Using Knowledge to
Optimally Achieve Coordination
in Distributed Systems*  

Gil Neiger
Rida A. Bazzi†

GIT–CC–93/26
April 9, 1993

Abstract

The problem of coordinating the actions of individual processors is fundamental in distributed computing. Researchers have long endeavored to find efficient solutions to a variety of problems involving coordination. Recently, processor knowledge has been used to characterize such solutions and to derive more efficient ones. Most of this work has concentrated on the relationship between common knowledge and simultaneous coordination. This paper takes an alternative approach, considering problems in which coordinated actions need not be performed simultaneously. This approach permits better understanding of the relationship between knowledge and the different requirements of coordination problems. This paper defines the ideas of optimal and optimum solutions to a coordination problem and precisely characterizes the problems for which optimum solutions exist. This characterization is based on combinations of eventual common knowledge and continual common knowledge. The paper then considers more general problems, for which optimal, but no optimum, solutions exist. It defines a new form of knowledge, called extended common knowledge, which combines eventual and continual knowledge, and shows how extended common knowledge can be used to both characterize and construct optimal protocols for coordination.

College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0280

*This work was supported in part by the National Science Foundation under grants CCR-8909663 and CCR-9106327. An earlier version of this paper appeared in Yoram Moses, editor, Proceedings of the Fourth Conference on Theoretical Aspects of Reasoning about Knowledge, pages 43–59. Morgan-Kaufmann, March 1993.
†This author was supported in part by a scholarship from the Hariri Foundation.
1 Introduction

Coordinating the activity of a set of independent processors is fundamental in distributed computing. One approach to this is to require the processors to agree on a common action to perform. In addition, they must ensure that the action chosen is legitimate given the context within which they are operating (e.g., with respect to their initial values). The purpose of this work is to explore the relationship between knowledge and coordination and to use this to derive efficient solutions to coordination problems.

This work specifically considers fault-tolerant coordination in a distributed computing system. It is assumed that some (but not all) of these processors may be faulty. A coordination protocol is an algorithm by which the nonfaulty processors successfully coordinate their actions despite the failures of others. There is a large body of literature that has studied fault-tolerant solutions to coordination problems, such as Reliable Broadcast and Distributed Consensus (Fischer [6] provides a survey of many such problems).

More recently, researchers have studied the relationship between simultaneous coordination and common knowledge [11]. Dwork and Moses [4] showed that achieving common knowledge was necessary for Simultaneous Byzantine Agreement. Moses and Tuttle [13] extended this result to a broad class of simultaneous coordination problems. Neiger and Tuttle [15] considered the more difficult class of consistent simultaneous coordination problems and showed that, in general, their solutions require a stronger form of common knowledge. This suggested that variations in the type of coordination desired may result in corresponding variations in the type of knowledge required. The three papers above used the necessity of common knowledge to construct optimum protocols to achieve coordination.¹ By having processors perform actions as soon as the required knowledge is attained, these protocols are guaranteed to match or outperform any other solution.

The requirement of simultaneous coordination is very strong, and this is why common knowledge is needed to achieve it. But common knowledge is difficult to attain. Halpern and Moses [11] showed that it cannot be attained in many practical distributed systems and that, therefore, simultaneous coordination is impossible in these systems. In addition, the requirement of simultaneity is so strong as to obscure the relationship between knowledge and other requirements of coordination problems. These facts motivate a study of the relationship between knowledge and problems requiring nonsimultaneous coordination. Halpern, Moses, and Waarts [12] considered one such problem, Eventual Byzantine Agreement, and developed a new form of knowledge, continual common knowledge, that could be used to develop optimal solutions.

Because the distinction between optimum and optimal solutions is central to many of the results of this paper, we describe it here briefly. Given a coordination problem, one solution dominates another if it always causes processors to choose an action at least as early as the other solution. This gives a partial order on solutions: two solutions may be incomparable if each outperforms the other in some environment. A solution is optimum if no solution strictly dominates it. In a given setting, a particular problem may or may not have an optimum solution, but it will always have one or more optimal solutions. It turns out that, for problems involving simultaneous coordination, there are often optimum solutions [4,13,15]. When simultaneity is not required, it is often the case that there are not.

¹These papers referred to the protocols they developed as optimal. As will be seen below, there is an important distinction between optimum and optimal protocols.
Previous work on knowledge and nonsimultaneous coordination considered protocols that guarantee that processor choices are correct (for example, with respect to processors' initial states) and in agreement with each other. Their knowledge-based analyses did not explicitly consider a third requirement of most coordination problems, termination. Such a requirement specifies the executions in which a nonfaulty processor must terminate the protocol by performing some action. Termination is thus a liveness property (as opposed to correctness and agreement, which are safety properties). Ideally, a problem's termination condition would require all nonfaulty processors to perform an action in every execution. Unfortunately, this requirement cannot be achieved in many practical systems [8]. In this paper, we consider the weaker termination condition developed by Gopal and Toueg [9]. This condition requires that, in any execution in which some processor performs an action, all nonfaulty processors must do so also. To reason about this kind of termination, one needs to consider eventual common knowledge [11,17].

This paper considers four types of coordination problems: consistent and nonconsistent problems that require termination and those that do not. For each, we establish the minimum knowledge necessary to perform an action. In some cases, this is simple knowledge or belief but, in the cases requiring termination, eventual common knowledge is required. We then consider the problem of deriving optimum solutions to these problems. Although some coordination problems have no optimum solution (e.g., Eventual Byzantine Agreement), such solutions do exist for others and this paper precisely characterizes those problems. This characterization uses both continual and eventual common knowledge; the type of knowledge used depends directly on the type of problem being considered.

We then consider optimal solutions, which do exist for all coordination problems. We show how these solutions can be characterized and constructed using different forms of knowledge. For problems requiring termination, this requires a new variant of common knowledge that combines the continual knowledge needed for agreement and the eventual knowledge needed for termination. We call this extended common knowledge. The development and use of extended common knowledge is one of the main contributions of this paper.

2 Definitions

This section defines a model of a distributed system. This model is similar to others used to study knowledge and coordination [4,11,12,13,15].

A distributed system consists of a finite set $P$ of $n$ processors and a communication network that connects them. All processors share a clock that starts at time 0 and advances in increments of one. Computation proceeds in a sequence of rounds, with round $l$ taking places between time $l-1$ and time $l$. At time 0, each processor starts in some initial state. Then, in every round, the processor performs some local computation (and, optionally, a coordination action), sends messages to other processors, and receives messages delivered to it in that round by the communication network.

The round-based model described above is usually used only for modeling synchronous systems, while this paper considers both synchronous and asynchronous systems. By assuming the existence of a global clock, we appear to be requiring processors (but not message-passing) to be synchronous. However, the global clock assumption is made only to simplify our presentation and does not limit our results. It would be easy to make processors asynchronous in

---

2 A processor's initial state is meant to model any input that a processor may receive from outside the system. It is also possible to model such inputs that may be received after time 0.
our model by adding to the operating environment (see below) an activation pattern, which specifies which processors are active in which rounds.

At any given time, a processor’s message history consists of the list of messages it has sent to and received from the other processors, each tagged with its sending or receiving time. A processor’s local state at any given time consists of its initial state, its message history, the time on the global clock, and the processor’s identity. A global state is a tuple $\langle s_1, \ldots, s_n \rangle$ of local states. A run of the system is an infinite sequence of global states, together with an operating environment (see below). An ordered pair $\langle r, l \rangle$, where $r$ is a run and $l$ is a natural number, is called a point and represents the state of the system after the first $l$ rounds of $r$. The global state at point $\langle r, l \rangle$ is denoted by $r(l)$ and the local state of processor $p$ at that point is denoted by $r_p(l)$. Processors follow a communication protocol $P$, which is a function of a processor’s local state that specifies the messages a processor is to send in that round. One communication protocol may produce different runs depending on how the system behaves during execution. Important factors are the processors’ initial states, how messages are delivered after they are sent, and how processors fail. Together these make up the operating environment of a run, described below.

Central to the results of this paper is the fact that one can compare the performance of different protocols with respect to the same behavior on the part of the “system.” The operating environment of an execution captures this behavior. It includes all information (besides the protocol) necessary to reconstruct the execution. There are two components to an operating environment: the initial states of the processors and message transmission information.\footnote{To completely model asynchronous processors, a third component would be necessary to indicate which processors are active in which rounds. One could call this the activation pattern. For the sake of simplicity, we do not formally model this.} Formally, an operating environment $o$ is a pair $(\mathbf{i}, \mathcal{M})$, where $\mathbf{i}$ is a vector of initial states ($\mathbf{i}[p]$ is the initial state of $p$) and $\mathcal{M}$ is a message transmission pattern, described below.

The message transmission pattern of a run specifies the following information for for every message that might be sent in a run: whether or not it is sent correctly; if and when it is delivered; and whether or not is correctly received. Formally, a message transmission pattern is a function $\mathcal{M} : \mathbb{N} \times \mathcal{P} \times \mathcal{P} \rightarrow \{y, n\} \times \mathbb{N} \times \{y, n\}$ (where $\mathbb{N}$ is the set of positive integers). Suppose that $\mathcal{M}(l, p, q) = \langle b_s, l_d, b_r \rangle$. If $b_s = n$, then processor $p$ does not send a message to $q$ in round $l$; if $b_s = y$, then the message is sent correctly if the protocol indeed specified that such a message be sent. If $b_s = y$ and $l_d = \infty$, then the message is lost by the communication system. If $b_s = y$ and $l_d \in \mathbb{N}$, then the message is delivered in round $l_d$. If $b_s = y$, $l_d \in \mathbb{N}$, and $b_r = n$, then $q$ omits to receive the message when it is delivered. If $b_s = y$, $l_d \in \mathbb{N}$, and $b_r = y$, then the message is correctly sent, delivered, and received.

The class of allowable message transmission patterns is determined by assumptions made about the communication network and about processor failures. For example, message-passing may be completely synchronous (messages are delivered in the round in which they are sent), completely asynchronous (there is no bound on the number of rounds required for delivery), or something in between. It may be reliable (all messages are delivered) or lossy (messages sent might never be delivered). Any of these restrictions can be specified by restricting the class of message transmission patterns that can occur.

Different failure models place different restrictions on the faulty behaviors that processors may exhibit (a formal specification of the failure patterns allowed by different failure models is beyond the scope of this paper). We can model any type of benign failures, specifically crash
(stopping) failures, send-omission failures, and general omission failures [14]. We cannot, however, model arbitrary processor failures [16]. Given a run \( r \) with operating environment \( \mathcal{O} = (t, \mathcal{M}) \), one can characterize the set of processors that are not faulty in \( r \):

\[
\mathcal{N}(r) = \{ p \in \mathcal{P} \mid \forall l \in \mathcal{N} \forall q \in \mathcal{P} \left[ m(l, p, q) = (y, l_d, b_r) \land \left( m(l, q, p) = (b_s, l_d, n) \Rightarrow (b_s = n \lor l_d = \infty) \right) \right] \}.
\]

The operating environment of a run must be consistent with the run in that it must
the initial states and message deliveries of the run. For example, consider an operating
environment \( \mathcal{O} \) that indicates that \( p \) can successfully send a message to \( q \) in round \( l \) (i.e., that
\( m(l, p, q) = (y, l_d, b_r) \)). If, in run \( r \), \( p \)'s protocol calls for it to send such a message but \( p \) omits
to do so, then \( \mathcal{O} \) is not consistent with \( r \). On the other hand, if the protocol did not call for any
such message to be sent, then \( \mathcal{O} \) (as described) is consistent. We assume that the operating
environment of every run is consistent. Two runs of two different communication protocols are corresponding runs if they have the same operating environment. Different protocols are
compared by comparing their behavior in corresponding runs.

This work identifies a system with the set of all runs of a communication protocol under
a given failure model and with specified assumptions about message-passing. Such a set of
runs is denoted by \( \mathcal{R}_P \), where \( P \) is the communication protocol being used; \( \mathcal{R} \) will be used
if \( P \) is obvious from context. If \( r \in \mathcal{R} \) and \( l \) is a natural number, then \( \langle r, l \rangle \) is a point in
\( \mathcal{R} \). In order to analyze systems, it is convenient to have a logical language in which one can
make statements about the system. A fact in this language is interpreted to be a property of
points: a fact \( \varphi \) will be either true or false at a given point \( \langle r, l \rangle \) in \( \mathcal{R} \), denoted \( \mathcal{R}, r, l \models \varphi \)
and \( \mathcal{R}, r, l \not\models \varphi \), respectively. Fact \( \varphi \) is valid in system \( \mathcal{R} \), denoted \( \mathcal{R} \models \varphi \), if it is true at
all points in \( \mathcal{R} \). A fact is valid if it is valid in all systems. Although facts are interpreted as
properties of points, it is often convenient to refer to facts that are about objects other than
points (e.g., properties of runs). In general, a fact \( \varphi \) is a fact about \( X \) if fixing \( X \) determines
the truth (or falsity) of \( \varphi \). A fact \( \varphi \) is stable in \( \mathcal{R} \) if, once it becomes true it remains so; for
all points \( \langle r, l \rangle \) in \( \mathcal{R} \), if \( \mathcal{R}, r, l \models \varphi \), then \( \mathcal{R}, r, l' \models \varphi \) for all \( l' \geq l \).

3 Coordination Problems

This section defines four classes of coordination problems using the model given in the previous
sections. Informally, a coordination problem requires processors to coordinate by choosing
a common action from a set of possible actions. In any given context, some subset of the
possible actions are enabled, and processors should only choose an enabled action. Formally,
a coordination problem is a finite set of actions \( \mathcal{C} = \{ a_1, \ldots, a_m \} \) together with a set of
associated enabling conditions \( \{ ok_1, \ldots, ok_m \} \) Each enabling condition is a fact about the
initial input and the identities of the faulty processors (thus, it is a fact about runs). The
processors must coordinate to choose a common action that is enabled. Processors need not
perform their actions simultaneously. One example is Eventual Byzantine Agreement, which
was considered by Halpern, Moses, and Waarts [12]. In this problem, all processors begin
with an initial value that is either 0 or 1. Processors choose from among two actions, \( a_0 \) and
\( a_1 \). The enabling conditions are given as follows: \( ok_0 \) holds if some processor began with 0
and \( ok_1 \) holds if some processor began with 1.

For a protocol to coordinate a choice of actions, there must be a mechanism by which it
can specify when an action is to be performed. An action protocol \( P(\Phi) \) is a communication
protocol \( P \) augmented by an action function \( \Phi \). For each \( a_i \in \mathcal{C} \) and \( p \in \mathcal{P} \), \( \Phi_{ip} \) is a fact

\footnote{It is not clear that such failures can be modelled using the methods presented here.}
about \( p \)'s local state (see Section 2 above). \( P(\Phi) \) has \( p \) perform \( a_i \) the first time \( \Phi_{i,p} \) becomes true. An action protocol \( P(\Phi) \) is a decision protocol if the following two properties hold of the action function \( \Phi \):

- For all \( a_i \in \mathcal{C} \) and \( p \in \mathcal{P} \), \( \Phi_{i,p} \) is stable; that is, a processor's choice is irrevocable.
- For all \( a_i \in \mathcal{C} \) and \( p \in \mathcal{P} \), if \( (\mathcal{R}_P, r, l) \models \Phi_{i,p} \), then for no \( j \neq i \) and \( l' \in \mathbb{N} \) does \( (\mathcal{R}_P, r, l') \models \Phi_{j,p} \); that is, a processor's choice is unique.

Note that the action function \( \Phi \) is completely orthogonal to the communication protocol \( P \). It controls only the actions of a processor, which are, technically speaking, not part of its local state. A processor does not stop running its communication protocol once it has made a choice. For this reason, the system \( \mathcal{R}_P \) is well-defined as the set of runs of the communication protocol \( P \) and is independent of any action function. The truth (or falsity) of different facts at different points of \( \mathcal{R}_P \) is always independent of the action function being used.

There are various ways to define the correctness of an action protocol with respect to a problem \( \mathcal{C} \). Informally, processors must agree and must choose an action that is enabled. In some cases, the actions taken by the faulty processors are not relevant; in others, their actions are subject to the same correctness criteria as those of the nonfaulty processors. We call these cases normal and consistent, respectively, and are discussed in the next two sections. The earlier literature on knowledge and coordination concentrated on general coordination [12,13]; more recently, researchers have begun to consider consistent coordination [9,15].

### 3.1 Normal Coordination

Most coordination problems defined in the literature require only that the nonfaulty processor coordinate their actions. For example, Eventual Byzantine Agreement places no restrictions on the values that may be chosen by the faulty processors. Such problems are easier to solve and are the only appropriate ones for systems in which the behavior of faulty processors is relatively unconstrained.

Formally, \( P(\Phi) \) normally satisfies \( \mathcal{C} \) (or \( N\)-satisfies \( \mathcal{C} \)) if the following hold:

- Validity. If an action is performed by a nonfaulty processor, then that action is enabled:
  \( \mathcal{R}_P \models \Phi_{i,p} \land (p \in \mathcal{N}) \Rightarrow ok_i \).
- Agreement. If two nonfaulty processors perform actions, they perform the same action:
  if \( (\mathcal{R}_P, r, l_p) \models \Phi_{i,p} \land (p \in \mathcal{N}) \) and \( (\mathcal{R}_P, r, l_q) \models \Phi_{i,q} \land (q \in \mathcal{N}) \), then \( i = j \).

Note that one can trivially solve any coordination problem as specified above by simply choosing \( \Phi_{i,p} = false \) for all \( a_i \in \mathcal{C} \) and \( p \in \mathcal{P} \). This is because Validity and Agreement are safety properties. Nevertheless, it still makes sense to consider optimum and optimal solutions to such problems (see below). In most cases, such solutions would not be trivial. Coordination problems can also be made nontrivial by adding a liveness property such as a termination condition.

Informally, a termination condition specifies when nonfaulty processors must perform an action (because of failures, one cannot require faulty processors to perform an action). Some problems simply require that all nonfaulty processors perform an action in every run. However, these problems cannot, in general, be solved in systems with asynchronous communication [8]. For that reason, Gopal and Toneg [9] introduced a weaker termination condition, which requires nonfaulty processors to act only if some other processor does. Formally, \( P(\Phi) \)
normally satisfies \( C \) with termination (or \( NT \)-satisfies \( C \)) if it \( G \)-satisfies \( C \) and the following condition holds:

- **Termination.** If a nonfaulty processor performs an action, then all nonfaulty processors perform that action: if \( (R_P, r, l) \models \Phi_{i,p} \wedge (p \in \mathcal{N}) \), then, for all \( q \in \mathcal{N}(r) \), there is some \( l' \) such that \( (R_P, r, l') \models \Phi_{i,q} \).

If \( P(\Phi) \) \( X \)-satisfies \( C \) (where \( X \) is either \( N \) or \( NT \)), then we say that \( P(\Phi) \) is an \( X \)-solution to \( C \).

Halpern, Moses, and Waarts compared normal solutions to *Eventual Byzantine Agreement* by comparing the solutions' behavior in corresponding runs. Their method is adapted here. Suppose that decision protocols \( P_1(\Phi_1) \) and \( P_2(\Phi_2) \) both \( N \)-satisfy some problem \( C \). \( P_1(\Phi_1) \) \( \mathcal{N} \)-dominates \( P_2(\Phi_2) \) if, in every pair of corresponding runs of the two protocols, \( P_2(\Phi_2) \) has no nonfaulty processor perform an action earlier than \( P_1(\Phi_1) \) (it may be that some processors perform actions in neither run). Formally, if \( P_1(\Phi_1) \) dominates \( P_2(\Phi_2) \) and \( r_1 \) and \( r_2 \) are corresponding runs of the two protocols, then \( (R_{P_2}, r_2, l) \models \Phi_{i,p} \) implies \( (R_{P_1}, r_1, l) \models \vee_{a_i \in \mathcal{C}} \Phi_{a_i} \) for all \( a_i \in \mathcal{C} \) and \( p \in \mathcal{N}(r_1) \) (since \( r_1 \) and \( r_2 \) are corresponding, \( \mathcal{N}(r_1) = \mathcal{N}(r_2) \)).

Notice that the \( \mathcal{N} \)-dominates relation is a partial orders on the space of \( N \)-solutions to a given problem. It may be that neither of \( P_1(\Phi_1) \) and \( P_2(\Phi_2) \) dominates the other; \( P_1(\Phi_1) \) may outperform \( P_2(\Phi_2) \) in one operating environment, while \( P_2(\Phi_2) \) outperforms \( P_1(\Phi_1) \) in another.

A protocol is \( N \)-optimum for \( \mathcal{C} \) (respectively, \( NT \)-optimum) if it \( N \)-satisfies \( \mathcal{C} \) (respectively, \( NT \)-satisfies) and \( \mathcal{N} \)-dominates every other protocol that does so. Because the \( \mathcal{N} \)-dominates order is partial, some problems may not have optimum solutions. For example, Moses and Tuttle [13] gave two incomparable \( N \)-solutions to *Eventual Byzantine Agreement*: one can decide on 0 very quickly (but is slow in deciding 1) and the other can decide 1 very quickly. They showed that there is no \( N \)-solution that can decide both values very quickly. Thus, there is no \( N \)-solution that \( \mathcal{N} \)-dominates both of the solutions described above and, hence, there is no \( N \)-optimum protocol. In contrast, protocol \( P(\Phi) \) is \( N \)-optimal for \( \mathcal{C} \) (respectively, \( NT \)-optimal) if it \( N \)-satisfies \( \mathcal{C} \) (respectively, \( NT \)-satisfies) and if every \( N \)-solution to \( \mathcal{C} \) (respectively, \( NT \)-solution) that \( \mathcal{N} \)-dominates \( P(\Phi) \) is in turn \( N \)-dominated by \( P(\Phi) \).

Although there is no optimum solution to *Eventual Byzantine Agreement*, there are coordination problems for which optimum solutions do exist. Section 6 precisely characterizes these problems. Section 9 shows how to construct optimal solutions to any of the problems defined here.

### 3.2 Consistent Coordination

Although normal coordination is appropriate for systems with unconstrained processor failures, much literature has studied systems with relatively benign failures. In these systems, it is appropriate to study coordination problems in which the actions of faulty processors (if any) must be consistent with those of the correct processors. This section defines a class of consistent coordination problems that parallels the normal coordination problem of the previous section.

\( P(\Phi) \) consistently satisfies \( C \) (or \( C \)-satisfies \( C \)) if the following hold:

- **Validity.** If an action is performed by any processor, then that action is enabled: \( R_P \models \Phi_{i,p} \Rightarrow ok_i \).
• Agreement. If two processors perform actions, they perform the same action: if $(R_P, r, l_p) \models \Phi_{i,p}$ and $(R_P, r, l_q) \models \Phi_{i,q}$, then $i = j$.

$P(\Phi)$ consistently satisfies $C$ with termination (or CT-satisfies $C$) if it $C$-satisfies $C$ and the following condition holds:

• Termination. If any processor performs an action, then all nonfaulty processors perform that action: if $(R_P, r, l) \models \Phi_{i,p}$, then, for all $q \in N(r)$, there is some $l'$ such that $(R_P, r, l') \models \Phi_{i,q}$.

If $P(\Phi)$ X-satisfies $C$ (where Y is either C or CT), then we say that $P(\Phi)$ is an X-solution to $C$.

If $P_1(\Phi_1)$ and $P_2(\Phi_2)$ are C-solutions to $C$, then $P_1(\Phi_1)$ C-dominates $P_2(\Phi_2)$ if, for all pairs $r_1, r_2$ of corresponding runs of the two protocols, $(R_{P_1}, r_2, l) \models \Phi_{i,p}$ implies $(R_{P_1}, r_1, l) \models \bigvee_{a_i \in C} \Phi_{i,a}$ for all $a_1 \in C$ and $p \in P$ (notice that this must be true for all $p$, including those that are faulty). A protocol is C-optimum for $C$ (respectively, CT-optimum) if it $C$-satisfies $C$ (respectively, CT-satisfies $C$) and $C$-dominates every other protocol that does so. $P(\Phi)$ is C-optimal for $C$ (respectively, CT-optimal) if it $C$-satisfies $C$ (respectively, CT-satisfies $C$) and if every $C$-solution to $C$ (respectively, CT-solution) that $C$-dominates $P(\Phi)$ is in turn $C$-dominated by $P(\Phi)$.

4 Definitions of Knowledge

The analysis in this paper depends on a processor’s knowledge at different points in an execution. This section formally defines processor knowledge. The treatment here is an adaptation of others [4,11,12,13,15].

4.1 Basic Definitions

This section gives a way to express processor knowledge by augmenting the logical language introduced in Section 2. Recall that a fact in this language is a property of points: a fact $\varphi$ is either true or false at a given point $(r, l)$ in system $R$, denoted $(R, r, l) \models \varphi$ or $(R, r, l) \not\models \varphi$, respectively. We assume that the language is powerful enough to represent all relevant ground facts—facts about the system that do not explicitly mention processors’ knowledge—and is closed under the standard boolean connectives.

Processor knowledge was first defined by Halpern and Moses [11] in the following way. Processor $p$ knows $\varphi$ at point $(r, l)$ in system $R$, denoted $(R, r, l) \models K_{p,\varphi}$, if $(R, r', l') \models \varphi$ for all runs $(r', l')$ in $R$ such that $r'_p(l') = r_p(l)$ (note that, because the global clock is part of a processor’s local state, $l'$ must equal $l$ for this equality to hold). Thus, a processor always knows any true fact about its local state (recall that an action protocol’s predicates $\Phi_{i,p}$ are all facts about $p$’s local state).

Because this paper deals with coordination among a group of processors, different forms of group knowledge are important. The particular setting (e.g., type of coordination problem) determines the group whose knowledge is of interest. We often consider sets of processors whose membership may vary from one run to another or over the course of a run. These are called indexical sets; their membership is determined by the point being considered. For example, if $S$ is an indexical set, then $S(r, l)$ refers to the contents of the set at point $(r, l)$.
Examples of indexical sets include the set of nonfaulty processors \( \mathcal{N} \) and sets of processors that know certain facts.\(^5\)

It is often useful to condition a processor’s knowledge on the processor’s membership in a specific set. We say that \emph{processor \( p \) believes \( \varphi \) conditional on \( S \)} if \( p \) knows that, if it is in \( S \), \( \varphi \) is true. That is,

\[
B_p^S \varphi \equiv K_p(p \in S \Rightarrow \varphi).
\]

It is easy to see that \((R, r, l) \models B_p^S \varphi\) if \((R, r', l) \models \varphi\) for all runs \( r' \) such that \( r'_p(l) = r_p(l) \) and \( p \in S(r', l) \). Processor knowledge, using the \( K_p \) operators will be used to define strong notions of group knowledge, while processor belief, using \( B_p^S \), will be used to define weaker notions.

Informally, a fact \( \varphi \) is \emph{common knowledge} to \( S \) if everyone in \( S \) knows \( \varphi \), everyone knows that everyone knows \( \varphi \), and so on.\(^5\) Common knowledge is necessary for the solution to simultaneous coordination problems \([4,13,15]\). The following is a brief overview of a more formal definition, based on logical fixed points. \emph{Everyone in indexical set \( S \) knows \( \varphi \), denoted} \( \mathcal{E}_S \varphi \), \emph{is defined to be} \( \bigwedge_{p \in S} K_p \varphi \). \emph{All processors in \( S \) believe \( \varphi \), denoted} \( \mathcal{A}_S \varphi \), \emph{is equivalent to} \( \bigwedge_{p \in S} B_p^S \varphi \). Based on this, two forms of common knowledge are defined, a strong one based on knowledge and a weak one based on belief. \emph{Strong common knowledge of fact \( \varphi \) by set \( S \), denoted} \( \mathcal{S}_S \varphi \), \emph{is the greatest fixed point of the equation}

\[
X \equiv \mathcal{E}_S(\varphi \land X).
\]

\emph{Weak common knowledge of fact \( \varphi \) by set \( S \), denoted} \( \mathcal{W}_S \varphi \), \emph{is the greatest fixed point of the equation}

\[
X \equiv \mathcal{A}_S(\varphi \land X).
\]

\( \mathcal{S}_S \varphi \) is equivalent to the infinite conjunction \( \bigwedge_{i \geq 1} \mathcal{E}_i^i \varphi \), while \( \mathcal{W}_S \varphi \) is equivalent to \( \bigwedge_{i \geq 1} \mathcal{A}_i^i \varphi \). Neiger and Tuttle \([15]\) showed that strong common knowledge was necessary for the solution of consistent simultaneous coordination, while Moses and Tuttle \([13]\) had earlier observed that achieving weak common knowledge was sufficient to achieve normal simultaneous coordination. The remainder of this section introduces two modifications of common knowledge that are appropriate to the study of nonsimultaneous coordination. Each has a strong and a weak version, which are appropriate to the analysis of consistent and normal coordination problems, respectively.

### 4.2 Eventual Common Knowledge

\emph{Eventual common knowledge} \([11,17]\) relaxes the simultaneity that is inherent in simple common knowledge. For this reason, it is appropriate in the study of problems that do not require simultaneous coordination. Informally, a fact is eventual common knowledge to a set of processors if they all eventually know it, all eventually know that all others eventually know it, and so forth. As will be seen below, eventual common knowledge is necessary for achieving termination in a solving coordination problems.

---

\(^5\) Recall that \( \mathcal{N}(r) \) was defined to be the set of processors nonfaulty in run \( r \). When convenient, \( \mathcal{N}(r, l) \) will be used to refer to the processors nonfaulty at point \( (r, l) \), but it should be understood then that \( \mathcal{N}(r, l) = \mathcal{N}(r, l') \) for all \( l \) and \( l' \).

\(^6\) When necessary to distinguish it from other forms of knowledge, we may refer to this as simple common knowledge.
The definition of eventual knowledge uses the temporal operator \( \text{eventually } \Box \). \((\mathcal{R}, r, l) \models \Box \varphi \) if and only if \((\mathcal{R}, r, l') \models \varphi \) for some \( l' \geq l \).\footnote{Note that these are linear-time semantics for \( \Box \).} Eventual common knowledge is defined in a manner analogous to that of simple common knowledge. \textit{Strong eventual common knowledge of fact} \( \varphi \) \textit{by set} \( \mathcal{S} \), denoted \( S^\Diamond \mathcal{S} \varphi \), is the greatest fixed point of the equation
\[
X \equiv \Box \mathcal{E}_\mathcal{S}(\varphi \land X). \footnote{This is slightly different from the original definition of Halpern and Moses \cite{halpern1990knowledge}. They defined eventual common knowledge to be the greatest fixed point of the equation \( X \equiv \bigwedge_{r \in \mathcal{S}} \Box_{r} (\varphi \land X) \). For all cases considered in this paper, their definition is equivalent to the one given here for strong eventual common knowledge.}
\]
\textit{Weak eventual common knowledge of fact} \( \varphi \) \textit{by set} \( \mathcal{S} \), denoted \( W^\Diamond \mathcal{S} \varphi \), is the greatest fixed point of the equation
\[
X \equiv \Box \mathcal{A}_\mathcal{S}(\varphi \land X).
\]

It is easy to see that \( S^\Diamond \mathcal{S} \varphi \) implies the infinite conjunction \( \bigwedge_{i \geq 1} (\Box \mathcal{E}_\mathcal{S})^i \varphi \). Similarly, \( W^\Diamond \mathcal{S} \varphi \) implies \( \bigwedge_{i \geq 1} (\Box \mathcal{A}_\mathcal{S})^i \varphi \). One should note that eventual common knowledge is \textit{weaker} that simple common knowledge. It does not require that processors gain their knowledge simultaneously or that all levels of knowledge will ever hold simultaneously. Eventual common knowledge does not, in general, imply “eventually” common knowledge.

Both forms of eventual common knowledge satisfy \textit{positive introspection}; if a fact is eventual common knowledge to a set, then all members of the set eventually know (or believe) this. Thus, the following are valid:

- \( S^\Diamond \mathcal{S} \varphi \Rightarrow \Box \mathcal{E}_\mathcal{S} S^\Diamond \mathcal{S} \varphi \); and
- \( W^\Diamond \mathcal{S} \varphi \Rightarrow \Box \mathcal{A}_\mathcal{S} W^\Diamond \mathcal{S} \varphi \).

(These follow directly from the fixed-point definitions.) Each form of eventual common knowledge satisfies an induction rule that can be used to show that certain facts are eventual common knowledge:

- If \( \varphi \Rightarrow \Box \mathcal{E}_\mathcal{S}(\varphi \land \psi) \) is valid in a system, then \( \varphi \Rightarrow S^\Diamond \mathcal{S} \psi \) is also valid in that system.
- If \( \varphi \Rightarrow \Box \mathcal{A}_\mathcal{S}(\varphi \land \psi) \) is valid in a system, then \( \varphi \Rightarrow W^\Diamond \mathcal{S} \psi \) is also valid in that system.

(Again, these follow from the fixed-point definitions.) This paper considers cases in which facts \( \varphi \) about runs (specifically, the enabling conditions of a coordination problem) become eventual common knowledge to the set \( \mathcal{N} \) of nonfaulty processors. Because we consider only systems in which this set is never empty, it is not hard to see that, in these systems, \( S^\Diamond \mathcal{N} \varphi \Rightarrow \varphi \) and \( W^\Diamond \mathcal{N} \varphi \Rightarrow \varphi \) are valid. These implications will simplify the presentation of some protocols below.

In a separate paper \cite{halpern1990knowledge}, we further explore eventual common knowledge, characterizing the systems in which it can be achieved and the complexity required to ascertain that it is true.

### 4.3 Continual Common Knowledge

Although eventual common knowledge is necessary for Termination, Halpern, Moses, and Waarts \cite{halpern1990knowledge} showed that it cannot be used to enforce Agreement. Intuitively, the reason for this is that, unlike simple common knowledge, different processors may learn of eventual common knowledge at different times. This lack of synchronization may lead to disagreement. They showed that, to ensure that no disagreement occurred at any time during a run, it is
necessary to use a kind of knowledge that was continual over all points of a run. They called this *continual common knowledge*.

This form of knowledge makes use of the temporal operator *always* □. This is a bidirectional variant of the temporal operator *henceforth* □' (\(\mathcal{R}, r, l) \models □' \varphi\) if and only if \((\mathcal{R}, r, l') \models \varphi\) for all \(l'\). Continual common knowledge can now be defined in a by-now familiar manner. **Strong continual common knowledge of fact \(\varphi\) by set \(\mathcal{S}\)**, denoted \(\mathsf{S}_\mathcal{S}^\mathcal{S} \varphi\), is the greatest fixed point of

\[
X \equiv □E_\mathcal{S}(\varphi \land X).
\]

**Weak continual common knowledge of fact \(\varphi\) by set \(\mathcal{S}\)**, denoted \(\mathsf{W}_\mathcal{S}^\mathcal{S} \varphi\) is the greatest fixed point of

\[
X \equiv □E_\mathcal{S}(\varphi \land X).
\]

Halpern, Moses, and Waarts note that \(\mathsf{W}_\mathcal{S}^\mathcal{S} \varphi\) is equivalent to \(\bigwedge_{i \geq 1} (□A_\mathcal{S})^i \varphi\); similarly, \(\mathsf{S}_\mathcal{S}^\mathcal{S} \varphi\) is equivalent to the infinite conjunction \(\bigwedge_{i \geq 1} (□E_\mathcal{S})^i \varphi\). That is, a fact \(\varphi\) is continual common knowledge to a set if it is always the case that everyone in the set knows \(\varphi\), it is always the case that everyone in the set knows that it is always the case that everyone in the set knows \(\varphi\), etc. Continual common knowledge is *stronger* than simple common knowledge. It guarantees that all members know a fact at all times and that all levels of knowledge hold at all times. Continual common knowledge implies "continually" common knowledge. Halpern, Moses, and Waarts used weak continual common knowledge to construct optimal solutions to *Eventual Byzantine Agreement*. Among other things, the current paper explores the use of strong continual common knowledge in the solutions of consistent coordination problems.

It may seem odd that a stronger form of knowledge (continual common knowledge) is necessary to solve a weaker problem (nonsimultaneous coordination). The reason for this apparent contradiction is the set of processes whose knowledge is relevant. When studying simultaneous coordination, common knowledge of the entire set of nonfaulty processes is important. For nonsimultaneous coordination, it is the continual common knowledge of a very different set of processes that is relevant.

The two forms of continual common knowledge have properties similar to eventual common knowledge. Both satisfy positive introspection; if a fact is continual common knowledge to a set, then it is always the case that all members of the set know this. Thus, the following are valid:

- \(\mathsf{S}_\mathcal{S}^\mathcal{S} \varphi \Rightarrow □E_\mathcal{S}(\varphi \land \mathsf{S}_\mathcal{S}^\mathcal{S} \varphi)\); and
- \(\mathsf{W}_\mathcal{S}^\mathcal{S} \varphi \Rightarrow □A_\mathcal{S}(\varphi \land \mathsf{W}_\mathcal{S}^\mathcal{S} \varphi)\).

(These follow directly from the fixed-point definition.) Both satisfy a kind of negative introspection:

**Theorem 1**: The following are valid:

- \(\neg \mathsf{S}_\mathcal{S}^\mathcal{S} \varphi \land K_p(p \in \mathcal{S}) \Rightarrow K_p \neg \mathsf{S}_\mathcal{S}^\mathcal{S} \varphi\); and
- \(\neg \mathsf{W}_\mathcal{S}^\mathcal{S} \varphi \land (p \in \mathcal{S}) \Rightarrow B_p \neg \mathsf{W}_\mathcal{S}^\mathcal{S} \varphi\).

**Proof**: The proofs of the two parts are similar, and only the first is presented here. Consider \(p \in \mathcal{P}\) and some point \((r, l)\) such that \((\mathcal{R}, r, l) \models \neg \mathsf{S}_\mathcal{S}^\mathcal{S} \varphi \land K_p(p \in \mathcal{S})\). It suffices to show that, for any run \(r'\) such that \(r'_p(l) = r_p(l)\), \((\mathcal{R}, r', l) \models \neg \mathsf{S}_\mathcal{S}^\mathcal{S} \varphi\). Suppose for a contradiction that \((\mathcal{R}, r', l) \models \mathsf{S}_\mathcal{S}^\mathcal{S} \varphi\). Since \((\mathcal{R}, r, l) \models K_p(p \in \mathcal{S})\), \(p \in \mathcal{S}(r', l)\). By positive introspection,
\[(\mathcal{R}, r', l) \models K_p \varphi. \text{ Thus, } (\mathcal{R}, r, l) \models S_S \psi, \text{ a contradiction.} \]

Each form of continual common knowledge satisfies an induction rule:

- If \( \varphi \Rightarrow \Box \psi \) is valid in a system, then \( \varphi \Rightarrow S_S \psi \) is also valid in that system.
- If \( \varphi \Rightarrow \Box A \psi \) is valid in a system, then \( \varphi \Rightarrow W_S \psi \) is also valid in that system.

(Again, these follow from the fixed-point definitions.) Finally, continual common knowledge is \textit{continual} in that, if it is true at any point in a run, it is true at every point in that run:

**Theorem 2:** The following are valid:

- \( S_S \varphi \Leftrightarrow \Box S_S \varphi \); and
- \( W_S \varphi \Leftrightarrow \Box W_S \varphi \).

**Proof:** The proofs of the two parts are similar, and only the second is presented here. Recall that \( W_S \varphi \Leftrightarrow \bigwedge_{i \geq 1} (\Box A) \varphi \). From the definition of \( \Box \), \( \Box \varphi \Leftrightarrow \Box \varphi \) and \( \bigwedge F \Box \varphi_i \Leftrightarrow \Box \bigwedge F \varphi_i \) (for any fact \( \varphi \) and finite set of facts \( F = \{ \varphi_1, \varphi_2, \ldots, \varphi_l \} \). This implies that \( \bigwedge_{i \geq 1} (\Box A) \varphi \Leftrightarrow \bigwedge_{i \geq 1} (\Box A) \varphi \Leftrightarrow \bigwedge_{i \geq 1} (\Box A) \varphi \Leftrightarrow \Box W_S \varphi \).

5 **Knowledge and Coordination**

This section shows some basic relationships between processor knowledge and solutions to the different types of coordination problems defined earlier. These relationships will be used to construct some very simple solutions to these problems that serve as the foundation of subsequent results.

Note first that, to perform an action, a processor must know (or believe) that the action is enabled:

**Theorem 3:** Let \( C \) be a coordination problem.

- If \( P(\Phi) \) \( N \)-satisfies \( C \), then \( \mathcal{R}_P \models \Phi_{i,p} \Rightarrow B_p^{\mathcal{N}} \text{ok}_{i} \).
- If \( P(\Phi) \) \( C \)-satisfies \( C \), then \( \mathcal{R}_P \models \Phi_{i,p} \Rightarrow K_p \text{ok}_{i} \).

**Proof:** For the first case, suppose for a contradiction that, for some point \( \langle r, l \rangle \), \( (\mathcal{R}_P, r, l) \models \Phi_{i,p} \land \neg B_p^{\mathcal{N}} \text{ok}_{i} \). Then there must be some other run \( r' \) such that \( r'_p(l) = r_p(l) \) and \( (\mathcal{R}_P, r', l) \models p \in \mathcal{N} \land \neg \text{ok}_{i} \). Since \( \Phi_{i,p} \) is a fact about \( p \)'s local state, \( (\mathcal{R}_P, r', l) \models \Phi_{i,p} \). This contradicts the fact that \( P(\Phi) \) \( N \)-satisfies \( C \) \( \langle r', l \rangle \) is a point at which nonfaulty processor \( p \) performs \( a_i \) despite the fact that \( \text{ok}_{i} \) is false.

The proof of the second case is similar, except that \( K_p \text{ok}_{i} \) can be shown because \( \text{ok}_{i} \) must be true if \( p \) performs \( a_i \), even if \( p \) is faulty.

For problems requiring termination, a processor must be sure, before taking an action, that all nonfaulty processors will eventually perform the same action. Each of these processors must in turn know or believe the same thing. This indicates that eventual common knowledge is necessary for problems requiring termination. To prove this, we begin with the following lemma:
Lemma 4: Let \( \mathcal{C} \) be a coordination problem.

- If \( P(\Phi) \) NT-satisfies \( \mathcal{C} \), then \( \mathcal{R}_P \models \Phi_{i,p} \land (p \in \mathcal{N}) \Rightarrow W^{\Phi}_{\mathcal{N}} ok_i \).
- If \( P(\Phi) \) CT-satisfies \( \mathcal{C} \), \( \mathcal{R}_P \models \Phi_{i,p} \Rightarrow S^{\Phi}_{\mathcal{N}} ok_i \).

Proof: For the first case, let \( \varphi_i \equiv \bigvee_{p \in \mathcal{N}} \Phi_{i,p} ; \varphi_i \) is true if and only if some nonfaulty processor performs \( a_i \). Since \( P(\Phi) \) NT-satisfies \( \mathcal{C} \), \( \varphi_i \) implies that all nonfaulty processors eventually perform \( a_i \). Thus, each processor eventually knows that, if it is nonfaulty, \( \varphi_i \land ok_i \) holds (\( \varphi_i \) by definition and \( ok_i \) by Theorem 3). Thus, \( \varphi_i \Rightarrow \Diamond_{\mathcal{N}} (\varphi_i \land ok_i) \) is valid in \( \mathcal{R}_P \). By induction, \( \varphi_i \Rightarrow W^{\Phi}_{\mathcal{N}} ok_i \) is also valid in \( \mathcal{R}_P \). Since \( \Phi_{i,p} \land (p \in \mathcal{N}) \Rightarrow \varphi_i \), we have \( \mathcal{R}_P \models \Phi_{i,p} \Rightarrow S^{\Phi}_{\mathcal{N}} ok_i \).

The proof of the second case is similar, except that \( \varphi_i \) is defined instead as \( \bigvee_{p \in \mathcal{P}} \Phi_{i,p} \).

Since \( P(\Phi) \) CT-satisfies \( \mathcal{C} \), this weaker \( \varphi_i \) also implies that all nonfaulty processors eventually perform \( a_i \). In this case, any nonfaulty processor that executes \( a_i \) unconditionally knows \( \varphi_i \). Furthermore, by Theorem 3, it also knows \( ok_i \). Thus, \( \varphi_i \Rightarrow \Diamond_{\mathcal{N}} (\varphi_i \land ok_i) \) is valid in the system; thus, by induction, \( \mathcal{R}_P \models \Phi_{i,p} \Rightarrow S^{\Phi}_{\mathcal{N}} ok_i \). \( \square \)

Arguments similar to those in the proof of Theorem 3 can now show that processors performing an action must actually know (or believe) the required eventual common knowledge:

Theorem 5: Let \( \mathcal{C} \) be a coordination problem.

- If \( P(\Phi) \) NT-satisfies \( \mathcal{C} \), then \( \mathcal{R}_P \models \Phi_{i,p} \Rightarrow B^{\Phi}_{\mathcal{N}} W^{\Phi}_{\mathcal{N}} ok_i \).
- If \( P(\Phi) \) CT-satisfies \( \mathcal{C} \), then \( \mathcal{R}_P \models \Phi_{i,p} \Rightarrow K^{\Phi}_{\mathcal{N}} S^{\Phi}_{\mathcal{N}} ok_i \).

Proof: For the first case, suppose for a contradiction that, for some point \( \langle r, l \rangle \), \( (\mathcal{R}_P, r, l) \models \Phi_{i,p} \land \neg B^{\Phi}_{\mathcal{N}} W^{\Phi}_{\mathcal{N}} ok_i \). Then there must be some other run \( r' \) such that \( r'_p(l) = r_p(l) \) and \( (\mathcal{R}_P, r', l) \models p \in \mathcal{N} \land \neg W^{\Phi}_{\mathcal{N}} ok_i \). Since \( \Phi_{i,p} \) is a fact about \( p \)'s local state, \( (\mathcal{R}_P, r', l) \models \Phi_{i,p} \). This contradicts Lemma 4: there is a point \( \langle r', l \rangle \) such that \( (\mathcal{R}_P, r', l) \models \Phi_{i,p} \land (p \in \mathcal{N}) \land \neg W^{\Phi}_{\mathcal{N}} ok_i \).

The proof of the second case is similar. \( \square \)

Theorems 3 and 5 suggest a family of very simple coordination protocols. Each endeavor to perform one action as quickly as possible while the others are never performed. While these protocols may be neither optimal nor optimum, they are important in the development of optimal protocols.

Theorem 6: Let \( \mathcal{C} \) be a coordination problem. For each \( a_i \in \mathcal{C} \), consider an action function \( \Phi^i \) defined so that \( \Phi^i_{j,p} \equiv \text{false} \) for all \( j \neq i \) and \( p \in \mathcal{P} \).

- \( P(\Phi^i) \) NT-satisfies \( \mathcal{C} \) if \( \Phi^i_{i,p} \equiv B^{\Phi^i}_{\mathcal{N}} ok_i \);
- \( P(\Phi^i) \) CT-satisfies \( \mathcal{C} \) if \( \Phi^i_{i,p} \equiv K^{\Phi^i}_{\mathcal{N}} ok_i \);
- \( P(\Phi^i) \) NT-satisfies \( \mathcal{C} \) if \( \Phi^i_{i,p} \equiv B^{\Phi^i}_{\mathcal{N}} W^{\Phi^i}_{\mathcal{N}} ok_i \); and
- \( P(\Phi^i) \) CT-satisfies \( \mathcal{C} \) if \( \Phi^i_{i,p} \equiv K^{\Phi^i}_{\mathcal{N}} S^{\Phi^i}_{\mathcal{N}} ok_i \).
Proof: Proofs are given for the first and fourth cases; the others are similar.

To prove that \( P(\Phi^i) \) N-satisfies \( C \) if \( \Phi_{i,p} \equiv B_p^N \), one must show that all runs satisfy the normal agreement and validity conditions. The agreement condition is obviously satisfied: no processor ever performs any action other than \( a_i \). Suppose now that some nonfaulty processor \( p \) performs \( a_i \) at point \( (r,l) \). By the definition of \( \Phi_{i,p} \), \( \Phi_{i,p} \equiv B_p^N \); thus, \( ok_i \) must hold of all runs \( r' \) such that \( r'_p(l) = r_p(l) \) and \( p \in N(r') \). \( \langle r,l \rangle \) is such a point, so the validity condition is satisfied.

To prove that \( P(\Phi^i) \) CT-satisfies \( C \) if \( \Phi_{i,p} \equiv K_p S_N^\circ \), one must show that all runs satisfy the consistent agreement, validity, and termination conditions. Again, the agreement condition is trivially satisfied and the validity condition follows from a proof similar to the above (recall that \( S_N^\circ \Rightarrow ok_i \) in the system because \( N \) is never empty). To show the termination condition, suppose that some processor \( p \) performs \( a_i \) at point \( (r,l) \). By the definition of \( \Phi_{i,p} \), \( (\mathcal{R}_P,r,l) \models S_N^\circ \). By the positive introspection of strong eventual common knowledge, it is the case that, for any \( q \in N(r) \), there is some \( l' \geq l \) such that \( (\mathcal{R}_P,r,l') \models K_q S_N^\circ \). Since this is \( \Phi_{i,q} \), it should be clear that any processor nonfaulty in run \( r \) will eventually perform \( a_i \), and the termination condition is satisfied. \( \square \)

6 Optimum Protocols

Moses and Tuttle [13] showed that there exists no N-optimum protocol for Eventual Byzantine Agreement. This section shows there are some coordination problems for which optimum protocols do exist. It precisely characterizes these problems and gives specifications of optimum solutions. The characterization is sufficient because it guarantees that a protocol can perform an action as soon as one of the necessary conditions of Theorems 3 and 5 becomes true (the resulting protocol must thus be optimum). The characterization is necessary because it is implied by the existence of a protocol that dominates all the simple protocols given in Theorem 6 (an optimum protocol would dominate all of these).

As has been noted elsewhere [2,4,13,15], there is an optimum solution to a problem if and only if there is a solution using a full-information communication protocol [2,7,10]. A full-information protocol is one in which each processor sends its local state to all others in each round and, at the end of a round, sets its local state to the vector of messages received in that round. Moses and Tuttle [13] showed that, if failures are benign, a full-information protocol can be simulated by one that uses messages of polynomial size.

Lemma 7 shows that the conditions shown necessary in Theorems 3 and 5 must be continual common knowledge whenever any action is taken by an optimum full-information protocol. In the statement of this lemma, \( \psi_i \) indicates that some (nonfaulty) processor is performing action \( a_i \). (Note that \( \psi_i \) is defined to depend only on \( N \) in the normal cases (where the actions of the faulty processors are unimportant); in the consistent cases, \( \psi_i \) depends on all of \( \mathcal{P} \).) The sets \( S_i \) to which the continual common knowledge is ascribed contain all processors that have the minimum knowledge necessary to perform some other action.

Lemma 7: Let \( C \) be a coordination problem and let \( F(\Psi) \) be a full-information action protocol. Then

- If \( F(\Psi) \) is N-optimum for \( C \), then \( \mathcal{R}_F \models \psi_i \Rightarrow W_{S_i}^\circ \), where \( \psi_i \equiv \bigvee_{p \in N} \psi_{i,p} \) and \( S_i = \{ p \in N \mid \forall j \neq i \ B_p^N \}. \)
• If \( F(\Psi) \) is C-optimum for \( C \), then \( R_F \models \psi_i \Rightarrow S_{S_i}^k o_k \), where \( \psi_i \equiv \bigvee_{p \in P} \Psi_{i,p} \) and \( S_i = \{ p \in P \mid \forall j \neq i, K_p o_j \} \).

• If \( F(\Psi) \) is NT-optimum for \( C \), then \( R_F \models \psi_i \Rightarrow W_{S_i}^N W_N^k o_k \), where \( \psi_i \equiv \bigvee_{p \in P} \Psi_{i,p} \) and \( S_i = \{ p \in P \mid \forall j \neq i, B_p^N W_N^k o_k \} \).

• If \( F(\Psi) \) is CT-optimum for \( C \), then \( R_F \models \psi_i \Rightarrow S_{S_i}^N S_N^k o_k \), where \( \psi_i \equiv \bigvee_{p \in P} \Psi_{i,p} \) and \( S_i = \{ p \in P \mid \forall j \neq i, K_p S_N^k o_k \} \).

**Proof:** The proofs of the four cases are similar; only the third is given here. Suppose that \( (R_F, r, l) \models \psi_i \) for some point \( (r, l) \). This means that some nonfaulty processor performs \( a_i \) by time \( l \) in run \( r \). By the agreement property, \( F(\Psi) \) can have no nonfaulty processor perform any action in \( r \) other than \( a_i \). The proof now shows that \( (R_F, r, l, l') \models \bigwedge_{j} A_{S_j} W_{N}^k o_k \). Consider any \( l' \) and let \( p \in S_i(r, l') \); this means that \( p \in N'(r) \) and \( (R_F, r, l') \models B_p^N W_N^k o_k \) for some \( j \neq i \). Recall the protocol \( F(\Phi) \) as defined in NT-case of Theorem 6 above (using the full-information communication protocol \( F \)). Since \( \Phi_{j,p} \equiv B_p^N W_N^k o_k, F(\Phi) \) has \( p \) act at point \( (r, l') \). Since \( F(\Psi) \) is NT-optimum and \( p \) is nonfaulty \( F(\Psi) \) must also have \( p \) act at point \( (r, l') \). Because only action \( a_i \) can be performed in run \( r \), it must be that \( \bigwedge_{j} A_{S_j} W_{N}^k o_k \). Since \( \bigwedge_{j} A_{S_j} W_{N}^k o_k \) is always valid, \( (R_F, r, l, l') \models B_p^N \psi_i \). Furthermore, \( (R_F, r, l, l') \models B_p^N W_N^k o_k \) by Theorem 5. This means that \( (R_F, r, l, l') \models B_p^N (W_N^k o_k \land \psi_i) \). Since \( l' \) and \( p \) were chosen arbitrarily, \( (R_F, r, l, l') \models \bigwedge_{j} A_{S_j} (W_N^k o_k \land \psi_i) \). Since \( (r, l, l') \) was chosen arbitrarily, we have \( R_F \models \psi_i \Rightarrow \bigwedge_{j} A_{S_j} (W_N^k o_k \land \psi_i) \). By induction, then, \( R_F \models \psi_i \Rightarrow \bigwedge_{j} W_{S_j}^N W_N^k o_k \), as desired.

Lemma 7 gives a property that holds of any optimum protocol: whenever some action \( a_i \) is performed, it is continual common knowledge that \( a_i \) can be performed to the set of processors that might perform another action. This is not possible for all coordination problems. Theorem 8 gives the conditions that are necessary and sufficient for the existence of optimum protocols. Informally, these conditions state that, whenever a processor has the minimum knowledge necessary to perform some action, then it also knows (or believes) that the continual common knowledge given in Lemma 7 also holds.

**Theorem 8:** Let \( C \) be a coordination problem.

- There is an N-optimum protocol for \( C \) if and only if \( R_F \models B_p^N o_k \Rightarrow \bigvee_{a_j \in C} B_p^N W_{S_j}^N o_k \), where \( S_j = \{ q \in N \mid \forall k \neq j, B_p^N o_k \} \).

- There is a C-optimum protocol for \( C \) if and only if \( R_F \models K_p o_k \Rightarrow \bigvee_{a_j \in C} K_p S_{S_j}^N o_k \), where \( S_j = \{ q \in P \mid \forall k \neq j, K_p o_k \} \).

- There is an NT-optimum protocol for \( C \) if and only if \( R_F \models B_p^N W_N^k o_k \Rightarrow \bigvee_{a_j \in C} B_p^N W_{S_j}^N W_N^k o_k \), where \( S_j = \{ q \in N \mid \forall k \neq j, B_p^N W_N^k o_k \} \).

- There is a CT-optimum protocol for \( C \) if and only if \( R_F \models K_p S_N^k o_k \Rightarrow \bigvee_{a_j \in C} K_p S_{S_j}^N S_N^k o_k \), where \( S_j = \{ q \in P \mid \forall k \neq j, K_p S_N^k o_k \} \).

**Proof:** We present only the proof of the third case; the others are similar. Assume that there is a total order ‘\(<\)’ on the actions in \( C \). In the following, we may use the following fact: \( R_F \models p \in S_j \Rightarrow B_p^N (p \in S_j) \). This is because \( \bigvee_{k \neq j} B_p^N W_N^k o_k \) is a fact about \( q \)’s local state.
Consider first the “if” direction and assume that the given condition holds. Consider now the protocol $F(\Phi)$, where $\Phi_{i,p} \equiv B^N_{p}(W^\bowtie_{\Delta} \land W^\bowtie_{\Delta} \land \land_{j<i} \lnot W^\bowtie_{\Delta} \land W^\bowtie_{\Delta} \land (j))$. We will show that $F(\Phi)$ is NT-optimum for $C$. Note first that it NT-satisfies $C$. Since $W^\bowtie_{\Delta} \Rightarrow ok_i$ (see Section 4.2 above), $\mathcal{R}_F \models \Phi_{i,p} \Rightarrow B^N_{p} \equiv ok_i$, so $F(\Phi)$ satisfies the validity condition. Now, suppose for a contradiction that, in some run $r$, two nonfaulty processors $p$ and $q$ perform actions $a_i$ and $a_j$ at times $l$ and $l'$, respectively, where $i < j$. This implies $(\mathcal{R}_F, r, l') \models \lnot W^\bowtie_{\Delta} \land W^\bowtie_{\Delta}$ and $(\mathcal{R}_F, r, l) \models W^\bowtie_{\Delta} \land W^\bowtie_{\Delta}$, which are contradictory because of the continuity of continual common knowledge. To prove termination, suppose that $(\mathcal{R}_F, r, l) \models \Phi_{i,p}$, where $p \in \mathcal{N}(r)$. By the definition of $\Phi$ and the continuity of eventual common knowledge, $(\mathcal{R}_F, r, l) \models \square \land_{j<i} \lnot W^\bowtie_{\Delta} \land W^\bowtie_{\Delta} \land W^\bowtie_{\Delta}$. The proof must show that, for all $q \in \mathcal{N}(r)$, there is some $l'$ such that $(\mathcal{R}_F, r, l') \models \Phi_{i,q}$. Since $W^\bowtie_{\Delta} \equiv ok_i$ holds at $(r, l)$, $B^N_{q} \equiv W^\bowtie_{\Delta} \land W^\bowtie_{\Delta}$ must hold at some point $(r, l')$. At that point, $q \in S_j$ for all $j \neq i$. Thus, by negative introspection for weak continual common knowledge, $(\mathcal{R}_F, r, l') \models B^N_{q} \land_{j<i} \lnot W^\bowtie_{\Delta} \land W^\bowtie_{\Delta}$. Since $(\mathcal{R}_F, r, l') \models B^N_{q} \equiv W^\bowtie_{\Delta}$, the hypothesis indicates that $\lor_{a_j \in C} B^N_{q} \equiv W^\bowtie_{\Delta} \land W^\bowtie_{\Delta}$. If $B^N_{q} \equiv W^\bowtie_{\Delta} \land W^\bowtie_{\Delta}$ for any $j \neq i$, then, because $q \in S_j$, $B^N_{q} \equiv W^\bowtie_{\Delta} \land W^\bowtie_{\Delta}$. In either case, $(\mathcal{R}_F, r, l') \models B^N_{q} \equiv W^\bowtie_{\Delta} \land W^\bowtie_{\Delta}$. This means that $(\mathcal{R}_F, r, l') \models \Psi_{i,q}$, completing the proof of termination.

The proof now shows that $F(\Phi)$ is NT-optimum. Consider any full-information protocol $F(\Psi)$ that NT-satisfies $C$. Suppose that $\Psi_{i,p}$ holds at some point $(r, l)$ for some $p \in \mathcal{N}(r)$. The proof must show that, for some $a_j \in C$, $(\mathcal{R}_F, r, l) \models \Phi_{j,p}$. By Theorem 5 applied to $F(\Psi)$, it must be that $B^N_{p} \equiv \land_{i<j} \lnot W^\bowtie_{\Delta} \land W^\bowtie_{\Delta}$. By the original hypothesis, $B^N_{p} \equiv \land_{j<i} \lnot W^\bowtie_{\Delta} \land W^\bowtie_{\Delta}$. Consider the least such $j$; we show that $(\mathcal{R}_F, r, l) \models B^N_{p} \equiv \land_{i<j} \lnot W^\bowtie_{\Delta} \land W^\bowtie_{\Delta}$. If $j = i$, then this is immediate. If $j \neq i$, then $p \in S_j$, so $B^N_{p} \equiv \land_{i<j} \lnot W^\bowtie_{\Delta} \land W^\bowtie_{\Delta}$ by positive introspection for continual common knowledge. Now consider any $k < j$; $(\mathcal{R}_F, r, l) \models \lnot B^N_{p} \equiv W^\bowtie_{\Delta} \land W^\bowtie_{\Delta}$. Since $B^N_{p} \equiv \land_{j<i} \lnot W^\bowtie_{\Delta} \land W^\bowtie_{\Delta}$ and $p \in \mathcal{N}(r)$, $p \in S_k$. By positive introspection again, it must be that $\lnot W^\bowtie_{\Delta} \land W^\bowtie_{\Delta}$ and, by negative introspection, $B^N_{p} \equiv \land_{i<j} \lnot W^\bowtie_{\Delta} \land W^\bowtie_{\Delta}$. Since this is true for all $k < j$, it is clear that $\Phi_{j,p}$ holds, as desired.

Next, consider the "only if" direction. Assume that there is some NT-optimum protocol $F(\Psi)$ for $C$. The proof must show that $\mathcal{R}_F \models B^N_{p} \equiv \land_{i<j} \lnot W^\bowtie_{\Delta} \land W^\bowtie_{\Delta}$. Suppose that $(\mathcal{R}_F, r, l) \models B^N_{p} \equiv \land_{i<j} \lnot W^\bowtie_{\Delta} \land W^\bowtie_{\Delta}$. Consider now two cases:

- For all runs $r'$ with $r'_i(l) = r_p(l), p \notin \mathcal{N}(r')$. This means that $(\mathcal{R}_F, r, l) \models B^N_{p} \equiv \land_{i<j} \lnot W^\bowtie_{\Delta} \land W^\bowtie_{\Delta}$ for all facts $\varphi$, and the result holds immediately.

- For some run $r'$ with $r'_i(l) = r_p(l), p \notin \mathcal{N}(r')$. Note that, since $p$ has the same local state at the two points, it believes the same facts at the two points. Thus, $(\mathcal{R}_F, r', l) \models B^N_{p} \equiv \land_{i<j} \lnot W^\bowtie_{\Delta} \land W^\bowtie_{\Delta}$. This means that, when executing the protocol $F(\Phi')$ defined in the NT-case of Theorem 6, $p$ executes $a_i$ at point $(r', l)$. Since $F(\Psi)$ is NT-optimum, it dominates $F(\Phi')$, so $(\mathcal{R}_F, r', l) \models \Phi_{j,p}$ for some $a_j \in C$. Consider now any run $r''$ such that $r''(l) = r_p(l)$ and $p \notin \mathcal{N}(r'')$. Since $\Phi_{j,p}$ is a fact about $p$’s local state, it must hold at $(r'', l)$, so $(\mathcal{R}_F, r'', l) \models \psi_j$, where $\psi_j$ is defined as in the NT-case of Lemma 7. By that lemma, $(\mathcal{R}_F, r'', l) \models B^N_{p} \equiv \land_{i<j} \lnot W^\bowtie_{\Delta} \land W^\bowtie_{\Delta}$. Thus, $(\mathcal{R}_F, r', l) \models B^N_{p} \equiv \land_{i<j} \lnot W^\bowtie_{\Delta} \land W^\bowtie_{\Delta}$. This means that $(\mathcal{R}_F, r, l) \models B^N_{p} \equiv \land_{i<j} \lnot W^\bowtie_{\Delta} \land W^\bowtie_{\Delta}$, implying $(\mathcal{R}_F, r, l) \models \lor_{a_j \in C} B^N_{p} \equiv \land_{i<j} \lnot W^\bowtie_{\Delta} \land W^\bowtie_{\Delta}$.

Since $(r, l)$ was chosen arbitrarily, this gives the desired validity. \qed
One can note at this point that the result of Moses and Tuttle that there can be no N-optimum protocol for Eventual Byzantine Agreement, can be seen as a corollary to Theorem 8. Eventual Byzantine Agreement is a coordination problem with two actions \( a_0 \) and \( a_1 \), where \( a_i \) indicates “decide \( i \).” The enabling conditions \( ok_i \) are “some processor began in initial state \( i \).”

The problem, as typically defined, considers N-solutions with a stronger notion of termination: the correct processors must perform actions in every run. Since there are, for synchronous systems, protocols that terminate in all runs, it is sufficient to consider N-optimum protocols, as they will do so also.

**Corollary 9:** There is no N-optimum protocol for Eventual Byzantine Agreement.

**Proof:** Consider a run \( r \) in which there are two nonfaulty processors \( p \) and \( q \) such that \( p \)'s initial state is 0 and \( q \)'s is 1. We will show that \((R_F, r, 0) \models B_p^N ok_0 \land \neg B_p^N W_{S_0} ok_0 \land \neg B_p^N W_{S_1} ok_1 \)

\[ \text{where } S_0 = \{ r \in \mathcal{N} \mid B_p^N ok_1 \} \text{ and } S_1 = \{ r \in \mathcal{N} \mid B_p^N ok_0 \}. \]

By Theorem 8, this will indicate that there can be no N-optimum protocol for Eventual Byzantine Agreement. Consider \( p \)'s belief at time 0. Clearly, \( B_p^N ok_0 \) because its initial state is 0. This implies \( p \in S_1(r, 0) \). Since there has been no communication, \( \neg B_p^N ok_1 \), so it must be that \( \neg W_{S_0} ok_1 \) and, therefore, \( \neg B_p^N W_{S_0} ok_1 \). Similarly, \( q \in S_0(r, 0) \) and \( \neg B_p^N ok_0 \), so \( \neg W_{S_0} ok_0 \). Since \( p \) is nonfaulty in this run, \( \neg B_p^N W_{S_0} ok_0 \). Thus, \( \neg B_p^N W_{S_0} ok_0 \land \neg B_p^N W_{S_0} ok_1 \), as desired.

Theorem 8 gives four conditions, one for each type of coordination, that are necessary and sufficient for the existence of an optimum solution. This theorem can be used to show that certain problems have optimum solutions regardless of the type of coordination required. These include problems whose enabling conditions of all actions are mutually exclusive. For example, suppose that one processor is seeking to broadcast a binary value. If a processor can decide on a value \( v \) only if that was the broadcaster’s value, then the enabling conditions for deciding 0 and deciding 1 are mutually exclusive and there is an optimum protocol.\(^9\)

In addition, there may be optimum solutions to problems in which the enabling conditions are related in certain ways. Consider a system and a problem with three possible actions such that the following implications are valid in the system: \( ok_1 \Rightarrow \neg ok_3; ok_3 \Rightarrow \neg ok_1; \) and \( ok_2 \Rightarrow ok_1 \). Such a problem always has an optimum solution.

Not all coordination problems admit optimum solutions. However, every coordination problem has a nonempty set of optimal solutions. The remainder of this paper considers the development of optimal solutions. Halpern, Moses, and Waarts [12] showed how the weak form of continual common knowledge could be used to construct N-optimal solutions to Eventual Byzantine Agreement. However, they did not explicitly consider the termination properties of the protocols they developed. When problems requiring termination are considered, it is necessary to combine eventual knowledge with continual knowledge. We call this combination extended common knowledge.

### 7 Extended Common Knowledge

The optimum protocols given in Section 6 for problems with termination required processors to gain continual common knowledge of eventual common knowledge of some enabling condition. Recall that eventual common knowledge is the greatest fixed point of the “everyone eventually knows” operator, while continual common knowledge is the greatest fixed point of

\(^9\)These enabling conditions are different from those classically given for Reliable Broadcast. Those conditions also permit deciding either value if the broadcaster is faulty, regardless of its initial value.
the “everyone always knows” operator. To characterize the domination relation between solutions to problems requiring termination, it becomes essential to develop a form of common knowledge that is the greatest fixed point of both these operators together. We call this extended common knowledge. Extended common knowledge pertains to two potentially different sets of processors: the set with eventual knowledge and the set with continual knowledge.

Strong extended common knowledge of fact \( \varphi \) with respect to sets \( S \) and \( T \), denoted \( S_{S,T}^- \varphi \), is the greatest fixed point of

\[
X \iff \diamond E_S(\varphi \land X) \land \Box E_T(\varphi \land X).
\]

Weak extended common knowledge of fact \( \varphi \) with respect to sets \( S \) and \( T \), denoted \( W_{S,T}^- \varphi \), is the greatest fixed point of

\[
X \iff \diamond A_S(\varphi \land X) \land \Box A_T(\varphi \land X).
\]

It is easy to see that \( S_{S,T}^- \varphi \) implies the infinite conjunction

\[
\diamond E_S \varphi \land \Box E_T \varphi \land (\diamond E_S)^2 \varphi \land \cdots.
\]

(A similar statement is true of weak extended common knowledge.) This is the first form of common knowledge that is the fixed point of two different knowledge operators. It turns out to be exactly what is necessary to capture the combined agreement and termination conditions of some coordination problems.

The two forms of extended common knowledge have some important properties that will be useful. Both forms of extended common knowledge satisfy positive introspection with respect to both eventual and continual knowledge. That is, it is easy to use the fixed-point definitions to show that the following are valid:

- \( S_{S,T}^- \varphi \Rightarrow \diamond E_S S_{S,T}^- \varphi \);
- \( S_{S,T}^- \varphi \Rightarrow \Box E_T (\varphi \land S_{S,T}^- \varphi ) \);
- \( W_{S,T}^- \varphi \Rightarrow \diamond A_S W_{S,T}^- \varphi \); and
- \( W_{S,T}^- \varphi \Rightarrow \Box A_T (\varphi \land W_{S,T}^- \varphi ) \).

(These can be stated more strongly; the forms given are sufficient for the results of this paper.) Both satisfy a kind of negative introspection with respect to continual knowledge, in that the following are valid:

- \( \neg S_{S,T}^- \varphi \land K_p (p \in T) \Rightarrow K_p \neg S_{S,T}^- \varphi \); and
- \( \neg W_{S,T}^- \varphi \land (p \in T) \Rightarrow B_p^T \neg W_{S,T}^- \varphi \).

(The proofs of these are similar to that of Theorem 1.) Extended common knowledge does not satisfy negative introspection with respect to eventual knowledge, which makes reasoning about it more difficult than reasoning about continual common knowledge.

Each form of extended common knowledge satisfies an induction rule that can be used to show that certain facts are extended common knowledge:

- If \( \varphi \Rightarrow \diamond E_S (\varphi \land \psi) \land \Box E_T (\varphi \land \psi) \) is valid in a system, then \( \varphi \Rightarrow S_{S,T}^- \psi \) is also valid in that system.
If $\varphi \Rightarrow \Diamond A_S(\varphi \land \psi) \land \Box A_T(\varphi \land \psi)$ is valid in a system, then $\varphi \Rightarrow W_{\vec{S},\vec{T}}^\varphi$ is also valid in that system.

(These follow from the fixed-point definitions.)

The sections below use extended common knowledge with the same kinds of sets that Section 6 used with eventual and continual common knowledge. The set used with eventual knowledge is the set $\mathcal{N}$ of nonfaulty processors, because this is the set of processors that must eventually perform an action. The sets used with continual knowledge are sets that know that performing some action is possible; these are the sets that must be brought into agreement with each other. Explicitly considered are cases in which facts $\varphi$ about runs (specifically, the enabling conditions of a coordination problem) become extended common knowledge. Because the first set $\mathcal{N}$ is never empty in the systems we consider, it is not hard to see that, in these systems, $S_{\vec{N},s}^\varphi \Rightarrow \varphi$ and $W_{\vec{N},s}^\varphi \Rightarrow \varphi$ are valid. These implications will be used to simplify the presentation of some protocols below.

It is important to understand the difference between extended common knowledge, as used below, and continual common knowledge of eventual common knowledge, which was used in Section 6. As noted above, strong eventual common knowledge to $S$ ($S_S^\Diamond \varphi$) implies the infinite conjunction $\bigwedge_{i \geq 1} (\Diamond E_S)^i \varphi$. Similarly, strong continual common knowledge to $T$ ($S_T^\Box \varphi$) is equivalent to the infinite conjunction $\bigwedge_{i \geq 1} (\Box E_T)^i \varphi$. Thus, $S_T^\Box S_S^\Diamond \varphi$, which is used in Section 6, implies the infinite conjunction $\bigwedge_{i,j \geq 1} (\Box E_T)^i (\Diamond E_S)^j \varphi$. Note that all $\Box E_T$ operators precede all $\Diamond E_S$ operators. In contrast, strong extended common knowledge of $\varphi$ by $S$ and $T$ ($S_{\vec{S},\vec{T}}^\varphi$) implies the infinite conjunction of all orderings of these operators as noted in Equation 1 above (thus, it is not hard to show that $S_{\vec{S},\vec{T}}^\varphi$ implies $S_T^\Box S_S^\Diamond \varphi$). It is the additional flexibility of the combination of knowledge operators that is needed to characterize domination as is done in the next section.

# Knowledge and Domination

This section exhibits a direct relationship between a dominating protocol and the knowledge that it must have about the protocol it dominates. Both continual common knowledge and extended common knowledge are used to characterize this relationship. We then show how these forms of knowledge can be used to construct a protocol that dominates a given protocol.

Theorem 10 shows that the domination relationship between two decision protocols (each using the same communication protocol) can be expressed using some form of common knowledge. In the case of normal solutions, a weak form is used, while a strong form is needed for consistent solutions. Solutions with termination use extended common knowledge, while the others use only continual common knowledge. Informally, Theorem 10 states that for a processor to perform an action in a dominating protocol, it must know (or believe) that the enabling condition for that action is continual (or extended) common knowledge. The set of processors requiring the continual knowledge contains exactly the processors performing some other action in the dominated protocol. In the case where extended common knowledge is needed, the set requiring eventual knowledge is the set of nonfaulty processors.

Formally, suppose that $\Phi$ is the action function used by the protocol to be dominated. For each $a_i \in \mathcal{C}$, define

- $\mathcal{P}_i = \{q \in \mathcal{P} \mid \bigvee_{j \neq i} \Phi_{j,p} \}$;
- $\mathcal{N}_i = \mathcal{P}_i \cap \mathcal{N}$.
\( \mathcal{P}_i \) is the set of processors performing some action other than \( a_i \); \( \mathcal{N}_i \) is the set of nonfaulty processors doing so. Because \( \Phi_{j,p} \) is a fact about \( p \)'s local state, it is easy to see that both \( (p \in \mathcal{P}_i) \Rightarrow K_p(p \in \mathcal{P}_i) \) and \( (p \in \mathcal{N}_i) \Rightarrow B^N_p(p \in \mathcal{N}_i) \) are valid.

**Theorem 10:** Let \( C \) be a coordination problem and let \( P(\Phi) \) and \( P(\Psi) \) be two decision protocols.

- If both protocols \( N \)-satisfy \( C \) and \( P(\Psi) \) \( N \)-dominates \( P(\Phi) \), then \( R_P |\psi_i \Rightarrow B^N_p (ok_i \land W^G_{\mathcal{N}_i} ok_i) \).
- If both protocols \( C \)-satisfy \( C \) and \( P(\Psi) \) \( C \)-dominates \( P(\Phi) \), then \( R_P |\psi_i \Rightarrow K_p (ok_i \land S^G_{\mathcal{P}_i} ok_i) \).
- If both protocols \( NT \)-satisfy \( C \) and \( P(\Psi) \) \( N \)-dominates \( P(\Phi) \), then \( R_P |\psi_i \Rightarrow B^N_p W^G_{\mathcal{N}_i \setminus \mathcal{P}_i} ok_i \).
- If both protocols \( CT \)-satisfy \( C \) and \( P(\Psi) \) \( C \)-dominates \( P(\Phi) \), then \( R_P |\psi_i \Rightarrow K_p S^G_{\mathcal{N}_i \setminus \mathcal{P}_i} ok_i \).

**Proof:** Proofs are supplied for the first and fourth cases.

Assume that both protocols \( N \)-satisfy \( C \) and that \( P(\Psi) \) \( N \)-dominates \( P(\Phi) \). By Theorem 3, \( R_P |\psi_i \Rightarrow B^N_p ok_i \). Let \( \psi_i \equiv \forall q \in \mathcal{N} (\Psi_{i,q} \land K_p ok_i) \). Since \( (\Psi_{i,p} \land p \in \mathcal{N}) \Rightarrow \psi_i \) and \( B^N_p (p \in \mathcal{N}) \) are valid, \( R_P |\psi_i \Rightarrow B^N_p \psi_i \) (recall that \( \Psi_{i,p} \) is a fact about \( p \)'s local state). Using techniques seen earlier (including induction), it is sufficient to show \( R_P |\psi_i \Rightarrow \square A_{\mathcal{N}_i} (\psi \land ok_i) \). Suppose that \( (R_P, r, l') \mid (\psi_i \land ok_i) \). Then \( R_P |\psi \Rightarrow \square A_{\mathcal{N}_i} (\psi \land ok_i) \). Since \( P(\Psi) \) \( N \)-dominates \( P(\Phi) \), the agreement condition ensures that \( (R_P, r, l') \mid \Psi_{i,p} \). Hence, \( R_P |\psi_i \Rightarrow \square A_{\mathcal{N}_i} (\psi \land ok_i) \) and, as argued earlier, \( (R_P, r, l') \mid B^N_p \psi_i \). Thus, \( (R_P, r, l) \mid \square A_{\mathcal{N}_i} (\psi \land ok_i) \), as desired.

Now assume that both protocols \( C \)-satisfy \( C \) and that \( P(\Psi) \) \( C \)-dominates \( P(\Phi) \). Let \( \psi_i \equiv \forall q \in \mathcal{P} (\Psi_{i,q} \land K_p \psi_i) \). Clearly, \( R_P |\psi_i \Rightarrow \psi_i \) is sufficient to show that \( R_P |\psi_i \Rightarrow \psi \land E_{\mathcal{N}_i} (\psi \land ok_i) \land E_{\mathcal{P}_i} (\psi \land ok_i) \). Assume that \( (R_P, r, l) \mid \psi \). The proof that \( (R_P, r, l) \mid \psi \land \square E_{\mathcal{N}_i} (\psi \land ok_i) \), consider some \( p \in \mathcal{N}(r) \). Since \( P(\Psi) \) has some processor perform \( a_i \) in \( r \), \( p \) must perform \( a_i \) in run \( r \); thus, \( (R_P, r, l') \mid \psi_i \), \( p \) is for some \( l' \). Clearly, \( (R, r, l') \mid K_p \psi_i \). Theorem 3 implies that \( (R_P, r, l') \mid K_p ok_i \), \( (R_P, r, l) \mid K_p (\psi \land ok_i) \). Since \( p \) was chosen arbitrarily, \( (R_P, r, l) \mid \psi_i \), completing the proof.

The first case above generalizes results observed earlier for *Eventual Byzantine Agreement* [12].

The following lemma is central both to the characterization and construction of optimal protocols given in Section 9. It shows how, given a coordination protocol, to construct another protocol that dominates it. This is done by improving the performance of a selected action so that it is performed as quickly as is possible by any protocol that dominates the original protocol.

**Lemma 11:** Let \( C \) be a coordination problem and let \( a_j \) be any action in \( C \).

- If \( P(\Phi) \) \( N \)-satisfies \( C \), then \( P(\Psi) \) \( N \)-satisfies \( C \) and \( N \)-dominates \( P(\Phi) \) if

\[
\Psi_{i,p} \equiv \begin{cases} 
B^N_p (ok_j \land W^G_{\mathcal{N}_j} ok_j) & \text{if } i = j \\
\Phi_{i,p} \land B^N_p (W^G_{\mathcal{N}_j} ok_j) & \text{if } i \neq j 
\end{cases}
\]
• If \( P(\Phi) \) C-satisfies \( C \), then \( P(\Psi) \) C-satisfies \( C \) and C-dominates \( P(\Phi) \) if

\[
\Psi_{i,p} \equiv \begin{cases} 
K_p(\text{ok}_j \land S_{p_j}^\Phi \text{ok}_j) & \text{if } i = j \\
\Phi_{i,p} \land K_p\neg S_{p_j}^\Phi \text{ok}_j & \text{if } i \neq j.
\end{cases}
\]

• If \( P(\Phi) \) NT-satisfies \( C \), then \( P(\Psi) \) NT-satisfies \( C \) and N-dominates \( P(\Phi) \) if

\[
\Psi_{i,p} \equiv \begin{cases} 
B^N_p W_{N_j}^\Psi \text{ok}_j & \text{if } i = j \\
\Phi_{i,p} \land B^N_p \neg W_{N_j}^\Psi \text{ok}_j & \text{if } i \neq j.
\end{cases}
\]

• If \( P(\Phi) \) CT-satisfies \( C \), then \( P(\Psi) \) CT-satisfies \( C \) and C-dominates \( P(\Phi) \) if

\[
\Psi_{i,p} \equiv \begin{cases} 
K_p S_{N_j}^\Psi \text{ok}_j & \text{if } i = j \\
\Phi_{i,p} \land K_p\neg S_{N_j}^\Psi \text{ok}_j & \text{if } i \neq j.
\end{cases}
\]

**Proof:** The proof considers only the fourth case; the remainder are similar. Suppose that \( P(\Phi) \) CT-satisfies \( C \) and let \( \Psi \) be as defined above. The proof must show that \( P(\Psi) \) CT-satisfies \( C \) and that it C-dominates \( P(\Phi) \). Consider first the validity condition. Suppose that \((R_P, r, l) \models \Psi_{i,p}\). If \( i = j \), then \((R_P, r, l) \models K_p S_{N_j}^{\Psi} \text{ok}_j\). Since \( S_{N_j}^{\Psi} \text{ok}_j \Rightarrow \text{ok}_j \) is valid in the system, \( a_j \) is an enabled action. If \( i \neq j \), then \((R_P, r, l) \models \Phi_{i,p} \land a_i \) is enabled because of the assumption that \( P(\Phi) \) CT-satisfies \( C \) (the correctness of \( P(\Phi) \) ensures validity). Next consider the agreement condition. Suppose that processors \( p \) and \( q \) perform actions \( a_i \) and \( a_k \), respectively, in run \( r \). If neither \( i \) nor \( k \) is \( j \), then \( \Phi_{i,p} \text{ and } \Phi_{k,q} \) hold in \( r \), implying that \( i = k \) (because \( P(\Phi) \) correctly gives agreement). Alternatively, assume without loss of generality that \( p \) performs \( a_j \) at time \( l \). Then \((R_P, r, l) \models K_p S_{N_j}^{\Psi} \text{ok}_j\). Suppose now that \((R_P, r, l') \models \Psi_{i,q} \) for some \( l' \) and some \( i \neq j \); by the definition of \( \Psi \), \((R_P, r, l') \models \Phi_{i,q} \) as well. Since \( q \in P_j(r, l') \), \((R_P, r, l') \models K_q S_{N_j}^{\Psi} \text{ok}_j \) by positive introspection. Thus, \((R_P, r, l') \models \neg \Psi_{i,q} \), giving the desired contradiction. Finally, consider the termination condition. Assume that \((R_P, r, l) \models \Psi_{i,p}\). The proof must show that, for every \( q \in N(r) \), there is some \( l' \) such that \((R_P, r, l') \models \Psi_{i,q}\). Consider two cases:

• \( i = j \). Then \((R_P, r, l) \models K_p S_{N_j}^{\Psi} \text{ok}_j\). By positive introspection, there is an \( l' \geq l \) such that \((R_P, r, l') \models K_p S_{N_j}^{\Psi} \text{ok}_j\). Since \( \Psi_{j,q} \equiv K_q S_{N_j}^{\Psi} \text{ok}_j\), this completes the proof.

• \( i \neq j \). Then \((R_P, r, l) \models \Phi_{i,p} \land K_p \neg S_{N_j}^{\Psi} \text{ok}_j\). By the termination of \( P(\Phi) \), there is an \( l' \) such that \((R_P, r, l') \models \Phi_{i,q}\). If \((R_P, r, l') \models S_{N_j}^{\Psi} \text{ok}_j\), then, since \( p \in P_j(r, l') \), \((R_P, r, l') \models K_p S_{N_j}^{\Psi} \text{ok}_j\) by positive introspection. This is a contradiction, so \((R_P, r, l') \models \neg S_{N_j}^{\Psi} \text{ok}_j\). Since \((R_P, r, l') \models K_q(q \in P_j)\), \((R_P, r, l') \models K_p \neg S_{N_j}^{\Psi} \text{ok}_j\) by negative introspection. Thus, \((R_P, r, l') \models \Phi_{i,q} \land K_p \neg S_{N_j}^{\Psi} \text{ok}_j\). Since \( \Psi_{i,q} \equiv \Phi_{i,q} \land K_p \neg S_{N_j}^{\Psi} \text{ok}_j\), this completes the proof.

In either case, \( q \) eventually performs \( a_i \), as desired.

To show that \( P(\Psi) \) dominates \( P(\Phi) \), assume that \((R_P, r, l) \models \Phi_{i,p}\). The proof must show that, for some \( a_k \in C,(R_P, r, l) \models \Psi_{k,p}\). Consider two cases:

• \( i = j \). Since \( P(\Phi) \) dominates itself, Theorem 10 implies that \((R_P, r, l) \models K_p S_{N_j}^{\Psi} \text{ok}_j\); thus, \((R_P, r, l) \models \Psi_{j,p}\).
Using Knowledge to Optimaly Achieve Coordination in Distributed Systems

- \( i \neq j \). This implies that \((\mathcal{R}_p, r, l) \models K_p(p \in \mathcal{P}_j)\). Consider now two subcases:
  
  - \((\mathcal{R}_p, r, l) \models S_{N^-}^{\mathcal{P}_j} ok_j\). Then \((\mathcal{R}_p, r, l) \models K_p S_{N^-}^{\mathcal{P}_j} ok_j\) by positive introspection, so \((\mathcal{R}_p, r, l) \models \Psi_{i,p}\).
  
  - \((\mathcal{R}_p, r, l) \models \neg S_{N^-}^{\mathcal{P}_j} ok_j\). Then \((\mathcal{R}_p, r, l) \models K_p \neg S_{N^-}^{\mathcal{P}_j} ok_j\) by negative introspection, so \((\mathcal{R}_p, r, l) \models \Psi_{i,p}\).

In all cases, \(P(\Phi)\) has \(p\) act at time \(l\), as desired, so \(P(\Psi)\) dominates \(P(\Phi)\).

Halpern, Moses, and Waarts [12] gave a different method, using continual common knowledge, to take an N-solution to *Eventual Byzantine Agreement* and construct from it an N-dominating N-solution; their technique can easily be extended to apply to coordination problems that do not require termination. It cannot, however, be applied to problems requiring termination (even by using extended common knowledge in place of continual common knowledge). Informally, this is because their techniques rely on negative introspection properties of continual common knowledge that are not possessed by extended common knowledge. Thus, the techniques presented here are more general because they can be applied to problems requiring termination. In addition, they are simpler in the following sense: the dominating protocol is constructed by replacing only one action predicate (the one for the action whose performance is to be improved); the others are simply augmented by adding a conjunct to the already existing predicate. This allows the new protocol to rely on the correctness of the original one.

Theorem 10 can be used to show that the generated protocol \(P(\Psi)\) performs the chosen action \(a_j\) as quickly as any protocol that dominates the original \(P(\Phi)\). Theorem 13 in the following section shows how Lemma 11 can be used iteratively to generate optimal protocols.

## 9 Optimal Protocols

This section provides a precise characterization of optimal protocols for coordination and a method by which any protocol can be converted into an optimal one. As in Section 6, this section concentrates on full-information protocols, in which a processor sends its local state (as a message) in each round and then sets its local state to the vector of messages it receives. If there is an optimal protocol that dominates a given protocol, then there is an optimal full-information protocol that does so also.

Theorem 12 gives the necessary and sufficient conditions for a full-information protocol to be optimal. These conditions are closely related to the dominating conditions established in Theorem 10.

**Theorem 12:** Let \(C\) be a coordination problem and let \(F(\Phi)\) be a full-information decision protocol.

- If \(F(\Phi)\) \(N\)-satisfies \(C\), then \(F(\Phi)\) is \(N\)-optimal for \(C\) if and only if \(\mathcal{R}_F \models \Phi_{i,p} \iff B_p^N(Ok_i \land \mathcal{W}_{N^-}^{\mathcal{P}_j} \neg \Phi_{j,p})\).

- If \(F(\Phi)\) \(C\)-satisfies \(C\), then \(F(\Phi)\) is \(C\)-optimal for \(C\) if and only if \(\mathcal{R}_F \models \Phi_{i,p} \iff K_p(Ok_i \land \mathcal{S}_{N^-}^{\mathcal{P}_j} \neg \Phi_{j,p})\).

- If \(F(\Phi)\) \(NT\)-satisfies \(C\), then \(F(\Phi)\) is \(NT\)-optimal for \(C\) if and only if \(\mathcal{R}_F \models \Phi_{i,p} \iff B_p^N \mathcal{W}_{N^-}^{\mathcal{P}_j} Ok_i \land \land_{j \neq i} \neg \Phi_{j,p}\).
• If $F(\Phi)$ CT-satisfies $C$, then $F(\Phi)$ is CT-optimal for $C$ if and only if $\mathcal{R}_F \models \Phi_{i,p} \iff K_p S_{\mathcal{N},p}^{-} o_{k_i} \land \bigwedge_{j \neq i} \neg \Phi_{j,p}$.

Proof: We prove only the “if” direction for N-optimality and the “only if” direction for CT-optimality. The remaining cases are similar.

Consider first the “if” direction for N-optimality. Assume that

$$\mathcal{R}_F \models \Phi_{i,p} \iff B_{p}^N (o_{k_i} \land \bigwedge_{j \neq i} \neg \Phi_{j,p}) \land \bigwedge_{j \neq i} \neg \Phi_{j,p}.$$  

We must show that $F(\Phi)$ dominates any N-solution to $C$ that N-dominates $F(\Phi)$. Let $F(\Psi)$ be such a protocol; the proof must show that $F(\Phi)$ N-dominates $F(\Psi)$. Suppose that $(\mathcal{R}_F, r, l) \models \Psi_{i,p}$. If $(\mathcal{R}_F, r, l) \models \bigvee_{j \neq i} \Phi_{j,p}$, the proof is complete; assume instead that $(\mathcal{R}_F, r, l) \models \bigwedge_{j \neq i} \neg \Phi_{j,p}$. By Theorem 3, $(\mathcal{R}_F, r, l) \models B_{p}^N o_{k_i}$. By Theorem 10, $(\mathcal{R}_F, r, l) \models B_{p}^N \bigwedge_{j \neq i} o_{k_i}$. Thus, $(\mathcal{R}_F, r, l) \models \Phi_{i,p}$ and $F(\Phi)$ is optimal.

Now consider the “only if” direction for CT-optimality. Assume that $F(\Phi)$ is CT-optimal for $C$; the proof must show that $\mathcal{R}_F \models \Phi_{i,p} \iff K_p S_{\mathcal{N},p}^{-} o_{k_i} \land \bigwedge_{j \neq i} \neg \Phi_{j,p}$. Suppose first that $(\mathcal{R}_F, r, l) \models \Phi_{i,p}$. Since $F(\Phi)$ is a decision protocol, $(\mathcal{R}_F, r, l) \models \bigwedge_{j \neq i} \neg \Phi_{j,p}$. Because $F(\Phi)$ C-dominates itself, Theorem 10 implies that $(\mathcal{R}_F, r, l) \models K_p S_{\mathcal{N},p}^{-} o_{k_i}$, giving the desired implication. Finally, suppose that $(\mathcal{R}_F, r, l) \models K_p S_{\mathcal{N},p}^{-} o_{k_i} \land \bigwedge_{j \neq i} \neg \Phi_{j,p}$. Let $F(\Psi)$ be the C-dominating protocol constructed from $F(\Phi)$ using the CT-case of Lemma 11 (using $j = i$). Note that $\Psi_{i,p} \equiv K_p S_{\mathcal{N},p}^{-} o_{k_i}$, so $(\mathcal{R}_F, r, l) \models \Psi_{i,p}$. Since $F(\Phi)$ is optimal, it C-dominates $F(\Psi)$, so $(\mathcal{R}_F, r, l) \models \Phi_{j,p}$ for some $j \in C$. Since all other actions are already excluded, it must be that $(\mathcal{R}_F, r, l) \models \Phi_{i,p}$, completing the proof.

The first case above is similar to a result observed earlier for Eventual Byzantine Agreement [12]. This characterization of optimal protocols, although precise, is somewhat lacking in that it does not indicate how to go about constructing such a protocol. This is in marked contrast to work on simultaneous coordination [4,13,15]. That work first exhibited the knowledge needed to achieve such coordination and then used it directly to construct optimum solutions.

To develop optimal protocols for nonsimultaneous coordination, one can iteratively apply Lemma 11 to some initial protocol. The idea is that each application of the lemma improves the performance of a particular action. After all actions have been improved, the result is an optimal protocol.

Theorem 13: Let $C$ be a coordination problem and let $\langle \prec \rangle$ be some total order of the actions in $C$ ($a_1 \prec a_2 \prec \cdots \prec a_m$). Let $F(\Phi)$ be a full-information decision protocol. If $F(\Phi)$ X-satisfies $C$ (where $X$ is either $N$, $C$, $NT$, or $CT$), then inductively define $F(\Phi^i)$ ($0 \leq i \leq m$) as follows: $F(\Phi^0)$ is $F(\Phi)$ and $F(\Phi^{i+1})$ is the Y-dominating protocol constructed from $F(\Phi^i)$ using the X-case of Lemma 11, using $j = i + 1$ (where $Y$ is $N$ if $X$ is $N$ or $NT$ and $C$ if $X$ is $C$ or $CT$). Then $F(\Phi^m)$ is X-optimal for $C$ and Y-dominates $F(\Phi)$.

Proof: The proof considers the case of CT-optimality; the others are similar. It follows by Lemma 11 that, for all $i$ ($1 \leq i \leq m$), $F(\Phi^i)$ CT-satisfies $C$ and C-dominates all $F(\Phi^j)$ with $j \leq i$. It remains only to show that $F(\Phi^m)$ is CT-optimal; this is done by showing that it C-dominates any protocol $F(\Psi)$ that CT-satisfies $C$ and C-dominates $F(\Phi^m)$. To show that $F(\Phi^m)$ C-dominates $F(\Phi)$, assume that $(\mathcal{R}_F, r, l) \models \Psi_{i,p}$. Since $F(\Psi)$ C-dominates $F(\Phi^m)$, it also C-dominates $F(\Phi^{i-1})$. Thus, by Theorem 10, $(\mathcal{R}_F, r, l) \models K_p S_{\mathcal{N},p}^{-} o_{k_i}$, where $\mathcal{P}_i$ is
Using Knowledge to Optimally Achieve Coordination in Distributed Systems

Based on $\Phi^{i-1}$. But this is precisely $\Phi_{i,p}^i$, so $F(\Phi^i)$ has $p$ perform $a_i$ at $(r,i)$. Since $F(\Phi^m)$ C-dominates $F(\Phi^i)$, it also has $p$ perform some action at $(r,i)$. Thus, $F(\Phi^m)$ C-dominates $F(\Psi)$ and is optimal.

Theorem 13 shows how any protocol can be converted into an optimal protocol. In particular, it can be applied to a degenerate protocol $F(\Phi)$ that performs no actions (i.e., with $\Phi_{i,p} = false$ for all $i$ and $p$). The first application of Lemma 11 results then in the following action function (for CT-satisfaction):

$$\Phi_{i,p}^1 \equiv \begin{cases} 
K_p S_N^\psi, p_1 ok_1 & \text{if } i = 1 \\
false & \text{if } i > 1
\end{cases}$$

where $P_1$ is based on $\Phi$ and is thus empty. Thus, $\Phi_{i,p}^1$ simplifies to $K_p S_N^\psi, ok_1$. Thus, by Theorem 5, $F(\Phi^1)$ performs $a_1$ as early as any protocol can. The second application results in the following:

$$\Phi_{i,p}^2 \equiv \begin{cases} 
K_p (S_N^\psi, p_1 \land S_N^\psi, p_2 \lor ok_2) & \text{if } i = 1 \\
K_p S_N^\psi, p_2 \lor ok_2 & \text{if } i = 2 \\
false & \text{if } i > 2
\end{cases}$$

where $P_2$ is based on $\Phi^1$ and is thus $\{q \in P \mid K_p S_N^\psi, ok_1\}$. If there are only two actions in $C$ (as in the case of Eventual Byzantine Agreement), then $F(\Phi^2)$ is optimal. Otherwise, Lemma 11 can be applied as many times as necessary.

10 Discussion and Conclusions

This paper considered four different types of coordination problems. For each problem, we determined the knowledge necessary to perform an action and used this to characterize the domination relationship between different solutions and to develop and characterize optimum and optimal solutions. In the past, researchers have used simple common knowledge to study simultaneous coordination [4,13,15]. When the simultaneity restriction is relaxed, weaker (but less intuitive) variants of common knowledge become more appropriate. These variants are the fixed points of certain knowledge operators. The operators used depended on the type of coordination desired:

- consistent coordination requires knowledge, whereas nonconsistent coordination requires only belief;
- the agreement condition of coordination requires continual knowledge to ensure that there is never a disagreement; and
- the termination condition of coordination requires eventual knowledge to ensure that all nonfaulty processors eventually decide.

A major contribution of this paper is the definition of extended common knowledge, which combines the continual and eventual knowledge needed for coordination problems with termination.

Necessary and sufficient conditions were given for the existence of an optimum solution to a problem in a given system; furthermore, a knowledge-based specification of such solutions
was given for cases in which the conditions were met. These conditions depended on the type of coordination desired. While some problems have optimum solutions regardless of the type of coordination required, it seems likely that the type of coordination will be important in some cases. Furthermore, it is quite possible that, for some problems, the existence of an optimum solution may depend also on the type and number of failures that can occur or on the synchrony of the communication network. In the future, we plan to further study these conditions to provide, when possible, a simpler characterization of coordination problems with optimum solutions.

Some of the optimum solutions given require action when some fact becomes eventual common knowledge. A better understanding of the semantics of this knowledge would facilitate the implementation of such protocols and an understanding of their complexity (Moses and Tuttle [13] analyze the complexity of computing simple common knowledge). Tuttle [17] gives a characterization of the semantics of eventual common knowledge based on game theory. In a separate paper [1], we study the relationship between eventual common knowledge and distributed knowledge [5,11]. For the purposes of this paper, distributed knowledge of a fact about the input is equivalent to weak eventual common knowledge of the same fact. Because it is easier to reason about distributed knowledge than eventual common knowledge, we can use this equivalence to simplify the implementation and analysis of some of the protocols discussed here. For example, we show that, in systems with general omission failures, testing for distributed knowledge is NP-hard.

We also consider cases in which the necessary knowledge is impossible to attain. Neiger and Tuttle [15] showed that strong common knowledge cannot be achieved in synchronous distributed systems with general omission failures in which \( n \), the number of processors, is less than equal to \( 2t \), where \( t \) is the maximum number of faulty processors. This shows that consistent simultaneous coordination cannot be achieved in these systems. We show that strong eventual common knowledge (of facts about the input) cannot be attained in these same systems, indicating that consistent nonsimultaneous coordination cannot be achieved in these systems. We also show that strong eventual common knowledge cannot be achieved in asynchronous systems with send omission failures in which \( n \leq 2t \), indicating again that consistent coordination cannot be achieved in such cases. In the future, we plan to study the systems in which the various forms of eventual common knowledge can be achieved so as to better understand when different forms of coordination are possible.

Our development of optimal protocols uses extended common knowledge. Implementation of these protocols will depend on gaining a better understanding of this new form of knowledge. It is possible that the semantics of extended common knowledge can be understood by combining the game-theoretic characterization of eventual common knowledge [17] with the graph-theoretic characterization of continual common knowledge [12]. Just as Moses and Tuttle [13] showed how a graph-theoretic characterization of common knowledge could be used to implement and analyze the complexity of simultaneous coordination protocols, a better understanding of extended common knowledge might be applied to the more general form of coordination considered here.

It should be noted that the results in this paper apply to systems with both synchronous and asynchronous communication. In the past, most papers involving knowledge and coordination have concentrated on systems with synchronous communication. Because we consider a new form of termination that is weaker, but more appropriate to asynchronous systems, than the one used earlier, our analysis applies to these more practical systems. For example, the results of this paper can be applied to the protocols developed by Gopal and Toueg [9] for coordination in asynchronous systems.
It is reasonable to ask why Halpern, Moses, and Waarts [12], who considered a problem that does require termination (Eventual Byzantine Agreement), used continual, instead of extended, common knowledge to construct optimal protocols. Actually, many of their results were developed for a weaker coordination problem (not requiring termination) and then applied to solutions to Eventual Byzantine Agreement (where processors terminate in every run). The termination property of the original solution to which their methods were applied is “inherited” by the resulting optimal protocol. Because the definition of termination considered in this paper is, in a sense, conditional and not absolute, it cannot be inherited in this way; extended common knowledge is needed in order to retain it.

Acknowledgements

The authors would like to thank Joseph Y. Halpern, Yoram Moses, Sam Toueg, Mark R. Tuttle, and Orli Waarts for discussions and comments that helped in the development of this work.

References


