FOUNDATIONS OF KNOWLEDGE FOR DISTRIBUTED SYSTEMS

Michael J. Fischer and Neil Immerman

YALEU/DCS/TR 450
Revision of TR 426
December, 1985

FOUNDATIONS OF KNOWLEDGE FOR DISTRIBUTED SYSTEMS

Michael J. Fischer[*] and Neil Immerman[†]
Computer Science Department
Yale University
New Haven, CT 06520

Abstract

We give a simple, yet very general definition for distributed protocols. We then define notions of knowledge and common knowledge appropriate for these protocols. We study how changes in the states of knowledge relate to more standard notions of computation. We find that by restricting our formulas to certain sets of global states we can realize different, appropriate definitions of knowledge with fundamentally different properties.

[*]Research supported in part by the National Science Foundation under Grant Number DCR-8405478 and by the Office of Naval Research under Contract Number N00014-82-K-0154.

[†]Research supported by an NSF postdoctoral fellowship and the Mathematical Sciences Research Insitute, Berkeley.

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER YALEU/DCS/TR 450 - Revision of TR426	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) FOUNDATIONS OF KNOWLEDGE FOR DISTRIBUTED SYSTEMS		5. TYPE OF REPORT & PERIOD COVERED Technical Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Michael J. Fischer and Neil Immerman		8. CONTRACT OR GRANT NUMBER(s) NSF- DCR-8405478 ONR- N00014-82-K-0154
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Computer Science/ Yale University 10 Hillhouse Avenue New Haven, CT 06520		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS NSF, Washington, DC, 20550/ Office of Naval Research 800 North Quincy, Arlington, VA 22217		12. REPORT DATE December, 1985
		13. NUMBER OF PAGES 18
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Office of Naval Research 800 North Quincy Arlington, VA 22217		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distributed 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)

Distributed system
Logic of knowledge

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

We give a simple, yet very general definition for distributed protocols. We then define notions of knowledge and common knowledge appropriate for these protocols. We study how changes in the states of knowledge relate to more standard notions of computation. We find that by restricting our formulas to certain sets of global states we can realize different, appropriate definitions of knowledge with fundamentally different properties.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

1 Introduction

Knowledge and common knowledge are intuitive concepts that help us reason about ordinary everyday situations in which we have only partial information. They become more complicated when other agents in the situation are intelligent and have reasoning power, for then our state of knowledge contains not only facts about the world but also facts about the state of knowledge of others, and how these states change over time depends on the agents' reasoning ability as well as on the occurrence of external events. (Cf. the "muddy children's problem" in [HM84].) It is appealing to use these concepts to reason about distributed protocols, for processors of a distributed system can be thought of as independent agents with only partial information about the global state of the system. To be sure our reasoning is correct, it is necessary to have rigorous and precise definitions of the intuitive concepts underlying such terms as "global state", "time", "knowledge", "message", etc. Only then can we be sure to avoid the circularities and inconsistencies that are all too common in informal reasoning.

Halpern and Moses informally define various notions of "knowledge" and "common knowledge" in the context of a particular model of distributed systems in which every processor has a clock and stores in its state the entire history of messages sent to it [HM84]. They argue that while common knowledge is desirable, it is unattainable in many realistic settings. They suggest a hierarchy of weakened versions of common knowledge and discuss conditions under which these can be achieved.

We find the assertion that "common knowledge is not attainable in real world systems"[1] to be at variance with our intuition, for it seems clear that "common knowledge" in the intuitive sense *is* attained in the real world. To understand this disparity between the formal model and our intuition, we examine, simplify, and make more precise the informal definitions given in [HM84].

We first give quite general and simple definitions of distributed protocol, knowledge and common knowledge. Under these simplified definitions, the arguments of [HM84], suitably formalized, still apply to show the impossibility of attaining common knowledge in systems without globally simultaneous transitions. We then show that it is not necessary to discard the notions of knowledge and common knowledge in favor of weaker ones in order to obtain realistic and useful definitions; rather, one can discard the assumption that formulas be interpretable at every global state and instead interpret them only at a subset of "safe" states. This is analogous to notions of database consistency in which the database is only required to be consistent at times when no transaction is in the middle of execution. Using the same definitions as before but restricted to safe states, we get a new and different notion of common knowledge which *can* be attained in situations where Halpern-Moses common knowledge cannot.

We conclude that formalizing these concepts is subtle, and seemingly innocuous assumptions can lead to unexpected results. Our desire is to formalize concepts of knowledge so that they may aid the design of distributed algorithms and clear proofs of their properties. We believe we have provided a solid base for future work in this area.

[1] [HM84], Conclusions.

2 Distibuted Protocols

2.1 A General Model

Definition 2.1 *A* distributed protocol,

$$\mathcal{P} = \langle n, Q, I, \tau \rangle,$$

consists of a number n of participants, a set Q of local states, a set $I \subseteq Q^n$ of initial global states, and a next move relation $\tau \subseteq Q^n \times Q^n$ on global states.

For any protocol $\mathcal{P}$, let $R_\mathcal{P}$ be the τ-reachable global states of $\mathcal{P}$, that is, the set of all global states we can reach by starting in I and taking any number of τ steps. By definition, only the reachable global states can occur in a run of $\mathcal{P}$. In general it may be a complex task to tell if a given element of Q^n is in $R_\mathcal{P}$; however, in this paper we will only be concerned with the reachable global states. For $p \in Q^n$ a global state and i a participant, we write $(p)_i$ to denote the i^{th} component of p. We will also use the notation $p \overset{i}{\sim} q$ to mean that $p, q \in R_\mathcal{P}$ and $(p)_i = (q)_i$, i.e. they are indistinguishable from i's point of view. Obviously each $\overset{i}{\sim}$ is an equivalence relation.[2]

Our definition of protocol is certainly simple and precise. Let us argue that it is also sufficiently general. Anything we would be willing to call a distributed system can be broken up into a finite number of logical entities which we call "participants". A participant may be any component of a system: a processor, a buffer, a clock, etc.[3] Each participant has some total configuration that we are calling its (local) state. Furthermore, the states of all the participants combined should determine the entire state of the system and thus which global states can next be entered.

It is easy to see for example that our model of distributed system is a generalization of the shared variable model of Lynch and Fischer [LF81]. In that model, the participants consist of shared variables and processors. Each action involves exactly one processor and one shared variable.

Similarly our model includes synchronous protocols in which every processor sends a message to every other during each round. One way to model this is to specify that the set of possible states is of the form $Q = M^n$, i.e. each processor's total configuration consists of an n-tuple. We can specify that the i^{th} entry of j's state is the value of the message sent from i to j during the previous round. This can be done as follows: for all processors i, j, and for all global states

[2]Our definition of knowledge given in Section 3 will be that of inherent knowledge—those facts that a participant could deduce given arbitrary computational power. Thus, in a global state $p \in R_\mathcal{P}$, a participant i will know that we are in some $q \in R_\mathcal{P}$ with $q \overset{i}{\sim} p$, and its knowledge will consist exactly of those facts true in all such q. The problem of determining membership in $R_\mathcal{P}$ is not relevant to the notions considered in this paper. It certainly is relevant, however, when considering knowledge in cryptographic protocols when an explicit bound is given on the computational resources of the participants.

[3]This approach is different from most of the other formal specifications of protocols of which we are aware, e.g. [CM85], [HM85], [HF85], [PR85]. The more usual approach is to say that after a message is sent it may sometime later be received by the addressee, but a message in transit is not explicitly modeled.

p, q, r, s, if $\langle p,q\rangle$ and $\langle r,s\rangle$ are in τ and if processor i has the same state in p as in r, then the i^{th} component of processor j's state is the same in q as in s.

The sense in which our model could be too general is that we allow any transition relation τ. Of course, for certain applications we can make appropriate restrictions. We have already seen that we can restrict our attention to processors which communicate with shared variables, or to synchronous message passing protocols. Similarly, instead of letting each processor's transitions be perfectly general, we can restrict our attention to processors with specified computing power, e.g. finite automata, polynomial time Turing machines, etc.

One interesting kind of restriction to place on the transition relation τ is locality.

Definition 2.2 *Let $T = \{i_1, \ldots, i_k\}$ be a set of participants, and let $E_T(p,q)$ hold if $(p)_i = (q)_i$ for all $i \in T$.*

- *We say that $\langle p,q\rangle$* affects only *participants in T if $E_{\overline{T}}(p,q)$ holds, where $\overline{T}$ is the set of participants not in T.*
- *We say that $\langle p,q\rangle \in \tau$* is enabled by *$T$ if for all p' such that $E_T(p',p)$, then $\langle p',q'\rangle \in \tau$ for the (unique) q' such that $E_T(q',q)$ and $E_{\overline{T}}(q',p')$.*
- *We say $\langle p,q\rangle \in \tau$* is local to *$T$ if it affects only participants in T and is enabled by T.*
- *We say that a protocol is* pairwise local *if every transition in τ is local to some set T of size two.*

Thus, a transition is local to T if only coordinates belonging to T are changed during the transition and if the local states of the participants not in T have no effect on whether or not this transition can occur. Note that a transition local to T does not necessarily affect all of the members of T. Note also that the same transition can be local to two sets T_1 and T_2 but not be local to their intersection. Consider for example the transition $\alpha = \langle(0,1,1,1),(1,1,1,1)\rangle$, and suppose τ contains every transition of the form $\langle(0,x_2,x_3,x_4),(1,x_2,x_3,x_4)\rangle$ *except* for $\langle(0,0,0,0),(1,0,0,0)\rangle$. Let $T_i = \{1,i\}$, $i = 2,3,4$. Then α affects only T_i (since only the first component changes), and α is enabled by T_i (since no matter how the components outside of T_i are changed, participant 1 can still make the transition from state 0 to 1). Thus, α is local to T_i for each i, but clearly α is not local to $\bigcap_i T_i = \{1\}$.

2.2 Comparison with Other Models

The shared variable model [LF81] is pairwise local: each transition is local to one processor and one shared variable. On the other hand the synchronous protocol described above is not pairwise local: each transition in general affects all n participants. We believe that if one models a distributed system at a sufficiently fine level then it will be pairwise local simply because it is difficult to insure that distant events occur simultaneously. However it is sometimes convenient to discuss synchronous protocols when the finer analysis would only obscure what is going on.

The models described in [CM85], [HM85], [HF85], [PR85] are all asynchronous message passing systems and thus are pairwise local when translated to our protocols. For readers more familiar with these other models, we will now consider one of them in more detail. Chandy and Misra [CM85] consider systems of n processors in which there are three disjoint sets of transitions: sends, receives, and local events. The relation between sends and receives is that each receive must correspond to a unique earlier send. The Chandy and Misra model forces the processors to remember their entire local history. Furthermore, the ability to perform a transition depends only on the local history of the affected processor except in the case of a receive, which also requires that the message to be received has already been sent. We can characterize the Chandy and Misra model in terms of our protocol model as follows:

Proposition 2.3 *The Chandy and Misra model [CM85] is isomorphic to the protocol model $\mathcal{P} = \langle n+1, Q, I, \tau \rangle$:*

1. *Participants* 1 *to* n *are the processors and participant* $n+1$ *is the message buffer.*
2. *There are sets* E *of local events and* M *of messages. A* message triple $\langle i, j, m \rangle$ *is a send from* i *and a receive by* j *of message* m. *The local state of processor* i *is a list of local events and message triples, each of which is a send from* i *or a receive by* i. *The local states of the buffer consist of any multiset of message triples.*
3. *The set of initial global states is the singleton* $I = \langle \lambda, \ldots, \lambda, \emptyset \rangle$ *in which all processors have the empty list* λ *as their local history and the buffer is empty.*
4. *Each transition is of one of the following three forms. Transition (a) only involves participant* i, *and transitions (b) and (c) only involve participants* i *and* $n+1$.
   (a) *Local event* $e \in E$ *at processor* i: e *is appended to* i*'s local state.*
   (b) *Send of* $m \in M$ *from processor* i *to processor* j: *the message triple* $\langle i, j, m \rangle$ *is appended to* i*'s local state and added to* $n+1$*'s local state.*
   (c) *Receive by processor* i *of the message* $m \in M$ *sent by processor* k: *a message triple* $\langle k, i, m \rangle$ *is deleted from* $n+1$*'s local state and appended to* i*'s local state.*

2.3 Two Examples

We conclude this section with two nontrivial examples of protocols, one asynchronous and the other synchronous. These protocols will be frequently referred to in the remainder of the paper.

Both protocols model the situation of a completely connected network of n processors which operate in rounds. On each round, every processor i sends a message $m_{i,j}$ to each other processor j. After receiving all of the messages sent to it on the given round, processor i changes state and chooses a new set of messages to send out on the next round. The new state and message set depend on the old state and on the messages received during the round.

In protocol $\mathcal{A}$, the messages are sent asynchronously, one at a time, via message buffers. A buffer either contains a message or is empty. A message can only be sent to an empty buffer and only received from a non-empty buffer. Sending a message makes the buffer non-empty, and receiving a message makes the buffer empty again. Thus, the execution of each round takes many steps.

In protocol $\mathcal{B}$, all messages are sent and received in one big synchronous step, that is, one step of $\mathcal{B}$ corresponds to an entire round of $\mathcal{A}$.

2.3.1 Protocol $\mathcal{A}$

Let $\mathcal{A} = \langle n^2, Q_{\mathcal{A}}, I_{\mathcal{A}}, \tau_{\mathcal{A}} \rangle$ be an asynchronous message passing protocol defined as follows: The first n participants of $\mathcal{A}$ are the processors $a_1, \ldots, a_n$; the remaining $(n-1)n$ participants are buffers. For a global state p, we abuse our previous notation slightly and write $(p)_{a_i}$ to denote the component corresponding to a_i and $(p)_{b_{i,j}}$ to denote the component corresponding to buffer $b_{i,j}$.

The set of possible local states of a buffer $b_{i,j}$, $i \neq j$, is $Q^b_{\mathcal{A}} = M \cup \{\lambda\}$, where M is a set of possible messages and λ is a special symbol denoting the null message. $(p)_{b_{i,j}} = m \in M$ indicates that the single message m was sent by i but not yet delivered to j in global state p, and $(p)_{b_{i,j}} = \lambda$ indicates that no message is waiting.

The set of possible local states of a processor a_i is $Q^a_{\mathcal{A}} = (D \times (M \cup \{\lambda\})^{n-1} \times \mathrm{N})$. State $\langle d, m_1, \ldots, m_{i-1}, m_{i+1}, \ldots, m_n, r\rangle$ indicates that the processor is in internal state d at round $\lfloor r/2 \rfloor$ with pending messages $m_1, \ldots, m_{i-1}, m_{i+1}, \ldots, m_n$. If r is even, then the processor is in a 'send' state, waiting to place each $m_j \neq \lambda$ into buffer $b_{i,j}$. If r is odd, then the processor is in a 'receive' state waiting to fetch a message from $b_{j,i}$ for each j such that $m_j = \lambda$.

Thus, the complete set of local states $Q_{\mathcal{A}}$ is $Q^a_{\mathcal{A}} \cup Q^b_{\mathcal{A}}$.

The transitions making up $\tau_{\mathcal{A}}$ are of four kinds:

1. $\langle p, q \rangle \in \mathit{Send}_{i,j}$ if
   - $p \overset{c}{\sim} q$ for all $c \notin \{a_i, b_{i,j}\}$;
   - $(p)_{b_{i,j}} = \lambda$;
   - $(q)_{b_{i,j}} = m_j \neq \lambda$;
   - $(p)_{a_i} = \langle d, \ldots, m_{j-1}, m_j, m_{j+1}, \ldots, 2k \rangle$;
   - $(q)_{a_i} = \langle d, \ldots, m_{j-1}, \lambda, m_{j+1}, \ldots, 2k \rangle$.
2. $\langle p, q \rangle \in \mathit{End\_send}_i$ if
   - $p \overset{c}{\sim} q$ for all $c \neq a_i$;
   - $(p)_{a_i} = \langle d, \lambda, \ldots, \lambda, 2k \rangle$;
   - $(q)_{a_i} = \langle d, \lambda, \ldots, \lambda, 2k+1 \rangle$.
3. $\langle p, q \rangle \in \mathit{Receive}_{i,j}$ if
   - $p \overset{c}{\sim} q$ for all $c \notin \{a_i, b_{j,i}\}$;
   - $(p)_{b_{j,i}} = m_j \neq \lambda$;
   - $(q)_{b_{j,i}} = \lambda$;
   - $(p)_{a_i} = \langle d, \ldots, m_{j-1}, \lambda, m_{j+1}, \ldots, 2k+1 \rangle$;

- $(q)_{a_i} = \langle d, \ldots, m_{j-1}, m_j, m_{j+1}, \ldots, 2k+1\rangle$.

4. $\langle p, q\rangle \in \mathit{End\_receive}_i$ if

   - $p \overset{c}{\sim} q$ for all $c \neq a_i$;
   - $(p)_{a_i} = \langle d, m_1, \ldots, m_n, 2k+1\rangle$, where $m_j \neq \lambda$ for all $j \neq i$;
   - $(q)_{a_i} = \langle d', m'_1, \ldots, m'_n, 2k+2\rangle$, where $m'_j \neq \lambda$ for all $j \neq i$, and d' and m'_j are functions of $(p)_{a_i}$.

Now we let $\tau_{\mathcal{A}}$ consist of all the above transitions:

$$\tau_{\mathcal{A}} = \bigcup_{i,j} \mathit{Send}_{i,j} \cup \mathit{End\_send}_i \cup \mathit{Receive}_{i,j} \cup \mathit{End\_receive}_i.$$

Finally let $I_{\mathcal{A}}$ be some nonempty set of global states in which the state of every processor a_i has the form $(d, m_1, \ldots, m_n, 0)$ with $m_j \neq \lambda$ for all $j \neq i$, and all the buffers are empty.

2.3.2 Protocol $\mathcal{B}$

Our second example of a protocol is a synchronous version of $\mathcal{A}$. Let $\mathcal{B} = \langle n^2, Q_{\mathcal{B}}, I_{\mathcal{B}}, \tau_{\mathcal{B}}\rangle$, where

$$\begin{aligned} Q^a_{\mathcal{B}} &= D \times M^{n-1} \times \{2r \mid r \in \mathrm{N}\}, \\ Q^b_{\mathcal{B}} &= \{\lambda\}, \\ Q_{\mathcal{B}} &= Q^a_{\mathcal{B}} \cup Q^b_{\mathcal{B}}. \end{aligned}$$

Let $\overline{Q}_{\mathcal{B}} = (Q^a_{\mathcal{B}})^n \times (Q^b_{\mathcal{B}})^{n(n-1)}$. Let the transitions $\tau_{\mathcal{B}}$ consist of all pairs $\langle p, q\rangle \in \tau^*_{\mathcal{A}} \cap (\overline{Q}_{\mathcal{B}} \times \overline{Q}_{\mathcal{B}})$ such that no $\tau_{\mathcal{A}}$ path in $\mathcal{A}$ from p to q goes through intermediate global states in $\overline{Q}_{\mathcal{B}}$. Finally let $I_{\mathcal{B}} = I_{\mathcal{A}}$.

We see that the reachable global states of $\mathcal{B}$ all belong to $\overline{Q}_{\mathcal{B}}$. Thus, all buffers are always empty and the local states of the a_i's are the corresponding states from $\mathcal{A}$ at the beginning of a round. It is not hard to see that $\mathcal{B}$ is a synchronous version of $\mathcal{A}$ such that in each round all processors send $n-1$ messages and then receive $n-1$ messages.

3 Definitions of Knowledge and Common Knowledge

It is convenient to picture a protocol $\mathcal{P}$ as a graph with nodes consisting of all the elements of $R_{\mathcal{P}}$. There is a directed edge labelled τ from p to q just if $\langle p, q\rangle \in \tau$. Furthermore there is an undirected edge labelled 'i' between p and q just if $p \overset{i}{\sim} q$.[4]

[4]The reader familiar with Kripke models will observe that an alternate description of a protocol $\mathcal{P}$ is as a Kripke model $\mathcal{K} = \langle R_{\mathcal{P}}, \tau, \overset{1}{\sim}, \ldots, \overset{n}{\sim}\rangle$ where the $\overset{i}{\sim}$'s are equivalence relations; and furthermore for all worlds $w, w' \in R_{\mathcal{P}}$ if $w \overset{i}{\sim} w'$ for all i then $w = w'$. See [FI85] where this characterization of protocols is used to obtain a simple proof that propositional logic of knowledge and branching time is EXPTIME complete.

Let E be an equivalence relation on the reachable global states $R_{\mathcal{P}}$ with equivalence classes $[p]_E$, $p \in R_{\mathcal{P}}$. Corresponding to E is a modal operator $\Box(E)$. For any sentence α,[5] it is natural to make the following definition of $\Box(E)\,\alpha$, which we read as "box E α":

$$\langle \mathcal{P}, p\rangle \models \Box(E)\,\alpha \quad \equiv \quad \forall q \in [p]_E(\langle \mathcal{P}, q\rangle \models \alpha).$$

Thus, $\Box(E)\,\alpha$ holds just if α is true in all the worlds E-equivalent to the current world.

Several equivalence relations will be of interest. $\overset{i}{\sim}$, the indistinguishability relation for participant i, has already been defined. We denote $\Box\left(\overset{i}{\sim}\right)$ by K_i, which we read "i knows." Thus, i knows α just if α is true in all worlds which are indistinguishable by i from the current world.

We generalize to a group of participants $G \subseteq \{1, \ldots, n\}$. Let

$$\overset{G}{\approx} \;=\; \left(\bigcup_{i \in G} \overset{i}{\sim}\right)^{*},$$

that is, the transitive closure of the union of the $\overset{i}{\sim}$ relations for $i \in G$. Thus, we have

$$p \overset{G}{\approx} q \Leftrightarrow (\exists r \geq 0)(\exists i_1, \ldots, i_r \in G)(\exists p_1, \ldots, p_{r-1})[p \overset{i_1}{\sim} p_1 \overset{i_2}{\sim} p_2 \ldots p_{r-1} \overset{i_r}{\sim} q].$$

We denote $\Box\left(\overset{G}{\approx}\right)$ by C_G, which we read "it is common knowledge among the members of G". Note that $\overset{\{i\}}{\approx} = \overset{i}{\sim}$, so also $\mathsf{C}_{\{i\}} \equiv \mathsf{K}_i$. We write $\approx$ for $\overset{G}{\approx}$ and C for C_G in the special case that G includes all participants.

The next result shows that $\mathsf{C}\alpha$ coincides with the definition current in the literature. (See for example [HM84].)

Theorem 3.1 *The following two statements are equivalent:*

1. $\langle \mathcal{P}, p\rangle \models \mathsf{C}_G\alpha$.
2. $(\forall r \geq 0)(\forall i_1, \ldots, i_r \in G)(\langle \mathcal{P}, p\rangle \models \mathsf{K}_{i_1}\mathsf{K}_{i_2}\ldots\mathsf{K}_{i_r}\alpha)$.

Proof

$(1 \Rightarrow 2)$: For any β, we have $\mathsf{C}_G\beta \rightarrow \beta$ since $p \overset{G}{\approx} p$. Thus, it suffices to show that for any β, if $\langle \mathcal{P}, p\rangle \models \mathsf{C}_G\beta$, then for all $i \in G$, $\langle \mathcal{P}, p\rangle \models \mathsf{C}_G\mathsf{K}_i\beta$. This is clear because if $q \overset{G}{\approx} p$ and $q' \overset{i}{\sim} q$ then $q' \overset{G}{\approx} p$; hence $\langle \mathcal{P}, q\rangle \models \mathsf{K}_i\beta$. Since $\mathsf{K}_i\beta$ holds for all $q \in [p]_{\overset{G}{\approx}}$, it is common knowledge in G at p, as desired.

[5]We have intentionally left the logical language unspecified from which the sentence α is drawn, for all that we require is that it be possible to interpret α at the pair $\langle \mathcal{P}, p\rangle$ (and we weaken even that requirement in Section 5). Of course the strongest such language would have a way to express each possible subset of $R_{\mathcal{P}}$. The languages we consider may be taken to be some unspecified subset of this strongest language.

(2 ⇒ 1): Suppose that $\langle \mathcal{P}, p\rangle \not\models \mathsf{C}_G\alpha$. It follows that there is a $q \in [p]_{\underset{\approx}{G}}$ such that $\langle \mathcal{P}, q\rangle \models \neg\alpha$. Let $i_1, \ldots, i_r \in G$ be such that there exists $p_1, \ldots, p_{r-1}$ with $p \overset{i_1}{\sim} p_1 \overset{i_2}{\sim} p_2 \ldots p_{r-1} \overset{i_r}{\sim} q$. It follows that $\langle \mathcal{P}, p\rangle \models \neg\mathsf{K}_{i_1}\mathsf{K}_{i_2}\ldots\mathsf{K}_{i_r}\alpha$. ∎

As an example of the above concepts, the next proposition shows that for the synchronous protocol $\mathcal{B}$ described at the end of the last section, whenever a processor is in round r, it is common knowledge among all the processors that they are in round r. We will see in the next section that this assertion is false for the asynchronous protocol $\mathcal{A}$.

Proposition 3.2 *Let $P = \{1, \ldots, n\}$ be the set of processors in protocol $\mathcal{B}$ (omitting the buffers). Let $i, j \in P$, let $\alpha(i, r)$ be a formula meaning that processor i is in round r, and let $q \in R_{\mathcal{B}}$ be any reachable global state. Then*

$$\langle \mathcal{B}, q\rangle \models \alpha(i, r) \rightarrow \mathsf{C}_P\alpha(j, r).$$

Proof It is trivial to show by induction that for all $p \in R_{\mathcal{B}}$, all the processors are in the same round. Therefore suppose that $\langle \mathcal{B}, q\rangle \models \alpha(i, r)$. It follows that for all $p \in [q]_{\underset{\approx}{P}}$, $\langle \mathcal{B}, p\rangle \models \alpha(j, r)$. ∎

4 Common Knowledge in Asynchronous Systems

Informally, an "asynchronous system" has two kinds of participants, "active" and "passive". Typically, the active elements are processors and the passive elements are memory cells or message ports. Every step of such a system consists of an interaction between an active and a passive element, and whether or not a step can occur depends only on the states of the element pair involved.

In our more general model, we have only one kind of participant, so we define a protocol to be ***asynchronous*** if it is pairwise local as defined in Section 2. Thus, every interaction "involves" at most two participants, and two steps involving disjoint sets of participants can occur in either order with the same effect.

The theorem that follows depends on much less than full asynchrony. Thus, we define a very weak notion of an asynchronous protocol that we call "nonsimultaneous".

Definition 4.1 *Let $G \subseteq \{1, \ldots, n\}$ be a specified set of participants in a protocol $\mathcal{P} = \langle n, Q, I, \tau\rangle$. We will call $\mathcal{P}$* nonsimultaneous *with respect to G if for all $\langle p, q\rangle \in \tau$, there is a participant $i \in G$ not affected by the transition $\langle p, q\rangle$. We say that $\mathcal{P}$ is* nonsimultaneous *if it is nonsimultaneous with respect to the set of all participants.*

As an example, note that a message passing protocol such as in [CM85] is nonsimultaneous with respect to its set of processors provided it has at least two processors. Note also that any asynchronous protocol with at least three participants is nonsimultaneous.

The following theorem shows that in a nonsimultaneous protocol, no new common knowledge can be acheived. (Cf. [HM84], Theorem 3.)

Theorem 4.2 *Let $\mathcal{P}$ be a protocol and G a set of participants. Let p be any global state in $R_{\mathcal{P}}$ and let p_0 be an intial state from which p is reachable by a sequence of τ steps, all of which are nonsimultaneous with respect to G. Let α be any sentence in a logic for $\mathcal{P}$. Then $\langle \mathcal{P}, p\rangle \models \mathsf{C}_G\alpha$ iff $\langle \mathcal{P}, p_0\rangle \models \mathsf{C}_G\alpha$.*

Proof It suffices to show that $q \overset{G}{\approx} r$ for any step $\langle q, r\rangle \in \tau$ that is nonsimultaneous with respect to G, for then $p_0 \overset{G}{\approx} p$ follows by considering the path of steps from p_0 to p, and the theorem then follows from the definition of common knowledge in G. But if $\langle q, r\rangle \in \tau$ is nonsimultaneous with respect to G, then there must exist a participant $j \in G$ unaffected by the transition, i.e. such that $q \overset{j}{\sim} r$. It follows that $q \overset{G}{\approx} r$. ∎

Corollary 4.3 *Let G be a set of participants in a protocol $\mathcal{P}$ that is nonsimultaneous with respect to G. Then it is impossible to gain new common knowledge among the members of G.*

As an example, consider the protocol $\mathcal{A}$ discussed at the end of the last section. It is easy to check that $\mathcal{A}$ is nonsimultaneous with respect to any set of G participants including at least two processors or at least two buffers. Thus, no new common knowledge among the members of any such G can arise in $\mathcal{A}$. This is in sharp contrast to the situation for $\mathcal{A}$'s cousin $\mathcal{B}$ (cf. Proposition 3.2).

It would seem at first glance that the difficulty in achieving common knowledge has to do with the problem of reaching an arbitrary depth of K's with only finitely many messages. We conclude this section with a look at finite state protocols where common knowledge is equivalent to a *bounded* stack of K's.

Theorem 4.4 *Let $\mathcal{P} = \langle n, Q, I, \tau\rangle$ be a finite state protocol, i.e. $|Q| < \infty$. For each i, let*

$$Q_i = \{(q)_i \mid q \in R_{\mathcal{P}}\}.$$

Thus, each processor is a $|Q_i|$-state automaton. Let $r = \min\{|Q_i| \mid 1 \le i \le n\}$. Let p be any global state and let α be any formula. Then the following are equivalent:

1. $\langle \mathcal{P}, p\rangle \models \mathsf{C}\alpha$.
2. *For all* $i_1, i_2, \ldots, i_{2r-1}, (\langle \mathcal{P}, p\rangle \models \mathsf{K}_{i_1} \ldots \mathsf{K}_{i_{2r-1}}\alpha)$.

Proof

$(1 \Rightarrow 2)$: By definition of C.

(2 $\Rightarrow$ 1): Suppose that $\langle \mathcal{P},p\rangle \not\models \mathsf{C}\alpha$. Then there must exist $q \in [p]_{\approx}$ such that $\langle \mathcal{P},q\rangle \models \neg\alpha$. Consider a minimum length $\sim$ chain from p to q:

$$p = p_0 \overset{i_1}{\sim} p_1 \overset{i_2}{\sim} p_2 \ldots p_{s-1} \overset{i_s}{\sim} p_s = q$$

No nonconsecutive pair p_j, p_k of global states agree on any component because if they did the chain could be shortened. It follows that in any given component, each state appears at most twice. Therefore $s \leq 2r - 1$. It follows that

$$\langle \mathcal{P},p\rangle \models \neg\mathsf{K}_{i_1}\mathsf{K}_{i_2}\ldots\mathsf{K}_{i_{2r-1}}\alpha.$$

∎

Example 4.5 *Consider the protocol* $\mathcal{P}_r = \langle 2, \{1,\ldots,r+1\}, \{\langle 1,1\rangle\}, \tau_r\rangle$ *where,*

$$\tau_r = \{(\langle i,i\rangle, \langle i+1,i\rangle) \mid 1 \leq i \leq r\} \cup \{(\langle i+1,i\rangle, \langle i+1,i+1\rangle) \mid 1 \leq i < r\}.$$

This protocol has the unique computation chain:

$$\langle 1,1\rangle,\ \langle 2,1\rangle,\ \langle 2,2\rangle,\ \langle 3,2\rangle,\ldots,\langle r,r-1\rangle,\ \langle r,r\rangle,\ \langle r+1,r\rangle.$$

Furthermore, for all reachable global states p and q, we have $p \approx q$. Thus, for any α,

$$\langle \mathcal{P}_r,p\rangle \models \mathsf{C}\alpha \Leftrightarrow \textit{for all } q \in R_{\mathcal{P}_r}, \langle \mathcal{P}_r,q\rangle \models \alpha.$$

Let α say that processor 1 is not in state 1. Then $\langle \mathcal{P}_r,\langle 1,1\rangle\rangle \not\models \alpha$, so $\langle \mathcal{P}_r,\langle r+1,r\rangle\rangle \not\models \mathsf{C}\alpha$. On the other hand, it is easily seen that

$$\langle \mathcal{P}_r,\langle r+1,r\rangle\rangle \models \mathsf{K}_1\underbrace{\mathsf{K}_2\mathsf{K}_1\mathsf{K}_2\mathsf{K}_1\ldots\mathsf{K}_2\mathsf{K}_1}_{2r-2}\alpha.$$

It follows that for all $i_1, i_2, \ldots, i_{2r-2} \in \{1,2\}$,

$$\langle \mathcal{P}_r,\langle r+1,r\rangle\rangle \models \mathsf{K}_{i_1}\ldots\mathsf{K}_{i_{2r-2}}\alpha,$$

showing that the bound in Theorem 4.4 cannot be improved.

5 Alternate Definitions of Knowledge

Halpern and Moses argue that, "If Cp is to be attained, all processors must start supporting it simultaneously."[6] Their conclusion is that in the absence of perfect global clocks and guaranteed exact message delivery times, one must settle for weaker notions than common knowledge. They suggest alternative notions and discuss when these can be acheived.

[6][HM84], Lemma 2.

We draw a differenct conclusion from the same problem, namely we believe that there is not a best or most desirable common knowledge which one would acheive if one could; but rather that different notions of knowledge and common knowledge may be appropriate for different protocols.

It is useful to consider our example protocols $\mathcal{A}$ and $\mathcal{B}$. We hope the reader will agree that they are realistic instances of an asynchronous and a synchronous message passing protocol, respectively. Recall that new common knowledge among the n processors is attainable in $\mathcal{B}$ but not in $\mathcal{A}$. This is confusing because in a very strong sense $\mathcal{A}$ and $\mathcal{B}$ are isomorphic protocols (cf. [CM85]).[7]

The difference between protocols $\mathcal{A}$ and $\mathcal{B}$ concerns the granularity at which processors in the two protocols may introspect. In $\mathcal{B}$, processors are only allowed to think about what they know at the start of each round. The fact that two structures whose observable behaviors are equivalent should differ so dramatically in terms of their knowledge gives us cause to reexamine our definitions of knowledge and common knowledge.

Let $\mathcal{P}$ be any protocol and let $S \subseteq R_{\mathcal{P}}$ be any subset of reachable global states. For each i, let $\overset{i}{\sim}_S$ be the restriction of $\overset{i}{\sim}$ to $S \times S$. As before, let $G \subseteq \{1, \ldots, n\}$ be a group of the participants in a protocol. Let

$$\overset{G}{\approx}_S \;=\; \left(\bigcup_{i \in G} \overset{i}{\sim}_S\right)^{*},$$

that is, the transitive closure of the union of the $\overset{i}{\sim}_S$ relations for $i \in G$. Thus, we have

$$\begin{aligned} p \overset{G}{\approx}_S q \quad \Leftrightarrow \quad & p, q \in S \text{ and} \\ & (\exists r \geq 0)(\exists i_1, \ldots, i_r \in G)(\exists p_1, \ldots, p_{r-1} \in S)[p \overset{i_1}{\sim}_S p_1 \overset{i_2}{\sim}_S p_2 \ldots p_{r-1} \overset{i_r}{\sim}_S q]. \end{aligned}$$

We generalize our previous definitions of knowledge by letting K_i^S denote the modal operator $\Box\left(\overset{i}{\sim}_S\right)$ and by letting C_G^S denote the modal operator $\Box\left(\overset{G}{\approx}_S\right)$. As before, we omit mention of G when G includes all participants.

The intuitive meaning of "$\mathsf{K}_i^S\alpha$" is, "Participant i knows that α holds, assuming we are in S," and the intuitive meaning of "$\mathsf{C}_G^S\alpha$" is, "It is common knowledge among the members of G that α holds, assuming we are in S." The following theorem makes this intuition precise.

Theorem 5.1 *Let $S \subseteq R_{\mathcal{P}}$, let $p \in S$, and let σ mean, "We are in S." Let $G \subseteq \{1, \ldots, n\}$. Then*

1. $\langle \mathcal{P}, p \rangle \models \mathsf{K}_i^S \alpha \Leftrightarrow \langle \mathcal{P}, p \rangle \models \mathsf{K}_i(\sigma \rightarrow \alpha)$
2. $\langle \mathcal{P}, p \rangle \models \mathsf{C}_G^S \alpha \Leftrightarrow (\forall r \geq 0)(\forall i_1, \ldots, i_r \in G)(\langle \mathcal{P}, p \rangle \models \mathsf{K}_{i_1}^S \mathsf{K}_{i_2}^S \ldots \mathsf{K}_{i_r}^S \alpha)$.

[7] We will call a pair of protocols such as $\mathcal{A}$ and $\mathcal{B}$, all of whose interactions are accomplished by a series of messages, *isomorphic* if the set of messages sequences they generate is identical up to permutations which do not switch the order of a send and a receive by the same participant, nor the order of a send and its corresponding receive.

Proof (1) is immediate from the definition of K_i^S. The proof of (2) is similar to the proof of Theorem 3.1. ∎

For any protocol $\mathcal{P}$ and any nonempty $S \subseteq R_{\mathcal{P}}$, the operators K^S and C^S seem to satisfy all requirements stated in [HM84] for such knowledge operators. In particular we note that they satisfy Kripke's S5 axioms (cf. [La77]).

Proposition 5.2 *For any protocol $\mathcal{P}$, any nonempty $S \subseteq R_{\mathcal{P}}$, and $G \subseteq \{1, \ldots, n\}$, the operators K_i^S and C_G^S satisfy the S5 axioms for modal operators.*

Proof This is immediate from the fact that each $\overset{i}{\sim}_S$ and $\overset{G}{\approx}_S$ is an equivalence relation. ∎

Let us now consider the protocol $\mathcal{A}$ with $S = R_{\mathcal{B}}$. The following proposition relates knowledge in $\mathcal{B}$ to knowledge with respect to S in $\mathcal{A}$.

Proposition 5.3 *Let $S = R_{\mathcal{B}}$ and $p \in S$. Let α be any knowledge formula all of whose K's and C's have the superscript S. Let α' be the formula resulting from α when we remove all of the superscripts S.*[8] *Then*

$$\langle \mathcal{A}, p\rangle \models \alpha \Leftrightarrow \langle \mathcal{B}, p\rangle \models \alpha'$$

Proof This is an easy induction on the length of α. The most interesting case is when $\alpha = \mathsf{K}_i^S \varphi$. In this case

$$\begin{array}{ll} \langle \mathcal{A}, p\rangle \models \mathsf{K}_i^S \varphi & \\ \Leftrightarrow & (\text{for all } q \in [p]_{\overset{i}{\sim}_S})\langle \mathcal{A}, q\rangle \models \varphi \\ \Leftrightarrow & (\text{for all } q \in [p]_{\overset{i}{\sim}_S})\langle \mathcal{B}, q\rangle \models \varphi' \\ \Leftrightarrow & \langle \mathcal{B}, p\rangle \models \mathsf{K}_i \varphi' \end{array}$$

∎

It follows from Propositions 5.2 and 5.3 that if we consider the protocol $\mathcal{A}$ with $S = R_{\mathcal{B}}$, then we get a quite reasonable definition of knowledge and common knowledge. Furthermore, with these definitions new common knowledge is attained in an asynchronous protocol.

Joe Halpern [Ha85] points out that we are in a sense cheating because when we consider K^S, we evaluate formulas only at the global states in S. This form of 'cheating' may however be useful and appropriate. An example of a useful restriction of attention to a set of safe states occurs in databases. A database must maintain some integrity constraints which can be violated in the middle of certain transactions. It is useful to assert that the constraints are always satisfied and to evaluate such assertions only at the safe states in which no transactions are incomplete. Note that during a typical run of the database system, no such safe states need occur.

[8]Recall that we haven't specified the syntax of the knowledge-free formulas. Any such knowledge-free subformula γ of α specifies a certain subset $T \subseteq R_{\mathcal{A}}$. We assume that γ is changed to γ' in α' where γ' specifies the same subset restricted to $R_{\mathcal{B}}$, i.e. $T' = T \cap R_{\mathcal{B}}$.

6 Conclusions

We have given precise formulations of distributed protocols. For any subset S of the reachable states, we have given a precise definition of knowledge and common knowledge with respect to S. We have presented theorems outlining some cases where new common knowledge can be attained and some cases where it cannot. Most strikingly, we have shown that in certain situations two plausible choices for S can give completely different results.

One can now ask the question, "For which sets of protocols is there a 'best' choice for S?" and thus a 'best' definition for knowledge and common knowledge. We suspect that in at least certain situations there may be such a best S, and that in this case knowledge and common knowledge with respect to S may be valuable tools.

Many arguments in distributed systems are first formulated at the intuitive level of what certain processors 'know' at certain points in the computation. With precise definitions for these concepts, it may be easier to formulate clear and correct proofs. We believe that considerable work is needed in order to develop logical tools and demonstrate their usefulness on problems of interest in distributed systems.

Acknowledgements

We are grateful to Dick Stearns for helpful comments on an earlier draft of this paper and to Joe Halpern, Yoram Moses, Albert Meyer, and Nancy Lynch for probing questions, constructive criticism, and valuable suggestions on this work.

References

[CM85] K. Mani Chandy and Jayadev Misra, "How Processes Learn," *Fourth ACM Symp. on Principles of Distributed Computing* (1985), 204–214.

[DM86] Cynthia Dwork and Yoram Moses, "Knowledge and Common Knowledge in a Byzantine Environment I: Crash Failures," *this volume.*

[FI85] Michael J. Fischer and Neil Immerman, "Propositional Logic of Knowledge and Time is Exponential Time Complete," manuscript (1985).

[Ha85] J. Y. Halpern, personal communication.

[HF85] J. Y. Halpern and Ron Fagin, "A Formal Model of Knowledge, Action, and Communication," *Fourth ACM Symp. on Principles of Distributed Computing* (1985), 224-236.

[HM84] J. Y. Halpern and Y. Moses, "Knowledge and Common Knowledge in a Distributed Environment," *Third ACM Symp. on Principles of Distributed Computing* (1984), 50–61.

[HM85] J. Y. Halpern and Y. Moses, "Knowledge and Common Knowledge in a Distributed Environment," revised manuscript (October, 1985).

[La77] Richard Ladner, "The Computational Complexity of Provability in Systems of Modal Propositional Logic," *SIAM J. Comput. 6,* 3 (1977), 467-480.

[LF81] Nancy A. Lynch and Michael J. Fischer, "On Describing the Behavior and Implementation of Distributed Systems," *Theoretical Comp. Sci. 13* (1981), 17–43.

[PR85] Rohit Parikh and R. Ramanujam, "Distributed Processes and the Logic of Knowledge (preliminary report)," *Proc. of Workshop on Logics of Programs,* Brooklyn (1985), 256-268.

DISTRIBUTION LIST

Office of Naval Research Contract N00014-82-K-0154

Michael J. Fischer, Principal Investigator

Defense Technical Information Center
Building 5, Cameron Station
Alexandria, VA 22314
(12 copies)

Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Dr. R.B. Grafton, Scientific
Officer (1 copy)

Information Systems Program (437)
(2 copies)

Code 200 (1 copy)
Code 455 (1 copy)
Code 458 (1 copy)

Office of Naval Research
Branch Office, Pasadena
1030 East Green Street
Pasadena, CA 91106
(1 copy)

Naval Research Laboratory
Technical Information Division
Code 2627
Washington, D.C. 20375
(1 copy)

Dr. A.L. Slafkosky
Scientific Advisor
Commandant of the Marine Corps
Code RD-1
Washington, D.C. 20380
(1 copy)

Naval Ocean Systems Center
Advanced Software Technology Division
Code 5200
San Diego, CA 92152
(1 copy)

Mr. E.R. Gleissner
Naval Ship Research and Development Center
Computation and Mathematics Department
Bethesda, MD 20084
(1 copy)

Captain Grace M. Hopper
Naval Data Automation Command
Washington Navy Yard
Building 166
Washington, D.C. 20374
(1 copy)

Defense Advance Research Projects Agency
ATTN: Program Management/MIS
1400 Wilson Boulevard
Arlington, VA 22209
(3 copies)