i a round $k+1$ message in r'. It thus follows that $g(p_i, k) = g'(p_i, k)$. Since this is true for all $p_i \in G(k+1)$, we have that $G(k) = G'(k)$, and the claim is proven. Notice that $G(k) \supseteq G(k+1)$. We now show by induction on k that for all $p_i \in G(k)$ it is the case that $v(p_i, r, k) = v(p_i, r', k)$. The case $k = 0$ follows from the fact that $(r, 0) = (r', 0)$ and $G(0) = G'(0)$. Assume inductively the claim holds for k; we prove it for $k+1$. Let $p_i \in G(k+1)$. Observe that p_i's view at $(r, k+1)$ is determined by its view at (r, k) and by the view of the group $g(p_i, k)$ at (r, k). Since by the inductive hypothesis we have that $g(p_i, k) = g'(p_i, k)$, and that $v(g(p_i, k), r, k) = v(g'(p_i, r', k)$, and that $v(p_i, r, k) = v(p_i, r', k)$, it follows that $v(p_i, r, k+1) = v(p_i, r', k+1)$. It now follows that $v(G(\ell), r, \ell) = v(G(\ell), r', \ell)$, and we are done with part (a).

For part (b), assume that $(S_{\hat{\mathcal{P}}}, r, k) \models I(d(k) \geq j)$. If round $k+1$ is seemingly clean then $E(d(k) \geq j)$ holds at $(r, k+1)$. Otherwise there must be (at least one) processor, say q, that fails in round $k+1$ by not sending a message to at least one processor, say p, that is active at time $k+1$. Thus, in particular, p knows at time $k+1$ that q has failed. Now, by requiring all processors to send messages to all of the other processors in every round, $\hat{\mathcal{P}}$ ensures that all processors that fail by (r, k) are known by everyone at $(r, k+1)$ to have failed. It follows that $d(k+1) \geq j$ is implicit knowledge at that time.

For part (c), assume that $\mathcal{W}(r, k) = j$. Then by part (a) there is some $k' \leq k$ such that $(S_{\hat{\mathcal{P}}}, r, k) \models I(d(k') \geq j)$. Without loss of generality let k' be the largest such number. If $k' < k$ then by (b) we have that at time $k'+1 \leq k$ everyone knows that $d(k') \geq j$. But $E(d(k') \geq j)$ is a stable fact because $d(k') \geq j$ is, and in this case $\mathcal{W}(r, k+1) \geq j$, and the claim of (c) holds. If $k' = k$ then part (b) implies that at time $k+1$ either everyone will know that $d(k) \geq j$ or it will be implicit knowledge that $d(k+1) \geq j$. In both cases we will have $\mathcal{W}(r, k+1) \geq j$, and we are done. $\bowtie$

We now have:

Theorem 6.12: Let $t \leq n-1$.

a) $\mathcal{W}(S_{\hat{\mathcal{P}}}, r) \geq j$ iff $(S_{\hat{\mathcal{P}}}, r, t+1-j) \models C(\mathcal{W}(S_{\hat{\mathcal{P}}}, \text{“the current run”}) \geq j)$.

b) Let φ be a fact about the initial configuration. If $\mathcal{W}(S_{\hat{\mathcal{P}}}, r) = j$ then $(S_{\hat{\mathcal{P}}}, r, t+1-j) \models I\varphi$ iff $(S_{\hat{\mathcal{P}}}, r, t+1-j) \models C\varphi$.

Proof: The "if" direction of part (a) is immediate from the fact that $C\varphi \supset \varphi$ is valid. We now show the other direction. Assume that $\mathcal{W}(S_\phi, r) \geq j$. We claim that there must be a seemingly clean round between the first time in which $I(D \geq j)$ first holds and time $t+1-j$. For some $\ell \geq 0$ it must be the case that $\mathcal{W}(S_\phi, r, \ell) \geq j$, and hence $(S_\phi, r, \ell) \models I(D \geq j)$. By Lemma 6.11(a) there is some $k \leq \ell$ for which $(S_\phi, r, \ell) \models I(d(k) \geq j)$. Let k' be the largest such k. Since $d(k') \geq j$ is a fact about the first k' rounds, we have by Theorem 6.5 that $(S_\phi, r, k') \models I(d(k') \geq j)$. Since $d(k') \geq j$ implies that at least $k'+j$ processors must have failed by time k', we have that $k' \leq t-j$. Furthermore, $(S_\phi, r, k'+1) \not\models I(d(k'+1) \geq j)$ implies that no new processor failure becomes visible to the active processors in round $k'+1$, which implies that round $k'+1$ must be seemingly clean. Since "$d(k') \geq j$" is a stable fact, it follows from Theorem 6.9(a) that $(S_\phi, r, k'+1) \models E(d(k') \geq j)$, and hence that $(S_\phi, r, \ell) \models E(d(k') \geq j)$ for all $\ell \geq k'+1$. In particular, since $t+1-j \geq k'+1$, we have that $(S_\phi, r, t+1-j) \models E(d(k') \geq j)$. Let ψ be the fact "$\mathcal{W}(S_\phi$, "the current run"$) \geq j$. By Corollary 6.6 we have that $E(d(k') \geq j) \supset E(I(d(k') \geq j))$, and since $(d(k') \geq j) \supset \psi$ is valid, we also have that $(S_\phi, r, t+1-j) \models E\psi$. It follows that $(S_\phi, r', t+1-j) \models \psi \supset E\psi$ for all runs $r' \in S_\phi$. Given that $t \leq n$, the only executions that are similar to an execution $(r', t+1-j)$ are of the form $(r'', t+1-j)$. Thus, by Proposition 6.7(a) we have that $(S_\phi, r', t+1-j) \models C(\psi \supset E\psi)$ for all $r' \in S_\phi$, and the induction axiom implies that all executions $(r, t+1-j)$ satisfy $\psi \supset C\psi$, which is the claim of part (a). For part (b), recall from the proof of part (a) that if $D \geq j$ then there must be a clean round by time $t+1-j$. By part (a), if $\mathcal{W}(S_\phi, r) = j$ then at time $t+1-j$ it is common knowledge that $I(D \geq j)$ and therefore in particular that $D \geq j$. It follows that at time $t+1-j$ it is common knowledge that a clean round (and hence also a seemingly clean round) has occurred. The claim now follows from Theorem 6.9(b). ⋈

Thus, certain patterns of failures help the processors to reach common knowledge of an identical view of the initial configuration early. In particular, if the wastefulness of the run is j, then the active processors obtain common knowledge of a common view of the initial configuration at time $t+1-j$. We now make precise our heretofore informal claim that it is the pattern of failures that determines the wastefulness of the runs of S_ϕ. Given a system S, a fact φ is said to be ***about the failure pattern*** $(S, r, k) \models \varphi$ iff $(S, r', k') \models \varphi$ for all runs $r, r' \in S$ that have the same failure pattern. Observe that $d(k)$ and D are facts about the failure pattern by this definition. We can now show:

Lemma 6.13: Let φ be a fact about the failure pattern. Let σ and σ' be initial configurations, let π be a failure pattern, and let $r = \hat{\mathcal{P}}(\sigma, \pi)$ and $r' = \hat{\mathcal{P}}(\sigma', \pi)$. Then $(S_{\hat{\mathcal{P}}}, r, \ell) \models I\varphi$ iff $(S_{\hat{\mathcal{P}}}, r', \ell) \models I\varphi$, for all $\ell \geq 0$.

Sketch of proof: Assume that $(S_{\hat{\mathcal{P}}}, r', k) \not\models I\varphi$, and let $G = A(r', k)$. It follows that there is a run r'' such that $v(G, r', k) = v(G, r'', k)$, and $(S_{\hat{\mathcal{P}}}, r'', k) \not\models \varphi$. Let Q be the set of processors on whose initial states σ and σ' disagree. Clearly $v(G, r', k)$ contains the view at time 0 (i.e., initial state) of none of the processors in Q. Thus, without loss of generality $r'' = \hat{\mathcal{P}}(\sigma', \pi'')$ for some π''. An inductive argument along the lines of the proof of Lemma 6.11(a) will now show that $v(G, r, k) = v(G, \hat{\mathcal{P}}(\sigma, \pi''), k)$. (Note that $A(r, k) = A(r', k) = G$).) But because φ is a fact about the failure pattern, it follows that $(S_{\hat{\mathcal{P}}}, \hat{\mathcal{P}}(\sigma, \pi''), k) \not\models \varphi$, and hence $(S_{\hat{\mathcal{P}}}, r, k) \not\models I\varphi$, and we are done with one direction. The other direction of the argument is symmetric. ⋈

We can now define the *wastefulness* of a failure pattern π, denoted $w(\pi)$, to be $\mathcal{W}(S_{\hat{\mathcal{P}}}, r)$ for a run r of the form $r = \hat{\mathcal{P}}(\sigma, \pi)$ for some σ. Lemma 6.13 implies that $w(\pi)$ is independent of the initial configuration σ chosen, and therefore $w(\pi)$ is well-defined. Theorem 6.12 can now be read to state that if the failure pattern of a run is π, then at time $t + 1 - w(\pi)$ the active processors have common knowledge of a common view of the initial configuration. A closer inspection of the proofs of Theorem 6.5(c) and of Theorem 6.12 actually shows that if $w(\pi) = j$ the at time $t + 1 - j$ there is a particular k' such that the active processors all know that $d(k') \geq j$, and for no $\ell > k'$ is it the case that an active processor knows that $d(\ell) \geq j$. By Theorem 6.12(a), $w(\pi) = j$ iff "$w = j$" is common knowledge at time $t + 1 - j$. It follows that the identity of this number k' is also *common knowledge* at time $t + 1 - j$. Consequently, the active processors obtain common knowledge of a common view of the first k' rounds of the run, and not only of the initial configuration. Furthermore, since k' is determined by the implicitly known values of $d(k)$, Lemma 6.13 implies that the value of k' is uniquely determined by π.

One of the consequences of Theorem 6.12 and Lemma 6.13 is:

Corollary 6.14: There is a t-resilient protocol for SBA that reaches SBA in $t + 1 - w(\pi)$ rounds in all runs of the protocol in which the failure pattern is π, for all failure patterns π in which and at most t processors fail.

Proof: The protocol (uniform for all processors p_i) is:

```
for $\ell \geq 0$ perform the following at time $\ell$:
    if $K(D \geq t + 1 - \ell)$
        then    halt (and send no messages in the following rounds);
                decide 0 if $K$("some initial value $v_j$ was 0");
                decide 1 otherwise.
        else    send the current view to all processors in round $\ell + 1$.
```

The K in the text of the protocol means "the processor knows", i.e., it is K_i in p_i's copy of the protocol. By Theorem 6.12(a) all correct processors halt after $t + 1 - \mathcal{W}(S_\phi, r)$ rounds. By Theorem 6.12(b) the active processors have common knowledge of the fact that they have an identical view of the initial configuration. Thus, their decisions are identical. The decision function clearly satisfies the requirements of SBA. ⋈

The above protocol is not a protocol in the traditional sense of the word, but rather a *knowledge-based protocol*, to use the terminology of Halpern and Fagin in [HF]: a processor's actions at any given point are determined by the processor's knowledge. As they point out, not every knowledge-based protocol can be implemented. However, if the only knowledge required in the protocol is knowledge about the past, it is implementable. Thus, the above protocol can be directly translated into a standard protocol.

Notice that in runs in which many failures become visible early it is the case that SBA is attained by this protocol significantly earlier than time $t + 1$. We are aware of no other protocol for SBA that stops before time $t + 1$ in some cases. In the next section we will show that the protocol of Corollary 6.14 is optimal in the sense that for any given pattern of failures, it attains SBA no later than any other protocol for SBA does.

Corollary 6.8 and Theorem 6.12 imply that the stopping condition $K(D \geq t + 1 - \ell)$ implies $C(D \geq t + 1 - \ell)$. In fact, we will be able to show that this protocol is equivalent to the following protocol:

```
for $\ell \geq 0$ perform the following at time $\ell$:
    if $C$("some initial value was 0")
        then    decide 0 and halt
        else if $C$("some initial value was 1")
            then    decide 1 and halt
            else    send the current view to all processors in round $\ell + 1$.
```

The number of bits of information required to describe a processor's view at round k is exponential in k. Thus, messages in the above protocols might be too long to be practical. The only properties that we really needed for the analysis were that the protocol require all processors to send all other (active) processors a message in every round, and that every processor relay all the information it has about the initial configuration and about the pattern of failures in the message. By modifying the protocol slightly so that messages specify only the sender's view of the initial configuration and of the failure pattern, we get a protocol for SBA with the same properties in which the length of each message is $O(n + t \log n)$.

6.4 Lower bounds

We are about to show that the only non-trivial facts that can become common knowledge in a run r of a t-uniform system S before time $t + 1 - \mathcal{W}(S, r)$ are facts about the wastefulness of the run. We do this by showing that all executions (r, ℓ) with $\mathcal{W}(S, r, \ell) \leq t - \ell$ are similar. We first prove a lemma that is necessary for our proof of this fact. Roughly speaking, this lemma says that if $D(r, \ell) \leq t - \ell$ and p is the last processor to fail in r, then (r, ℓ) is similar to an execution in which p doesn't fail, and all other processors behave as they do in r. To make this precise we make the following definition: Given a failure pattern π, the failure pattern π^{-p} is defined to be $\pi - \langle p, k(p), Q(p)\rangle$ if there is a triple of the form $\langle p, k(p), Q(p)\rangle$ in π, and to be π otherwise. Given a run $r = \mathcal{P}(\sigma, \pi)$, we define r^{-p} to be $\mathcal{P}(\sigma, \pi^{-p})$. If $\mathcal{P}$ does not require all processors to send messages to all other processors in every round, r can be said to display a number of failure patterns. I.e., we may have $\mathcal{P}(\sigma, \pi) = \mathcal{P}(\sigma, \pi')$ for $\pi \neq \pi'$. However, it is easy to check that if $\mathcal{P}(\sigma, \pi) = \mathcal{P}(\sigma, \pi')$ then $\mathcal{P}(\sigma, \pi^{-p}) = \mathcal{P}(\sigma, \pi'^{-p})$, so that r^{-p} is well defined. We can now show:

Lemma 6.15: Let $t \leq n - 2$, and let S be a t-uniform system for $\mathcal{P}$, with $r = \mathcal{P}(\sigma, \pi) \in S$. If $D(r, \ell) \leq t - \ell$ and no processor fails in r in a later round than p does, then $(r, \ell) \sim (r^{-p}, \ell)$.

Proof: If p does not fail in r then $r = r^{-p}$, and the claim trivially holds. Thus, let k be the round in which p fails in r, and notice that by assumption no processor fails in r at a later round. If $k > \ell$ then $(r, \ell) = (r^{-p}, \ell)$ and thus clearly $(r, \ell) \sim (r^{-p}, \ell)$. We still need to show the claim for $k \leq \ell$. We do this by induction on $j = \ell - k$.

Case $j = 0$ (i.e., $k = \ell$): Let $q_i \neq q_j \in A(r, \ell)$ be two processors active at (r, ℓ). Such processors exist by the assumption that $t \leq n - 2$. Clearly, q_i's view at (r, ℓ) is independent of whether or not p sent a message to q_j in round ℓ. Thus,

$(r, \ell) \sim (r', \ell)$, where (r', ℓ) differs from (r, ℓ) only in that p does send a message to q_j in round ℓ of (r', ℓ). (If p sends q_j a message in round ℓ of r, then $r = r'$.) Now, since p does send q_j a message in round ℓ of (r', ℓ), processor q_j's view at (r', ℓ) is independent of whether p fails in (r', ℓ) (it is consistent with q_j's view at (r', ℓ) that p sends messages to all processors in round ℓ), and thus $(r', \ell) \sim (r^{-p}, \ell)$. By transitivity of $\sim$ we also have that $(r, \ell) \sim (r^{-p}, \ell)$.

Case $j > 0$ (i.e., $k < \ell$): Assume inductively that the claim holds for $j - 1$. Again, let $Q = \{q_1, \ldots, q_s\}$ be the set of processors active at (r, ℓ) to whom p fails to send a message in round k of (r, ℓ). We prove our claim by induction on s. If $s = 0$ then no processor active in (r, ℓ) can distinguish whether p failed in round k or in round $k+1$. Thus, $(r, \ell) \sim (r', \ell)$, where (r', ℓ) differs from (r, ℓ) only in that rather than failing in round k, processor p fails in round $k + 1$ of (r', ℓ) before sending any messages. Since $\ell - (k+1) = j - 1$, we have by the inductive hypothesis that $(r', \ell) \sim (r^{-p}, \ell)$. By transitivity of $\sim$ we have that $(r, \ell) \sim (r^{-p}, \ell)$. Now assume that $s > 0$ and that the claim is true for $s - 1$. Let r_s be a run such that $(r_s, k) = (r, k)$, processor q_s fails in round $k + 1$ of r_s before sending any messages, and no other processor fails in r_s after round k. Clearly $D(r_s, \ell) \leq t - \ell$, since $d(r_s, k') = d(r, k') \leq t - \ell$ for all $k' \leq k$, and $d(r_s, k+1) = N(r_s, k+1) - (k+1) = N(r, k) + 1 - (k+1) = d(r, k) \leq t - \ell$. Notice also that no processor fails in (r_s, ℓ) after round $k + 1$. Thus, $r = r_s^{-q_s}$, and by the inductive assumption on $j - 1$, we have that $(r_s, \ell) \sim (r, \ell)$. Let $p_i \in A(r_s, \ell)$. Clearly p_i's view at (r_s, ℓ) is independent of whether p sent a message to q_s in round k of (r_s, ℓ). Thus, $(r_s, \ell) \sim (r'_s, \ell)$, where r'_s differs from r_s in that p does send a message to q_s in round k of r'_s. Again by the inductive hypothesis for $j - 1$ we have that $(r'_s, \ell) \sim (r', \ell)$, where $r' = r_s'^{-q_s}$. Processor p fails to send round k messages only to $s - 1$ processors in r', and thus by the inductive hypothesis for $s - 1$ we have that $(r', \ell) \sim (r^{-p}, \ell)$. By the symmetry and transitivity of $\sim$, we have that $(r, \ell) \sim (r^{-p}, \ell)$, and we are done. $\bowtie$

The proof of Lemma 6.15 is a generalization and simplification of the basic inductive argument in the lower bound proofs of [DS], [LF], and [CD]. Notice that the run r^{-p} in the statement of Lemma 6.15 has the following properties: (i) if r is not free of failures, then the number of processors that fail in r^{-p} is one fewer than in r; (ii) $D(r^{-p}, \ell) \leq t - \ell$, and (iii) $(r^{-p}, 0) = (r, 0)$. We can now use Lemma 6.15 to show:

Theorem 6.16: Let $t \leq n-2$ and let S be an independent t-uniform system.

a) If $\ell \leq t$ then all failure-free executions $(r,\ell) \in S \times \{\ell\}$ are similar.

b) If $\mathcal{W}(S,r,\ell) \leq t-\ell$ and $\mathcal{W}(S,r',\ell) \leq t-\ell$, then $(r,\ell) \sim (r',\ell)$.

Proof: (a) Assume that $\ell \leq t$ and let (r,ℓ) and $(\hat{r},\ell)$ be failure-free executions. We wish to show that $(r,\ell) \sim (\hat{r},\ell)$. Let $Q = \{q_1,\ldots,q_s\}$ be the set of processors whose initial states in r and $\hat{r}$ differ. We prove by induction on s that $(r,\ell) \sim (\hat{r},\ell)$. If $s=0$ then $(r,\ell) = (\hat{r},\ell)$ and we are done. Let $s>0$ and assume inductively that all failure-free executions that differ from $(\hat{r},\ell)$ in the initial state of no more than $s-1$ processors are similar to it. Let (r_s,ℓ) be an execution such that $(r,0)=(r_s,0)$, in which q_s fails in the first round without sending any messages, and no other processor fails. Clearly $D(r_s,\ell) = 0 \leq t-\ell$, and by Lemma 6.15 we have that $(r_s,\ell) \sim (r,\ell)$. Let $p_i \in A(r_s,\ell)$. Given that S is an independent t-uniform system, processor p_i's view at (r_s,ℓ) does not determine whether the initial state of q_s in r_s is as in r or as in $\hat{r}$. Thus, $(r_s,\ell) \sim (r'_s,\ell)$, where r'_s differs from r_s only in that the initial state of q_s in r'_s is as in $\hat{r}$. Again by Lemma 6.15 we have that $(r'_s,\ell) \sim (r',\ell)$, where $(r'_s,0)=(r',0)$, and (r',ℓ) is failure-free. Since (r',ℓ) differs from $(\hat{r},\ell)$ only in the initial states of $s-1$ processors, by the inductive assumption we have that $(r',\ell) \sim (\hat{r},\ell)$, and by the symmetry and transitivity of $\sim$ we have $(r,\ell) \sim (\hat{r},\ell)$, and we are done with part (a).

(b) If $\mathcal{W}(S,r,\ell) \leq t-\ell$ then in particular it is not implicit knowledge at (r,ℓ) that $d(k) > t-\ell$ for some $k \leq \ell$. It follows that $(r,\ell) \sim (\bar{r},\ell)$, for some $\bar{r} \in S$ satisfying $D(\bar{r},\ell) \leq t-\ell$. Using Lemma 6.15, a straightforward induction on the number of processors that fail in $(\bar{r},\ell)$ shows that $(\bar{r},\ell) \sim (\hat{r},\ell)$, where $(\hat{r},\ell)$ is failure-free. By transitivity of $\sim$ we have that $(r,\ell) \sim (\hat{r},\ell)$. The same argument applies to (r',ℓ), and the claim now follows from part (a). ⋈

Observe that the assumption of independence of the set of initial configurations is essential to this lower bound. Lemma 6.15 can also be used to characterize non-independent systems. Lemma 6.15 and Theorem 6.16(a) generalize and somewhat simplify the $t+1$ round lower bound on the worst-case behavior of SBA in our model (see [DLM], [DS], [FL], [H], [CD]). As we will see in the sequel, Theorem 6.16(b) allows us to completely characterize the runs in which $t+1$ rounds are necessary for attaining SBA, as well as those that require k rounds, for all k. More generally, Proposition 6.7(a) and Theorem 6.16(b) provide us with a lower bound on the time by which facts can become common knowledge in t-uniform systems. Formally, we have:

Theorem 6.17: Let $t \leq n-2$, and let S be an independent t-uniform system. If $(S,r',\ell) \not\models \varphi$ holds for some $r' \in S$ satisfying $\mathcal{W}(S,r') \leq t-\ell$, then $(S,r,\ell) \not\models C\varphi$ for all $r \in S$ satisfying $\mathcal{W}(S,r) \leq t-\ell$. ⋈

Theorem 6.17 and Theorem 6.12(b) completely characterize when non-trivial facts about the initial configuration become common knowledge in the runs of $S_{\hat{\mathcal{P}}}$. In a precise sense, they imply that the only fact that is common knowledge at (r,k), for $k \leq t - \mathcal{W}(S_{\hat{\mathcal{P}}},r)$, is that the wastefulness is less than $t+1-k$. Formally, we have:

Corollary 6.18: Let $t \leq n-2$, let $S_{\hat{\mathcal{P}}}$ be an independent t-uniform system for $\hat{\mathcal{P}}$, and let $\mathcal{W}(S_{\hat{\mathcal{P}}},r) \leq t-\ell$. Then $(S_{\hat{\mathcal{P}}},r,\ell) \models C\varphi$ iff for all $r' \in S_{\hat{\mathcal{P}}}$ such that $\mathcal{W}(S_{\hat{\mathcal{P}}},r',\ell) \leq t-\ell$ it is the case that $(S_{\hat{\mathcal{P}}},r',\ell) \models \varphi$. ⋈

Furthermore, Corollary 6.8 and Theorem 6.17 immediately imply:

Corollary 6.19: Let $t \leq n-2$, let $\mathcal{P}$ be a t-resilient protocol for SBA, and let S be a t-uniform system for $\mathcal{P}$, with $r \in S$. Then SBA is not attained in r in fewer than $t+1-\mathcal{W}(S,r)$ rounds. ⋈

Corollary 6.19 proves that SBA cannot be attained in the runs of $\hat{\mathcal{P}}$ any earlier than it is attained by the protocol of Corollary 6.14. However, it still seems possible that using another protocol SBA will be attainable in fewer rounds than in the protocol of Corollary 6.14. We now show that this protocol is optimal in a rather strong sense: for any given initial configuration and failure pattern, no protocol attains SBA in fewer rounds than the protocol of Corollary 6.14. This fact follows from the following theorem, which states that the wastefulness of a run resulting from a given initial configuration and failure pattern is no greater than its wastefulness in $S_{\hat{\mathcal{P}}}$. Given Corollary 6.19, this will imply that the protocol of Corollary 6.14 always attains SBA at the earliest possible time, given the initial configuration and failure pattern.

Theorem 6.20: Let S be a t-uniform system for a protocol $\mathcal{P}$, let $r = \mathcal{P}(\sigma, \pi)$, and let $\hat{r} = \hat{\mathcal{P}}(\sigma, \pi)$. Then $\mathcal{W}(S, r) \leq \mathcal{W}(S_{\hat{\mathcal{P}}}, \hat{r})$.

Proof: We will show a more general fact from which the theorem will follow. Given an initial configuration σ', and a failure pattern π', let $r' = \mathcal{P}(\sigma', \pi')$ and $\hat{r}' = \hat{\mathcal{P}}(\sigma', \pi')$. Notice that $A(r, k') = A(\hat{r}, k')$ for all k'. We claim that for all k and all $p_i \in A(r, k)$ it is the case that if $v(p_i, \hat{r}, k) = v(p_i, \hat{r}', k)$ then $v(p_i, r, k) = v(p_i, r', k)$. We argue by induction on k. The case $k = 0$ is immediate. Let $k > 0$ and assume inductively that the claim holds for all processors in $A(r, k-1)$ at time $k-1$. Thus, if $v(p_i, \hat{r}, k) = v(p_i, \hat{r}', k)$ and p_j sends a round k message to p_i in $\hat{r}$, then p_j has the same view at $(\hat{r}, k-1)$ and $(\hat{r}', k-1)$, and p_j also sends p_i a round k message in $\hat{r}'$. In this case both π and π' determine that round k messages from p_j to p_i are delivered. By the inductive assumption p_j also has the same view in $(r, k-1)$ and in $(r', k-1)$. It follows that $\mathcal{P}$ requires p_j to act identically in round k of both r and r'. And if p_j is required to send p_i a round k message in r then it is required to send p_i the same message in round k of r'. Processor p_j does not send a round k message to p_i in $\hat{r}$ only if π determines that p_j cannot send p_i such a message. But then for similar reasons π' must also determine that p_j does not send p_i a round k message. It follows that in this case p_j does not send p_i a round k message in r or in r'. Thus, for all processors p_j it is the case that p_i receives a round k message from p_j in r iff p_i receives an identical message from p_j in round k of r'. The inductive assumption also implies that $v(p_i, r, k-1) = v(p_i, r', k-1)$, and it now follows that $v(p_i, r, k) = v(p_i, r', k)$ and we are done with the claim. We now show how the theorem follows from this claim. Assume that $\mathcal{W}(S, r) = j$ and that $\mathcal{W}(S_{\hat{\mathcal{P}}}, \hat{r}) < j$. Then there is a time k such that $(S, r, k) \models I(D \geq j)$, and $(S_{\hat{\mathcal{P}}}, \hat{r}, k) \not\models I(D \geq j)$. Let $G = A(\hat{r}, k)$ (notice that $G = A(r, k)$ as well). It follows that there is a run $\hat{r}' \in S_{\hat{\mathcal{P}}}$ such that $v(G, \hat{r}, k) = v(G, \hat{r}', k)$ and $D(\hat{r}', k) < j$. Let σ' and π' be the initial configuration and failure pattern in $\hat{r}'$. Let r' be $\mathcal{P}(\sigma', \pi')$. Since $v(G, \hat{r}, k) = v(G, \hat{r}', k)$, our claim implies that $v(G, r, k) = v(G, r', k)$. But since $D(r', k) = D(\hat{r}', k) < j$ and $A(r, k) = G$, we have that $(S, r, k) \not\models I(D \geq j)$, contradicting our original assumption. ⋈

Theorem 6.20 and Corollary 6.19 now imply that the protocol of Corollary 6.14 is indeed optimal in the strong sense we intended: given any initial configuration and failure pattern, it attains SBA as early as any t-resilient protocol for SBA can. In light of Theorem 6.20, we can talk about the inherent *wastefulness* $w(\pi)$ of a failure pattern π, defined to be $\mathcal{W}(S_{\hat{\mathcal{P}}}, \hat{\mathcal{P}}(\sigma, \pi))$. That $w(\pi)$ is well defined follows from the fact that runs r of $S_{\hat{\mathcal{P}}}$ have the property that $\mathcal{W}(S_{\hat{\mathcal{P}}}, r, k)$ depends only on the pattern of failures and is independent of the initial configuration. This can be proved by a somewhat tedious but straightforward induction on k, and is left to the reader. Theorem 6.16 through Corollary 6.19 can now be viewed as statements about the effect of the failure pattern on the similarity of executions and on what facts can become common knowledge at various times in the execution of an arbitrary t-resilient protocol. Corollaries 14 and 19 tell us that exactly $t+1-w(\pi)$ rounds are necessary and sufficient to attain SBA in runs of any t-resilient protocol for SBA that have pattern failure π (in the rest of the chapter we will use π to refer to the failure pattern of the run in question). This provides a complete characterization of the number of rounds required to reach SBA in a run, given the pattern in which failures occur.

We have seen that the only facts that can become common knowledge before time $t+1-w(\pi)$ are facts about the wastefulness of the run. In the previous section we saw that in runs of $S_{\mathcal{P}}$ the processors attain common knowledge of an identical view of the initial configuration at time $t+1-w(\pi)$. Thus, we have a complete description of when facts about the initial configuration become common knowledge. It is interesting to ask the more general question of when arbitrary facts become common knowledge. As we have remarked in the previous section, the proofs of Lemma 6.11 and Theorem 6.12 can be used to show that at time $t+1-w(\pi)$ in a run of $S_{\mathcal{P}}$ the active processors do not attain common knowledge only of the fact that they have a identical view of initial configuration, Rather, there is a natural number $k \geq 0$ such that at time $t+1-w(\pi)$ they attain common knowledge of an identical view of the state of the system at time k. We denote this number k by $k_1(\pi)$. There is some number, say f_1, of processors that are commonly known at time $t+1-w(\pi)$ to have failed by time $k_1(\pi)$. Let $t_1 = t - f_1$. Roughly speaking, time $k_1(\pi)+1$ can now be regarded as the start of a new run, and for appropriate definitions of $d_1(k)$ and $w_1(\pi)$, we get that at time $(k_1(\pi)+1)+t_1+1-w_1(\pi)$ the system will attain common knowledge of a common view of the state of the system at time $k_1(\pi)+1$. Interestingly, it can be shown that $(k_1(\pi)+1)+t_1+1-w_1(\pi) = t+2-w(\pi)$. That is, one round after the processors attain common knowledge of (a common view of) the state of the run at time $k_1(\pi)$, they attain common knowledge of a common view of the state of the run at $k_1(\pi)+1$. In fact, again we have some number $k'' \geq 0$ such that the processors have common knowledge at time $t+2-w(\pi)$ of a common view of the state of the system at time k''. Denoting this number by k_2, the above analysis can be repeated. We leave further details to the interested reader.

The result of the analysis discussed in the preceding paragraph is that at any point after time $t+w(\pi)$ in a run of $\hat{\mathcal{P}}$ the active processors have common knowledge of a common view of the first k rounds, for a number k that can be computed given the failure pattern π. Following every round after time $t+1-w(\pi)$ the active processors attain common knowledge of a common view of at least one additional round. Consequently, there is a *window of common plausibility* of a number of the most recent rounds about which no non-trivial facts are common knowledge, and a common view of all preceding rounds is common knowledge. The size of this window at a given point is t minus the number of processors that (at that point) are not commonly known to have failed. This classification of what facts are common knowledge in the runs of $S_{\mathcal{P}}$ provide good upper bounds on when a simultaneous action that depends on the first k rounds can then be carried out bv all active processors in a consistent way. The lower bound results of this section can imply that these bounds are tight *in all runs*, and thus we have a complete characterization of when simultaneous actions that depend on the first k rounds can be carried out, as a function of the failure pattern.

6.5 Applications

Throughout the chapter we have shown how our results regarding when common knowledge of various facts is attained in a Byzantine system affect the SBA problem. In this section we discuss some further consequences of the analysis presented in the previous sections. This is intended to illustrate the types of applications that the analysis can be used for. We start by considering some problems that are closely related to SBA.

The problem of *Weak* SBA, which differs from SBA in that clause (4) is changed so that the active processors are required to decide on a value v only if all initial values were v and *no processor fails,* was introduced by Lamport as a weakening of SBA. However, Theorem 6.16(b) immediately implies that the active processoıs do not have common knowledge of whether any processors failed before time $t+1-w(\pi)$, in any run of a t-resilient protocol for WSBA with failure pattern π. And since SBA can already be performed at time $t+1-w(\pi)$, we have that t-resilient protocols cannot attain WSBA any earlier than they can SBA. Theorem 6.16 also describes why the variant of SBA used in this chapter (which was introduced by [FL]) is essentially equivalent to the original version of the *Byzantine Generals* problem of [PSL], in which only one processor initially has a value, and the processors need to decide on this value if the processor does not fail, and on a consistent value otherwise.

It has been a folk conjecture that a t-resilient protocol that guarantees that a non-trivial action is performed simultaneously must require $t+1$ rounds in the worst case. We now show that this is not the case. Let *bivalent agreement* be defined by clauses (1)–(3) of SBA, and replacing clause (4) by:

4′. At least one run of the protocol decides 0, and at least one run decides 1.

Thus, a t-resilient protocol for bivalent agreement is a protocol $\mathcal{P}$ with the property that all runs of the independent t-uniform system S for $\mathcal{P}$ in which the set of initial configurations is $\{0,1\}^n$ satisfy clauses (1)–(3), and at least one run of S decides 0, and at least one run decides 1. Proposition 6.7 implies that any action that is guaranteed to be performed simultaneously requires some fact to become common knowledge before the action can be performed. Theorem 6.12(b) implies that at the end of round 2 of $S_{\hat{\mathcal{P}}}$ it is common knowledge whether or not the wastefulness of the run is $t-1$ (i.e., whether t processors were seen to have failed in the first round). Thus, we can easily derive a t-resilient protocol for bivalent agreement: Each processor follows $\hat{\mathcal{P}}$ for the first two rounds, and then decides 0 if it knows that t processors failed in the first round, and 1 otherwise. This protocol attains bivalent agreement in two rounds, and Theorem 6.17 implies that there is no faster protocol for bivalent agreement so long as $t \leq n-2$. Furthermore, it implies that in a precise sense this is the only two-round protocol for bivalent agreement. We leave it to the reader to check that if $t \geq n-1$ then there is a protocol for bivalent agreement that requires only one round. Thus, bivalent agreement is a truly easier problem than SBA. We note that [FLP] and [DDS] prove that in an asynchronous system there is no 1-resilient protocol for an even weaker variant of bivalent agreement.

We have stressed the connection between common knowledge and simultaneous actions. Interestingly, the lower bounds on the time required for attaining common knowledge imply worst-case bounds on the behavior of t-resilient protocols that perform coordinated actions that are not required to be performed simultaneously. For example, *Eventual Byzantine Agreement* (EBA) is defined by clauses (1), (2), and (4) of SBA: the processors' decisions need not be simultaneous (cf. [DRS]). There are well-known protocols that attain EBA after two rounds in failure-free runs (for which $w(\pi) = 0$). However, using Proposition 6.7 and Theorems 6.17 and 6.20 it is not hard to show that a t-resilient protocol for EBA must require $t+1$ rounds in some runs with $w(\pi) = 0$. More generally, these theorems show that such a protocol must require $t+1-j$ rounds in some runs with $w(\pi) = j$. This is a slight refinement of the well-known fact that EBA requires $t+1$ rounds in the worst case (cf. [DRS]). Many very relevant and interesting aspects of EBA are not covered by our analysis. We believe that an analysis of EBA should involve a study of when the states of ϵ-common knowledge and eventual common knowledge (cf. Chapter 2) are attained in a Byzantine environment. This is an interesting open problem.

As our investigation centered around t-resilient protocols, we now briefly discuss some other possible reliability assumptions. Recall that Corollary 6.10 states that all active processors are guaranteed to have an identical view of the system's initial configuration at time $t+1$ in every run of a t-uniform system for $\hat{\mathcal{P}}$. This follows simply from the fact that at time $t+1$ it is common knowledge that one of the previous rounds was clean. Instead of t-resiliency, we could require that a protocol for SBA be guaranteed to attain SBA so long as no more than k consecutive rounds are dirty. In the system corresponding to all the runs of $\hat{\mathcal{P}}$ in which at most k consecutive rounds are dirty, it is common knowledge at time $k+1$ that a clean round has occurred, and $\hat{\mathcal{P}}$ can be converted in to a protocol for SBA that is guaranteed to attain SBA in no more than $k+1$ rounds. This means, for example, that if processors in a Byzantine system are known to fail at least two at a time, SBA can be achieved in $t/2+1$ rounds. Having a bound of k consecutive dirty rounds seems in many cases to be a more appropriate assumption about a system than having a bound of t on the total number of failures possible, since the latter is not a local assumption. Of course, these two assumptions are not mutually exclusive, and we may often have a small bound on the possible number of consecutive dirty rounds, and only a much larger bound holds for the total number of failures. The bound on the number of consecutive dirty rounds implies a good upper bound on SBA in the case of crash failures.

Another way we can consider varying the reliability assumptions about the system is by restricting the number of possible processor failures that can occur in a round. For example, let us consider the assumption that at most one processor can fail in any given round of the computation, and at most t processors might fail overall. We are interested in the question of whether such assumptions allow us to attain SBA quickly. Unfortunately, the lower bound proofs of Lemma 6.15 and Theorem 6.16 work very well for this reliability model. In fact, since all of the runs of such a system are guaranteed to have wastefulness 0, even bivalent agreement cannot be attained in any run of the system in less than $t+1$ rounds! SBA and WSBA clearly require $t+1$ rounds in *all* runs of the system. We now present a somewhat artificial variant of this assumption that provides us with a non-uniform reliability assumption whose behavior is interesting and somewhat counter-intuitive: We say that a protocol for SBA is *one visible failure resistant* (1-VFR) if it is guaranteed to attain SBA so long as no more than one processor failure becomes visible to the active processors in any given round. The set of possible runs of a protocol $\mathcal{P}$ that display such behavior will be called a *visibly restrained* system for $\mathcal{P}$. It is possible to show that in the visibly restrained system for the simple protocol $\hat{\mathcal{P}}$ of Section 6.3 it is common knowledge at time 2 whether round 1 is clean, and therefore WSBA can be attained in two rounds. However, SBA can be shown to require $n-1$ rounds in runs of $\hat{\mathcal{P}}$ in which one processor fails in every round except possibly the $(n-1)$st round. (If one adds a bound of $t \leq n-2$ on the total number of failures possible, $n-1$ is replaced by $t+1$.) Interestingly, there is a 1-VFR protocol for SBA that is guaranteed to attain SBA in three rounds (in all runs)! Thus, for the 1-VFR reliability model, our simple protocol is no longer a most general protocol. The reason for the odd behavior of 1-VFR protocols is that the patterns of failures of the runs that satisfy 1-VFR are intimately related to the structure of the protocol. Thus, the protocol can restrict the patterns of failures possible and make effective use of the 1-VFR assumption.

6.6 Discussion

We have analtzed th states of knowledge attainable in the course course of the execution of various protocols in th system, for the case of a particular simple model of unreliable distributed systems that is fairly popular in the literature. Motivated by Chapters 1 and 2, the analysis focused mainly on when various facts about the system become common knowledge given an upper bound of t on the number of possible faulty processors. This problem is shown to directly correspond to the question of when simultaneous actions of various types can be performed by the processors in such a system. In particular, this is a generalization of Simultaneous Byzantine Agreement and related problems. By deriving exact bounds on the question of when facts become common knowledge, we immediately got exact bounds for SBA and many other problems. An interesting fact that came out of the analysis was that the pattern in which processors fail in a given run determines a lower bound on the time in which facts about the system's initial configuration become common knowledge, with different patterns determining different bounds. Ironically, facts become common knowledge faster in cases when many processors fail early in the run. The somewhat paradoxical argument for this is that, given an upper bound on the total number of failures possible, if many processors fail early then only few can fail later. The protocol can make effective use of the fact that the rest of the run is relatively free of failures. As a by-product of the analysis, we were able to derive a simple improved protocol for SBA that is optimal in *all* runs.

Our analysis shows that the essential driving force behind many of the phenomena in unreliable systems seems to be the inherent uncertainty that a particular site in such a system has about the global state of the system. We come to grips with this uncertainty by performing a knowledge-based analysis of such a system. We stress that our analysis was by and large restricted to protocols for simultaneous actions in a rather clean and simple model of unreliable systems: synchronous systems with global clocks and crash failures. We believe that performing similar analyses for nastier models of failures will prove very exciting, and will provide a much better understanding of the true structure underlying the richer failure models, and of the differences between the failure models. The ideas and techniques developed in this chapter should provide a sound basis on which to build such an analysis, although it is clear that a number of additional ideas would be required.

In summary, the analysis in this chapter makes explicit and essential use of reasoning about knowledge in order to obtain insight into a well-known problem in distributed systems. The generality and applicability of our results suggest that this is a promising approach.

CHAPTER 7

CONCLUSIONS

Knowledge seems to play an important role in distributed computing. A major advantage protocol designers obtain by intuitively reasoning about processors' knowledge in a distributed system is the fact that personal experience from everyday life can be used to facilitate the design of distributed protocols. This thesis suggests that it is worthwhile to carry this approach one step further, and ascribe knowledge to processors in a precise and formal way. This is useful because there are many subtleties involved in distinguishing relevant states of knowledge that may initially seem similar but are in fact very different. Understanding what the relevant states of knowledge are and how they relate to one another is a major problem for future work. We have shown a close relationship between certain states of knowledge and the ability to perform coordinated actions of various kinds. In particular, the inability to perform certain types of actions under particular conditions was captured in a rather general way by our negative results in Chapters 3, 4, and 6 regarding the inability to attain the related states of knowledge in those circumstances. Thus, it seems that reasoning about the attainability of states of knowledge under various conditions may be a good way to study the properties of sustems of various kinds.

Whereas our treatment dealt mainly with a rather general model of distributed systems, it seems desirable and promising to investigate particular types of distributed systems that are of interest from the point of view of the states of knowledge that runs of various protocols attain in such a system. A good example of this is the work of Chandy and Misra in [ChM], in which they capture some essential properties of totally asynchronous systems by performing a knowledge-based analysis of such systems. Our analysis in Chapter 6 also provides a general setting and some new insights into the properties of a particular model of systems of unreliable processors. Much more work in this direction needs to be done.

Using the language of knowledge to specify, present, and perhaps synthesize distributed protocols seems attractive in a number of ways. First, it is often an intuitive way to think anout the protocols. Furthermore, it may be a more unified and portable way of communicating the protocol. The work of Afrati, Papadimitriou, and Papageorgiou in [APP] shows a case in which it is not clear how to specify the goals of a protocols other than in terms of attaining a particular state of knowledge, and Halpern and Fagin in [HF] define and discuss knowledge-based protocols in a formal way. This is another area requiring further work.

In Chapter 2 we presented a very general framework for ascribing knowledge to processors in a distributed system. However, in later chapters we used only state-based interpretations in which processors are ascribed a great deal of knowledge without being required to perform any kind of computation to attain it. The reason for that was that we were working on a rather coarse level of investigation, looking at problems in which the computational complexity was negligible, and the information-theoretic aspects played a major role. However, for many applications in fields such as AI and cryptography the computational problems involved are of major importance. In such circumstances it is essential to develop a good theory for computationally-based knowledge. We regard this as one of the harder problems in the area, and one in which progress is urgently necessary. Fagin and Halpern in [FH] take a significant first step towards such a theory for AI applications. Goldwasser, Micali, and Rackoff in [GMR] present a theory of *Knowledge complexity*, that promises to be a good starting point for developing a computationally-based theory of knowledge for cryptography.

In summary, the study of knowledge in distributed environments is still in its infancy, but promises to provide insights into many aspects of distrbuted environments. We have shown that it is possible to make reasoning about knowledge in a distributed environment precise, and used such reasoning to obtain new insight into some well-known problems. I would like to close this thesis with two quotes. The first — a fortune cookie that I got while working with Cynthia Dwork on the material of Chapter 6:

"Imagination is more important than knowledge",

and the second a quote from Marc Chagall at age 90:

"All I know is that one understands only what one loves".

Of course, these sayings suggest further inspiring directions for future work...

APPENDIX A

SYSTEMS OF MODAL LOGIC

This appendix contains a brief summary of the axioms of the modal systems S4 and S5, refered to in the thesis. For a more detailed exposition of these systems and their role in modal logics of knowledge, see [HM]. We use M for a generic modal operator (corresponding to K_i, C_G, I_G, etc.).

The system S5 consists of the axioms:

A1. $M(\varphi \supset \psi) \supset (M\varphi \supset M\psi)$ (Consequence Closure)

A2. $M\varphi \supset \varphi$ (Knowledge)

A3. $M\varphi \supset MM\varphi$ (Positive Introspection)

A4. $\neg M\varphi \supset M\neg M\varphi$ (Negative Introspection)

And the rules of inference:

R1. From $\vdash \varphi$ and $\vdash (\varphi \supset \psi)$ infer ψ (Modus Ponens)

R2. From $\vdash \varphi$ infer $\vdash M\varphi$ (Generalization)

Under the total view interpretation of knowledge, K_i, C_G, and I_G all satisfy the axioms of S5. In systems of unreliable processors (the case of Chapter 6), the common knowledge operator C corresponding to "common knowledge among the *active* processors" also satisfies the axioms of S5.

The system consisting of axioms A1–A3 and inference rules R1 and R2 is called S4. The implicit knowledge operator of Chapter 6, corresponding to "implicit knowledge among the *active* processors" satisfies the axioms of S4, but does not satisfy the axiom A4, and hence does not satisfy S5.

APPENDIX B

A LOGIC WITH FIXPOINT DEFINITIONS

In this appendix we present a logic with greatest fixpoint definitions and illustrate how common knowledge and variants of common knowledge can be formally defined as greatest fixpoints. Our presentation follows Kozen's in [Koz]. For other treatments of modal logics with fixpoint constructions, we refer the reader to [Koz] and [Fit].

Before we define the logic, we need to review a number of relevant facts about fixpoints. Given a set S, we will be considering operators f mapping subsets of S to subsets of S. A subset A of S is said to be a *fixpoint* of f if $f(A) = A$. A *greatest* (respectively, *least*) fixpoint of f is a set B that is a fixpoint of f and that for all fixpoints A of f satisfies $A \subseteq B$ (resp. $B \subseteq A$). It follows that if f has a greatest fixpoint B, then $B = \bigcup\{A : f(A) = A\}$. The operator f is said to be *monotone* if $f(A) \subseteq f(B)$ whenever $A \subseteq B$. The Knaster-Tarski theorem (cf. [T]) implies that a monotone operator has a greatest (and a least) fixpoint. Given an operator f and a subset A, define $f^0(A) = A$ and $f^{i+1}(A) = f(f^i(A))$. f is said to be *continuous* if $f(\bigcup_i A_i) = \bigcup_i f(A_i)$ for all sequences $A_1, A_2, \ldots$. Given a monotone and continuous operator f, the greatest fixpoint of f is the set

$$\bigcap_{n<\omega} f^n(S).$$

Similarly, the least fixpoint of f is $\bigcup\limits_{n<\omega} f^n(\emptyset)$.

We can now define a state-based propositional logic of knowledge for a given distributed system, with temporal operators and a greatest fixpoint definition construct.

B.1 Syntax

The primitive nonlogical symbols of our language consist of a set $\Phi = \{P, Q, \ldots\}$ of primitive propositions, a single distinguished propositional variable Z, and auxiliary propositional variables $X, Y, X_1, Y_1, \ldots$. Formulas of the language are defined inductively by:

a) Z is a formula.

b) P is a formula for all primitive propositions $P \in \Phi$.

c) $\neg\varphi$ is a formula if φ is a formula.

d) $\varphi \wedge \psi$ is a formula if both φ and ψ are formulas.

e) $\bigcirc^{\delta}\varphi$ is a formula if φ is a formula and δ is a real number.

f) $\Diamond\varphi$ is a formula if φ is a formula.

g) $K_i\varphi$ is a formula if φ is a formula and $i \in \{1, \ldots, n\}$.

h) $\nu X.\varphi[Z/X]$ is a formula if $\varphi[Z]$ is a formula in which all occurrences of Z are positive (i.e., are within the scope of an even number of negation signs $\neg$), X is an auxiliary variable that does not occur in $\varphi[Z]$, and $\varphi[Z/X]$ is the result of replacing all occurrences of Z in $\varphi[Z]$ by X.

B.2 Semantics

Given a distributed system represented by its set of runs S, let $\mathcal{S} = S \times (-\infty, \infty)$. A model $\mathcal{M}$ is a triple $(\mathcal{S}, \pi, \sigma)$, where $\mathcal{S}$ is as above, $\pi : \mathcal{S} \times \Phi \rightarrow \{\mathbf{true}, \mathbf{false}\}$ is an assignment of truth values to the primitive propositions at the points of $\mathcal{S}$, and $\sigma : \{1, \ldots, n\} \times \mathcal{S} \rightarrow \Sigma$ is an assignment of the states (from a set of statcs Σ) to the processors at the points of $\mathcal{S}$. Formally, a formula φ in our language is interpreted as an operator $\varphi^{\mathcal{M}}$ from subsets of $\mathcal{S}$ to subsets of $\mathcal{S}$. The operator $\varphi^{\mathcal{M}}$ is defined inductively as follows:

a) $Z^{\mathcal{M}}(A) = A$.

b) $\varphi^{\mathcal{M}}(A) = \{s \in \mathcal{S} \,:\, \pi(s, \mathcal{P}) = \mathbf{true}\}$.

c) $(\neg\varphi)^{\mathcal{M}}(A) = \mathcal{S} - \varphi^{\mathcal{M}}(\mathcal{A})$.

d) $(\varphi \wedge \psi)^{\mathcal{M}}(A) = \varphi^{\mathcal{M}}(A) \cap \psi^{\mathcal{M}}(A)$.

e) $(\bigcirc^{\delta}\varphi)^{\mathcal{M}}(A) = \{s \in A : \, s = (r, t) \text{ and } (r, t + \delta) \in \varphi^{\mathcal{M}}(A)\}$.

f) $(\Diamond\varphi)^{\mathcal{M}}(A) = \bigcup_{\delta \geq 0} (\bigcirc^{\delta}\varphi)^{\mathcal{M}}(A)$.

g) $(K_i\varphi)^{\mathcal{M}}(A) = \{s \in A : \text{for all } s' \in \mathcal{S},\ \sigma(i, s) = \sigma(i, s') \text{ implies } s' \in \varphi^{\mathcal{M}}(A)\}$.

h) $(\nu X.\varphi[Z/X])^{\mathcal{M}}(A) = \bigcup\{B : \varphi^{\mathcal{M}}(B) = B\}$.

We define $\mathcal{M}, (r, t) \models \varphi$ if $(r, t) \in \varphi^{\mathcal{M}}(\emptyset)$. It can be checked that for formulas φ in which no variables or greatest fixpoint operators appear, this definition is consistent with our previous definition. I.e., $\mathcal{M}, (r, t) \models \varphi$ iff $(\mathcal{I}_{\sigma}, r, t) \models \varphi$.

We can extend the syntax of our language by defining $\vee$, $\supset$, and $\equiv$ in terms of $\wedge$ and $\neg$, defining $E\varphi$ ("everyone knows φ") and $E^\diamond\varphi$ ("everyone will eventually know φ") as $K_1\varphi \wedge K_2\varphi \wedge \cdots \wedge K_m\varphi$ and $\Diamond K_1\varphi \wedge \Diamond K_2\varphi \wedge \cdots \wedge \Diamond K_m\varphi$ respectively, and defining the least fixpoint construct $\mu X.\varphi[Z/X]$ as $\neg\nu X.\neg\varphi[Z/\neg X]$. (Notice that if all occurrences of Z in $\varphi[Z]$ are positive, then all occurrences of X in $\neg\varphi[Z/\neg X]$ are also positive.)

We state without proof the following facts (cf. [Koz]):

(i) If all occurrences of Z in $\varphi[Z]$ are positive then $(\varphi[Z])^\mathcal{M}$ is a monotone operator. By the Knaster-Tarski theorem, this implies that $\nu X.\varphi[Z/X]$ is well defined.

(ii) If all occurrences of Z in $\varphi[Z]$ are positive then $(\varphi[Z])^\mathcal{M}$ is also continuous, and thus

$$(\nu X.\varphi[Z/X])^\mathcal{M} = \bigcap_{n<\omega}(((\varphi[Z])^\mathcal{M})^n(\mathcal{S}).$$

(iii) A formula in which Z does not appear is called *closed*. If φ is a closed formula then $\varphi^\mathcal{M}$ is a constant operator, i.e., there is a set $B \subseteq \mathcal{S}$ such that $\varphi^\mathcal{M}(A) = B$ for all subsets A of $\mathcal{S}$. (In this case we denote $\varphi^\mathcal{M}(A)$ by $\varphi^\mathcal{M}$.) In particular, $(\mathbf{true})^\mathcal{M} = \mathcal{S}$ and $(\mathbf{false})^\mathcal{M} = \emptyset$.

Given the machinery at our disposal, we can now define $C\varphi$ as $\nu X.(\varphi \wedge EX)$, define $C^\epsilon\varphi$ as $\nu X.(\varphi \wedge \bigcirc^\epsilon EX)$, and define $C^\diamond\varphi$ as $\nu X.(\varphi \wedge E^\diamond X)$. By continuity, we get that

$$C\varphi \equiv \varphi \wedge (\varphi \wedge E\varphi) \wedge (\varphi \wedge E\varphi \wedge EE\varphi) \wedge \ldots.$$

The analogous fact holds for $C^\epsilon\varphi$ and $C^\diamond\varphi$. Notice that in the case of $C\varphi$ and $C^\epsilon\varphi$ we have that

$$C\varphi \equiv \varphi \wedge E\varphi \wedge EE\varphi \wedge \ldots, \text{ and that}$$

$$C^\epsilon\varphi \equiv \varphi \wedge (\bigcirc^\epsilon E)\varphi \wedge (\bigcirc^\epsilon E)^2\varphi \wedge \ldots.$$

However, $C^\diamond\varphi$ is *not* equivalent to $\varphi \wedge E^\diamond\varphi \wedge (E^\diamond)^2\varphi \wedge \ldots$. We encourage the reader to check that $C\varphi$, $C^\epsilon\varphi$, and $C^\diamond\varphi$ indeed satisfy the axioms (1)–(3) of Section 4.1, and that $C\varphi$ satisfies $\neg C\varphi \supset C\neg C\varphi$.

It is straightforward to extend the above framework to include reference to sets G of processors, in order to define variants of common knowledge parameterized by G. It is also possible to extend it to include explicit individual clock times in order to define $C^T\varphi$, and to add likelihood, probability, etc., in order to define all of the other variants of common knowledge introduced in the thesis.

REFERENCES

[APP] F. Afrati, C. H. Papadimiriou, and G. Papageorgiou, The synthesis of communication protocols, *To appear, Proceedings of the Fifth ACM Symposium on the Principles of Distributed Computing*, 1986.

[A] R. J. Aumann, Agreeing to disagree, *Annals of Statistics*, **4**:6, 1976.

[B] J. Barwise, Scenes and other situations, *Journal of Philosophy*, **LXXVIII**:7, 1981, pp. 369-397.

[CE] E. M. Clarke and E. A. Emerson, Synthesis of synchronization skeletons from branching time temporal logic, *Proceedings of the Workshop on Logics of Programs* (Yorktown-Heights, N.Y.), Lecture Notes in Computer Science **131**, Springer-Verlag, New York, 1981, pp. 52-71.

[CD] B. Coan and C. Dwork, Simultaneity is harder than agreement, *Proceedings of the Fifth Symposium on Reliability in Distributed Software and Database Systems*, 1986, pp. 141-150.

[CL] K. M. Chandy and L. Lamport, Distributed snapshots: Determining global states of distributed systems, *ACM Transactions on Computer Systems*, **3**:1, February 1985.

[ChM] K. M. Chandy and J. Misra, How processes learn, *Proceedings of the Fourth ACM Symposium on the Principles of Distributed Computing*, 1985, pp. 204-214.

[ClM] H. H. Clark and C. R. Marshall, Definite reference and mutual knowledge, in A. K. Joshi, B. L. Webber, and I. A. Sag (Eds.), *Elements of Discourse Understanding*, Cambridge University Press, 1981.

[DeLM]R. DeMillo, N. A. Lynch, and M. Merritt, Cryptographic Protocols, *Proceedings of the Fourteenth Annual ACM Symposium on the Theory of Computing*, 1982, pp. 383-400.

[DDS] D. Dolev, C. Dwork, and L. Stockmeyer, On the minimal synchronization needed for distributed consensus, *Proceedings of the 24th Annual Symposium on Foundations of Computer Science*, 1983, pp. 369-397.

[DHS] D. Dolev, J. Y. Halpern, and H. R. Strong, On the possibility and impossibility of achieving clock synchronization, *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, 1984, pp. 504-511.

[DRS] D. Dolev, R. Reischuk, and H. R. Strong, Eventual is earlier than immediate, *Proceedings of the 23th Annual Symposium on Foundations of Computer Science*, 1982, pp. 196-203

[DS] D. Dolev H. R. Strong, Polynomial algorithms for multiple processor agreement, *Proceedings of the Fourteenth Annual ACM Symposium on the Theory of Computing*, 1982, pp. 401-407.

[DM] C. Dwork and Y. Moses, Knowledge and common knowledge in a Byzantine environment I: Crash failures, Proceedings of the Conference on Theoretical Aspects of Reasoning About Knowledge, Monterey, 1986, to appear.

[EO] A. El Gamal, and A. Orlitsky, Interactive data compression, *Proceedings of the 25th Annual Symposium on Foundations of Computer Science*, 1984, pp. 100–108.

[FH] R. Fagin and J. Y. Halpern, Belief, awareness, and limited reasoning, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, 1985, pp. 491-501.

[FHV] R. Fagin, J. Y. Halpern, and M. Y. Vardi, A model-theoretic analysis of knowledge: preliminary report, *Proceedings of the 25th IEEE Symposium on Foundations of Computer Science*, 1984, pp. 268-278.

[FV] R. Fagin and M. Y. Vardi, Knowledge and implicit knowledge in a distributed environment, *Proceedings of the Conference on Theoretical Aspects of Reasoning About Knowledge*, Monterey, 1986.

[Fis] M. J. Fischer, The consensus problem in unreliable distributed systems (A brief survey), *Yale University Technical Report YALEU/DCS/RR-273*, 1983.

[FI] M. J. Fischer and N. Immerman, Foundations of knowledge for distributed systems, *YALEU/DCS/TR 426,* September, 1985.

[FL] M. J. Fischer and N. A. Lynch, A lower bound for the time to assure interactive consistency, *Information Processing Letters*, 14:4, 1982, pp. 183-186.

[FLP] M. J. Fischer, N. A. Lynch, and M. S. Paterson, Impossibility of distributed consensus with one faulty process, *Proceedings of the 2nd ACM Symposium on the Principles of Database Systems*, 1983.

[Fit] M. Fitting, Extending modal logics by fixpoint operators, unpublished note, 1984.

[Gal] R. G. Gallager, Seminar on Computer Communication Networks, *Office of Industrial Liason*, MIT, January 1979.

[GS] G. Gamow and M. Stern, Forty unfaithful wives, *Puzzle Math*, The Viking Press, New York, 1958, pp. 20–23.

[Gar] M. Gardner, *Puzzles from other worlds*, Vintage, 1984.

[Gr] J. Gray, Notes on data base operating systems, *IBM Research Report* RJ 2188, 1978.

[H] V. Hadzilacos, A lower bound for Byzantine agreement with fail-stop processors, *Harvard University Technical Report TR-21-83.*

[HF] J. Y. Halpern and R. Fagin, A formal model of knowledge, action, and communication in distributed systems, *Proceedings of the Fourth ACM Symposium on the Principles of Distributed Computing*, 1985, pp. 224-236.

[HMM] J. Y. Halpern, N. Megiddo, and A. Munshi, Optimal precision in the presence of uncertainty, *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, 1985, pp. 346-355; to appear in the *Journal of Complexity*.

[HM] J. Y. Halpern and Y. Moses, A guide to the modal logic of knowledge and belief, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, 1985, pp. 480-490.

[HM2] J. Y. Halpern and Y. Moses, A guide to the modal logic of knowledge and belief, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, 1985, pp. 480-490.

[HR] J. Y. Halpern and M. O. Rabin, A logic to reason about likelihood, *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, 1983, pp. 310-319.

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

[Kon] K. Konolige, Belief and incompleteness, *SRI Artificial Intelligence Note 319,* SRI International, Menlo Park, California, 1984.

[Koz] D. Kozen, Results on the propositional μ-calculus, *Theoretical Computer Science* **27**, 1983, pp. 333-354.

[KP] D. Kozen and R. Parikh, An elementary proof of the completeness of PDL, *Theoretical Computer Science*, **14**:1, 1981, pp. 113-118.

[LF] L. Lamport and M. J. Fischer, Byzantine grenerals and transaction commit protocols, *SRI Technical Report Op.62*, 1982.

[Le] D. Lehmann, Knowledge, common knowledge, and related puzzles, *Proceedings of the Third Annual ACM Symposium on Principles of Distributed Computing*, 1984, pp. 62-67.

[Li] B. G. Lindsay, P. G. Selinger, C. Galtieri, J. N. Gray, R. A. Lorie, T. G. Price, F. Potzulo, and B. W. Wade, Notes on distributed databases, *IBM Research Report RJ* 2571, 1979.

[MP] Z. Manna and A. Pnueli, Verificaion of concurrent programs: The temporal framework, *The Correctness Problem in Computer Science* (R.S. Boyer and J.S. Moore, eds.), International Lecture series in Computer Science, Academic Press, London, 1981.

[MW] Z. Manna and P. Wolper, Synthesis of communicating processes from temporal logic specifications, *ACM Transactions on Programming Languages and Systems*, **6**:1, 1984, pp. 68-93.

[Mc] J. McCarthy, Lecture Notes 1968 — 1974.

[Mi] P. Milgrom, An axiomatic characterization of common knowledge, *Econometica*, **49**:1, 1981.

[MDH] Y. Moses, D. Dolev, and J. Y. Halpern, Cheating husbands and other stories: a case study of knowledge, action, and communication, *Proceedings of the Fourth ACM conference on the Principles of Distributed Computing*, 1985, pp. 215-223.

[PR] R. Parikh and R. Ramanujam, Distributed processes and the logic of knowledge (preliminary report), *Proceedings of the Workshop on Logics of Programs*, 1985, pp. 256-268.

[PSL] M. Pease, R. Shostak, and L. Lamport, Reaching agreement in the presence of faults, *JACM*, **27**:2, 1980, pp. 228-234.

[R] S. J. Rosenschein, Formal theories of knowledge in AI and robotics, unpublished manuscript, 1985.

[Sa] M. Sato, A study of Kripke-type models of some modal logics by Gentzen's sequential method, *Publications of the Research Institute for Mathematical Sciences, Kyoto University* **13**:2, 1977.

[T] A. Tarski, A lattice-theoretic fixpoint theorem and its applications, *Pacific Journal of Mathematics*, **5**, 1955, pp. 285-309.

[YC] Y. Yemini and D. Cohen, Some issues in distributed processes communication, *Proceedings of the 1st International Conference on Distributed Computing Systems*, 1979, pp. 199-203.