Using Knowledge to Optimally Achieve Coordination in Distributed Systems* (Extended Abstract)

Gil Neiger
Rida Bazzi†
College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0280

Abstract

Fundamental to many disciplines is the problem of coordinating the actions of a group of independent agents. Researchers in distributed computing systems have long endeavored to find efficient solutions to a variety of problems involving coordination among the processors in such a system. Recently, processor *knowledge* has been used to characterize such solution and to derive more efficient ones. Most 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.

1 Introduction

The problem of coordinating the activity of a set of independent agents is fundamental to many disciplines. In such problems, the agents are required 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. The purpose of this work is to explore the relationship between knowledge and coordination and use this to derive efficient solutions to coordination problems.

This work specifically considers *fault-tolerant* coordination in a distributed computing system. The agents are individual processors that communicate via a communication network. 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 within computer science 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 [17] considered the more difficult class of *consistent* simultaneous coordination problems and showed that, in general, their solutions require a stronger form of common

*This work was supported in part by the National Science Foundation under grants CCR-8909663 and CCR-9106627.
†This author was supported in part by a scholarship from the Hariri Foundation.

knowledge. This suggested that variations in the type of coordination desired may result in corresponding variations in the type of knowledge required. These three papers 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 was 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. For these reasons, researchers have begun to consider the relationship between knowledge and *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. These are protocols that are not outperformed by any other solution. Neiger [14] extended these results to the more difficult class of consistent coordination problems and produced similar results (consequently requiring a stronger form of continual common knowledge).

The previous papers on nonsimultaneous coordination considered protocols that guarantee that processor choices were correct and in agreement. 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 explicitly reason about satisfaction this termination requirement, one needs to consider *eventual common knowledge* [11,19].

This paper considers four types of coordination problems: consistent and nonconsistent problems that require termination and those that do not. For each type, 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. Recall that an optimum solution is a protocol that matches or outperforms all other solutions. Although some coordination problems have no optimum solution (e.g., *Eventual Byzantine Agreement*), such solutions do exist for some 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 [4,11,12,13,17].

A distributed system consists of a finite set $\mathcal{P}$ 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.[2] Computation proceeds in a sequence of *rounds*, with round r taking places between time $r-1$ and time r. At time 0, each

[1] These papers referred to the protocols they developed as *optimal*. As will be seen below, there is an important distinction between optimum and optimum protocols.

[2] Although this paper considers both synchronous and asynchronous systems, it is reasonable to assume that processors share a global clock. It has been shown [15,20] that synchronous processors can be simulated in systems in which processors and message passing are asynchronous.

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. At any given time, a processor's *message history* consists of the list of messages it has sent and received to and from the other processors.

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. 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 $r_p(l)$.

The *delivery pattern* of a run specifies, for every message sent in the run, whether or not it is delivered and, if it is, the round in which this occurs. The class of allowable delivery patterns is determined by assumptions made about the communication network. For example, the system may support *synchronous* message passing (in which messages are delivered in the round in which they are sent) or *totally asynchronous* message passing (in which there is no bound on the number of rounds required for delivery). It may be reliable (all messages are delivered) or lossy.

Processors follow a *communication protocol* P, which is a function that, given a processor's local state at the beginning of a round, specifies the messages a processor is required to send in that round. While nonfaulty processors send all messages required by the protocol and receive all messages delivered to them, some processors may be *faulty*. Such a processor may omit to send any of the messages required by its protocol or may fail to receive some of the messages delivered to it.[3] We assume that, of the n processors, at most t ($t < n$) can be faulty in any execution. The *faulty behavior* of a processor specifies, for every round, the processors to which that processor fails to send messages and the delivered messages that the processor fails to receive. A *failure pattern* is a collection of faulty behaviors, one for each processor. 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). The results presented in this paper apply to any *benign* failure model; specifically, they apply to crash (stopping) failures, send-omission failures, and general omission failures [16]. Given a run r, let $\mathcal{N}(r)$ refer to the set of processors that are *nonfaulty* in r. This set is constant throughout a run.

In this paper, it is important to be able to compare two different protocols that solve the same problem. This is done by comparing their performance under the same conditions; these conditions are the initial states of the processors and the way that failures and message deliveries occur. Formally, an *operating environment* is a triple $\langle \mathrm{I}, \mathrm{D}, \mathrm{F} \rangle$, where I is a vector of initial states ($\mathrm{I}[p]$ is that of p), D is a delivery pattern, and F is a failure pattern. An operating environment is *consistent* with a run if it matches the initial states and message deliveries of that run. For example, consider an operating environment that indicates that p does not fail to send a message to q in round l. If, in run r, p tries but fails to send a message to q in round l, then the operating environment is *not* consistent with r. On the other hand, if p did not attempt to send a message to q in round l of run r' (because its protocol called for no message to be sent), then the operating environment (as described) is consistent with r'. The system being run limits the operating environments that are consistent with runs of the system; for example, no run of a system with crash failures will be consistent with an operating environment in which a processor sends two different messages in a round. Two runs of two different communication protocols are *corresponding runs* if there is some operating environment that is consistent with both of them. 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 φ 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 φ is *valid in system* $\mathcal{R}$, denoted $\mathcal{R} \models \varphi$, if it is true at all points in

[3]This paper does not consider processors that may fail *arbitrarily* [18]. Neiger and Toueg [16] give formal definitions of a range of different failure models.

$\mathcal{R}$. 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 φ is a *fact about X* if fixing X determines the truth (or falsity) of φ.

3 Coordination Problems

This section defines four classes of coordination problems using the model given in the previous section. A *coordination problem* is a finite set of actions $\mathcal{C} = \{a_1, \ldots, a_m\}$. Each action has associated with it an *enabling condition* ok_i, which 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.

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* Φ. For each $a_i \in \mathcal{C}$ and $p \in \mathcal{P}$, $\Phi_{i,p}$ is a fact 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 a processor's choice is irrevocable and unique.

There are several 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 are called *nonconsistent* and *consistent*, respectively. Formally, $P(\Phi)$ *nonconsistently satisfies* $\mathcal{C}$ (or *N-satisfies* $\mathcal{C}$) if the following conditions hold:

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

$P(\Phi)$ *consistently satisfies* $\mathcal{C}$ (or *C-satisfies* $\mathcal{C}$) if the above conditions hold with the word "nonfaulty" and the conjuncts "$p \in \mathcal{N}$" and "$q \in \mathcal{N}$" deleted. The earlier literature on knowledge and coordination concentrated on nonconsistent coordination [12,13]; more recently, researchers have begun to consider consistent coordination [14,17].

In some cases, it is desirable to add a *termination condition* that specifies when nonfaulty processors *must* perform an action (it is assumed that, 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. Such problems cannot, however, be solved in systems with asynchronous communication [8]. For that reason, this paper considers a weaker termination condition, which requires nonfaulty processors to act only if some other processor does. Formally, $P(\Phi)$ *nonconsistently satisfies* $\mathcal{C}$ *with termination* (or *NT-satisfies* $\mathcal{C}$) if it N-satisfies $\mathcal{C}$ and the following condition holds:

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

$P(\Phi)$ *consistently satisfies* $\mathcal{C}$ *with termination* (or *CT-satisfies* $\mathcal{C}$) if it C-satisfies $\mathcal{C}$ and the above condition holds with the first word "nonfaulty" and the conjunct "$p \in \mathcal{N}$" deleted.

Halpern, Moses, and Waarts compared solutions to *Eventual Byzantine Agreement* by comparing their behavior in corresponding runs. Their method is adapted here. Suppose that decision protocols $P_1(\Phi_1)$ and $P_2(\Phi_2)$ both coordinate some problem $\mathcal{C}$. $P_1(\Phi_1)$ *dominates* $P_2(\Phi_2)$ if, in every pair of corresponding runs of the two protocols, $P_2(\Phi_2)$ has no processor perform an action earlier than $P_1(\Phi_1)$. Notice that the "dominates" relation is a partial order on the space of 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 $P_2(\Phi_2)$ may outperform $P_1(\Phi_1)$ in another.

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

4 Definitions of Knowledge

The analysis in this paper depends on a processor's knowledge at different points in an execution. This section defines such a notion of knowledge. The treatment here is an adaptation of others [4,11,12,13,17].

This section gives a way to express processor knowledge by augmenting the logical language introduced in Section 2. 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 $\wedge$ and $\neg$.

Processor knowledge was first defined by Halpern and Moses [11] in the following way. *Processor p knows φ at point $\langle r,l\rangle$ in system $\mathcal{R}$*, denoted $(\mathcal{R},r,l)\models \mathsf{K}_p\varphi$, if $(\mathcal{R},r',l')\models\varphi$ for all runs $\langle r',l'\rangle$ in $\mathcal{R}$ 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. Most interesting are 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 $\mathcal{S}$ is an indexical set, then $\mathcal{S}(r,l)$ refers to the contents of the set at point $\langle r,l\rangle$. Examples of indexical sets include the set of nonfaulty processors $\mathcal{N}$ and sets of processors that know certain facts.[4] It is often useful to condition a processor's knowledge on the processor's membership in a specific set. We say that *processor p believes φ conditional on $\mathcal{S}$* if p knows that, if it is in $\mathcal{S}$, φ is true. That is, $\mathsf{B}_p^{\mathcal{S}}\varphi\equiv\mathsf{K}_p(p\in\mathcal{S}\Rightarrow\varphi)$. It is easy to see that $(\mathcal{R},r,l)\models\mathsf{B}_p^{\mathcal{S}}\varphi$ if $(\mathcal{R},r',l)\models\varphi$ for all runs r such that $r'_p(l)=r_p(l)$ and $p\in\mathcal{S}(r',l)$. Processor knowledge, using the K_p operators will be used to define strong notions of group knowledge, while processor belief, using $\mathsf{B}_p^{\mathcal{S}}$, will be used to define weaker notions.

Researchers studying simultaneous coordination problems [4,13,17] have found that *common knowledge* is necessary for the solution of such problems. Informally, a fact φ is common knowledge to $\mathcal{S}$ if everyone in $\mathcal{S}$ knows φ, everyone knows that everyone knows φ, and so on. The following is a brief overview of a more formal definition, based logical fixed points. *Everyone in indexical set $\mathcal{S}$ knows φ*, denoted $\mathsf{E}_{\mathcal{S}}\varphi$, is defined to be $\bigwedge_{p\in\mathcal{S}}\mathsf{K}_p\varphi$. *All processors in $\mathcal{S}$ believe φ*, denoted $\mathsf{A}_{\mathcal{S}}\varphi$, is equivalent to $\bigwedge_{p\in\mathcal{S}}\mathsf{B}_p^{\mathcal{S}}\varphi$. Based on this, two forms of common knowledge are defined, a strong one based on knowledge and a weak one based on belief. *Strong common knowledge of fact φ by set $\mathcal{S}$*, denoted $\mathsf{S}_{\mathcal{S}}\varphi$, is the greatest fixed point of the equivalence $X\Leftrightarrow\mathsf{E}_{\mathcal{S}}(\varphi\wedge X)$. *Weak common knowledge of fact φ by set $\mathcal{S}$*, denoted $\mathsf{W}_{\mathcal{S}}\varphi$, is the greatest fixed point of the equivalence $X\Leftrightarrow\mathsf{A}_{\mathcal{S}}(\varphi\wedge X)$. $\mathsf{S}_{\mathcal{S}}\varphi$ is equivalent to the infinite conjunction $\bigwedge_{i\geq 1}\mathsf{E}_{\mathcal{S}}^i\varphi$, while $\mathsf{W}_{\mathcal{S}}\varphi$ is equivalent to $\bigwedge_{i\geq 1}\mathsf{A}_{\mathcal{S}}^i\varphi$. Neiger and Tuttle [17] showed that strong common knowledge was necessary for the solution of consistent simultaneous coordination, while Moses and Tuttle [13] observed that achieving weak common knowledge was sufficient to achieve nonconsistent 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 nonconsistent coordination problems, respectively.

Eventual common knowledge [11,19] relaxes the simultaneity that is inherent in true common knowledge. For this reason, it is more 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

[4] 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 $\langle r,l\rangle$, but it should be understood then that $\mathcal{N}(r,l)=\mathcal{N}(r,l')$ for all l and l'.

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 a coordination problem. The definition of eventual knowledge uses the temporal operator *eventually* $\Diamond$. $(\mathcal{R}, r, l) \models \Diamond\varphi$ if and only if $(\mathcal{R}, r, l') \models \varphi$ for some $l' \geq l$. Eventual common knowledge is also defined using fixed points. *Strong eventual common knowledge of fact* φ *by set* $\mathcal{S}$, denoted $\mathsf{S}_{\mathcal{S}}^{\Diamond}\varphi$, is the greatest fixed point of the equivalence $X \Leftrightarrow \Diamond\mathsf{E}_{\mathcal{S}}(\varphi \wedge X)$.[5] *Weak eventual common knowledge of fact* φ *by set* $\mathcal{S}$, denoted $\mathsf{W}_{\mathcal{S}}^{\Diamond}\varphi$, is the greatest fixed point of the equivalence $X \Leftrightarrow \Diamond\mathsf{A}_{\mathcal{S}}(\varphi \wedge X)$. For the cases considered in this paper, $\mathsf{S}_{\mathcal{S}}^{\Diamond}\varphi$ implies (but is not equivalent to) the infinite conjunction $\bigwedge_{i\geq 1}(\Diamond\mathsf{E}_{\mathcal{S}})^i\varphi$. A similar statement is true for weak eventual common knowledge. One should note that eventual common knowledge is *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 positive introspection; if a fact is eventual common knowledge to a set, then all members of the set eventually know (or believe) this. That is, $\mathsf{S}_{\mathcal{S}}^{\Diamond}\varphi \Rightarrow \Diamond\mathsf{E}_{\mathcal{S}}\mathsf{S}_{\mathcal{S}}^{\Diamond}\varphi$ and $\mathsf{W}_{\mathcal{S}}^{\Diamond}\varphi \Rightarrow \Diamond\mathsf{A}_{\mathcal{S}}\mathsf{W}_{\mathcal{S}}^{\Diamond}\varphi$ are valid. Each form of eventual common knowledge satisfies an induction rule that can be used to show that certain facts are eventual common knowledge:

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

This paper considers cases in which facts φ about runs (specifically, the enabling conditions of a coordination problem) become eventual common knowledge to the set $\mathcal{N}$ of nonfaulty processors. Because this set is assumed to never be empty, it is not hard to see that, in these cases, $\mathsf{S}_{\mathcal{N}}^{\Diamond}\varphi \Rightarrow \varphi$ and $\mathsf{W}_{\mathcal{N}}^{\Diamond}\varphi \Rightarrow \varphi$ are valid.

Although eventual common knowledge is necessary for *termination*, Halpern, Moses, and Waarts [12] showed that it cannot be used to achieve the *agreement* needed to solve a coordination problem. Intuitively, the reason for this is that, unlike true 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* $\Box$. $(\mathcal{R}, r, l) \models \Box\varphi$ 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* φ *by set* $\mathcal{S}$, denoted $\mathsf{S}_{\mathcal{S}}^{\Box}\varphi$, is the greatest fixed point of the equivalence $X \Leftrightarrow \Box\mathsf{E}_{\mathcal{S}}(\varphi \wedge X)$. *Weak continual common knowledge of fact* φ *by set* $\mathcal{S}$, denoted $\mathsf{W}_{\mathcal{S}}^{\Box}\varphi$ is the greatest fixed point of the equivalence $X \Leftrightarrow \Box\mathsf{E}_{\mathcal{S}}(\varphi \wedge X)$. As defined here, $\mathsf{S}_{\mathcal{S}}^{\Box}\varphi$ is equivalent to the infinite conjunction $\bigwedge_{i\geq 1}(\Box\mathsf{E}_{\mathcal{S}})^i\varphi$ (and similarly for weak continual common knowledge). That is, a fact φ is continual common knowledge to a set if it is always the case that everyone in the set knows φ, it is always the case that everyone in the set knows that it is always the case that everyone in the set knows φ, 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.* Neiger [14] explored the use of strong continual common knowledge in the solutions of consistent coordination problems. Continual common knowledge, as used in these papers, is *not* stronger than the simple common knowledge used in the papers on simultaneous coordination. The reason for this is the subtle definition of the set to which the knowledge is ascribed.

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: $\mathsf{S}_{\mathcal{S}}^{\Box}\varphi \Rightarrow \Box\mathsf{E}_{\mathcal{S}}(\varphi \wedge \mathsf{S}_{\mathcal{S}}^{\Box}\varphi)$ and $\mathsf{W}_{\mathcal{S}}^{\Box}\varphi \Rightarrow \Box\mathsf{A}_{\mathcal{S}}(\varphi \wedge \mathsf{W}_{\mathcal{S}}^{\Box}\varphi)$ are both valid. Both satisfy a kind of negative introspection, in that the following implications are valid: $\neg\mathsf{S}_{\mathcal{S}}^{\Box}\varphi \wedge \mathsf{K}_p(p \in \mathcal{S}) \Rightarrow \mathsf{K}_p\neg\mathsf{S}_{\mathcal{S}}^{\Box}\varphi$ and $\neg\mathsf{W}_{\mathcal{S}}^{\Box}\varphi \wedge (p \in \mathcal{S}) \Rightarrow \mathsf{B}_p^{\mathcal{S}}\neg\mathsf{W}_{\mathcal{S}}^{\Box}\varphi$. Each form of continual common knowledge satisfies an induction rule:

[5] This is slightly different from the original definition of Halpern and Moses [11]. They defined eventual common knowledge to be the greatest fixed point of the equivalence $X \Leftrightarrow \bigwedge_{p\in\mathcal{S}} \Diamond\mathsf{K}_p(\varphi \wedge X)$. For all cases considered in this paper, this definition is equivalent to that given here for strong eventual common knowledge.

- If $\varphi \Rightarrow \Box\mathsf{E}_S(\varphi \wedge \psi)$ is valid in a system, then $\varphi \Rightarrow \mathsf{S}_S^{\Box}\psi$ is also valid in that system.
- If $\varphi \Rightarrow \Box\mathsf{A}_S(\varphi \wedge \psi)$ is valid in a system, then $\varphi \Rightarrow \mathsf{W}_S^{\Box}\psi$ is also valid in that system.

Finally, continual common knowledge is *continual* in that, if it is true at any point in a run, it is true at every point in that run. Thus, $\mathsf{S}_S^{\Box}\varphi \Leftrightarrow \Box\mathsf{S}_S^{\Box}\varphi$ and $\mathsf{W}_S^{\Box}\varphi \Leftrightarrow \Box\mathsf{W}_S^{\Box}\varphi$ are both valid.

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 will 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 1: *Let C be a coordination problem.*

1. *If $P(\Phi)$ N-satisfies C, then $\mathcal{R}_P \models \Phi_{i,p} \Rightarrow \mathsf{B}_p^{\mathcal{N}} ok_i$.*
2. *If $P(\Phi)$ C-satisfies C, then $\mathcal{R}_P \models \Phi_{i,p} \Rightarrow \mathsf{K}_p ok_i$.*

For problems requiring termination, a processor must ensure, before taking an action, that all nonfaulty processors will eventually perform the same action. Each of these processors must in turn know the same thing. This relationship indicates that eventual common knowledge is necessary for problems requiring termination:

Theorem 2: *Let C be a coordination problem.*

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

Theorems 1 and 2 suggest a family of very simple coordination protocols. Each endeavors 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 optimum protocols.

Theorem 3: *Let C be a coordination problem. For each $a_i \in C$, consider an action function Φ^i defined so that $\Phi^i_{j,p} \equiv false$ for all $j \neq i$ and $p \in \mathcal{P}$.*

1. *$P(\Phi^i)$ N-satisfies C if $\Phi^i_{i,p} \equiv \mathsf{B}_p^{\mathcal{N}} ok_i$;*
2. *$P(\Phi^i)$ C-satisfies C if $\Phi^i_{i,p} \equiv \mathsf{K}_p ok_i$;*
3. *$P(\Phi^i)$ NT-satisfies C if $\Phi^i_{i,p} \equiv \mathsf{B}_p^{\mathcal{N}}\mathsf{W}_{\mathcal{N}}^{\Diamond} ok_i$; and*
4. *$P(\Phi^i)$ CT-satisfies C if $\Phi^i_{i,p} \equiv \mathsf{K}_p\mathsf{S}_{\mathcal{N}}^{\Diamond} ok_i$.*

6 Optimum Protocols

Moses and Tuttle [13] showed that there exists no 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 1 and 2 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 3 (an optimum protocol would dominate all of these).

As has been noted elsewhere [2,4,13,17], 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 4 shows that the conditions shown necessary in Theorems 1 and 2 must be continual common knowledge whenever any action is taken by an *optimum* full-information protocol. In the statement of this lemma, φ_i indicates that some (nonfaulty) processor is performing action a_i. The sets $\mathcal{S}_i$ to which the continual common knowledge is ascribed contain all processors that have the minimum knowledge necessary to perform some *other* action.

Lemma 4: *Let $\mathcal{C}$ be a coordination problem.*

1. *If $F(\Phi)$ is N-optimum for $\mathcal{C}$, then $\mathcal{R}_F \models \varphi_i \Rightarrow \mathsf{W}^{\boxdot}_{\mathcal{S}_i} ok_i$, where $\varphi_i \equiv \bigvee_{p\in\mathcal{N}} \Phi_{i,p}$ and $\mathcal{S}_i = \{p \in \mathcal{N} \mid \exists a_j \in \mathcal{C}[j \neq i \wedge \mathsf{B}^{\mathcal{N}}_p ok_j]\}$.*
2. *If $F(\Phi)$ is C-optimum for $\mathcal{C}$, then $\mathcal{R}_F \models \varphi_i \Rightarrow \mathsf{S}^{\boxdot}_{\mathcal{S}_i} ok_i$, where $\varphi_i \equiv \bigvee_{p\in\mathcal{P}} \Phi_{i,p}$ and $\mathcal{S}_i = \{p \in \mathcal{P} \mid \exists a_j \in \mathcal{C}[j \neq i \wedge \mathsf{K}_p ok_j]\}$.*
3. *If $F(\Phi)$ is NT-optimum for $\mathcal{C}$, then $\mathcal{R}_F \models \varphi_i \Rightarrow \mathsf{W}^{\boxdot}_{\mathcal{S}_i}\mathsf{W}^{\diamond}_{\mathcal{N}} ok_i$, where $\varphi_i \equiv \bigvee_{p\in\mathcal{N}} \Phi_{i,p}$ and $\mathcal{S}_i = \{p \in \mathcal{N} \mid \exists a_j \in \mathcal{C}[j \neq i \wedge \mathsf{B}^{\mathcal{N}}_p \mathsf{W}^{\diamond}_{\mathcal{N}} ok_j]\}$.*
4. *If $F(\Phi)$ is CT-optimum for $\mathcal{C}$, then $\mathcal{R}_F \models \varphi_i \Rightarrow \mathsf{S}^{\boxdot}_{\mathcal{S}_i}\mathsf{S}^{\diamond}_{\mathcal{N}} ok_i$, where $\varphi_i \equiv \bigvee_{p\in\mathcal{P}} \Phi_{i,p}$ and $\mathcal{S}_i = \{p \in \mathcal{P} \mid \exists a_j \in \mathcal{C}[j \neq i \wedge \mathsf{K}_p \mathsf{S}^{\diamond}_{\mathcal{N}} ok_j]\}$.*

Lemma 4 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 5 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 4 also holds.

Theorem 5: *Let $\mathcal{C}$ be a coordination problem.*

1. *There is an N-optimum protocol for $\mathcal{C}$ if and only if $\mathcal{R}_F \models \mathsf{B}^{\mathcal{N}}_p ok_i \Rightarrow \bigvee_{a_j\in\mathcal{C}} \mathsf{B}^{\mathcal{N}}_p \mathsf{W}^{\boxdot}_{\mathcal{S}_j} ok_j$, where $\mathcal{S}_j = \{q \in \mathcal{N} \mid \exists k \in \mathcal{C}[k \neq j \wedge \mathsf{B}^{\mathcal{N}}_q ok_k]\}$.*
2. *There is a C-optimum protocol for $\mathcal{C}$ if and only if $\mathcal{R}_F \models \mathsf{K}_p ok_i \Rightarrow \bigvee_{a_j\in\mathcal{C}} \mathsf{K}_p \mathsf{S}^{\boxdot}_{\mathcal{S}_j} ok_j$, where $\mathcal{S}_j = \{q \in \mathcal{P} \mid \exists k \in \mathcal{C}[k \neq j \wedge \mathsf{K}_q ok_k]\}$.*
3. *There is an NT-optimum protocol for $\mathcal{C}$ if and only if $\mathcal{R}_F \models \mathsf{B}^{\mathcal{N}}_p \mathsf{W}^{\diamond}_{\mathcal{N}} ok_i \Rightarrow \bigvee_{a_j\in\mathcal{C}} \mathsf{B}^{\mathcal{N}}_p \mathsf{W}^{\boxdot}_{\mathcal{S}_j}\mathsf{W}^{\diamond}_{\mathcal{N}} ok_j$, where $\mathcal{S}_j = \{q \in \mathcal{N} \mid \exists k \in \mathcal{C}[k \neq j \wedge \mathsf{B}^{\mathcal{N}}_q \mathsf{W}^{\diamond}_{\mathcal{N}} ok_k]\}$.*
4. *There is a CT-optimum protocol for $\mathcal{C}$ if and only if $\mathcal{R}_F \models \mathsf{K}_p \mathsf{S}^{\diamond}_{\mathcal{N}} ok_i \Rightarrow \bigvee_{a_j\in\mathcal{C}} \mathsf{K}_p \mathsf{S}^{\boxdot}_{\mathcal{S}_j}\mathsf{S}^{\diamond}_{\mathcal{N}} ok_j$, where $\mathcal{S}_j = \{q \in \mathcal{P} \mid \exists k \in \mathcal{C}[k \neq j \wedge \mathsf{K}_q \mathsf{S}^{\diamond}_{\mathcal{N}} ok_k]\}$.*

Proof Sketch: This proof gives only the third case; the others are similar. Assume that there is a total order '$<$' on the actions in $\mathcal{C}$.

Consider first the "if" direction and assume that the given condition holds. Consider now the protocol $F(\Phi)$, where $\Phi_{i,p} \equiv \mathsf{B}^{\mathcal{N}}_p(\mathsf{W}^{\diamond}_{\mathcal{N}} ok_i \wedge \mathsf{W}^{\boxdot}_{\mathcal{S}_i}\mathsf{W}^{\diamond}_{\mathcal{N}} ok_i \wedge \bigwedge_{j<i} \neg\mathsf{W}^{\boxdot}_{\mathcal{S}_j}\mathsf{W}^{\diamond}_{\mathcal{N}} ok_j)$. We will show that $F(\Phi)$ is NT-optimum for $\mathcal{C}$. Note first that it NT-satisfies $\mathcal{C}$. Clearly, it 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 $i < j$. These imply $(\mathcal{R}_F, r, l') \models \neg\mathsf{W}^{\boxdot}_{\mathcal{S}_i}\mathsf{W}^{\diamond}_{\mathcal{N}} ok_i$ and $(\mathcal{R}_F, r, l) \models \mathsf{W}^{\boxdot}_{\mathcal{S}_i}\mathsf{W}^{\diamond}_{\mathcal{N}} ok_i$, which are contradictory because of the continuality of $\mathsf{W}^{\boxdot}_{\mathcal{S}_i}\mathsf{W}^{\diamond}_{\mathcal{N}} ok_i$. To prove termination, suppose that $(\mathcal{R}_F, r, l) \models \Phi_{i,p}$, where $p \in \mathcal{N}(r)$. By the definition of Φ, $(\mathcal{R}_F, r, l) \models \boxdot \bigwedge_{j<i} \neg\mathsf{W}^{\boxdot}_{\mathcal{S}_j}\mathsf{W}^{\diamond}_{\mathcal{N}} ok_j$. The proof must show that, for all $q \in \mathcal{N}(r)$, there is some l' such that $(\mathcal{R}_F, r, l') \models \Phi_{i,q}$. This can be done using several applications of both positive and negative introspection.

The proof now shows that $F(\Phi)$ is NT-optimum. Consider any protocol $F(\Psi)$ that NT-satisfies $\mathcal{C}$. Suppose that $\Psi_{i,p}$ holds at some point $\langle r,l\rangle$. The proof must show that, for some $a_j \in \mathcal{C}$, $\Phi_{j,p}$ must hold at $\langle r,l\rangle$. By Theorem 2 applied to $F(\Psi)$, it must be that $\mathsf{B}_p^{\mathcal{N}}\mathsf{W}_{\mathcal{N}}^{\diamond} ok_i$. By the original hypothesis, $\mathsf{B}_p^{\mathcal{N}}\mathsf{W}_{\mathcal{S}_j}^{\boxdot}\mathsf{W}_{\mathcal{N}}^{\diamond} ok_j$ for some $a_j \in \mathcal{C}$. Consider the least such j. If $j = i$, then $\mathsf{B}_p^{\mathcal{N}}\mathsf{W}_{\mathcal{N}}^{\diamond} ok_j$ by assumption. If $j \neq i$, then $p \in \mathcal{S}_j$, so $\mathsf{B}_p^{\mathcal{N}}\mathsf{W}_{\mathcal{N}}^{\diamond} ok_j$ by positive introspection. Now consider any $k < j$. Since $\mathsf{B}_p^{\mathcal{N}}\mathsf{W}_{\mathcal{N}}^{\diamond} ok_j$, $\mathsf{B}_p^{\mathcal{N}}(p \in \mathcal{S}_k)$. By positive introspection, it must be that $\neg\mathsf{W}_{\mathcal{S}_k}^{\boxdot}\mathsf{W}_{\mathcal{N}}^{\diamond} ok_j$ and, by negative introspection, $\mathsf{B}_p^{\mathcal{N}}\neg\mathsf{W}_{\mathcal{S}_k}^{\boxdot}\mathsf{W}_{\mathcal{N}}^{\diamond} ok_j$. 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 $F(\Phi)$ for $\mathcal{C}$. The proof must show that $\mathcal{R}_F \models \mathsf{B}_p^{\mathcal{N}}\mathsf{W}_{\mathcal{N}}^{\diamond} ok_i \Rightarrow \bigvee_{a_j\in\mathcal{C}} \mathsf{B}_p^{\mathcal{N}}\mathsf{W}_{\mathcal{S}_j}^{\boxdot}\mathsf{W}_{\mathcal{N}}^{\diamond} ok_j$. Suppose that $(\mathcal{R}_F,r,l) \models \mathsf{B}_p^{\mathcal{N}}\mathsf{W}_{\mathcal{N}}^{\diamond} ok_i$. This means that, when executing the protocol $F(\Phi^i)$ defined in the NT-case of Theorem 3, p executes a_i at point $\langle r,l\rangle$. Since $F(\Phi)$ is NT-optimum, it dominates $F(\Phi^i)$, so $(\mathcal{R}_F,r,l) \models \Phi_{j,p}$ for some $a_j \in \mathcal{C}$. Consider now any run r' such that $r'_p(l) = r_p(l)$ and $p \in \mathcal{N}(r')$. Since $\Phi_{j,p}$ is a fact about p's local state, it must hold at $\langle r',l\rangle$, so $(\mathcal{R}_F,r',l) \models \varphi_j$, where φ_j is defined as in the NT-case of Lemma 4. By that lemma, $(\mathcal{R}_F,r',l) \models \mathsf{W}_{\mathcal{S}_j}^{\boxdot}\mathsf{W}_{\mathcal{N}}^{\diamond} ok_j$. Thus, $(\mathcal{R}_F,r,l) \models \mathsf{B}_p^{\mathcal{N}}\mathsf{W}_{\mathcal{S}_j}^{\boxdot}\mathsf{W}_{\mathcal{N}}^{\diamond} ok_j$, giving the desired validity. □

One can note at this point that the result of Moses and Tuttle, that there can be no optimum protocol for *Eventual Byzantine Agreement*, can be seen as a corollary to Theorem 5. *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-satisfaction (since there are, for synchronous systems, protocols that terminate in all runs, any optimum protocol will do so also).

Corollary 6: *There is no N-optimum protocol for* Eventual Byzantine Agreement.

Proof: Consider a run r in which there are two correct processors p and q such that p's initial state is 0 and q's is 1. We will show that $(\mathcal{R}_F,r,0) \models \mathsf{B}_p^{\mathcal{N}} ok_0 \wedge \neg\mathsf{B}_p^{\mathcal{N}}\mathsf{W}_{\mathcal{S}_0}^{\boxdot} ok_0 \wedge \neg\mathsf{B}_p^{\mathcal{N}}\mathsf{W}_{\mathcal{S}_1}^{\boxdot} ok_1$, where $\mathcal{S}_0 = \{r \in \mathcal{N} \mid \mathsf{B}_r^{\mathcal{N}} ok_1\}$ and $\mathcal{S}_1 = \{r \in \mathcal{N} \mid \mathsf{B}_r^{\mathcal{N}} ok_0\}$. By Theorem 5, this will indicate that there can be no N-optimum protocol for *Eventual Byzantine Agreement.* Consider p's belief at time 0. Clearly, $\mathsf{B}_p^{\mathcal{N}} ok_0$ because its initial state is 0. This implies $p \in \mathcal{S}_1(r,0)$. Since $\neg\mathsf{B}_p^{\mathcal{N}} ok_1$ (there has been no communication), it must be that $\neg\mathsf{W}_{\mathcal{S}_1}^{\boxdot} ok_1$. Similarly, $q \in \mathcal{S}_0(r,0)$ and $\neg\mathsf{B}_q^{\mathcal{N}} ok_0$, so $\neg\mathsf{W}_{\mathcal{S}_0}^{\boxdot} ok_0$. Thus, $\neg\mathsf{B}_p^{\mathcal{N}}\mathsf{W}_{\mathcal{S}_0}^{\boxdot} ok_0 \wedge \neg\mathsf{B}_p^{\mathcal{N}}\mathsf{W}_{\mathcal{S}_1}^{\boxdot} ok_1$, as desired. □

Theorem 5 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.[6] In addition, there may be optimum solutions to problems in which the enabling conditions are related in certain ways. Consider a problem with three possible actions such that the following implications are valid: $ok_1 \Rightarrow \neg ok_3$; $ok_3 \Rightarrow \neg ok_1$; 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 optimal solutions to *Eventual Byzantine Agreement.* Neiger [14] used a stronger form to derive solutions to a class of consistent coordination problems. Neither of these works explicitly considered the termination properties of the protocols they developed. When problems requiring termination are considered, it is necessary to combine eventual knowledge to continual knowledge. We call this combination *extended common knowledge.*

[6] 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.

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 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 φ to sets $\mathcal{S}$ and $\mathcal{T}$, denoted $\mathsf{S}^{\leftrightarrow}_{\mathcal{S},\mathcal{T}}\varphi$, is the greatest fixed point of

$$X \Leftrightarrow \Diamond\mathsf{E}_{\mathcal{S}}(\varphi \wedge X) \wedge \boxdot\mathsf{E}_{\mathcal{T}}(\varphi \wedge X).$$

Weak extended common knowledge of fact φ to sets $\mathcal{S}$ and $\mathcal{T}$, denoted $\mathsf{W}^{\leftrightarrow}_{\mathcal{S},\mathcal{T}}\varphi$, is the greatest fixed point of

$$X \Leftrightarrow \Diamond\mathsf{A}_{\mathcal{S}}(\varphi \wedge X) \wedge \boxdot\mathsf{A}_{\mathcal{T}}(\varphi \wedge X).$$

$\mathsf{S}^{\leftrightarrow}_{\mathcal{S},\mathcal{T}}\varphi$ implies (but is not equivalent to) the infinite conjunction

$$\Diamond\mathsf{E}_{\mathcal{S}}\varphi \wedge \boxdot\mathsf{E}_{\mathcal{T}}\varphi \wedge (\Diamond\mathsf{E}_{\mathcal{S}})^2\varphi \wedge \Diamond\mathsf{E}_{\mathcal{S}} \boxdot \mathsf{E}_{\mathcal{T}}\varphi \wedge \boxdot\mathsf{E}_{\mathcal{T}}\Diamond\mathsf{E}_{\mathcal{S}}\varphi \wedge (\boxdot\mathsf{E}_{\mathcal{T}})^2\varphi \wedge \cdots. \tag{1}$$

(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.

Both forms of extended common knowledge satisfy positive introspection with respect to both eventual and continual knowledge. Thus, the following are valid:

- $\mathsf{S}^{\leftrightarrow}_{\mathcal{S},\mathcal{T}}\varphi \Rightarrow \Diamond\mathsf{E}_{\mathcal{S}}\mathsf{S}^{\leftrightarrow}_{\mathcal{S},\mathcal{T}}\varphi \wedge \boxdot\mathsf{E}_{\mathcal{T}}(\varphi \wedge \mathsf{S}^{\leftrightarrow}_{\mathcal{S},\mathcal{T}}\varphi)$; and
- $\mathsf{W}^{\leftrightarrow}_{\mathcal{S},\mathcal{T}}\varphi \Rightarrow \Diamond\mathsf{A}_{\mathcal{S}}\mathsf{W}^{\leftrightarrow}_{\mathcal{S},\mathcal{T}}\varphi \wedge \boxdot\mathsf{A}_{\mathcal{T}}(\varphi \wedge \mathsf{W}^{\leftrightarrow}_{\mathcal{S},\mathcal{T}}\varphi)$.

(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:

- $\neg\mathsf{S}^{\leftrightarrow}_{\mathcal{S},\mathcal{T}}\varphi \wedge \mathsf{K}_p(p \in \mathcal{T}) \Rightarrow \mathsf{K}_p\neg\mathsf{S}^{\leftrightarrow}_{\mathcal{S},\mathcal{T}}\varphi$; and
- $\neg\mathsf{W}^{\leftrightarrow}_{\mathcal{S},\mathcal{T}}\varphi \wedge (p \in \mathcal{T}) \Rightarrow \mathsf{B}^{\mathcal{T}}_p\neg\mathsf{W}^{\leftrightarrow}_{\mathcal{S},\mathcal{T}}\varphi$.

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 $\varphi \Rightarrow \Diamond\mathsf{E}_{\mathcal{S}}(\varphi \wedge \psi) \wedge \boxdot\mathsf{E}_{\mathcal{T}}(\varphi \wedge \psi)$ is valid in a system, then $\varphi \Rightarrow \mathsf{S}^{\leftrightarrow}_{\mathcal{S},\mathcal{T}}\psi$ is also valid in that system.
- If $\varphi \Rightarrow \Diamond\mathsf{A}_{\mathcal{S}}(\varphi \wedge \psi) \wedge \boxdot\mathsf{A}_{\mathcal{T}}(\varphi \wedge \psi)$ is valid in a system, then $\varphi \Rightarrow \mathsf{W}^{\leftrightarrow}_{\mathcal{S},\mathcal{T}}\psi$ is also valid in that system.

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 φ about runs (specifically, the enabling conditions of a coordination problem) become extended common knowledge. Because the first set $\mathcal{N}$ is assumed to never be empty, it is not hard to see that, in these cases, $\mathsf{S}^{\leftrightarrow}_{\mathcal{N},\mathcal{S}}\varphi \Rightarrow \varphi$ and $\mathsf{W}^{\leftrightarrow}_{\mathcal{N},\mathcal{S}}\varphi \Rightarrow \varphi$ 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 $\mathcal{S}$ ($\mathsf{S}^{\Diamond}_{\mathcal{S}}\varphi$) implies the infinite conjunction $\bigwedge_{i\geq 1}(\Diamond\mathsf{E}_{\mathcal{S}})^i\varphi$. Similarly, strong continual common knowledge to $\mathcal{T}$ ($\mathsf{S}^{\boxdot}_{\mathcal{T}}\varphi$) is equivalent to the infinite conjunction $\bigwedge_{i\geq 1}(\boxdot\mathsf{E}_{\mathcal{T}})^i\varphi$. Thus, $\mathsf{S}^{\boxdot}_{\mathcal{T}}\mathsf{S}^{\Diamond}_{\mathcal{S}}\varphi$, which is used in Section 6, implies the infinite conjunction $\bigwedge_{i,j\geq 1}(\boxdot\mathsf{E}_{\mathcal{T}})^i(\Diamond\mathsf{E}_{\mathcal{S}})^j\varphi$. Note that all $\boxdot\mathsf{E}_{\mathcal{T}}$ operators precede all $\Diamond\mathsf{E}_{\mathcal{S}}$ operators. In contrast, strong extended common knowledge of φ by $\mathcal{S}$ and $\mathcal{T}$ ($\mathsf{S}^{\leftrightarrow}_{\mathcal{S},\mathcal{T}}$) implies the infinite conjunction of all orderings of these operators as noted in Equation 1 above. $\mathsf{S}^{\leftrightarrow}_{\mathcal{S},\mathcal{T}}$ thus implies $\mathsf{S}^{\boxdot}_{\mathcal{T}}\mathsf{S}^{\Diamond}_{\mathcal{S}}\varphi$. It is the additional flexibility of the combination of knowledge operators that is needed to characterize domination as is done in the next section.

Many of the results of the next sections depend upon the knowledge of the set of processors performing any action other than a specified one. Given an action function Φ and some action $a_i \in \mathcal{C}$, let $\mathcal{P}_i = \{p \in \mathcal{P} \mid \exists a_j \in \mathcal{C}[j \neq i \wedge \Phi_{j,p}]\}$ be the set of processors performing some action other than a_i. Let $\mathcal{N}_i = \{p \in \mathcal{N} \mid \exists a_j \in \mathcal{C}[j \neq i \wedge \Phi_{j,p}]\}$ be the set of nonfaulty processors performing some action other than a_i. Note that, for all $p \in \mathcal{P}$ and $a_i \in \mathcal{C}$, both $(p \in \mathcal{P}_i) \Rightarrow \mathsf{K}_p(p \in \mathcal{P}_i)$ and $(p \in \mathcal{N}_i) \Rightarrow \mathsf{B}^{\mathcal{N}}_p(p \in \mathcal{N}_i)$ are valid.

8 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.

The following theorem shows that the domination relationship between two action protocols (each using the same communication protocol) can be expressed using some form of common knowledge. In the case of nonconsistent problems, a weak form is used, while a strong form is needed for consistent problems. Problems requiring termination use extended common knowledge, while the others require only continual common knowledge. Informally, Theorem 7 states that for a processor to perform an action in a *dominating* protocol, it must know (or believe) that continual or extended common knowledge holds, where the set possessing the continual knowledge contains processors performing some other action in the *dominated* protocol.

Theorem 7: *Let C be a coordination problem and let $P(\Phi)$ and $P(\Psi)$ be such that $P(\Psi)$ dominates $P(\Phi)$.*

1. *If both protocols N-satisfy C, then $\mathcal{R}_P \models \Psi_{i,p} \Rightarrow \mathsf{B}^{\mathcal{N}}_p\mathsf{W}^{\boxdot}_{\mathcal{N}_i} ok_i$.*
2. *If both protocols C-satisfy C, then $\mathcal{R}_P \models \Psi_{i,p} \Rightarrow \mathsf{K}_p\mathsf{S}^{\boxdot}_{\mathcal{P}_i} ok_i$.*
3. *If both protocols NT-satisfy C, then $\mathcal{R}_P \models \Psi_{i,p} \Rightarrow \mathsf{B}^{\mathcal{N}}_p\mathsf{W}^{\leftrightarrow}_{\mathcal{N},\mathcal{N}_i} ok_i$.*
4. *If both protocols CT-satisfy C, then $\mathcal{R}_P \models \Psi_{i,p} \Rightarrow \mathsf{K}_p\mathsf{S}^{\leftrightarrow}_{\mathcal{N},\mathcal{P}_i} ok_i$.*

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

Assume that both protocols N-satisfy $\mathcal{C}$. Let $\psi_i \equiv \bigvee_{q\in\mathcal{N}}\Psi_{i,q}$. Using techniques seen earlier (including induction), it is sufficient to show that $\mathcal{R}_P \models \psi_i \Rightarrow \boxdot\mathsf{A}_{\mathcal{N}_i}(\psi_i \wedge ok_i)$. Suppose that $(\mathcal{R}_P, r, l) \models \psi_i$ and assume that $p \in \mathcal{N}_i(r,l')$ for some l'. By the definition of $\mathcal{N}_i$, $p \in \mathcal{N}(r)$ and, for some $j \neq i$, $(\mathcal{R}_P, r, l') \models \Phi_{j,p}$. Since $P(\Psi)$ dominates $P(\Phi)$, the agreement condition ensures that $(\mathcal{R}_P, r, l') \models \Psi_{i,p}$. By Theorem 1, $(\mathcal{R}_P, r, l') \models \mathsf{B}^{\mathcal{N}}_p ok_i$ and clearly, $(\mathcal{R}_P, r, l') \models \mathsf{B}^{\mathcal{N}}_p\psi_i$. Thus, $(\mathcal{R}_P, r, l) \models \boxdot\mathsf{A}_{\mathcal{N}_i}(\psi_i \wedge ok_i)$, as desired.

Now assume that both protocols CT-satisfy $\mathcal{C}$. Let $\psi_i \equiv \bigvee_{q\in\mathcal{P}}\Psi_{i,q}$. It is sufficient to show that $\mathcal{R}_P \models \psi_i \Rightarrow \Diamond\mathsf{E}_{\mathcal{N}}(\psi_i \wedge ok_i) \wedge \boxdot\mathsf{E}_{\mathcal{P}_i}(\psi_i \wedge ok_i)$. Assume that $(\mathcal{R}_P, r, l) \models \psi_i$. The proof that $(\mathcal{R}_P, r, l) \models \boxdot\mathsf{E}_{\mathcal{P}_i}(\psi_i \wedge ok_i)$ is similar to the one given above for $\mathsf{A}_{\mathcal{N}_i}$. Since $P(\Psi)$ has some processor perform a_i in r, all processors $p \in \mathcal{N}(r)$ must perform a_i at some time l'. Thus, $(\mathcal{R}_P, r, l') \models \Psi_{i,p}$; using Theorem 1, $(\mathcal{R}_P, r, l') \models \mathsf{K}_p(\psi_i \wedge ok_i)$, completing the proof. □

The first two cases above are essentially the same as results observed earlier [12,14]. The results regarding problems requiring termination are new.

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 a protocol that dominates it. This is done by improving the performance of selected action so that is performed as quickly as is possible by any protocol that dominates the original one.

Lemma 8: *Let C be a coordination problem and let a_j be any action in C.*

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

$$\Psi_{i,p} \equiv \begin{cases} \mathsf{B}_p^{\mathcal{N}}(ok_j \wedge \mathsf{W}_{\mathcal{N}_j}^{\boxdot} ok_j) & \text{if } i = j \\ \Phi_{i,p} \wedge \mathsf{B}_p^{\mathcal{N}} \neg \mathsf{W}_{\mathcal{N}_j}^{\boxdot} ok_j & \text{if } i \neq j. \end{cases}$$

2. *If $P(\Phi)$ C-satisfies C, then $P(\Psi)$ C-satisfies C and dominates $P(\Phi)$ if*

$$\Psi_{i,p} \equiv \begin{cases} \mathsf{K}_p(ok_j \wedge \mathsf{S}_{\mathcal{P}_j}^{\boxdot} ok_j) & \text{if } i = j \\ \Phi_{i,p} \wedge \mathsf{K}_p \neg \mathsf{S}_{\mathcal{P}_j}^{\boxdot} ok_j & \text{if } i \neq j. \end{cases}$$

3. *If $P(\Phi)$ NT-satisfies C, then $P(\Psi)$ NT-satisfies C and dominates $P(\Phi)$ if*

$$\Psi_{i,p} \equiv \begin{cases} \mathsf{B}_p^{\mathcal{N}} \mathsf{W}_{\mathcal{N},\mathcal{N}_j}^{\leftrightarrow} ok_j & \text{if } i = j \\ \Phi_{i,p} \wedge \mathsf{B}_p^{\mathcal{N}} \neg \mathsf{W}_{\mathcal{N},\mathcal{N}_j}^{\leftrightarrow} ok_j & \text{if } i \neq j. \end{cases}$$

4. *If $P(\Phi)$ CT-satisfies C, then $P(\Psi)$ CT-satisfies C and dominates $P(\Phi)$ if*

$$\Psi_{i,p} \equiv \begin{cases} \mathsf{K}_p \mathsf{S}_{\mathcal{N},\mathcal{P}_j}^{\leftrightarrow} ok_j & \text{if } i = j \\ \Phi_{i,p} \wedge \mathsf{K}_p \neg \mathsf{S}_{\mathcal{N},\mathcal{P}_j}^{\leftrightarrow} ok_j & \text{if } i \neq j. \end{cases}$$

Proof: The proof considers only the fourth case; the remainder are similar. Suppose that $P(\Phi)$ CT-satisfies C and let $P(\Psi)$ be as defined above. The proof must show that $P(\Psi)$ CT-satisfies C and that it dominates $P(\Phi)$. Consider first the validity condition. Suppose that $(\mathcal{R}_P, r, l) \models \Psi_{i,p}$. If $i = j$, then $(\mathcal{R}_P, r, l) \models \mathsf{K}_p \mathsf{S}_{\mathcal{N},\mathcal{P}_j}^{\leftrightarrow} ok_j$. Since $\mathsf{S}_{\mathcal{N},\mathcal{P}_j}^{\leftrightarrow} ok_j \Rightarrow ok_j$ is valid, a_j is an enabled action. If $i \neq j$, then $(\mathcal{R}_P, r, l) \models \Phi_{i,p}$ and a_i is enabled because of the assumption that $P(\Phi)$ CT-satisfies C. Next consider the agreement condition. Suppose that processors p and q perform actions a_{i_p} and a_{i_q}, respectively, in run r. If neither i_p nor i_q is j, then $\Phi_{i_p,p}$ and $\Phi_{i_q,q}$ hold in r, implying that $i_p = i_q$. Alternatively, assume without loss of generality that p performs a_j at time l. Then $(\mathcal{R}_P, r, l) \models \mathsf{K}_p \mathsf{S}_{\mathcal{N},\mathcal{P}_j}^{\leftrightarrow} ok_j$. Suppose now that $(\mathcal{R}_P, r, l') \models \Phi_{i,q}$ for some l' and some $i \neq j$. Since $q \in \mathcal{P}_j(r, l')$, $(\mathcal{R}_P, r, l') \models \mathsf{K}_q \mathsf{S}_{\mathcal{N},\mathcal{P}_j}^{\leftrightarrow} ok_j$ by positive introspection. Thus, it is impossible for q to perform a_i and agreement is preserved. Finally, consider the termination condition. Assume that $(\mathcal{R}_P, r, l) \models \Psi_{i,p}$. The proof must show that, for every $q \in \mathcal{N}(r)$, there is some l' such that $(\mathcal{R}_P, r, l') \models \Psi_{i,q}$. Consider two cases:

- $i = j$. Then $(\mathcal{R}_P, r, l) \models \mathsf{K}_p \mathsf{S}_{\mathcal{N},\mathcal{P}_j}^{\leftrightarrow} ok_j$. By positive introspection, there is an $l' \geq l$ such that $(\mathcal{R}_P, r, l') \models \mathsf{K}_q \mathsf{S}_{\mathcal{N},\mathcal{P}_j}^{\leftrightarrow} ok_j$.
- $i \neq j$. Then $(\mathcal{R}_P, r, l) \models \Phi_{i,p} \wedge \mathsf{K}_p \neg \mathsf{S}_{\mathcal{N},\mathcal{P}_j}^{\leftrightarrow} ok_j$. By the termination of $P(\Phi)$, there is an l' such that $(\mathcal{R}_P, r, l') \models \Phi_{i,q}$. If $(\mathcal{R}_P, r, l') \models \mathsf{S}_{\mathcal{N},\mathcal{P}_j}^{\leftrightarrow} ok_j$, then, since $p \in \mathcal{P}_j(r, l)$, $(\mathcal{R}_P, r, l) \models \mathsf{K}_p \mathsf{S}_{\mathcal{N},\mathcal{P}_j}^{\leftrightarrow} ok_j$. This is a contradiction, so $(\mathcal{R}_P, r, l') \models \neg \mathsf{S}_{\mathcal{N},\mathcal{P}_j}^{\leftrightarrow} ok_j$. Since $(\mathcal{R}_P, r, l') \models \mathsf{K}_q(q \in \mathcal{P}_j)$, $(\mathcal{R}_P, r, l') \models \mathsf{K}_p \neg \mathsf{S}_{\mathcal{N},\mathcal{P}_j}^{\leftrightarrow} ok_j$ by negative introspection.

In either case, q eventually performs a_i, as desired.

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

- $i = j$. Since $P(\Phi)$ dominates itself, Theorem 7 implies that $(\mathcal{R}_P, r, l) \models \mathsf{K}_p \mathsf{S}^{\leftrightarrow}_{\mathcal{N},\mathcal{P}_j} ok_j$; thus, $(\mathcal{R}_P, r, l) \models \Psi_{j,p}$.
- $i \neq j$. Recall that $(\mathcal{R}_P, r, l) \models \mathsf{K}_p(p \in \mathcal{P}_j)$. Consider now two subcases:
  - $(\mathcal{R}_P, r, l) \models \mathsf{S}^{\leftrightarrow}_{\mathcal{N},\mathcal{P}_j} ok_j$. Then $(\mathcal{R}_P, r, l) \models \mathsf{K}_p \mathsf{S}^{\leftrightarrow}_{\mathcal{N},\mathcal{P}_j} ok_j$ by positive introspection, so $(\mathcal{R}_P, r, l) \models \Psi_{j,p}$.
  - $(\mathcal{R}_P, r, l) \models \neg \mathsf{S}^{\leftrightarrow}_{\mathcal{N},\mathcal{P}_j} ok_j$. Then $(\mathcal{R}_P, r, l) \models \mathsf{K}_p \neg \mathsf{S}^{\leftrightarrow}_{\mathcal{N},\mathcal{P}_j} ok_j$ 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] and Neiger [14] gave methods, using continual common knowledge, of constructing dominating protocols for coordination problems that did not require termination. Their techniques, however, cannot be applied to problems requiring termination (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 7 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 10 in the following section shows how Lemma 8 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 9 gives the necessary and sufficient conditions for a full-information protocol to be optimal. It turns out that these conditions are closely related to the dominating conditions established in Theorem 7.

Theorem 9: *Let $\mathcal{C}$ be a coordination problem and let $F(\Phi)$ be a full-information protocol.*

1. *$F(\Phi)$ is N-optimal for $\mathcal{C}$ if and only if $\mathcal{R}_F \models \Phi_{i,p} \Leftrightarrow \mathsf{B}^{\mathcal{N}}_p(ok_i \wedge \mathsf{W}^{\Box}_{\mathcal{N}_i} ok_i) \wedge \bigwedge_{j \neq i} \neg \Phi_{j,p}$.*
2. *$F(\Phi)$ is C-optimal for $\mathcal{C}$ if and only if $\mathcal{R}_F \models \Phi_{i,p} \Leftrightarrow \mathsf{K}_p(ok_i \wedge \mathsf{S}^{\Box}_{\mathcal{P}_i} ok_i) \wedge \bigwedge_{j \neq i} \neg \Phi_{j,p}$.*
3. *$F(\Phi)$ is NT-optimal for $\mathcal{C}$ if and only if $\mathcal{R}_F \models \Phi_{i,p} \Leftrightarrow \mathsf{B}^{\mathcal{N}}_p \mathsf{W}^{\leftrightarrow}_{\mathcal{N},\mathcal{N}_i} ok_i \wedge \bigwedge_{j \neq i} \neg \Phi_{j,p}$.*
4. *$F(\Phi)$ is CT-optimal for $\mathcal{C}$ if and only if $\mathcal{R}_F \models \Phi_{i,p} \Leftrightarrow \mathsf{K}_p \mathsf{S}^{\leftrightarrow}_{\mathcal{N},\mathcal{P}_i} ok_i \wedge \bigwedge_{j \neq i} \neg \Phi_{j,p}$.*

Proof: Two cases are proven; the rest are similar.

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

$$\mathcal{R}_F \models \Phi_{i,p} \Leftrightarrow \mathsf{B}^{\mathcal{N}}_p(ok_i \wedge \mathsf{W}^{\Box}_{\mathcal{N}_i} ok_i) \wedge \bigwedge_{j \neq i} \neg \Phi_{j,p}.$$

Clearly, $F(\Phi)$ N-satisfies $\mathcal{C}$. Let $F(\Psi)$ be a protocol that N-satisfies $\mathcal{C}$ and that dominates $F(\Phi)$; the proof must show that $F(\Phi)$ 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 1, $(\mathcal{R}_F, r, l) \models \mathsf{B}^{\mathcal{N}}_p ok_i$. By Theorem 7, $(\mathcal{R}_F, r, l) \models \mathsf{B}^{\mathcal{N}}_p \mathsf{W}^{\Box}_{\mathcal{N}_i} ok_i$. 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 $\mathcal{C}$; the proof must show that $\mathcal{R}_F \models \Phi_{i,p} \Leftrightarrow \mathsf{K}_p\mathsf{S}^{\leftrightarrow}_{\mathcal{N},\mathcal{P}_i} ok_i \wedge \bigwedge_{j\neq i} \neg\Phi_{j,p}$. Suppose first that $(\mathcal{R}_F, r, l) \models \Phi_{i,p}$. Clearly, $(\mathcal{R}_F, r, l) \models \bigwedge_{j\neq i} \neg\Phi_{j,p}$. Because $F(\Phi)$ dominates itself, Theorem 7 implies that $(\mathcal{R}_F, r, l) \models \mathsf{K}_p\mathsf{S}^{\leftrightarrow}_{\mathcal{N},\mathcal{P}_i} ok_i$, giving the desired implication. Finally, suppose that $(\mathcal{R}_F, r, l) \models \mathsf{K}_p\mathsf{S}^{\leftrightarrow}_{\mathcal{N},\mathcal{P}_i} ok_i \wedge \bigwedge_{j\neq i} \neg\Phi_{j,p}$. Let $F(\Psi)$ be the dominating protocol constructed from $F(\Phi)$ using the CT-case of Lemma 8 (using $j = i$). Note that $\Psi_{i,p} \equiv \mathsf{K}_p\mathsf{S}^{\leftrightarrow}_{\mathcal{N},\mathcal{P}_i} ok_i$, so $(\mathcal{R}_F, r, l) \models \Psi_{i,p}$. Since $F(\Phi)$ is optimal, it dominates $F(\Psi)$, so $(\mathcal{R}_F, r, l) \models \Phi_{j,p}$ for some $a_j \in \mathcal{C}$. 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 two cases above are essentially the same as results observed earlier [12,14]. The results regarding problems requiring termination are new. 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 the papers on simultaneous coordination [4,13,17]. These 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, it is necessary to iteratively apply Lemma 8 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 10: *Let $\mathcal{C}$ be a coordination problem and let '$<$' be some total order of the actions in $\mathcal{C}$ ($a_1 < a_2 < \cdots < a_m$). Let $F(\Phi)$ be a full-information action protocol. If $F(\Phi)$ X-satisfies $\mathcal{C}$ (where X is either N, C, NT, or CT), then inductively define $F(\Phi^i)$ ($0 \le i \le m$) as follows: $F(\Phi^0)$ is $F(\Phi)$ and $F(\Phi^{i+1})$ is the dominating protocol constructed from $F(\Phi^i)$ using the X-case of Lemma 8, using $j = i+1$. Then $F(\Phi^m)$ is X-optimal for $\mathcal{C}$ and dominates $F(\Phi)$.*

Proof: The proof considers the case of CT-optimality; the others are similar. It follows by Lemma 8 that, for all i ($1 \le i \le m$), $F(\Phi^i)$ CT-satisfies $\mathcal{C}$ and dominates all $F(\Phi^j)$ with $j \le i$. It remains only to show that $F(\Phi^m)$ is CT-optimal; this is done by showing that it dominates any protocol $F(\Psi)$ that CT-satisfies $\mathcal{C}$ and that dominates $F(\Phi^m)$. To show that $F(\Phi^m)$ dominates $F(\Psi)$, assume that $(\mathcal{R}_F, r, l) \models \Psi_{i,p}$. Since $F(\Psi)$ dominates $F(\Phi^m)$, it also dominates $F(\Phi^{i-1})$. Thus, by Theorem 7, $(\mathcal{R}_F, r, l) \models \mathsf{K}_p\mathsf{S}^{\leftrightarrow}_{\mathcal{N},\mathcal{P}_i} ok_i$, where $\mathcal{P}_i$ is based on Φ^{i-1}. But this is precisely $\Phi^i_{i,p}$, so $F(\Phi^i)$ has p perform a_i at $\langle r, l\rangle$. Since $F(\Phi^m)$ dominates $F(\Phi^i)$, it also has p perform some action at $\langle r, l\rangle$. Thus, $F(\Phi^m)$ dominates $F(\Psi)$ and is optimal. □

Theorem 10 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 $\Phi_{i,p} = \mathit{false}$ for all i and p). The first application of Lemma 8 results then in the following action function (for CT-satisfaction):

$$\Phi^1_{i,p} \equiv \begin{cases} \mathsf{K}_p\mathsf{S}^{\leftrightarrow}_{\mathcal{N},\mathcal{P}_1} ok_1 & \text{if } i = 1 \\ \mathit{false} & \text{if } i > 1 \end{cases}$$

where $\mathcal{P}_1$ is based in Φ and is thus empty. Thus, $\Phi^1_{1,p}$ simplifies to $\mathsf{K}_p\mathsf{S}^{\diamond}_{\mathcal{N}} ok_1$. Thus, by Theorem 2, $F(\Phi^1)$ performs a_1 as early as any protocol can. The second application results in the following:

$$\Phi^2_{i,p} \equiv \begin{cases} \mathsf{K}_p(\mathsf{S}^{\diamond}_{\mathcal{N}} ok_1 \wedge \neg\mathsf{S}^{\leftrightarrow}_{\mathcal{N},\mathcal{P}_2} ok_2) & \text{if } i = 1 \\ \mathsf{K}_p\mathsf{S}^{\leftrightarrow}_{\mathcal{N},\mathcal{P}_2} ok_2 & \text{if } i = 2 \\ \mathit{false} & \text{if } i > 2 \end{cases}$$

where $\mathcal{P}_2$ is based on Φ^1. This is $\{q \in \mathcal{P} \mid \mathsf{K}_q\mathsf{S}^{\diamond}_{\mathcal{N}} ok_1\}$. If there are only two actions in $\mathcal{C}$ (as in the case of *Eventual Byzantine Agreement*), then $F(\Phi^2)$ is optimal. Otherwise, Lemma 8 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 [4,13,17]. 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 true 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 a problem to have an optimum solution. 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 [19] 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 [17] 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 cannot be attained in these same systems, indicating that consistent nonsimultaneous coordination cannot be achieved in these systems. Furthermore, we conjecture 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 likely that the semantics of extended common knowledge can be understood by combining the game-theoretic characterization of eventual common knowledge [19] 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 based on common knowledge, a better understanding of extended common knowledge could do the same for 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 that 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. It turns out that 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.

Note that the construction of optimal protocols depends on a total ordering of the m actions in a coordination problem. There are $m!$ such orderings and thus $m!$ ways to derive an optimal protocol. We conjecture that, for any coordination problem with m actions, there are at most $m!$ optimal protocols. This would imply that the construction of an optimal protocol could always begin with a degenerate protocol (that performs no actions) and that it would not be necessary to first find a correct protocol and then optimize 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

[1] Rida Bazzi and Gil Neiger. On the complexity of using eventual common knowledge to achieve coordination. In progress.

[2] Brian A. Coan. A communication-efficient canonical form for fault-tolerant distributed protocols. In *Proceedings of the Fifth ACM Symposium on Principles of Distributed Computing*, pages 63–72, August 1986. A revised version appears in Coan's Ph.D. dissertation [3].

[3] Brian A. Coan. *Achieving Consensus in Fault-Tolerant Distributed Computer Systems: Protocols, Lower Bounds, and Simulations.* Ph.D. dissertation, Massachusetts Institute of Technology, June 1987.

[4] Cynthia Dwork and Yoram Moses. Knowledge and common knowledge in a Byzantine environment: Crash failures. *Information and Computation*, 88(2):156–186, October 1990.

[5] Ronald Fagin and Moshe Y. Vardi. Knowledge and implicit knowledge in a distributed environment: Preliminary report. In Joseph Y. Halpern, editor, *Proceedings of the First Conference on Theoretical Aspects of Reasoning about Knowledge*, pages 187–206. Morgan-Kaufmann, March 1986. Also appears as IBM Research Laboratory Technical Report RJ4990.

[6] Michael J. Fischer. The consensus problem in unreliable distributed systems (a brief survey). Technical Report 273, Department of Computer Science, Yale University, June 1983.

[7] Michael J. Fischer and Nancy A. Lynch. A lower bound for the time to assure interactive consistency. *Information Processing Letters*, 14:183–186, 1982.

[8] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, April 1985.

[9] Ajei Gopal and Sam Toueg. Reliable broadcast in synchronous and asynchronous environments (preliminary version). In J.-C. Bermond and M. Raynal, editors, *Proceedings of the Third International*

Workshop on Distributed Algorithms, volume 392 of *Lecture Notes on Computer Science*, pages 110–123. Springer-Verlag, September 1989.

[10] Vassos Hadzilacos. *Issues of Fault Tolerance in Concurrent Computations.* Ph.D. dissertation, Harvard University, June 1984. Department of Computer Science Technical Report 11-84.

[11] Joseph Y. Halpern and Yoram Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37(3):549–587, July 1990.

[12] Joseph Y. Halpern, Yoram Moses, and Orli Waarts. A characterization of eventual Byzantine agreement. In *Proceedings of the Ninth ACM Symposium on Principles of Distributed Computing*, pages 333–346, August 1990.

[13] Yoram Moses and Mark R. Tuttle. Programming simultaneous actions using common knowledge. *Algorithmica*, 3(1):121–169, 1988.

[14] Gil Neiger. Consistent coordination and continual common knowledge. Unpublished manuscript, January 1991.

[15] Gil Neiger and Sam Toueg. Substituting for real time and common knowledge in asynchronous distributed systems (preliminary version). In *Proceedings of the Sixth ACM Symposium on Principles of Distributed Computing*, pages 281–293, August 1987. A revised and expanded version appears as Georgia Institute of Technology School of Information and Computer Science Technical Report 90/05 under the title "Simulating Synchronized Clocks and Common Knowledge in Distributed Systems." To appear in the *Journal of the ACM.*

[16] Gil Neiger and Sam Toueg. Automatically increasing the fault-tolerance of distributed algorithms. *Journal of Algorithms*, 11(3):374–419, September 1990.

[17] Gil Neiger and Mark R. Tuttle. Common knowledge and consistent simultaneous coordination. In J. van Leeuwen and N. Santoro, editors, *Proceedings of the Fourth International Workshop on Distributed Algorithms*, volume 486 of *Lecture Notes on Computer Science*, pages 334–352. Springer-Verlag, September 1990. To appear in *Distributed Computing.*

[18] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, April 1980.

[19] Mark R. Tuttle. A game-theoretic characterization of eventual common knowledge. Unpublished manuscript, October 1988.

[20] Jennifer Lundelius Welch. Simulating synchronous processors. *Information and Computation*, 74(2):159–171, August 1987.