FOUNDATIONS OF KNOWLEDGE FOR DISTRIBUTED SYSTEMS

Michael J. Fischer and Neil Immerson

YALEU/DCS/TR 426
September, 1985
FOUNDATIONS OF KNOWLEDGE FOR DISTRIBUTED SYSTEMS

Michael J. Fischer and Neil Immerman

YALEU/DCS/TR 426
September, 1985
**Title:** Foundations of Knowledge for Distributed Systems

**Authors:** Michael J. Fischer and Neil Immerman

**Performing Organization:** Department of Computer Science/ Yale University

**Contract or Grant Numbers:**
- NSF: DCR-8405478
- ONR: N00014-82-K-0154

**Report Date:** September, 1985

**Number of Pages:** 11

**Distribution Statement:** Approved for public release; distributed unlimited.

**Abstract:** General and precise definitions are given of distributed protocol and of various notions of knowledge and common knowledge in distributed systems. Whether or not a process in a distributed protocol can obtain new common knowledge during execution is shown to depend on which notion of common knowledge chosen, showing that the problem of capturing intuitive concepts about knowledge is more subtle than was previously believed.
Foundations of Knowledge for Distributed Systems

Michael J. Fischer* and Neil Immerman†

Computer Science Department
Yale University
New Haven, CT 06520

September 19, 1985

1 Introduction

In [HM84], Halpern and Moses present an interesting discussion of knowledge and common knowledge for distributed systems. They argue that while common knowledge is desirable, it is unattainable in certain settings. They suggest a hierarchy of weakened versions of common knowledge and discuss when these can be achieved.

One difficulty with the Halpern and Moses paper is that it is informal and the concepts dealt with are not rigorously defined. Given one interpretation, their theorems are true as claimed, but given another, the opposite occurs, as we show with appropriate counterexamples. Thus, their theorems are not wrong in spirit, but the concepts are subtle and the terms must be more carefully defined.

In the main body of this paper, we give quite general and precise definitions of distributed protocol, knowledge, and common knowledge. We also provide precise settings and rigorous proofs for many of the results in [HM84]. We then propose some alternate definitions in which the same results become false.

Our desire is to develop a way to design clear distributed algorithms and write clear proofs about them. We believe we have provided a solid base for future work in this area.

*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.
2 Definitions

Definition 2.1 A distributed protocol,

\[ P = (n, Q, I, r) \]

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 \( r \subseteq Q^n \times Q^n \) on global states.

Our definition of protocol is certainly simple and precise. Let us first argue that it is sufficiently general. Anything we would be willing to call a distributed system can be broken up into a finite number of logical entities. Each such entity must be in some total configuration that we are calling its state. Furthermore the states of all the components combined should determine the entire state of the system and thus which global states can be next 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 components 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 \( p, q, r, s \), if \( (p, q) \) and \( (r, s) \) are in \( r \) 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 \( r \). 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 automaton, polynomial time Turing machine, etc.

For any protocol \( P \), let \( R_p \) be the \( r \)-reachable global states of \( P \), that is, the set of all global states we can reach by starting in \( I \) and taking any number of \( r \) steps. For \( p \in Q^n \) a global state and \( 1 \leq i \leq n \), we write \( (p)_i \) to denote the \( i^{th} \) component of \( p \).

For any two states \( p, q \in R_p \) and any participant \( i \), we will use the notation \( p \overset{\sim}{\rightarrow} q \) to mean that \( (p)_i = (q)_i \), i.e. they are indistinguishable from \( i \)'s point of view. Obviously each \( \overset{\sim}{\rightarrow} \) is an equivalence relation. For any reachable global state \( p \), define the \( i \)-neighborhood of \( p \) as follows:

\[ N_i(p) = \{ q \mid q \overset{\sim}{\rightarrow} p \} \]
If we are in state \( p \), then all \( i \) "knows" is that we are in \( N_i(p) \). Therefore, for any sentence \( a \), it is natural to make the following definition of \( K_i a \), which we read as "\( i \) knows \( a \)"

\[
(P,p) \models K_i a \iff \forall q \in N_i(p)((P,q) \models a)
\]

Intuitively \( i \) knows \( a \) just if \( a \) is true in all the worlds which are indistinguishable by \( i \) from the current world.

It is convenient to picture a protocol \( P \) as a graph with nodes consisting of all the elements of \( R_P \). There is a directed edge labelled \( r \) from \( p \) to \( q \) just if \((p,q) \in r \). Furthermore there is an undirected edge labelled 'i' between \( p \) and \( q \) just if \( p \sim q \).

Let \( G \subseteq \{1, \ldots, n\} \) be a group of participants in a protocol. For any \( p \in R_P \), define the \( G \)-neighborhood of \( p \) as follows:

\[
N_G(p) = \{ q \mid (\exists r \geq 0)(\exists i_1, \ldots, i_r \in G)(\exists p_1, \ldots, p_{r-1})[p \leadsto p_1 \leadsto p_2 \cdots p_{r-1} \leadsto q] \}
\]

This generalizes our previous definition since \( N_i(p) = N_{\{i\}}(p) \).

Analogously to our definition of \( K_i a \), we define \( C_G a \), which we read, "it is common knowledge among the members of \( G \) that \( a \)"

\[
(P,p) \models C_G a \iff \forall q \in N_G(p)((P,q) \models a)
\]

We write \( C \) for \( C_G \) and \( N \) for \( N_G \) in the special case that \( G \) includes all participants.

The next result shows that \( C \) coincides with the intuitive definition current in the literature. (See for example [HM84].)

**Theorem 2.2** The following two statements are equivalent:

1. \((P,p) \models C_G a.\)
2. \((\forall r \geq 0)(\forall i_1, \ldots, i_r \in G)((P,p) \models K_i_1 K_i_2 \cdots K_i_r a).\)

**Proof**

(1 \( \Rightarrow \) 2): For any \( \beta \), we have \( C_G \beta \rightarrow \beta \) since \( p \in N_G(p) \). Thus, it suffices to show that for any \( \beta \), if \((P,p) \models C_G \beta \), then for all \( i \in G \), \((P,p) \models C_G K_i \beta \). This is clear because if \( q \in N_G(p) \) and \( q' \in N_i(q) \) then \( q' \in N_G(p) \); hence \((P,q') \models K_i \beta \). Since \( K_i \beta \) holds for all \( q \in N_G(p) \), it is common knowledge in \( G \) at \( p \), as desired.

(2 \( \Rightarrow \) 1): Suppose that \((P,p) \not\models C_G a \). It follows that there is a \( q \in N_G(p) \) such that \((P,q) \models \neg a \). Let \( i_1, \ldots, i_r \in G \) be such that there exists \( p_1, \ldots, p_{r-1} \) with \( p \leadsto p_1 \leadsto p_2 \cdots p_{r-1} \leadsto q \). It follows that \((P,p) \models \neg K_i_1 K_i_2 \cdots K_i_r a.\)
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.

Let $A = \langle n^2, Q_A, I_A, r_A \rangle$ be an asynchronous message passing protocol defined as follows: The first $n$ participants of $A$ 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 $M \cup \{\lambda\}$, where $M$ is a set of possible messages and $\lambda$ is a special symbol denoting the null message. $b_{i,j} = m \in M$ indicates that the single message $m$ was sent by $i$ but not yet delivered to $j$. $b_{i,j} = \lambda$ indicates that no message is waiting.

The set of possible local states of a processor $a_i$ is $(D \times M^{n-1} \times N)$. State $(d, m_1, \ldots, m_{i-1}, m_{i+1}, \ldots, m_n, r)$ indicates that the processor is in internal state $d$ at round $r$ 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 $b_{i,j}$ for each $j$ such that $m_j = \lambda$.

Thus, the complete set of local states $Q_A$ is $(M \cup \{\lambda\}) \cup (D \times M^{n-1} \times N)$.

The transitions making up $r_A$ are of four kinds:

1. $(p, q) \in Send_{i,j}$ if
   - $p \preceq q$ for all $c \not\in \{a_i, b_{i,j}\}$;
   - $(p)_{b_{i,j}} = \lambda$;
   - $(q)_{b_{i,j}} = m_j \neq \lambda$;
   - $(p)_{a_i} = (d, \ldots, m_{i-1}, m_j, \ldots, 2k)$;
   - $(q)_{a_i} = (d, \ldots, m_{i-1}, \lambda, \ldots, 2k)$.

2. $(p, q) \in Receive_{i,j}$ if
   - $p \preceq q$ for all $c \not\in \{a_i, b_{i,j}\}$;
   - $(p)_{b_{i,j}} = m_j \neq \lambda$;
   - $(q)_{b_{i,j}} = \lambda$;
   - $(p)_{a_i} = (d, \ldots, m_{i-1}, \lambda, \ldots, 2k + 1)$;
   - $(q)_{a_i} = (d, \ldots, m_{i-1}, m_j, \ldots, 2k + 1)$.

3. $(p, q) \in Stop_i$ if
   - $p \preceq q$ for all $c \neq a_i$;
   - $(p)_{a_i} = (d, \lambda, \ldots, \lambda, 2k)$;
   - $(q)_{a_i} = (d, \lambda, \ldots, \lambda, 2k + 1)$.
4. \((p, q) \in \text{Start}\), if

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

Now we let \(r_A\) consist of all the above transitions:

\[ r_A = \bigcup_{i,j} \text{Send}_{i,j} \cup \text{Receive}_{i,j} \cup \text{Stop} \cup \text{Start}, \]

Finally let \(I_A\) be some nonempty set of global states in which the state of every processor \(\alpha_i\) 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.

Our second example of a protocol is a synchronous version of \(A\). Let \(B = (n^2, Q_B, r_B, I_B)\), where

\[ Q_B = \{\lambda\} \cup (D \times M^{n-1} \times \{2r \mid r \in N\}) \]

Thus in \(B\) all buffers are empty and the local states of the \(\alpha_i\)'s are the corresponding states from \(A\) at the beginning of a send phase. Let the transitions \(r_B\) consist of all pairs \((p, q)\) such that there exists a \(r_A\) path in \(A\) from \(p\) to \(q\) such that none of the intermediate steps go through global states of \(B\). Finally let \(I_B = I_A\).

It is not hard to see that \(B\) is a synchronous version of \(A\) such that in each round all processes send \(n - 1\) messages and then receive \(n - 1\) messages.

3 Common Knowledge in Asynchronous Systems

**Definition 3.1** We will call a protocol, \(P = (n, Q, I, r)\), totally asynchronous if for all \((p, q) \in r, p\) and \(q\) differ on at most two components.

The following theorem shows that in a totally asynchronous protocol, no new common knowledge can be achieved.

**Theorem 3.2** Let \(P\) be a totally asynchronous protocol and \(G\) a set of at least three participants. Let \(p\) be any global state of \(P\) and let \(p_0\) be an initial state from which \(p\) is reachable by a sequence of \(r\) steps. Let \(\alpha\) be any sentence in a logic for \(P\). If \((P, p) \models_G \alpha\) then \((P, p_0) \models_G \alpha\).

**Proof** We first show that if \(p\) is reachable from \(p_0\) by a sequence of \(r\) steps then \(N_G(p) = N_G(p_0)\). It suffices to consider the case where \((p_0, p) \in r\). By
the definition of a totally asynchronous protocol, p and p₀ must agree on all but at most two components. Thus, there exists a participant j ∈ G with the same local state in p₀ as in p, i.e. p₀ ∼ p. It follows that p ∈ NG(p₀) and thus NG(p) = NG(p₀). The theorem now follows from the definition of common knowledge in G.

As an example, consider the protocol A discussed at the end of the last section. It is easy to check that A satisfies the definition of totally asynchronous and thus no new common knowledge can arise in A. By way of contrast, if we look at A’s cousin B, then one observes that all reachable global states in B have all processors in the same round. Thus, if two reachable global states are ∼ equivalent for some i, 1 ≤ i ≤ n, then they are both in the same round. It follows that if we let G = {a₁, ..., aₙ} be the set of processors—i.e. we don’t care what the buffers know—then at any round r, ‘CG(we’re at round r)’ holds, i.e. it is common knowledge in G that all processors are at round r.

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 finite stack of K’s.

**Theorem 3.3** Let P = (n, Q, I, r) be a finite state protocol, i.e. |Q| < ∞. For each i, let

\[ Q_i = \{(q), q ∈ R_i\} \]

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

1. \( (P, p) \models Cα \).
2. For all \( i_1, i_2, ..., i_{2r-1}, ((P, p) \models K_{i_1} K_{i_2} ... K_{i_{2r-1}} α) \).

**Proof**

(1 ⇒ 2): By definition of C.

(2 ⇒ 1): Suppose that \( (P, p) \not\models Cα \). Then there must exist \( q ∈ N(p) \) such that \( (P, q) \models \neg α \). Consider a minimum length ~ chain from p to q:

\[ p = p₀ \sim p₁ \sim p₂ ... \sim p_{2r-1} \sim p_r = q \]

Note that no nonconsecutive pair \( p_i, p_k \) can agree on some component because if they did the chain could be shortened. It follows that in any given component each state appears at most twice. Therefore \( r ≤ 2r - 1 \). It follows that

\[ (P, p) \models \neg K_{i_1} K_{i_2} ... K_{i_{2r-1}} α \]

\(^3\text{This assumes of course that our logical language is powerful enough to express the property 'we're at round r'.}\)
Example 3.4 Consider the protocol \( P_r = (2, \{1, \ldots, r + 1\}, \{(i, i)\}, r) \) where, 
\( r = \{(i, i), (i + 1, i) \mid 1 \leq i \leq r\} \cup \{(i + 1, i), (i + 1, i + 1) \mid 1 \leq i < r\}. \)
This protocol has the unique computation chain:
\[(1, 1), (2, 1), (2, 2), (3, 2), \ldots, (r, r), (r, r), (r + 1, r).\]
Furthermore, for all global states \( p \) and \( q \) we have \( N(p) = N(q) \). Thus, for any \( \alpha \),
\[\langle P_r, p \rangle \models C \alpha \iff \text{for all } q \in R_{P_r}, \langle P_r, q \rangle \models \alpha.\]
Let \( \alpha \) say that processor 1 is not in state 1. Then \( \langle P_r, (1, 1) \rangle \not\models \alpha \), so \( \langle P_r, (r + 1, r) \rangle \not\models C \alpha \). On the other hand, it is easily seen that
\[\langle P_r, (r + 1, r) \rangle \models K_{i_1}K_{i_2}K_1 \ldots K_2 K_1 \alpha\]
It follows that for all \( i_1, i_2, \ldots, 2r - 1 \in \{1, 2\}, \)
\[\langle P_r, (r + 1, r) \rangle \models K_{i_1} \ldots K_{i_{2r-1}} \alpha,\]
showing that the bound in Theorem 3.3 cannot be improved.

4 Alternate Definitions of Knowledge

According to Halpern and Moses, "If \( C \) is to be attained, all processors must start supporting it simultaneously."\(^3\) Unfortunately the notion of two distant events occurring simultaneously has no meaning in modern physics. Do Halpern and Moses plus Einstein imply that no real distributed system ever achieves new common knowledge? A corollary would be that no real, synchronous distributed system can exist.

A look at our example protocols \( A \) and \( B \) reveals that they are realistic. Recall that new common knowledge among the \( n \) processors is attainable in \( B \) but not in \( A \). This is all the more confusing because in a very strong sense \( A \) and \( B \) are isomorphic protocols (cf. [CM85]).\(^4\)

The difference between protocols \( A \) and \( B \) concerns the granularity at which processors in the two protocols may introspect. In \( B \), processors are only allowed

\(^3\)[HM84], Lemma 2.

\(^4\)We will call a pair of protocols such as \( A \) and \( B \), 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.
to think about what they know at the start of each write phase. When two isomorphic structures differ on some property, we become very suspicious about whether or not that property is well defined. In the present case we must reexamine our definitions of knowledge and common knowledge.

Let \( P \) be any protocol and let \( S \subseteq R_P \) be any subset of reachable global states. For each \( i \), let \( \sim_S \) be the restriction of \( \sim \) to \( S \times S \). We can now generalize our previous definition of neighborhood. Let \( G \subseteq \{1, \ldots, n\} \) be a group of the participants in a protocol. For any \( p \in S \), define the \( G \)-neighborhood of \( p \) with respect to \( S \) as follows:

\[
N^S_G(p) = \{ q \mid (\exists r \geq 0)(\exists i_1, \ldots, i_r \in G)(\exists p_1, \ldots, p_{r-1} \in S) \\ [p \sim_S p_1 \sim_S p_2 \ldots \sim_S p_{r-1} \sim_S q]\}
\]

Intuitively, \( i \) knows only about the global states in \( S \). We thus define \( K^S_i \alpha \) as follows: for \( p \in S \),

\[
\langle p, p \rangle \models K^S_i \alpha \iff \forall q \in N^S_G(p)(\langle p, q \rangle \models \alpha).
\]

Similarly, we can define common knowledge in \( G \) with respect to \( S \):

\[
\langle p, p \rangle \models C^S_G \alpha \iff \forall q \in N^S_G(p)(\langle p, q \rangle \models \alpha).
\]

It is easy to see that the following generalization of Theorem 2.2 holds:

**Theorem 4.1** Let \( S \subseteq R_P \) and let \( G \subseteq \{1, \ldots, n\} \). For \( p \in S \) the following two statements are equivalent:

1. \( \langle p, p \rangle \models K^S_G \alpha \).
2. \( (\forall r \geq 0)(\forall i_1, \ldots, i_r \in G)(\langle p, p \rangle \models K^S_i K^S_{i_1} \ldots K^S_G \alpha)\).

The following theorem shows that for any protocol \( P \) and any nonempty \( S \subseteq R_P \), the operators \( K^S \) and \( C^S \) satisfy the standard S5 axioms for knowledge operators. It follows that if we consider the protocol \( A \) with \( S = R_\emptyset \), then we get a quite reasonable definition of knowledge and common knowledge for which the asynchronous protocol \( A \) does attain new common knowledge. This contradicts Theorem 3 of [HM84]. More importantly, these observations show the definitions of knowledge and common knowledge needed to make useful progress in the understanding of distributed protocols are much more subtle than one might have at first thought.

**Theorem 4.2** For any protocol \( P \), any nonempty \( S \subseteq R_P \), and \( G \subseteq \{1, \ldots, n\} \), the operators \( K^S_i \) and \( C^S_G \) satisfy the standard S5 axioms for modal operators.

**Proof** This is immediate from the fact that each \( \sim_S \) is an equivalence relation.
5 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 some 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.

6 References

References


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)

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)

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)

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

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

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

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

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