

Analyzing Distributed Commitment
By
Reasoning About Knowledge

Murray S. Mazer Frederick H. Lochovsky*
Digital Equipment Corporation
Cambridge Research Lab

CRL 90/10

30 November 1990

Abstract

We present an analysis of distributed, negotiated commitment. This is the problem of ensuring that processes in a distributed negotiation commit consistently to the outcome, even in the face of system failures. Our analysis is based on reasoning in a temporal, epistemic logic about the knowledge of the processes in any solution to the problem.

In our analysis, we present necessary levels of knowledge for commitment in settings that admit process or communication failures; we also consider settings that must be nonblocking or guarantee termination. From the necessary knowledge, we derive interprocess communication requirements, via a result linking knowledge and communication; this yields the underlying communication structure in any protocol that supports negotiated commitment. We then give a message lower bound for achieving commitment and several other impossibility results, showing that certain desirable commitment behaviours cannot be supported by any protocol. These results are based on new techniques, which use the knowledge and communication requirements that we derive from the specification of negotiated commitment.

This paper contributes a detailed and precise specification and analysis of the generalized distributed commitment problem. Further, the paper shows new ways in which one can use reasoning about knowledge to gain insight into distributed problems.

Keywords: Commitment, fault tolerance, process knowledge, logic of knowledge, knowledge and communication, negotiation, bidding.

©Digital Equipment Corporation 1990. All rights reserved.

*Department of Computer Science and Computer Systems Research Institute, University of Toronto, Toronto ON, Canada M5S 1A4.

Think globally, act locally.
Environmental credo and
distributed systems slogan.

1 Introduction

Negotiation is a useful form of coordination in distributed computer systems, for dynamically establishing commitments to joint courses of action. Forms of computer-based negotiation have appeared or been proposed in the literature for various purposes, such as resource allocation, task allocation, task scheduling, transaction atomic commitment, distributed planning, stock trading, security pass allocation, and travel reservations (see [Maze89] for a survey). The problem of *negotiated commitment* is to ensure that the processes in a distributed negotiation commit consistently to the outcome, even in the face of unpredictability, such as system failures. All negotiating systems must solve this problem. Previous appearances of computer-based negotiation, however, lacked a formal definition of commitment, used an informal model of computation, or did not clearly state systems assumptions.¹

We give a formal specification and analysis of negotiated commitment, using a modal logic of knowledge as the main analysis tool. Distributed problems are typically stated in terms of global behaviours, yet processes must act locally, based on their inherently incomplete view of the global state of the system. The abstraction of knowledge is simply a precise way to reason about the extent to which a process's local state accurately reflects important aspects of the global state. Several researchers have demonstrated the value in reasoning formally about the knowledge of processes in distributed computations, as a precise way to specify, analyze, and derive protocols for distributed problems (see, for example, [Hadz90, Halp87, HaMo90, HaZu89, Maze89, Maze90, MoTu88, Tutt89]). From a knowledge-theoretic perspective, a group of processes acquires and disseminates knowledge about the system, through various events, as a system computation evolves. Intuitively, a process's actions depend on its knowledge, and its knowledge changes as a result of actions [HaFa89].

Reasoning formally about knowledge may offer a useful conceptual abstraction and an elegant formalism for expressing reasoning which is often either intuitive and operational or formal but opaque [HaMo90]. For example, informal arguments often proceed as follows: "The coordinator must send a message to each of the others, so that each may learn the joint decision. Each must know the decision in order to carry out the corresponding actions. Once the coordinator has received an acknowledgement from a participant, it knows that the participant knows the decision and . . ." Such reasoning, if formalized, is transformed into complex combinatorial arguments which obscure the relationship to the problem specification and the knowledge of the participants. Our approach seeks to retain, in the formal reasoning process, the relationship

¹The exception is atomic commitment—see below.

between the problem specification and the informal reasoning. For a distributed problem, one first shows, formally, the knowledge each process requires to solve the problem, and then one derives communication requirements from the knowledge requirements.

This approach helps the protocol designer in several ways. By determining the knowledge required by processes to solve a distributed problem, one gains insight into the propositional structure of the set of possible protocols for the problem. Further, deriving interprocess communication requirements from the knowledge requirements yields insight into the underlying communication structure of any protocol to solve the problem. By using the knowledge and communication requirements derived from a problem specification, one can then show impossibility results and design protocols for the given problem.

The problem of negotiated commitment involves two kinds of processes: ① a distinguished process, historically called the *manager*, which coordinates the commitment; and ② the set of *contractors*, or bidders. Each of the contractors chooses whether to bid or not on an announced contract. The manager selects from among the bidding contractors to establish a *dependency set*, representing those contractors which the manager wants to commit to performing the announced task. The manager then relays its decision to the contractors, and the dependency set members commit accordingly. The period of contractor uncertainty about the manager's decision, and the potential for process or communication failures, makes consistent negotiated commitment nontrivial to achieve.²

Negotiated commitment is related, but incomparable, to the problem of atomic commitment in distributed transaction systems [BeHG87, Gray79, Lamp81]; they differ in two main aspects. First, in negotiated commitment, the manager coordinates the commitment; in atomic commitment, there need not be a single coordinator. Second, in negotiated commitment, commitment may be established among subsets of the participating processes; in atomic commitment, a commitment must include all processes. The specifications of these two problems reflect these differences (cf. [Hadz90]). Throughout this paper, we comment on how results on negotiated commitment translate to results on atomic commitment.

Our analysis yields several kinds of results. First, we show the levels of knowledge each process requires to achieve different kinds of commitment behaviour. Second, we identify restrictions on distributed computations, including underlying communication patterns, needed to facilitate the identified states of knowledge. Third, we give a message lower bound for achieving commitment, based on the impossibility of achieving the required knowledge in fewer messages than the lower bound. Fourth, we show the impossibility of achieving commitment under certain assumptions of system characteristics; these results are based on the impossibility of the infinite communication needed to achieve the required knowledge transfer. We include settings that admit process recovery from failures, that must be nonblocking, or that must guarantee protocol termination. In particular, we show the impossibility of commitment protocols

²Asynchrony, regardless of system failures, also makes commitment nontrivial to achieve, but we focus our analysis on systems which admit failures; cf. [Maze89].

which (1) support independent process recovery, (2) are terminating under process recovery and bounded communication time, (3) are nonblocking under permanent communication failures, or (4) are nonblocking and terminating under communication failures.

Impossibility result (2) is new. Dwork and Skeen (1983) showed the message lower bound for the related problem of atomic commitment, basing their arguments on the message passing graphs produced by a “best-case,” failure-free instance of an atomic commitment protocol. Impossibility results (1) and (3) have proofs for atomic commitment, based upon an examination of the plausible state transitions in atomic commitment protocols given in a finite state machine model [Skee82, SkSt83]. One cannot always easily determine, however, how the combinatorial proofs reflect the problem being solved. Unlike these proofs, our knowledge-theoretic proofs first determine the propositional content [RoKa86] of the problem solutions, based on the problem specification. From this semantic analysis, we then argue that certain message passing patterns are needed. By using this approach, we have derived results that hold for more circumstances than the previous results. Hadzilacos (1987, 1990) gives a knowledge-theoretic treatment of atomic commitment. He shows the minimum knowledge levels that hold in two-phase and three-phase atomic commitment protocols, the impossibility of nonblocking protocols under the assumptions in (3) and (4) above, and a message lower bound. Our treatment is similar to that of Hadzilacos, although it differs in several important ways. For example, we allow processes to recover from process failures, and we admit systems in which messages may spend only a limited time in transit. Our treatment of negotiated commitment attacks in more depth a problem slightly different from atomic commitment, and the results for atomic commitment follow naturally from the results for negotiated commitment. We prove our results using a uniform underlying strategy which differs from that used by Hadzilacos for his results.

The paper proceeds as follows. In Section 2, we give a model of distributed computation. Section 3 presents a logic of knowledge in which one can express problem specifications that include temporal and epistemic (that is, knowledge) assertions; we also show some useful properties of systems. In Section 4, we specify negotiated commitment. In Section 5, we analyze the specification to determine knowledge requirements. We use these results in Section 6 to determine the communication requirements for commitment. In Section 7, we give the impossibility results, which follow from some further knowledge and communication analysis. Section 8 discusses knowledge requirements in general nonblocking systems, and Section 9 is our summary.

2 A Model of Distributed Systems

In this section, we give a model of distributed computation in which we will ground the definition of knowledge and our analysis of negotiated commitment.

2.1 Executions

Adapting [Hadz90], we consider distributed systems which comprise two types of elements: ① processes, which execute events (let Π represent the set of n processes); and ② a communication system, \mathcal{N} , which contains a set of message packets (of the form $\langle p, \underline{m}, q, i \rangle$, representing the message \underline{m} sent from p to q at time i). The events are of two kinds: communicative and noncommunicative. The communicative events are SEND(\underline{m}, q) (the executing process sends message \underline{m} to process q , where $\underline{m} \in \underline{M}$, a message vocabulary) and RECV(\underline{m}, q) (the executing process receives message \underline{m} from process q ; \underline{m} may be the null message λ or a message from \underline{M}). These are the only two events by which a process may communicate externally; all other process events are local and have no effect on the communication system.

A possible joint behaviour over time of the processes and the communication system is modelled by an *execution* (or *run*). Each execution e is a function mapping time to a global state tuple of the form $\langle \text{time}, \text{history}_{p_1}, \text{history}_{p_2}, \dots, \text{history}_{p_n}, \text{packets} \rangle$. *time* represents the time at which the system is observed; *history* $_{p_i}$ represents the finite sequence of events executed by process p_i in execution e up to the observation instant; and *packets* is the set of message packets in transit at that instant. As is common, we take “time” to be the natural numbers, N . The *points* of an execution set \mathcal{E} , $\text{Pts}(\mathcal{E})$, are $\{(e, f) \mid e \in \mathcal{E} \text{ and } f \in N\}$.

Here is some notation on executions, required for the sequel. Throughout this paper, we use the letter ‘ e ’ to refer to an execution. ‘ p ’ and ‘ q ’ refer to processes. We use other letters, notably ‘ f ’, ‘ g ’, and ‘ i ’, to refer to times. Any of these may appear superscripted or subscripted. For $p \in \Pi$, we write $e(f, p)$ for *history* $_p$, p ’s history element in the tuple at point (e, f) ; similarly, we write $e(f, \mathcal{N})$ for *packets*, the set of message packets in the communication system at point (e, f) . $d \vdash e(f, p)$ asserts that d is the last event in the sequence *history* $_p$ at point (e, f) ; $d \in e(f, p)$ indicates that event d appears in the sequence; $|e(f, p)|$ indicates the number of events in the sequence; and $e(f, p) \cdot d$ indicates the concatenation of event d to the sequence. We write $d \sqsubset (e, f + 1, p)$ to say that process p has just executed event d at point $(e, f + 1)$, i.e., $d \sqsubset (e, f + 1, p)$ iff $e(f + 1, p) = e(f, p) \cdot d$. $e(f + 1, p) \geq e(f, p)$ denotes that p ’s event sequence up to time $f + 1$ in e has, as a prefix, p ’s event sequence up to f in e . For $P, Q \subseteq \Pi$, $e(f, \mathcal{N})[P, Q] \stackrel{\text{def}}{=} \{\langle p, \underline{m}, q, i \rangle \in e(f, \mathcal{N}) \mid p \in P \text{ and } q \in Q\}$. That is, $e(f, \mathcal{N})[P, Q]$ is the set of messages in transit from processes in P to processes in Q at instant f of execution e . $\bar{P} \stackrel{\text{def}}{=} \Pi \setminus P$.

For each $p \in \Pi$, a relation \sim_p on the points in system \mathcal{E} captures when p has the same event sequence in two points. For $(e, f), (e', g) \in \text{Pts}(\mathcal{E})$, we write $(e, f) \sim_p (e', g)$ iff $e(f, p) = e'(g, p)$. For process set $P \subseteq \Pi$, $(e, f) \sim_P (e', g)$ iff $(e, f) \sim_p (e', g)$ for all $p \in P$. Similarly, the communication system is the same in both points, written $(e, f) \sim_{\mathcal{N}} (e', g)$, iff $e(f, \mathcal{N}) = e'(g, \mathcal{N})$.

Given two executions $e', e \in \mathcal{E}$ and instant $f \in N$, (e', f) and (e, f) are *historically equivalent*, written $(e', f) \equiv (e, f)$, iff the two executions have the same global states through time f : iff, for all $0 \leq g \leq f$, $e'(g) = e(g)$. A point (e', g) *extends* (e, f) , written $(e', g) \geq (e, f)$, iff $(e', f) \equiv (e, f)$ and $g \geq f$. An execution e' *extends* a point (e, f) iff $(e', f) \equiv (e, f)$.

Executions conform to the following informal operational behaviour. At the beginning of time (system initialization), the communication system is empty, and no process has executed any events. Each process executes at most one event between successive observation instants. A message is removed from the communication system if the message is received or lost. Only messages which were sent but not yet removed may appear in the communication system. We describe this behaviour axiomatically in [Maze89].

2.2 Systems of Executions

Informally speaking, one often characterizes the behaviours of a distributed protocol by a set of executions \mathcal{E} over Π and \mathcal{N} (see [HaFa89, Maze89] for more discussion). In order to link communication to knowledge gain (in Section 6.1), we require an execution set to exhibit some natural closure properties which ensure that, if the set represents certain behaviours, then it represents certain other behaviours; these properties capture the ways in which one process's event sequence and the behaviour of the communication system affect another process's event sequence. We call such a closed execution set a *system*. Intuitively, the ability of a process p to execute some event should not depend on the events executed so far by other processes or on the behaviour of the communication system, unless p 's event is a receive—a process can execute a RECV event only if there is an appropriate message in the communication system, and such a message must have been sent by some process. We give the three required closure properties in Appendix A.

2.3 System Characteristics

We now describe the system characteristics we will consider in our analysis. Informally, a system \mathcal{E} is *weakly terminating* if every point of \mathcal{E} can be extended to a point beyond which no process executes any more events in any extension [KoTo88].

Definition 1 A system \mathcal{E} is *weakly terminating* if, for each point $(e, f) \in \mathbf{Pts}(\mathcal{E})$, there is (e', g) extending (e, f) such that, for all $p \in \Pi$ and all $(e'', h) \in \mathbf{Pts}(\mathcal{E})$ extending (e', g) , $e''(h, p) = e'(g, p)$. (e', g) is a *terminating extension* of (e, f) and a *terminating point* of \mathcal{E} . \square

We capture process crash failures and message loss by another set of closure properties. Informally, a system is subject to *process failures* if any process subset may fail at any time. A failure of process p is modelled by a FAIL event in an event sequence for p . We define $\mathbf{fail}(e, 0, P) = \emptyset$, for all $e \in \mathcal{E}$ and $P \subseteq \Pi$ (no process is initially failed). For $f > 0$ and $p \in \Pi$, we collect into $\mathbf{fail}(e, f, P)$ each execution e' in an execution set \mathcal{E} such that ① e' extends $(e, f - 1)$; ② processes other than those in P execute the same events at (e', f) as at (e, f) ; ③ each nonterminated member of P is failed; and ④ any message that $p \in P$ sends at (e, f) does not appear in $e'(f, \mathcal{N})$, and any message that $p \in P$ receives at (e, f) appears in $e'(f, \mathcal{N})$.

Definition 2 A system \mathcal{E} is *subject to process failures* if, for any $(e, f) \in \mathbf{Pts}(\mathcal{E})$: (any process subset may fail) for any $P \subseteq \Pi$, $\mathbf{fail}(e, f, P) \neq \emptyset$. \square

Let $Failed(e, f)$ represent the set of failed processes at point (e, f) :

$$Failed(e, f) = \{p \mid p \in \Pi \text{ and } FAIL \vdash e(f, p)\}.$$

A system is subject to process failures and *recovery* if, at any time, any process subset may fail, and any subset of failed processes may recover.

Definition 3 A system \mathcal{E} is *subject to process failures and recovery* if, for any $(e, f) \in \mathbf{Pts}(\mathcal{E})$:

- the system is subject to process failures, and
- for each nonempty $P \subseteq Failed(e, f)$, there is $(e', g) \in \mathbf{Pts}(\mathcal{E})$ properly extending (e, f) such that $FAIL \not\vdash e'(g, p)$, for all $p \in P$. \square

Some thought will show that, in any system \mathcal{E} subject to process failures and recovery, no terminated process is failed: if $p \in \Pi$ has terminated at $(e, f) \in \mathbf{Pts}(\mathcal{E})$, then $p \notin Failed(e, f)$.

A system is subject to *communication failures* if any subset of messages in transit at any time may be lost. Let M be a subset of the messages in transit at point (e, f) : $M \subseteq e(f, \mathcal{N})$. We collect into the set $lose(e, f, M)$ the set of executions $e' \in \mathcal{E}$ such that e' extends $(e, f - 1)$, $(e', f) \sim_{\Pi}(e, f)$, and $e'(f, \mathcal{N}) = e(f, \mathcal{N}) \setminus M$.

Definition 4 A system \mathcal{E} is *subject to communication failures* if, for any $(e, f) \in \mathbf{Pts}(\mathcal{E})$ and any $M \subseteq e(f, \mathcal{N})$: $lose(e, f, M) \neq \emptyset$. \square

In a system which is subject to *permanent* communication failures, any subset of messages in transit at any time may be lost, *and*, for any subset of processes at any time, it is possible that all messages sent to that subset from that time forward will be lost.

Definition 5 A system \mathcal{E} is *subject to permanent communication failures* if:

- the system is subject to communication failures, and
- for any $(e, f) \in \mathbf{Pts}(\mathcal{E})$ and $P \subseteq \Pi$, there is some e_1 extending (e, f) such that, for all $g \geq f$, $e_1(g, \mathcal{N})[\Pi, P] = \emptyset$. \square

As we shall see, permanent communication failures preclude certain behaviour, such as nonblocking behaviour in commitment systems. What if we assume that communication failures are never permanent, but instead are transient? In a system which is subject to *transient* communication failures, any subset of messages in transit at any time may be lost but, for each execution, there is a point beyond which no more messages are lost. We call (e, f) a *lossless point* if no messages are lost at (e, f) , i.e., if

$$e(f, \mathcal{N}) = e(f - 1, \mathcal{N}) \cup \{ \langle p, \underline{m}, q, f \rangle \mid p, q \in \Pi \text{ and } SEND(\underline{m}, q) \sqsubset (e, f, p) \} \setminus \\ \{ \langle p, \underline{m}, q, i \rangle \mid p, q \in \Pi \text{ and } RECV(\underline{m}, p) \sqsubset (e, f, q) \}.$$

$no_{loss}(e, f)$ is the set of executions which are historically equivalent to e up to time f , except that no messages are lost at time f : $no_{loss}(e, f) = \{e_1 \mid e_1 \text{ extends } (e, f - 1) \text{ such that } (e_1, f) \sim_{\Pi}(e, f) \text{ and } (e_1, f) \text{ is a lossless point}\}$. Note that e is not necessarily in $no_{loss}(e, f)$.

Definition 6 A system \mathcal{E} is *subject to transient communication failures* if

- the system is subject to communication failures
- for all $e \in \mathcal{E}$, there is $f \geq 0$ such that, for all $g \geq f$, $e \in \text{noLoss}(e, g)$. \square

This is a strong assumption about the behaviour of the communication system, because it guarantees, for example, that if any message is sent repeatedly, it can eventually be received [KoTo88]. As we shall see, even though some behaviours which are impossible under permanent communication failures are possible under transient communication failures, certain other system behaviour is still unattainable under transient communication failures.

In the sequel, if a system under discussion is not explicitly identified as being subject to a kind of failure, we assume that the system is free of those failures.

We also model systems in which a message has a maximum lifetime in transit. Informally, a system is *k-transit bounded* if any message sent disappears from the communication system at most k time units after being sent.

Definition 7 A system \mathcal{E} is *k-transit bounded* (for some finite $k \geq 0$) if, for any $\underline{m} \in \underline{M}$, $q, p \in \Pi$, $e \in \mathcal{E}$: if $\text{SEND}(\underline{m}, q) \sqsubset (e, f, p)$, then $\langle p, \underline{m}, q, f \rangle \notin e(f + k, \mathcal{N})$. \square

Round-based protocols typically assume k -transit bounded systems.

3 Problem Specifications and a Logic of Knowledge

To specify a problem to be solved, one gives a set of properties which any protocol solving the problem must exhibit. We will express those properties in the epistemic (knowledge) logic of Halpern and Moses (1990). The intuition used for defining knowledge is based on *possible worlds*: at any given moment, an agent considers several worlds, including the real one, to be possible, because the agent is uncertain of the state of other parts of the system. Informally, we say that, in a given state of the system, an agent p *knows* a fact ϕ if ϕ is true in all worlds p considers possible (that is, in all global states in which p has its current local state). Epistemic specifications are surprisingly common: any problem specification which asserts that a property or value is private to some process *is* an epistemic specification, because it asserts that the property or value depends only on the process's local state (for example, a contractor's bid choice). We are also interested in epistemic propositions to capture assertions on the extent to which a process's local state accurately reflects the system state, such as "the manager knows whether the contractors have bid." The logic also allows one to express temporal properties, in order to capture assertions about the behaviour of a system over time, such as "the protocol eventually terminates" or "the contractor eventually knows the manager's decision."

3.1 A Logic of Knowledge

The language of the logic has the following symbols: a set Φ of primitive propositions; a finite set Π of process names; $\{\neg, \vee, \square, (\cdot, \cdot)\}$; $\{K_x \mid x \in \Pi\}$; and $\{K_X \mid X \subseteq \Pi, X \neq \emptyset\}$. The set of well-formed formulae (or *wffs*) $\mathcal{L}_\Pi(\Phi)$ is the smallest set such that (1) every member of Φ is a well-formed formula, and (2) if ϕ and ψ are well-formed formulae, then so are $(\neg\phi)$, $(\phi \vee \psi)$, $\diamond\phi$, $\square\phi$, $K_x\phi$, $K_X\phi$. We abbreviate $(\neg((\neg\phi) \vee (\neg\psi)))$ by $(\phi \wedge \psi)$ and $((\neg\phi) \vee \psi)$ by $(\phi \supset \psi)$ ³.

We interpret wffs via possible worlds semantics relative to an *interpreted system* (or *model*), a structure $\mathbf{M} = (\mathcal{E}, \mathcal{I})$ in which \mathcal{E} is a system over Π and \mathcal{N} , and $\mathcal{I}: \Phi \rightarrow 2^{\text{Pts}(\mathcal{E})}$ is an interpretation mapping each primitive proposition to the set of points in \mathcal{E} in which the proposition holds. Intuitively, \mathcal{E} represents the set of possible executions of a protocol of interest, and \mathcal{I} interprets primitive propositions of interest with respect to \mathcal{E} .⁴ The points of the system are the possible worlds. Knowledge is based on a complete history interpretation [HaMo90]; that is, each process's view of the system consists of all of the events it has executed, and so each process knows as much as it can—no other encapsulation of a process's state can give a process more knowledge.

Given a model \mathbf{M} , we write $(\mathbf{M}, e, f) \models \phi$ to express that wff ϕ holds in point (e, f) of the model. (If \mathbf{M} is understood from context, we write $(e, f) \models \phi$.) We define \models as follows (assume $\phi, \psi \in \mathcal{L}_\Pi(\Phi)$):

- [Primitives] For $\phi \in \Phi$, $(\mathbf{M}, e, f) \models \phi$ iff $(e, f) \in \mathcal{I}(\phi)$.
- [Negation] $(\mathbf{M}, e, f) \models (\neg\phi)$ iff $(\mathbf{M}, e, f) \not\models \phi$ does not hold.
- [Disjunction] $(\mathbf{M}, e, f) \models (\phi \vee \psi)$ iff $(\mathbf{M}, e, f) \models \phi$ or $(\mathbf{M}, e, f) \models \psi$ (inclusively).
- [Eventually] $(\mathbf{M}, e, f) \models \diamond\phi$ iff, for all $e' \in \mathcal{E}$ such that $(e, f) \equiv (e', f)$, there is some $h \geq f$ such that $(\mathbf{M}, e', h) \models \phi$. (“eventually ϕ ” holds in point (e, f) iff ϕ is true now or will be in any execution extending (e, f) , i.e. iff ϕ will hold at some future point no matter what the future is.)
- [Henceforth] $(\mathbf{M}, e, f) \models \square\phi$ iff, for all $e' \in \mathcal{E}$ such that $(e, f) \equiv (e', f)$, $(\mathbf{M}, e', g) \models \phi$ for all $g \geq f$. (“henceforth ϕ ” holds in point (e, f) iff ϕ holds now and in any possible extension of (e, f) .)
- [Process Knowledge] For $p \in \Pi$, $(\mathbf{M}, e, f) \models K_p\phi$ iff $(\mathbf{M}, e', g) \models \phi$, for all $(e', g) \in \text{Pts}(\mathcal{E})$ such that $(e, f) \sim_p (e', g)$. (“ p knows ϕ ” iff ϕ is true in all points which look to p similar to the current one.)

³In the sequel, we elide the parentheses “(” and “)” in the usual way in formulae in which no ambiguity results. Furthermore, for clarity, we sometimes use “[” for “(” and “]” for “)”. “ $((\phi \supset \psi) \wedge (\psi \supset \phi))$ ” abbreviates “ $\phi \equiv \psi$ ” (read “ ϕ is equivalent to ψ ”). To discuss a formula which appears repeatedly, once for each member of a set of processes or process sets, we use the following abbreviations. For $X = \{x_1, x_2, \dots, x_m\}$, and $\psi_{(x)}$ a wff mentioning x , $\bigwedge_{x \in X} \psi_{(x)}$ is defined as $\psi_{(x_1)} \wedge \psi_{(x_2)} \wedge \dots \wedge \psi_{(x_m)}$; that is, the conjunction of instances of ψ with all appearances of x in each instance of ψ replaced uniformly by an element of X . For example, $\bigwedge_{x \in X} \text{FAILED}_x$ expresses that all of the processes in X are failed. Similarly, $\bigvee_{x \in X} \psi_{(x)}$ is defined as $\psi_{(x_1)} \vee \psi_{(x_2)} \vee \dots \vee \psi_{(x_m)}$. If X is the empty set, then $\bigwedge_{x \in X} \psi_{(x)}$ and $\bigvee_{x \in X} \psi_{(x)}$ are defined to be trivially true.

⁴The interpretation is usually simple and straightforward, based on a mapping of each primitive proposition to the set of global states in which it holds.

[Collective Knowledge] For $P \subseteq \Pi$, $(M, e, f) \models K_P \phi$ iff $(M, e', g) \models \phi$,

for all $(e', g) \in \text{Pts}(\mathcal{E})$ such that $(e, f) \sim_P (e', g)$. (“the members of P collectively know ϕ ” iff ϕ holds in all points which the members of P collectively think possible.)

A wff ϕ is *valid in structure* M , written $M \models \phi$, iff $(M, e, f) \models \phi$ for all points $(e, f) \in \text{Pts}(\mathcal{E})$. A problem specification is a set of wffs, each of which must be valid in any model which purports to solve the problem.

Note that processes in this logic have the *introspection* property: for any wff ϕ , $P \subseteq \Pi$, and model M , $M \models K_P \phi \supset K_P K_P \phi$ and $M \models \neg K_P \phi \supset K_P \neg K_P \phi$.

Lemma 8 states that, if ψ is necessary for ϕ and p knows ϕ , then p knows ψ . The proof of this simple lemma illustrates the use of the possible worlds definition of knowledge.

Lemma 8 For any model M , $p \in \Pi$, wffs ϕ, ψ , if $M \models \phi \supset \psi$, then $M \models K_p \phi \supset K_p \psi$.

Proof: Assume by way of contradiction (bwoc) that there is $(e, f) \in \text{Pts}(\mathcal{E})$ such that $(e, f) \models K_p \phi \wedge \neg K_p \psi$. Therefore, there is $(e_1, g) \in \text{Pts}(\mathcal{E})$ such that $(e_1, g) \sim_p (e, f)$ and $(e_1, g) \models \neg \psi$. By the semantics of knowledge, $(e_1, g) \models \phi \wedge \neg \psi$, violating the antecedent. \square

3.2 Useful Properties of Models

For any system which we will henceforth model, we include in the set of primitive propositions the following ones: ① for each $p \in \Pi$, $FAILED_p$, which is interpreted to mean that p is currently failed; ② $FAILURE$, which is interpreted to mean that a process failure or a communication failure has occurred [Hadz90]; ③ $PROCFAIL$, which is interpreted to mean that any of the processes has failed at some point up to the current one; ④ $INIT$, representing the assertion that the system is in an initial state [Lamp80]; and ⑤ for each $p \in \Pi$, $TERM_p$, which is interpreted to mean that p has terminated. Precisely,

Definition 9 (Standard Interpretation)

Given any model $M = (\mathcal{E}, \mathcal{I})$, \mathcal{I} is a *standard interpretation* iff

$$\mathcal{I}(FAILED_p) = \{(e, f) \mid (e, f) \in \text{Pts}(\mathcal{E}) \text{ and } FAIL \vdash e(f, p)\}.$$

$$\mathcal{I}(PROCFAIL) = \{(e, f) \mid (e, f) \in \text{Pts}(\mathcal{E}) \text{ and there is } p \in \Pi \text{ such that } FAIL \in e(f, p)\}.$$

$$\mathcal{I}(FAILURE) = \{(e, f) \mid (e, f) \in \text{Pts}(\mathcal{E}) \text{ and, for some } p \in \Pi, FAIL \in e(f, p) \text{ or, for some } q, p \in \Pi, \text{ message } \underline{m}, SEND(q, \underline{m}) \in e(f, p), RECV \notin e(f, q), \text{ and } \langle p, \underline{m}, q \rangle \notin e(f, \mathcal{N})\}.$$

$$\mathcal{I}(INIT) = \{(e, 0) \mid e \in \mathcal{E}\}.$$

$$\mathcal{I}(TERM_p) = \{(e, f) \mid \text{for all } e_1 \text{ extending } (e, f), \text{ for all } g \geq f, e_1(g, p) = e(f, p)\}. \square$$

We will henceforth assume that all models are standard interpretations. Note that $INIT$, $PROCFAIL$, $FAILED_p$ for all $p \in \Pi$, and $FAILURE$ are all initially false in any model.

We now identify a set of useful and important properties of formulae in interpreted systems; propositions in the specification and analysis of negotiated commitment will exhibit these properties. We also relate these concepts to each other.

Stable: A wff ϕ is *stable* (in \mathbf{M}) if the following property holds: $\mathbf{M} \models \phi \supset \Box\phi$. A stable wff stays true forever after it becomes true [ChLa85]. Stability is useful for expressing immutable properties and decisions, such as a system deadlock or the choice to commit to a contract. Note that *FAILURE*, *PROCFAIL*, and *TERM_p* are stable.

Local: A formula ϕ is *local to P* (in \mathbf{M}), for $P \subseteq \Pi$, if $\mathbf{M} \models K_P\phi \vee K_P\neg\phi$. That is, P always knows the truth value of ϕ [ChMi86]. Local formulae are intended to model predicates whose value is controlled by or locally testable by the actions of the processes to which the formulae are local. If $P = \{p\}$, we write that ϕ is local to p instead of $\{p\}$. Note that *FAILED_p* and *TERM_p* are local to p .

P -failure-dissociated: A formula ϕ is called *P -failure-dissociated* (in \mathbf{M}), for $P \subseteq \Pi$, if, whenever ϕ is false, ϕ remains false as long as a process in P is failed [Hadz90]. That is, for any $(e, f) \in \text{Pts}(\mathcal{E})$, if $(e, f) \models \neg\phi$ and $\text{FAIL} \dashv e(f+1, p)$, for any $p \in P$, then $(e, f+1) \models \neg\phi$. If $P = \{p\}$, we write that ϕ is *p -failure-dissociated* instead of $\{p\}$ -failure-dissociated.

P -receive-dependent: A formula ϕ is called *P -receive-dependent* (in \mathbf{M}) if, when it is false, it can become true only if some process in P receives a nonnull message from a process not in P [Hadz90]. That is, for any $(e, f) \in \text{Pts}(\mathcal{E})$, if $(e, f) \models \neg\phi$ and $(e, f+1) \models \phi$, then $\text{RCV}(\underline{m}, q) \sqsubset (e, f+1, p)$, for some $p \in P$, $\underline{m} \neq \lambda$, and $q \in \bar{P}$. If $P = \{p\}$, we write that ϕ is *p -receive-dependent* instead of $\{p\}$ -receive-dependent.

Nontrivial: A formula ϕ is called *nontrivial* (in \mathbf{M}) iff, whenever it is false, it could stay false forever; i.e., $\mathbf{M} \models \neg\phi \supset \neg\Diamond\phi$. That is, for any $(e, f) \in \text{Pts}(\mathcal{E})$, if $(e, f) \models \neg\phi$, then there is $e' \in \mathcal{E}$ such that e' extends (e, f) and $(e', g) \models \neg\phi$ for all $g > f$.

Pointwise nontrivial: A formula ϕ is called *pointwise nontrivial* (in \mathbf{M}) iff, whenever it is false, it may remain false in the next time instant; i.e., for any $(e, f) \in \text{Pts}(\mathcal{E})$, if $(e, f) \models \neg\phi$, then there is $e' \in \mathcal{E}$ such that e' extends (e, f) and $(e', f+1) \models \neg\phi$.

Note that any nontrivial formula is perforce pointwise nontrivial, but not vice versa. Lemma 10 shows that a nontrivial formula is eventually true exactly when it is true (note that Lemma 10 does not hold for pointwise nontrivial formulae).

Lemma 10 Let \mathbf{M} be a model, ϕ a nontrivial formula, and $(e, f) \in \text{Pts}(\mathcal{E})$.

Then $(e, f) \models \phi$ iff $(e, f) \models \Diamond\phi$.

Proof: $\phi \supset \Diamond\phi$ holds by definition. $\Diamond\phi \supset \phi$ holds by contraposition of the definition of nontrivial. \square

Intuitively, locality means that the proposition is “about” the process set to which the proposition is local. For example, if we were modelling the outcome of a coin toss by process p , and wff ϕ represents the proposition “ p has flipped a heads”, then we expect ϕ to be local to p .

Furthermore, we do not expect ϕ to become true while p is failed, so ϕ is p -failure-dissociated. We expect the outcome of p 's coin toss to be fair and not forced to be either heads or tails, so ϕ is nontrivial (and, therefore, pointwise nontrivial). Finally, if another process q must receive a message sent by p after the flip in order for q to learn that p flipped a heads, then the proposition $K_p\phi$ is q -receive-dependent.

As the following lemma shows, these concepts are strongly related.

Lemma 11 Given any model M , wff ϕ , and $P \subseteq \Pi$, ϕ is pointwise nontrivial in M if either of the following holds:

- M is subject to process failures and ϕ is P -failure-dissociated.
- M is subject to communication failures and ϕ is P -receive-dependent.⁵

Further, if M is subject to process failures and ϕ is P -receive-dependent, then ϕ is P -failure-dissociated. \square

In the interest of space, we omit this proof. The reader may find this and any other omitted proofs in [Maze89].

Theorem 12 states that, in any weakly terminating system which is either (1) subject to communication failures or (2) k -transit bounded and subject to process failures, every point can be extended to a terminating point without any process receiving any further messages. Koo and Toueg (1988) showed this for weakly terminating systems subject to communication failures; instead of communication failures, we use the combination of k -transit boundedness and process failures to ensure that messages may disappear without being received.

Theorem 12 Let M be either (1) a weakly terminating model subject to communication failures, or (2) a weakly terminating model which is k -transit bounded and subject to process failures. For any $(e, f) \in \mathbf{Pts}(\mathcal{E})$, there is a terminating extension (e_1, g) such that no process receives a nonnull message after (e_1, f) .

Proof: Koo and Toueg (1988) showed the result for clause (1) as Theorem 3.1. (They proved the result for asynchronous systems, but they note that the result holds for a system with any synchrony property. Further, they prove the result for initial points $(e, 0)$ for all $e \in \mathcal{E}$, but the result generalizes for all points.) [Maze89, Theorem 4.2] shows the result for clause (2). \square

Lemma 13 characterizes some system conditions under which a receive-dependent wff is nontrivial and, therefore, eventually holds only when it holds already.

Lemma 13 In any model M which is

- (1) subject to permanent communication failures,
- (2) weakly terminating and subject to communication failures, or

⁵This also holds for models with asynchronous processes and ϕ local to P . We did not formally define models with asynchronous processes, so we leave that case out of the statement of this result.

(3) weakly terminating, subject to process failures and recovery, and k -transit bounded,

any wff which is q -receive-dependent, for some $q \in \Pi$, is nontrivial in \mathbf{M} . (Consequently, by Lemma 10, $\mathbf{M} \models \diamond\phi \equiv \phi$.)

Proof: Pick any $(e, f) \in \text{Pts}(\mathcal{E})$ such that $(e, f) \models \neg\phi$. (If there is none, then ϕ is valid and, therefore, nontrivial.) Because the system is subject to one of the three conditions, there is $e_1 \in \mathcal{E}$ extending (e, f) such that $(e_1, g) \models \neg\phi$, for all $g > f$ (because q does not receive at or after $(e_1, f + 1)$ any message which would establish ϕ ; this is possible under each condition: (1) because the message may be lost and no more messages received by q , (2) by Theorem 12, or (3) by Theorem 12.) Therefore, ϕ is nontrivial. \square

4 Specification of Negotiated Commitment

The specification is a set of propositions which must be valid in the model of a system induced by a protocol that solves the problem.⁶ We call any such model a C -system.

We divide the processes in the system into two disjoint sets: the manager, $\{m\}$, and the *contractors*, or bidders, \mathcal{C} . Informally, each of the contractors chooses whether to bid or not on an announced contract. The manager selects from among the bidding contractors to establish a *dependency set*, representing those contractors which the manager wants to commit to performing the announced task; contractors not in the dependency set must not carry out the task. We represent contractor c 's choice to bid by a primitive proposition BID_c ; we represent its choice not to bid by $NO-BID_c$. We represent the manager's possible dependency set choices by the primitive propositions $DEPEND_m^x$ for each nonempty $x \subseteq \mathcal{C}$. For each $c \in \mathcal{C}$, we define the allowed dependencies set $\mathcal{D}_c \subseteq \{x \mid x \in 2^{\mathcal{C}} \text{ and } c \in x\}$. $NOT-CHOSE_m^c$ represents the manager's choice not to make c a codependent. The manager records locally a decision outcome for each contractor, either $AWARD_m^c$, representing that m expects c to carry out the task, or $REJECT_m^c$, representing that m expects c not to carry out the task. Similarly, each contractor records locally a decision outcome, either $ACCEPT_c$, representing that c will carry out the contract, or $REFUSE_c$, that c will not carry out the contract. Informally, the processes reach consistent commitment if, for each $c \in \mathcal{C}$, the manager decides $AWARD_m^c$ and c decides $ACCEPT_c$, or m decides $REJECT_m^c$ and c decides $REFUSE_c$. Each BID_c , $NO-BID_c$, $ACCEPT_c$, and $REFUSE_c$ proposition is stable and local to c ; BID_c is also c -failure-dissociated. Each $DEPEND_m^x$, $NOT-CHOSE_m^c$, $AWARD_m^c$, and $REJECT_m^c$ proposition is stable and local to m ; each $DEPEND_m^x$ is also m -failure-dissociated. All of these propositions are initially false.

A C -system is an interpreted system, with the primitive propositions described above, which satisfies the following additional properties of negotiated commitment under process or communication failures.

⁶The specification of negotiated commitment is more complicated than that of atomic commitment (which requires ten properties), because of the possibility of "subatomic" dependencies and the singularity of the coordinator.

Necessity properties

Dependent Acceptance: For all $c \in \mathcal{C}$, $M \models \text{ACCEPT}_c \supset \bigvee_{x \in \mathcal{D}_c} \text{DEPEND}_m^x$.

(An accepted contractor must be a codependent.)

Dependent Award: For all $c \in \mathcal{C}$, $M \models \text{AWARD}_m^c \supset \bigvee_{x \in \mathcal{D}_c} \text{DEPEND}_m^x$.

(An awarded contractor must be a codependent.)

No Unilateral Dependencies: For all $c \in \mathcal{C}$, $M \models \bigvee_{x \in \mathcal{D}_c} \text{DEPEND}_m^x \supset \text{BID}_c$.

(A codependent must have bid.)

No Predetermined Bids: For all $c \in \mathcal{C}$, $M \models \text{INIT} \supset \neg(\text{BID}_c \vee \text{NO-BID}_c)$.

(No contractor starts with its bid choice made.)

No Predetermined Dependencies: For all $c \in \mathcal{C}$,
 $M \models \text{INIT} \supset \neg(\bigvee_{x \in \mathcal{D}_c} \text{DEPEND}_m^x \vee \text{NOT-CHOSE}_m^c)$.

(The manager starts without having made any dependency choices.)

Exclusivity properties

Exclusive Bid: For all $c \in \mathcal{C}$, $M \models \neg(\text{BID}_c \wedge \text{NO-BID}_c)$.

(A contractor may choose only one of the two bidding options.)

Exclusive Dependencies: For all $c \in \mathcal{C}$,

$M \models \neg(\text{NOT-CHOSE}_m^c \wedge (\bigvee_{x \in \mathcal{D}_c} \text{DEPEND}_m^x))$.

(The manager may not both exclude c from any dependency set in \mathcal{D}_c and include c in a dependency set.)

Nonintersecting Dependencies:

For each $c \in \mathcal{C}$, for each pair $x, y \in \mathcal{D}_c$ such that $x \neq y$,

$M \models \neg(\text{DEPEND}_m^x \wedge \text{DEPEND}_m^y)$.

(c may be involved in at most one dependency set at any time in any one negotiation.)

Total Decision Harmony: For all $c \in \mathcal{C}$,

$M \models \neg(\text{AWARD}_m^c \wedge \text{REFUSE}_c)$

$M \models \neg(\text{REJECT}_m^c \wedge \text{ACCEPT}_c)$

(The manager and each contractor can never decide inconsistently.)

$M \models \neg(\text{AWARD}_m^c \wedge \text{REJECT}_m^c)$

$M \models \neg(\text{ACCEPT}_c \wedge \text{REFUSE}_c)$.

(Only one of two possible decisions is allowed for each process.)

For all $x \in \bigcup_{c \in \mathcal{C}} \mathcal{D}_c$, $c, d \in x$,

$$\mathbf{M} \models \text{DEPEND}_m^x \supset \neg(\text{AWARD}_m^c \wedge \text{REJECT}_m^d).$$

(if contractors c and d are both in the dependency set x ,
then m cannot decide for them inconsistently ...)

$$\mathbf{M} \models \text{DEPEND}_m^x \supset \neg(\text{ACCEPT}_c \wedge \text{REFUSE}_d)$$

(... and c and d cannot decide inconsistently with each other ...)

$$\mathbf{M} \models \text{DEPEND}_m^x \supset \neg(\text{AWARD}_m^c \wedge \text{REFUSE}_d).$$

(... and m cannot award to c and a codependent d refuse ...)

$$\mathbf{M} \models \text{DEPEND}_m^x \supset \neg(\text{REJECT}_m^c \wedge \text{ACCEPT}_d).$$

(... and m cannot reject c and a codependent d accept.)

Nontriviality properties

Nontrivial Process Failure:

$$\text{For all } P \subseteq \Pi, \mathbf{M} \models \neg \forall p \in P \text{ FAILED}_p \supset \neg \diamond \forall p \in P \text{ FAILED}_p.$$

(If a process is not failed, then it need not fail.)

Nontrivial System Failure:

$$\mathbf{M} \models \neg \text{FAILURE} \supset \neg \diamond \text{FAILURE}.$$

(If no failure has yet occurred, then a failure does not have to occur.)

Jointly Nontrivial Bid Choice:

$$\text{For all } c \in \mathcal{C}, \mathbf{M} \models (\neg \text{BID}_c \wedge \neg \text{NO-BID}_c) \supset (\neg \diamond \text{BID}_c \wedge \neg \diamond \text{NO-BID}_c).$$

(If c has not yet chosen whether to bid, then both bid choices are open.)

Jointly Nontrivial Dependencies:

$$\text{For all } c \in \mathcal{C}, \mathbf{M} \models (\neg \forall x \in \mathcal{D}_c \text{ DEPEND}_m^x \wedge \neg \text{NOT-CHOSE}_m^c) \supset$$

$$(\neg \diamond \forall x \in \mathcal{D}_c \text{ DEPEND}_m^x \wedge \neg \diamond \text{NOT-CHOSE}_m^c)$$

(If the manager has not yet chosen to make c a codependent, then m is not forced
either to make c a codependent or to ensure c will not be a codependent.)

System-failure-free Dependency Mix:

For all nonempty $Q \subseteq \mathcal{C}$, $P \subseteq Q$,

$$\mathbf{M} \models (\neg \text{FAILURE} \wedge [\wedge_{c \in Q} (\text{BID}_c \wedge \neg [\forall x \in \mathcal{D}_c \text{ DEPEND}_m^x \vee \text{NOT-CHOSE}_m^c])]) \supset$$

$$(\neg \square \neg [\neg \text{FAILURE} \wedge (\wedge_{c \in P} [\forall x \in \mathcal{D}_c \text{ DEPEND}_m^x]) \wedge (\wedge_{c \in Q \setminus P} \text{NOT-CHOSE}_m^c)]).$$

(If a system failure has not yet occurred and m has not yet made its dependency choices
about some bidding subset Q of \mathcal{C} , then, for all subsets P of those contractors, there is
an extension in which a system failure still has not occurred and each member P is a
codependent and each member of Q not in P is not chosen.)

Decision completion properties

Failure or Decision: $M \models \diamond(\text{FAILURE} \vee$
 $[\wedge_{c \in \mathcal{C}}(\text{BID}_c \vee \text{NO-BID}_c) \wedge$
 $([\vee_{x \in \mathcal{D}_c} \text{DEPEND}_m^x] \wedge [\text{AWARD}_m^c \wedge \text{ACCEPT}_c]) \vee$
 $(\text{NOT-CHOSE}_m^c \wedge [\text{REJECT}_m^c \wedge \text{REFUSE}_c])])$
 $]$).

(If there are no failures, then all contractors should make a bid choice, all codependents should establish commitments, and all noncodependents should establish “noncommitments”.)

Post-Failure Termination: For all $(e, f) \in \text{Pts}(\mathcal{E})$,
 if $\text{Failed}(e, f) = \emptyset$ and $\text{noLoss}(e, f) \neq \emptyset$, then there is $e_1 \in \text{noLoss}(e, f)$ and $h \in N$ such
 that there are no process or communication failures in (e_1, g) for $f \leq g \leq h$
 (i.e., $\text{Failed}(e_1, g) = \emptyset$ and $e_1 \in \text{noLoss}(e_1, g)$) and
 $(e_1, h) \models \wedge_{c \in \mathcal{C}}(\text{ACCEPT}_c \vee \text{REFUSE}_c) \wedge \wedge_{c \in \mathcal{C}}(\text{AWARD}_m^c \vee \text{REJECT}_m^c)$.

(If there are presently no failures, then it is possible for no process or communication failures to occur for sufficiently long that all processes decide.)

The following resulting properties of C-systems are straightforward: if no process has failed yet, then no process is forced to fail; there are executions without failures; for each contractor $c \in \mathcal{C}$, there is an execution in which c establishes commitment without any system failures having occurred and an execution in which c establishes noncommitment (i.e., refuses) without any system failures having occurred; for each $c \in \mathcal{C}$, neither commitment nor noncommitment is predetermined; establishing commitment and establishing noncommitment is each possible for each contractor in any C-system; each of the BID_c , NO-BID_c , DEPEND_m^x , and NOT-CHOSE_m^c propositions is nontrivial (and therefore each is pointwise nontrivial); and $\vee_{x \in \mathcal{D}_c} \text{DEPEND}_m^x$ is m -failure-dissociated and pointwise nontrivial.

5 Initial Knowledge Analysis

Given the specification of negotiated commitment, we now wish to determine levels of knowledge which each process needs to commit. It is straightforward to show the following simple knowledge requirements. The first one, for example, states that, for a dependency set x which includes contractor c , if a process p knows that the dependency set is established, then p knows that c bid.

Lemma 14 For any C-system M , $c \in \mathcal{C}$, $p \in \Pi$,

1. $M \models K_p(\vee_{x \in \mathcal{D}_c} \text{DEPEND}_m^x) \supset K_p \text{BID}_c$.
2. $M \models K_p \text{AWARD}_m^c \supset K_p(\vee_{x \in \mathcal{D}_c} \text{DEPEND}_m^x)$.

3. $M \models K_p \text{AWARD}_m^c \supset K_p \text{BID}_c$.
4. $M \models K_p \text{ACCEPT}_c \supset K_p (\forall x \in \mathcal{D}_c \text{DEPEND}_m^x)$.
5. $M \models K_p \text{ACCEPT}_c \supset K_p \text{BID}_c$.

Proof: Follows from: (1) Lemma 8 and **No Unilateral Dependencies**, (2) Lemma 8 and **Dependent Awards**, (3) items 2 and 1, (4) Lemma 8 and **Dependent Acceptance**, and (5) items 4 and 1. \square

This matches our intuition about the problem; the important point is that we are able to formalize and validate that intuition directly.⁷ These knowledge requirements are also enough to show the message lower bound. As we shall see in following sections, the knowledge requirements (and the corresponding communication requirements) are not always so simple.

6 Communication Requirements

One of the goals of a problem analysis is to determine the message passing structure of protocols to solve the problem. We now derive some communication requirements for commitment solutions, based on the knowledge requirements of Lemma 14. First, we show the following underlying communication structure for any negotiated commitment protocol: if $P \subseteq \mathcal{C}$ contractors have accepted, then there must have been a message chain from each $c \in P$ to m and a *subsequent* message chain from m to each c . Then we show the following message lower bound for commitment: if some subset P of contractors has accepted, then the number of nonnull RECVs is at least twice the number of contractors which accepted; and if the manager has awarded to some subset P of contractors, then the number of nonnull RECV events is at least the number of awarded contractors.

To get these results, we need a theorem given by Mazer (1989) which identifies circumstances under which processes in faulty distributed systems must communicate for one process to gain knowledge about another. One can use this theorem as a high-level link between knowledge and communication; the theorem hides detailed, combinatorial arguments from the high-level view.

6.1 Message Chain Theorem

Informally, this result says that, if at some time a proposition ϕ about process p is false and at some later time another process q knows that ϕ is true, then q received a message through some chain of message passing which originated at p , given one of the following conditions: processes

⁷The analogous result for atomic commitment is that, for process p to commit, p must know that every site voted to commit the transaction [Hadz90].

can crash-fail and ϕ cannot become true while p is failed; or messages can be lost and ϕ is never forced to become true.⁸ The result is formalized as follows.

Given an execution e , a *message chain from process p to process q in interval (e, f) to (e, g)* is a sequence of send/receive pairs such that ($f \leq f_1$; $f_i < f_{i+1}$, for $1 \leq i < 2n$; $f_{2n} \leq g$):

SEND(\underline{m}_1, p_1) \sqsubset (e, f_1, p); RECV(\underline{m}_1, p) \sqsubset (e, f_2, p_1); SEND(\underline{m}_2, p_2) \sqsubset (e, f_3, p_1);

RECV(\underline{m}_2, p_1) \sqsubset (e, f_4, p_2); ... SEND(\underline{m}_n, q) \sqsubset (e, f_{2n-1}, p_{n-1}); RECV(\underline{m}_n, p_{n-1}) \sqsubset (e, f_{2n}, q).

The abbreviation $P \xrightarrow{+} Q$ indicates a message chain of length at least one from P to Q (execution and interval will be clear from context). We abbreviate $\{p\} \xrightarrow{+} \{q\}$ as $p \xrightarrow{+} q$.

For any model M , $e \in \mathcal{E}$, nonempty process sets $P \subset \Pi$ and $Q \subset \Pi$ such that $P \cap Q = \emptyset$, and wff ϕ local to P , a ϕ -message chain from P to Q in interval (e, f) to (e, i) is a message chain from some $p \in P$ to some $q \in Q$ in an interval (e, g) to (e, i) such that $f < g$, $(M, e, g-1) \models \neg\phi$, $(M, e, g) \models \phi$, and $(M, e, i) \models K_Q\phi$.

Here is the knowledge gain result.

Theorem 15 (The Message Chain Theorem)[Maze89, Maze90]

Fix a model M , any nonempty $P \subset \Pi$ and $Q \subset \Pi$ such that $P \cap Q = \emptyset$, and wff ϕ local to P in M . Further, let one of the following two conditions hold: (1) M is subject to process failures and ϕ is P -failure-dissociated in M , or (2) M is subject to communication failures and ϕ is pointwise nontrivial in M . Fix point (e, f) in \mathcal{E} and $i > f$ such that $(M, e, f) \models \neg\phi$ and $(M, e, i) \models K_Q\phi$.

Then there is a ϕ -message chain from P to Q in (e, f) to (e, i) . \square

Recall from Lemma 11 that, if ϕ is P -failure-dissociated in a model which is subject to process failures, then ϕ is pointwise nontrivial. Pointwise nontriviality is a key concept in understanding the Message Chain Theorem. Intuitively, pointwise nontriviality causes uncertainty; Q requires a ϕ -message chain in order to learn ϕ because Q must be able to distinguish between worlds in which ϕ holds and those in which ϕ does not hold (these latter worlds are possible because ϕ earlier did not hold and was not forced to hold). Without the ϕ -message chain, Q has no basis upon which to make the required distinction. See [Maze89, Maze90] for the detailed proof.

Corollary 16 and Lemma 17 below use the Message Chain Theorem and some earlier lemmas to prove results about knowledge gain in three kinds of systems which we will examine again later (in the context of commitment systems).

Corollary 16 Let M be any model which is either (1) subject to process failures or (2) subject to communication failures. For any $P \subset \Pi$ and wff ϕ which is local to $Q \subseteq \bar{P}$, Q -receive-dependent, and initially false, $K_P\phi$ is P -receive-dependent.

Proof: We treat the two cases separately:

⁸Chandy and Misra (1986) showed such a result for systems with asynchronous processes, regardless of failures. We use a result which holds for systems with failures, regardless of synchrony.

process failures: By Lemma 11, ϕ is Q -failure dissociated. Therefore, by the Message Chain Theorem, ϕ requires a message chain $Q \xrightarrow{\phi} P$, so $K_P\phi$ is P -receive-dependent.

communication failures: By Lemma 11, ϕ is pointwise nontrivial in \mathbf{M} . Then, by the Message Chain Theorem, ϕ requires $Q \xrightarrow{\phi} P$. \square

Lemma 17 In any model \mathbf{M} which is

1. subject to permanent communication failures,
2. weakly terminating and subject to communication failures, or
3. weakly terminating, subject to process failures and recovery, and k -transit bounded,

for any $P \subset \Pi$ and wff ϕ which is local to $Q \subseteq \bar{P}$, Q -receive-dependent, and initially false, $K_P\Diamond\phi$ is P -receive-dependent.

Proof: By Corollary 16, $K_P\phi$ is P -receive-dependent. By Lemmas 13 and 10, $\phi \equiv \Diamond\phi$. Therefore, $K_P\Diamond\phi$ is P -receive-dependent. \square

6.2 Communication Structure and Message Lower Bound

We now determine the communication structure, and a lower bound on the number of messages required (excluding the contract announcements) to establish commitment, in any negotiated commitment protocol. These results are important, because they tell the protocol designer that any protocol that supports negotiated commitment must ensure that at least the lower bound number of messages passes among processes, according to the determined communication structure; further, the propositional content of these messages comes from the knowledge requirements.

Lemma 14 identified some of the knowledge a contractor needs to accept or a manager needs to award. We now use the Message Chain Theorem, **No Predetermined Bids**, **No Predetermined Dependencies**, the pointwise nontriviality of BID_c and $\bigvee_{x \in \mathcal{D}_c} DEPEND_m^x$, the specification of the primitive propositions, and the failure assumptions for C-systems, to derive communication requirements from the knowledge requirements. It is easy to show, for systems with process failures or communication failures, that, for process p to know that contractor c bid, there must be a message chain from c to p (Lemma 18), and for c to know that m selected c as a codependent, there must be a subsequent message chain from m to c (Lemma 19).

Lemma 18 For any C-system \mathbf{M} , $c \in \mathcal{C}$, $p \in \Pi$ such that $p \neq c$, $e \in \mathcal{E}$, and $i \in N$, if $(e, i) \models K_p BID_c$, then there is a BID_c -message chain $c \xrightarrow{+} p$ in $(e, 0)$ to (e, i) . \square

Lemma 19 For any C-system \mathbf{M} , $c \in \mathcal{C}$, $p \in \Pi$ such that $p \neq c$, and $(e, i) \in \text{Pts}(\mathcal{E})$, if $(e, i) \models K_p \bigvee_{x \in \mathcal{D}_c} DEPEND_m^x$, then there is a $\bigvee_{x \in \mathcal{D}_c} DEPEND_m^x$ -message chain $m \xrightarrow{+} p$ in $(e, 0)$ to (e, i) . \square

Therefore, each of $K_p BID_c$ and $K_p \vee_{x \in \mathcal{D}_c} DEPEND_m^x$ is p -receive-dependent.

From this simple analysis and Lemma 14, we get the underlying communication structure of any protocol which supports negotiated commitment: if $P \subseteq \mathcal{C}$ contractors have accepted, then there must have been a message chain from each $c \in P$ to m and a *subsequent* message chain from m to each c .⁹ Adapting the *linear two-phase protocol* for atomic commitment (see [Gray79, BeHG87]) to negotiated commitment illustrates that the message chains required may overlap; for example, the message chain from the manager to a contractor can include as a subchain the message chain from the manager to a contractor earlier in the linear order. For atomic commitment, the analogous result is that, for p to commit, there must be a message chain from every other process to p ; further, for the others to commit, there must be a message chain from p to each other process. Centralized and linear two-phase commit protocols illustrate that the chains may overlap or converge through a single coordinator; the decentralized two-phase commit protocol illustrates that the message chains may be independent. Further, all of the “flexible” two-phase atomic commitment protocols discussed informally by Bürger (1989) implicitly respect this underlying communication structure.

Finally, we can get our lower bound result. First, if some subset P of contractors has accepted, then the number of nonnull RECVs is at least twice the number of contractors which accepted. Further, if the manager has awarded to some subset P of contractors, then the number of nonnull RECV events is at least the number of awarded contractors. Note that the proof is couched in terms of the high-level concepts of knowledge and message chains; much of the combinatorial detail is hidden under these concepts.

Theorem 20 For any C-system M , $(e, f) \in \mathbf{Pts}(\mathcal{E})$, $P \subseteq \mathcal{C}$,

1. if $(e, f) \models \wedge_{c \in P} ACCEPT_c$,
then the number of nonnull RECV events in (e, f) is at least $2 | P |$.
2. if $(e, f) \models \wedge_{c \in P} AWARD_m^c$,
then the number of nonnull RECV events in (e, f) is at least $| P |$.

Proof:

1. We note that $(e, f) \models \wedge_{c \in P} (K_c \vee_{x \in \mathcal{D}_c} DEPEND_m^x)$ (by locality of $ACCEPT_c$ and Lemma 14) and that a $\vee_{x \in \mathcal{D}_c} DEPEND_m^x$ -message chain exists from m to each c (by Lemma 19). Furthermore, because $M \models (\vee_{x \in \mathcal{D}_c} DEPEND_m^x) \supset K_m BID_c$, there must be a BID_c -message chain $c \xrightarrow{+} m$ for each c (by Lemma 18). The $\vee_{x \in \mathcal{D}_c} DEPEND_m^x$ -message chain

⁹One could extend this analysis as follows. It is easy to show that $M \models K_p DEPEND_m^x \supset \wedge_{c \in x} K_p BID_c$. Therefore, when dependency set x has accepted, for each pair $c, d \in x$, $K_c BID_d$ holds and $K_d BID_c$ holds. Therefore there must be a BID_c -message chain from c to d and a BID_d -message chain from d to c . We do not pursue this line of reasoning, because the BID_c -message chain from c to d is embedded in the BID_c -message chain from c to m concatenated with the $DEPEND_m^x$ -message chain from m to d (and symmetrically from d to c).

from m to any c must *strictly follow* the BID_c -message chain from that c to m (i.e., there are consecutive message chains $c \xrightarrow{+} m \xrightarrow{+} c$)¹⁰. Therefore, each $c \in P$ must send a message which is received (along c 's BID_c -message chain to m), and each $c \in P$ must receive a nonnull message (along the $\forall_{x \in \mathcal{D}_c} DEPEND_m^x$ -message chain from m).

Let $P = \{c_1, c_2, \dots, c_k\}$. For each $c_i \in P$, call the message it sends on its BID_{c_i} -message chain to m $\underline{m}_{c_i, B}$; also call the message it receives on the $\forall_{x \in \mathcal{D}_{c_i}} DEPEND_m^x$ -message chain from m $\underline{m}_{c_i, D}$. Therefore, in (e, f) , there are the following nonnull, received messages:

$$\underline{m}_{c_1, B}, \underline{m}_{c_2, B}, \dots, \underline{m}_{c_k, B},$$

and

$$\underline{m}_{c_1, D}, \underline{m}_{c_2, D}, \dots, \underline{m}_{c_k, D}.$$

Therefore, there are $2 | P |$ distinct, nonnull messages received unless, for some distinct $c_i, c_j \in P$, $\underline{m}_{c_i, B} = \underline{m}_{c_j, D}$; that is, if there are fewer than $2 | P |$ nonnull messages received, then there must be at least one c_i, c_j pair such that the bid message sent by c_i is the dependency message received by c_j . We now show that, for such a c_i, c_j pair, either $(e, f) \models \neg ACCEPT_{c_i}$, contradicting the statement of the theorem, or other messages must be received in (e, f) .

$\underline{m}_{c_j, D}$ is the message on the $\forall_{x \in \mathcal{D}_{c_j}} DEPEND_m^x$ -message chain from m to c_j which allows c_j to attain $K_{c_j}(\forall_{x \in \mathcal{D}_{c_j}} DEPEND_m^x)$. Therefore, c_i receives a nonnull message \underline{m} on the $(\forall_{x \in \mathcal{D}_{c_j}} DEPEND_m^x)$ -message chain from m to c_j . We note that $\underline{m} \neq \underline{m}_{c_i, D}$, because c_i receives \underline{m} before initiating the BID_{c_i} -message chain from c_i to m (with $\underline{m}_{c_i, B}$), and c_i must receive $\underline{m}_{c_i, D}$ *after* sending $\underline{m}_{c_i, B}$. Thus, we have an additional nonnull message \underline{m} received, which compensates for the distinct message “lost” by the fact that $\underline{m}_{c_i, B} = \underline{m}_{c_j, D}$.

Therefore, we conclude that at least $2 | P |$ nonnull messages are received.

2. We know that $(e, f) \models \bigwedge_{c \in P} K_m BID_c$ (by Lemma 14). Therefore, there is a BID_c -message chain $c \xrightarrow{+} m$ in $(e, 0)$ to (e, f) for each $c \in P$. Therefore, each $c \in P$ must send a nonnull message which is received (along $c \xrightarrow{+} m$), so there are at least $| P |$ nonnull messages received. \square

For an atomic commitment, one in which all contractors commit, $| P | = n - 1$; Theorem 20 tells us that any execution in which all participants decide to commit requires at least $2(n - 1)$ nonnull messages received. This matches the known result for atomic commitment, given first by Dwork and Skeen (1983). To show their lower bound result, Dwork and Skeen (1983) use a tightly synchronous computation model with permanent process failures and an argument based on the message passing graphs produced by a “best-case,” failure-free instance of an atomic commitment protocol. We suggest that the knowledge-theoretic approach yields a more

¹⁰This is true because, by the definition of a $\forall_{x \in \mathcal{D}_c} DEPEND_m^x$ -message chain, $\forall_{x \in \mathcal{D}_c} DEPEND_m^x$ must hold at the point (e, f) when m sends the first message in the chain. In that case, $(e, f) \models K_m BID_c$, and thus the RECV by which m learns BID_c must occur earlier than time f .

elegant and intuitive proof and a more generally applicable result (applicable under process failures, communication failures, or asynchrony). Hadzilacos (1990) also gives a knowledge-theoretic proof of this result for atomic commitment. Although his proof differs significantly in approach from ours, Hadzilacos also determines requisite knowledge levels for decision and message passing requirements for attaining the required knowledge, from which the lower bound follows.

7 Impossibility Results

An impossibility result proves that no protocol can guarantee the behaviour addressed in the result; for example, we will soon show that no protocol can guarantee that an undecided process, recovering from a failure, can decide consistently without receiving further messages. Impossibility results save the protocol designer from the futile effort of writing a protocol to support the desired behaviour.

All of our impossibility proofs have the same form: ① determine that the desired commitment behaviour requires arbitrarily deeply nested knowledge in the specific type of system; ② determine that establishing that knowledge requires arbitrarily many consecutive message chains in any protocol; and ③ argue that the communication is unattainable. This common structure demonstrates the power of the knowledge-theoretic approach.

7.1 Independent Recovery

Independent recovery is the ability of a process to decide consistently, upon recovery from a process failure, without executing any nonnull receive events. Independent recovery is desirable, because it allows failed processes, if they recover, to decide consistently based on local state, without blocking and without communicating with others; processes that are not failed can ignore failed ones. The lack of independent recovery means that at least one “live” process must have the knowledge and longevity to assist recovering processes in deciding, regardless of how long those processes can remain failed.

In this section, we show that no negotiated commitment protocol can support independent recovery. The proof of this result proceeds as follows. First, we show that all processes have decided at any terminating point of a system which supports independent recovery. Then we derive knowledge levels required to establish a commitment (award or accept) in such a system. Then we show that certain sequences of consecutive message chains are needed to attain the relevant knowledge levels. Finally, we argue that establishing a commitment in a system which supports independent recovery requires an infinite sequence of consecutive message chains.

In particular, we show that, in a C-system which supports independent recovery, ① an accepting contractor c must have arbitrarily deeply nested knowledge about the manager’s knowledge about c ’s knowledge that m made c a codependent, and ② an awarding manager must have arbitrarily deeply nested knowledge about c ’s knowledge that m made c a codependent.

We show this, in Lemma 25, by induction on the knowledge nesting level. Then the Message Chain Theorem allows us to show that the required knowledge cannot be gained in finite time (Theorem 27), by showing that arbitrarily many consecutive message chains are needed to gain the required knowledge (Lemma 26). In order to show Lemma 25, we must show ① how m 's award to c depends on c 's acceptance knowledge and c 's acceptance depends on m 's award knowledge (Lemma 23), and ② that each of the nested levels of knowledge in Lemma 25 can be attained only by message receipt (Lemma 24). Lemma 23 shows that ① if c must know a c -receive-dependent proposition ϕ in order to accept, then an awarding m must know that c knows ϕ , and ② if m must know some m -receive-dependent proposition ϕ in order to award, then, in order to accept, c must know that m knows ϕ . Lemma 23 and Lemma 24 allow us to show the interleaved knowledge requirement in Lemma 25, using, as a basis, the fact that an accepting c must know that it is a codependent (Lemma 14).

Definition 21 A C-system \mathbf{M} *supports independent recovery* if \mathbf{M} is subject to process failures and recovery, and for all $(e, f) \in \text{Pts}(\mathcal{E})$, $f > 0$,

- if, for $c \in \mathcal{C}$, $\text{FAIL} \dashv e(f-1, c)$ and $\text{FAIL} \not\vdash e(f, c)$, then either

- ① there is $g \geq f$ such that $(e, g) \models \text{ACCEPT}_c \vee \text{REFUSE}_c$
and $\text{RCV}(\underline{m}, p) \notin e(g, c) - e(f-1, c)$
for all messages $\underline{m} \neq \lambda$ and $p \in \Pi \setminus \{c\}$, or
- ② there is a $g \geq f$ such that $\text{FAIL} \dashv e(g, c)$

and

- if $\text{FAIL} \dashv e(f-1, m)$ and $\text{FAIL} \not\vdash e(f, m)$, then either

- ① there is $g \geq f$ such that $(e, g) \models \bigwedge_{c \in \mathcal{C}} (\text{AWARD}_m^c \vee \text{REJECT}_m^c)$
and $\text{RCV}(\underline{m}, p) \notin e(g, m) - e(f-1, m)$
for all messages $\underline{m} \neq \lambda$ and $p \in \Pi \setminus \{m\}$, or
- ② there is a $g \geq f$ such that $\text{FAIL} \dashv e(g, m)$. \square

We first prove that, in a C-system subject to process failures and recovery, all processes have decided at any terminating point.

Lemma 22 In a C-system subject to process failures and recovery, if (e, f) is a terminating point, then $(e, f) \models \bigwedge_{c \in \mathcal{C}} (\text{ACCEPT}_c \vee \text{REFUSE}_c) \wedge \bigwedge_{c \in \mathcal{C}} (\text{AWARD}_m^c \vee \text{REJECT}_m^c)$.

Proof: Assume bwoc that (e, f) is a terminating point, so $(e, f) \models \bigwedge_{p \in \Pi} \text{TERM}_p$, but $(e, f) \not\models \neg [\bigwedge_{c \in \mathcal{C}} (\text{ACCEPT}_c \vee \text{REFUSE}_c) \wedge \bigwedge_{c \in \mathcal{C}} (\text{AWARD}_m^c \vee \text{REJECT}_m^c)]$.

Because no process is failed at a terminating point, $Failed(e, f) = \emptyset$. Without loss of generality, assume $\text{no loss}(e, f + 1) \neq \emptyset$ (this is without loss of generality because there is a finite number of messages, say j , in transit at (e, f) , so one of $(e, f + 1)$, $(e, f + 2)$, \dots , $(e, f + j + 1)$ must be lossless). Note that for any $e_1 \in \text{no loss}(e, f + 1)$, $Failed(e_1, f + 1) = \emptyset$ and $(e_1, f + 1) \models \bigwedge_{p \in \Pi} TERM_p$ (i.e., all processes must also be terminated in the lossless points corresponding to $(e, f + 1)$, because $TERM_p$ is a stable, local predicate). Also, $(e_1, f + 1) \models \neg[\bigwedge_{c \in \mathcal{C}} (ACCEPT_c \vee REFUSE_c) \wedge \bigwedge_{c \in \mathcal{C}} (AWARD_m^c \vee REJECT_m^c)]$, because $ACCEPT_c$, $REFUSE_c$, $AWARD_m^c$, and $REJECT_m^c$ is each local to the (terminated) process subscripting it.

By **Post-Failure Termination**, there is $e_2 \in \text{no loss}(e, f + 1)$ and $h > f + 1$ such that $Failed(e_2, g) = \emptyset$ and $e_2 \in \text{no loss}(e_2, g)$, for $f + 1 \leq g \leq h$, and $(e_2, h) \models \bigwedge_{c \in \mathcal{C}} (ACCEPT_c \vee REFUSE_c) \wedge \bigwedge_{c \in \mathcal{C}} (AWARD_m^c \vee REJECT_m^c)$.

Therefore, at least one $p \in \Pi$ executes an event in $(e_2, h) - (e_2, f)$ (by the definition of knowledge and some process's local knowledge changing), and $(e_2, f) \models \neg TERM_p$. Because $(e_2, f) \sim_q (e, f)$ and $TERM_q$ is local to q , for all $q \in \Pi$, we have that $(e, f) \models \neg TERM_p$, so (e, f) is not a terminating point, contradicting our assumption. \square

Lemma 23 shows how ① m 's award to c depends on c 's acceptance knowledge, and ② c 's decision to accept depends on m 's awarding knowledge. The first part of Lemma 23 below says intuitively that, in order for m to award to c at some point, m must be sure that c has received enough information to accept. This is because otherwise c may fail, recover, and need to decide without receiving any more messages; then c cannot gain the knowledge it needs to accept and must therefore refuse, violating the **Decision Harmony** property.¹¹ The second part of Lemma 28 has the corresponding assertion needed for c to accept.

Lemma 23 In any C-system M which supports independent recovery, for any $c \in \mathcal{C}$,

1. if $K_c \phi$ is c -receive-dependent and $M \models ACCEPT_c \supset K_c \phi$,
then $M \models AWARD_m^c \supset K_m K_c \phi$.
2. if $K_m \phi$ is m -receive-dependent and $M \models AWARD_m^c \supset K_m \phi$,
then $M \models ACCEPT_c \supset K_c K_m \phi$.

Proof: We prove the first; the second follows analogously. We prove this result using a series of three support claims (assume $M \models ACCEPT_c \supset K_c \phi$, as stated above):

Claim 1: $M \models (AWARD_m^c \wedge FAILED_c) \supset [K_c \phi \vee \square \diamond FAILED_c]$.

(If the manager has awarded to c and c is failed,
then either c knows ϕ or c will keep failing (infinitely often).)

¹¹Unlike "agreement" problems, in which the post-failure decisions of faulty processes are irrelevant [Hadz89], commitment problems impose the same consistency requirements on all processes, whether they decide before or after failures. This motivates m , when it awards to c , to know that c has received enough information to decide consistently.

Claim 2: $M \models (AWARD_m^c \wedge FAILED_c) \supset K_c \phi$.

(If m has awarded to c and c is failed, then c knows ϕ .)

Claim 3: $M \models AWARD_m^c \supset K_c \phi$.

(If m has awarded to c , then c knows ϕ .)

Proof of claim 1: $M \models (AWARD_m^c \wedge FAILED_c) \supset [K_c \phi \vee \Box \Diamond FAILED_c]$.

(Because $AWARD_m^c$ remains true in any extension, the only way to prevent a failed c from needing to know ϕ (in order to accept independently) is by ensuring that c , in every extension, fails infinitely often (and, therefore, need not decide)).

Assume bwoc that there is some $(e, f) \in \mathbf{Pts}(\mathcal{E})$ such that

$(e, f) \models AWARD_m^c \wedge FAILED_c \wedge \neg K_c \phi \wedge \neg \Box \Diamond FAILED_c$. Therefore, there is (e_1, g) extending (e, f) such that $(e_1, g) \models \neg \Diamond FAILED_c$. Therefore, there is (e_2, h) extending (e_1, g) such that $(e_2, h') \models \neg FAILED_c$, for all $h' \geq h$. Now (e_2, h) extends (e, f) , and $(e_2, h) \models AWARD_m^c$ (by stability).

[Find an “earliest” point at which c recovers after (e, f) ; this is guaranteed to exist, by the above.] Let (e_3, i) extend (e, f) such that $(e_3, j) \models FAILED_c$, for all $f \leq j < i$, $(e_3, i) \models \neg FAILED_c$, and there is no e_4 extending (e, f) and $f \leq k < i$ such that $(e_4, k) \models \neg FAILED_c$. Therefore, $RECV(\underline{m}, p) \notin e_3(i-1, c) - e_3(f, c)$, for all messages $\underline{m} \neq \lambda$ and $p \in \Pi \setminus \{c\}$. Now, because ϕ is c -receive-dependent, $(e_3, i-1) \models \neg K_c \phi$. By stability, $(e_3, i-1) \models AWARD_m^c$.

(*) Because M supports **Nontrivial Process Failure**, there is e_5 extending (e_3, i) such that $(e_5, j) \models \neg FAILED_c$, for all $j \geq i$.

(**) Because M supports independent recovery, there is $k > i$ such that $(e_5, k) \models ACCEPT_c \vee REFUSE_c$, and $RECV(\underline{m}, p) \notin e_5(k, c) - e_5(i-1, c)$, for all messages $\underline{m} \neq \lambda$ and $p \in \Pi \setminus \{c\}$ (i.e., c receives no message before it decides). Because $K_c \phi$ is c -receive-dependent, $(e_5, k) \models \neg K_c \phi$. Therefore, $(e_5, k) \models REFUSE_c$, and, by stability, $(e_5, k) \models AWARD_m^c \wedge REFUSE_c$, violating **Decision Harmony**. \square

Proof of claim 2: $M \models (AWARD_m^c \wedge FAILED_c) \supset K_c \phi$.

In any system which is subject to process failure and recovery (Definition 3), any subset of failed processes may recover. Therefore, given any point $(e, f) \in \mathbf{Pts}(\mathcal{E})$ such that $(e, f) \models FAILED_c$, there is at least one point $(e_1, g) \in \mathbf{Pts}(\mathcal{E})$ extending (e, f) such that $(e_1, g) \models \neg FAILED_c$. By **Nontrivial Process Failure**, $(e_1, g) \models \neg \Diamond FAILED_c$, and because (e_1, g) extends (e, f) , $(e, f) \models \neg \Diamond FAILED_c$. For any point $(e_2, h) \in \mathbf{Pts}(\mathcal{E})$ such that $(e_2, h) \models \neg FAILED_c$, $(e_2, h) \models \neg \Diamond FAILED_c$, by **Nontrivial Process Failure**. Therefore, $M \models \neg \Diamond FAILED_c$, so $M \models \neg \Box \Diamond FAILED_c$.

Therefore, $\mathbf{M} \models (\text{AWARD}_m^c \wedge \text{FAILED}_c) \supset [K_c\phi \vee \square \diamond \text{FAILED}_c]$

(from claim 1) reduces to

$\mathbf{M} \models (\text{AWARD}_m^c \wedge \text{FAILED}_c) \supset K_c\phi. \square_2$

Proof of claim 3: $\mathbf{M} \models \text{AWARD}_m^c \supset K_c\phi.$

Bwoc, assume there is $(e, f) \in \text{Pts}(\mathcal{E})$ such that $(e, f) \models \text{AWARD}_m^c \wedge \neg K_c\phi.$ If $(e, f) \models \text{FAILED}_c,$ then we have a contradiction of claim (2). Assume, instead, that $(e, f) \models \neg \text{FAILED}_c.$

Take any $e_1 \in \text{fail}(e, f, \{c\}).$ $(e_1, f) \models \text{AWARD}_m^c$ (because AWARD_m^c is local to m and $(e_1, f) \sim_m (e, f)$). Further, either $(e_1, f) \models \text{FAILED}_c,$ or $(e_1, f) \models \text{TERM}_c$ (by the definition of fail). In either case, $(e_1, f) \models \neg K_c\phi$ (because $K_c\phi$ is c -receive-dependent, by hypothesis).

In the case that $(e_1, f) \models \text{FAILED}_c,$ we have $(e_1, f) \models \text{AWARD}_m^c \wedge \text{FAILED}_c \wedge \neg K_c\phi,$ violating claim (2). In the case that $(e_1, f) \models \text{TERM}_c,$ we have $(e_1, f) \models \text{ACCEPT}_c$ (by Lemma 22 and Decision Harmony), but then $(e_1, f) \models K_c\phi,$ a contradiction.

Therefore, $(e, f) \models K_c\phi. \square_3$

Therefore, $\mathbf{M} \models \text{AWARD}_m^c \supset K_c\phi,$ and by the locality of AWARD_m^c to m and Lemma 8, $\mathbf{M} \models \text{AWARD}_m^c \supset K_m K_c\phi. \square$

For the purpose of following lemmas, we will call part 1 of Lemma 23 the Award Knowledge Rule 23 and part 2 the Accept Knowledge Rule 23.

Henceforth, we abbreviate $\bigvee_{x \in \mathcal{D}_c} \text{DEPEND}_m^x$ by $\text{DEPEND}_m^x.$ Lemma 24 shows that ① c must receive a message for the knowledge level $(K_c K_m)^j K_c \text{DEPEND}_m^x$ ¹², to hold; and ② m must receive a message for $(K_m K_c)^i \text{DEPEND}_m^x$ to hold.

Lemma 24 In a C-system \mathbf{M} subject to process failures,

- $(K_c K_m)^j K_c \text{DEPEND}_m^x$ is c -receive-dependent and initially false, for all $j \geq 0$; and
- $(K_m K_c)^i \text{DEPEND}_m^x$ is m -receive-dependent and initially false, for all $i \geq 1.$

Proof: We prove the first by induction on the knowledge nesting level j ; the second follows similarly by induction on the knowledge nesting level $i.$

¹²For any $p, q \in \Pi, j \geq 0,$ we abbreviate $\underbrace{K_p K_q}_{1} \underbrace{K_p K_q}_{2} \dots \underbrace{K_p K_q}_{j} \phi$ by $(K_p K_q)^j \phi.$ Similarly, we abbreviate $\underbrace{K_p \diamond K_q}_{1} \underbrace{K_p \diamond K_q}_{2} \dots \underbrace{K_p \diamond K_q}_{j} \phi$ by $(K_p \diamond K_q)^j \phi.$ Finally, we abbreviate $p \xrightarrow{+} q \xrightarrow{+} p \xrightarrow{+} q \xrightarrow{+} p \xrightarrow{+} q \dots \xrightarrow{+} p \xrightarrow{+} q$ by $p \xrightarrow{+} q (\xrightarrow{+} p \xrightarrow{+} q)^j.$

Base case: $j = 0$. We claim that $K_c \text{DEPEND}_m^x$ is c -receive-dependent. This holds from the requirement of a DEPEND_m^x -message chain (Lemma 19). $K_c \text{DEPEND}_m^x$ is initially false because DEPEND_m^x is so (by **No Predetermined Dependencies**).

Inductive step: $j > 0$. Assume the inductive hypothesis holds for $j - 1$.

Therefore, $(K_c K_m)^{j-1} K_c \text{DEPEND}_m^x$ is c -receive-dependent and initially false. We now claim $K_c K_m (K_c K_m)^{j-1} K_c \text{DEPEND}_m^x$ is c -receive-dependent. By Lemma 11, introspection, and the Message Chain Theorem, $K_m (K_c K_m)^{j-1} K_c \text{DEPEND}_m^x$ is m -receive-dependent. Further, $K_m (K_c K_m)^{j-1} K_c \text{DEPEND}_m^x$ is initially false, because DEPEND_m^x is (by **No Predetermined Dependencies**). Therefore, by Lemma 11, introspection, and the Message Chain Theorem, $K_c K_m (K_c K_m)^{j-1} K_c \text{DEPEND}_m^x$ is c -receive-dependent; further, $K_c K_m (K_c K_m)^{j-1} K_c \text{DEPEND}_m^x$ is initially false, because DEPEND_m^x is (by **No Predetermined Dependencies**). \square

Lemma 25 shows that ① $(K_c K_m)^j K_c \text{DEPEND}_m^x$ must hold for arbitrarily deep nesting in order for c to accept, and ② $(K_m K_c)^i \text{DEPEND}_m^x$ must hold for arbitrarily deep nesting in order for m to award to c .

Lemma 25 For any C-system \mathbf{M} supporting independent recovery,

1. $\mathbf{M} \models \text{ACCEPT}_{c \supset} (K_c K_m)^j K_c \text{DEPEND}_m^x$, for any $c \in \mathcal{C}$ and for all $j \geq 0$.
2. $\mathbf{M} \models \text{AWARD}_{m \supset}^c (K_m K_c)^i \text{DEPEND}_m^x$, for any $c \in \mathcal{C}$ and for all $i \geq 1$.

Proof: We show the first; the second follows immediately. We prove this by induction on j .

Base case: $j = 0$. The claim that $\mathbf{M} \models \text{ACCEPT}_{c \supset} K_c \text{DEPEND}_m^x$ holds by locality of $\text{ACCEPT}_{c \supset}$ and Lemma 14.

Inductive hypothesis: $j > 0$. Assume the inductive hypothesis holds for $j - 1$. Therefore, $\mathbf{M} \models \text{ACCEPT}_{c \supset} (K_c K_m)^{j-1} K_c \text{DEPEND}_m^x$. We now claim that $\mathbf{M} \models \text{ACCEPT}_{c \supset} K_c K_m (K_c K_m)^{j-1} K_c \text{DEPEND}_m^x$. This holds immediately from (a) an application of the Accept Knowledge Rule 23 on $(K_c K_m)^{j-1} K_c \text{DEPEND}_m^x$ (which is c -receive-dependent by Lemma 24), and (b) an application of the Award Knowledge Rule 23 on the result of application (a) (which is m -receive-dependent, also by Lemma 24). \square

Lemma 26 establishes that consecutive message chains are required to establish each knowledge level in Lemma 25.

Lemma 26 In a C-system \mathbf{M} subject to process failures,

- for all $j \geq 0$, if, for some $(e, f) \in \text{Pts}(\mathcal{E})$ and $c \in \mathcal{C}$, $(e, f) \models (K_c K_m)^j K_c \text{DEPEND}_m^x$, then there is a sequence of consecutive message chains $m \xrightarrow{+} c (\xrightarrow{+} m \xrightarrow{+} c)^j$ in $(e, 0)$ to (e, f) ; and

- for all $i \geq 0$, if, for some $(e, f) \in \text{Pts}(\mathcal{E})$ and $c \in \mathcal{C}$, $(e, f) \models (K_m K_c)^i \text{DEPEND}_m^x$, then there is a sequence of consecutive message chains $c \xrightarrow{+} m(\xrightarrow{+} c \xrightarrow{+} m)^i$ in $(e, 0)$ to (e, f) .

Proof: We prove the first by induction on the length of the sequence of chains, j ; the second follows analogously by induction on i .

Base case: $j = 0$. The claim is that, for any $(e, f) \in \text{Pts}(\mathcal{E})$, if $(e, f) \models K_c \text{DEPEND}_m^x$, then there is a DEPEND_m^x -message chain from m to c in interval $(e, 0)$ to (e, f) . This holds by Lemma 19.

Inductive step: $j > 0$. Assume the inductive hypothesis holds for $j - 1$. Now we claim that $K_c K_m (K_c K_m)^{j-1} K_c \text{DEPEND}_m^x$ requires a sequence of message chains $m \xrightarrow{+} c(\xrightarrow{+} m \xrightarrow{+} c)^j$. That is, if $(e, f) \models K_c K_m (K_c K_m)^{j-1} K_c \text{DEPEND}_m^x$, then there is a sequence of consecutive message chains $m \xrightarrow{+} c(\xrightarrow{+} m \xrightarrow{+} c)^j$ in interval $(e, 0)$ to (e, f) . From the inductive hypothesis, we can assert the existence of the chain sequence $m \xrightarrow{+} c(\xrightarrow{+} m \xrightarrow{+} c)^{j-1}$, required to establish $(K_c K_m)^{j-1} K_c \text{DEPEND}_m^x$. By Lemma 24, $(K_c K_m)^{j-1} K_c \text{DEPEND}_m^x$ is c -receive-dependent in \mathbf{M} . Therefore, by Lemma 11, $(K_c K_m)^{j-1} K_c \text{DEPEND}_m^x$ is c -failure-dissociated in \mathbf{M} . By Lemma 24, $(K_c K_m)^{j-1} K_c \text{DEPEND}_m^x$ is initially false. By introspection, $(K_c K_m)^{j-1} K_c \text{DEPEND}_m^x$ is local to c . Therefore, by the Message Chain Theorem, there is a $(K_c K_m)^{j-1} K_c \text{DEPEND}_m^x$ -message chain $c \xrightarrow{+} m$ in $(e, 0)$ to (e, f) , to establish $K_m (K_c K_m)^{j-1} K_c \text{DEPEND}_m^x$. Now this chain must strictly follow $m \xrightarrow{+} c(\xrightarrow{+} m \xrightarrow{+} c)^{j-1}$, because $(K_c K_m)^{j-1} K_c \text{DEPEND}_m^x$ must hold at the start of the new $c \xrightarrow{+} m$, and $(K_c K_m)^{j-1} K_c \text{DEPEND}_m^x$ cannot hold any earlier than the end of $m \xrightarrow{+} c(\xrightarrow{+} m \xrightarrow{+} c)^{j-1}$. From this, we conclude the existence of $m \xrightarrow{+} c(\xrightarrow{+} m \xrightarrow{+} c)^{j-1} \xrightarrow{+} m$. By similar reasoning, there is a $K_m (K_c K_m)^{j-1} K_c \text{DEPEND}_m^x$ -message chain, to establish $K_c K_m (K_c K_m)^{j-1} K_c \text{DEPEND}_m^x$. We can conclude the existence of $m \xrightarrow{+} c(\xrightarrow{+} m \xrightarrow{+} c)^{j-1} \xrightarrow{+} m \xrightarrow{+} c$, or $m \xrightarrow{+} c(\xrightarrow{+} m \xrightarrow{+} c)^j$. \square

Theorem 27 There is no C-system which supports independent recovery.

Proof: Bwoc, fix any C-system \mathbf{M} which supports independent recovery. From Lemmas 25 and 26, we may conclude that, in order for c to decide to accept, there must be an infinitely long sequence of message chains from c to m and back in \mathbf{M} . If there is some $(e, f) \in \text{Pts}(\mathcal{E})$ such that $(e, f) \models \text{ACCEPT}_c$, the largest possible consecutive message chain length is $\frac{f}{2}$, so at most there exists $m \xrightarrow{+} c(\xrightarrow{+} m \xrightarrow{+} c)^{\frac{f}{2}-1}$ in $(e, 0)$ to (e, f) . By Lemma 25, however, $m \xrightarrow{+} c(\xrightarrow{+} m \xrightarrow{+} c)^j$ also exists in $(e, 0)$ to (e, f) for all $j > \frac{f}{2} - 1$. This is a contradiction. Essentially, the required infinitely long sequence of message chains cannot occur in a finite portion of execution e . Therefore, c may never accept in \mathbf{M} .

Accept decisions must be possible in \mathbf{M} . Therefore, \mathbf{M} is not a C-system. \square

7.2 Weak Termination, Process Recovery, and Bounded Time Communication

The result we give below states that there is no protocol for negotiated commitment which guarantees weak termination in a system in which processes may fail and recover and in which messages may spend a bounded amount of time in transit. If such protocols existed, then a decided process could stop executing events (terminate) regardless of the state of other, failed, processes. As with independent recovery, this impossibility result means that, in any negotiated commitment protocol, at least one “live” process must have the knowledge and longevity to communicate with recovering processes to help them decide, regardless of how long those processes remain failed. The proof of this result uses lemmas similar to those in the proof of the impossibility of independent recovery.

The first part of Lemma 28 below says intuitively that, in order for m to award to c at some point, m must be sure that c has received enough information to accept. This is because otherwise the system may terminate without any more messages being received (by Theorem 12), so that c cannot gain the knowledge it needs to accept and must therefore refuse (by Lemma 22), violating the **Decision Harmony** property. The second part of Lemma 28 has the corresponding assertion needed for c to accept.

Lemma 28 In any weakly terminating C-system M which is k -transit bounded and subject to process failure and recovery, for any $c \in \mathcal{C}$,

1. if $K_c\phi$ is c -receive-dependent and $M \models \text{ACCEPT}_c \supset K_c\phi$,
then $M \models \text{AWARD}_m^c \supset K_m K_c\phi$.
2. if $K_m\phi$ is m -receive-dependent and $M \models \text{AWARD}_m^c \supset K_m\phi$,
then $M \models \text{ACCEPT}_c \supset K_c K_m\phi$.

Proof: We prove the first; the second follows analogously. We use the following claim:

$$M \models \text{AWARD}_m^c \supset K_c\phi.$$

Proof of claim: Assume bwoc that there is $(e, f) \in \text{Pts}(\mathcal{E})$ such that $(e, f) \models \text{AWARD}_m^c \wedge \neg K_c\phi$. Therefore, $(e, f) \models \neg \text{ACCEPT}_c$. By Theorem 12, there is $(e_1, h) \in \text{Pts}(\mathcal{E})$ which is a terminating extension of (e, f) such that no process receives a nonnull message after (e_1, f) . By Lemma 22, $(e_1, h) \models \text{ACCEPT}_c \vee \text{REFUSE}_c$, and by $K_c\phi$ being c -receive-dependent, $(e_1, h) \models \text{REFUSE}_c$. Therefore, $(e_1, h) \models \text{AWARD}_m^c \wedge \text{REFUSE}_c$, violating **Decision Harmony**_{Claim}.

Because AWARD_m^c is local to m , whenever AWARD_m^c holds in a particular point, it holds in all points which m considers similar. Therefore, $K_c\phi$ also holds in those points, so $M \models \text{AWARD}_m^c \supset K_m K_c\phi$. \square

For the purpose of the next lemma, we will call part 1 of Lemma 28 the Award Knowledge Rule 28 and part 2 the Accept Knowledge Rule 28. Lemma 29 shows that $\textcircled{1} (K_c K_m)^j K_c \text{DEPEND}_m^x$

must hold for arbitrarily deep nesting in order for c to accept, and ② $(K_m K_c)^i \text{DEPEND}_m^x$ must hold for arbitrarily deep nesting in order for m to award to c .

Lemma 29 For any weakly terminating C-system M which is k -transit bounded and subject to process failures and recovery,

1. $M \models \text{ACCEPT}_c \supset (K_c K_m)^j K_c \text{DEPEND}_m^x$, for any $c \in \mathcal{C}$ and for all $j \geq 0$;
2. $M \models \text{AWARD}_m^c \supset (K_m K_c)^i \text{DEPEND}_m^x$, for any $c \in \mathcal{C}$ and for all $i \geq 1$;

Proof: This proof is the same as that for Lemma 25, except that the Award Knowledge Rule 28 and the Accept Knowledge Rule 28 are used here. \square

From the fact that commitment between c and m requires infinitely long sequences of message chains between c and m and the fact that commitment must be possible in any C-system, we conclude our impossibility result.

Theorem 30 There is no weakly terminating C-system which is k -transit bounded and subject to process failures and recovery. \square

There is a strong connection between a system supporting independent recovery and a system being weakly terminating, k -transit-bounded, and subject to process failures and recovery. The reader may have noticed the great similarities between Lemmas 23 and 28, Lemmas 25 and 29, and Theorems 27 and 30 (respectively). This strong connection is related to the *possibility* in each system of deciding independently; see Appendix B.

The analogue of Theorem 30 for atomic commitment tells us the following: if a round-based atomic commitment protocol is resilient to process failures and recovery and such that a message may be received only in the round in which it is sent, then the protocol may run forever.

From the knowledge levels in Lemma 29, one might think that one can prove Theorem 30 using common knowledge [HaMo90]. Indeed, another way to prove this theorem would be to show that commitment in the given systems requires common knowledge among $\{x\}$ and m of DEPEND_m^x , which, as a direct corollary of the Message Chain Theorem, is impossible. The current results on attaining common knowledge do not address systems with process failures (although this extension should not be difficult). The Message Chain Theorem allows us to reason about both finite and infinite knowledge levels in several kinds of systems, including systems in which one cannot attain common knowledge. Further, we can reason about incremental gains in a process's knowledge through communication. Finally, the Message Chain Theorem applies to a broad class of problems.

We note that commitment under other system assumptions may require a *variant* of common knowledge of some proposition; for example, [Hadz88] argues the analogue of Theorem 38 for nonblocking atomic commitment by showing ① the need for *eventual* common knowledge of a

particular proposition, and ② the impossibility of attaining that common knowledge in the given systems. Theorem 38 uses our proof technique using nested knowledge levels and consecutive message chains.

7.3 Nonblocking Behaviour, Termination, and Communication Failures

Informally, we say that a process is *blocked* when it must await the repair of failures before proceeding [Skee82, SkSt83, BeHG87]. Blocking is undesirable, because it may cause participants to wait for an arbitrarily long time before deciding, making a contract undecided for arbitrarily long, uselessly holding any resources which might be required for commitment. Therefore, non-blocking commitment systems are preferred over blocking ones. We will show some conditions under which nonblocking behaviour is impossible to achieve.

Definition 31 A C-system M is called *nonblocking* if

$$M \models \bigwedge_{c \in \mathcal{C}} \diamond (FAILED_c \vee ACCEPT_c \vee REFUSE_c) \wedge \\ \bigwedge_{c \in \mathcal{C}} \diamond (FAILED_m \vee AWARD_m^c \vee REJECT_m^c). \quad \square$$

That is, in a nonblocking C-system, all nonfailed processes eventually decide one way or the other. Notice that this covers a situation in which a process fails, recovers, and does not fail again—such a process must decide.

We show here that no negotiated commitment protocol can achieve nonblocking behaviour if the system is subject to permanent communication failures (shown in [Skee82, SkSt83] for atomic commitment) or the system is weakly terminating and subject to (even transient) communication failures (shown in [Hadz90] for atomic commitment). Call a system which has either of these two properties a *target* system. The development of this result proceeds in essentially two parts. First, we demonstrate that we can derive from any target C-system a process-failure-free counterpart target C-system, and then we derive knowledge levels required to establish a commitment (award or accept) in a nonblocking, process-failure-free C-system. Second, we show that certain sequences of consecutive message chains are required in order to attain certain relevant knowledge levels in any target C-system. Then, to bring the two parts together, we argue that, in order to establish a commitment in a nonblocking target C-system, an infinite sequence of consecutive message chains is required. Lemmas 32-34 constitute the first part of the development, Lemmas 35-37 constitute the second part, and Theorem 38 establishes the overall result.

Lemma 32 shows that, from any target C-system, we can derive another target C-system whose executions are process-failure-free and a subset of the executions of the original system, and whose points support primitive propositions iff they did in the original system.

Lemma 32 Given any C-system $M = (\mathcal{E}, \mathcal{I})$, there is a process-failure-free C-system $M_{PFF} = (\mathcal{E}_{PFF}, \mathcal{I}_{PFF})$ such that

1. $\mathcal{E}_{PFF} \subseteq \mathcal{E}$,

2. for any $(e, f) \in \mathbf{Pts}(\mathcal{E}_{PFF})$ and primitive proposition $\phi \in \Phi$,
 $(\mathbf{M}_{PFF}, e, f) \models \phi$ iff $(\mathbf{M}, e, f) \models \phi$, and
3. $\mathbf{M}_{PFF} \models \neg \text{FAILED}_p$, for all $p \in \Pi$.

Proof: We take $\mathcal{E}_{PFF} = \{e \mid e \in \mathcal{E} \text{ and } \text{FAIL} \notin e(f, p), \text{ for all } f \in N \text{ and } p \in \Pi\}$. Now $\mathcal{E}_{PFF} \subseteq \mathcal{E}$. Further, \mathcal{E}_{PFF} is a system. For any $\phi \in \Phi$, we take $\mathcal{I}_{PFF}(\phi) = \mathcal{I}(\phi) \cap \mathbf{Pts}(\mathcal{E}_{PFF})$ (that is, any such primitive proposition ϕ holds in those points of \mathcal{E} (in which it held) which are now in \mathcal{E}_{PFF}). Now it is straightforward to verify that \mathcal{E} is a system and that all specifications hold in \mathbf{M}_{PFF} , because they hold in \mathbf{M} . \square

In any process-failure-free C-system \mathbf{M}_{PFF} , the nonblocking property becomes $\mathbf{M}_{PFF} \models \bigwedge_{c \in \mathcal{C}} \diamond (\text{ACCEPT}_c \vee \text{REFUSE}_c) \wedge \bigwedge_{c \in \mathcal{C}} \diamond (\text{AWARD}_m^c \vee \text{REJECT}_m^c)$. Lemma 32 implies that it is enough, for our impossibility result, to show that there is no process-failure-free nonblocking target C-system.

Now we identify some knowledge states required for decision in *any* process-failure-free nonblocking C-system.¹³ Lemma 33 shows how ① c 's decision to accept depends on m 's awarding knowledge; and ② m 's award to c depends on c 's acceptance knowledge.

Lemma 33 For any process-failure-free, nonblocking C-system \mathbf{M}_{PFF} , $c \in \mathcal{C}$, and wff ϕ ,

1. if $\mathbf{M}_{PFF} \models \text{AWARD}_m^c \supset K_m \phi$, then $\mathbf{M}_{PFF} \models \text{ACCEPT}_c \supset K_c \diamond K_m \phi$; and
2. if $\mathbf{M}_{PFF} \models \text{ACCEPT}_c \supset K_c \phi$, then $\mathbf{M}_{PFF} \models \text{AWARD}_m^c \supset K_m \diamond K_c \phi$.

Proof: We prove the first; the second follows analogously. Assume bwoc that there is some \mathbf{M}_{PFF} such that $\mathbf{M}_{PFF} \models \text{AWARD}_m^c \supset K_m \phi$, but $\mathbf{M}_{PFF} \not\models \text{ACCEPT}_c \supset K_c \diamond K_m \phi$. Then there is at least one $(e, f) \in \mathbf{Pts}(\mathcal{E}_{PFF})$ such that $(e, f) \models \text{ACCEPT}_c \wedge \neg K_c \diamond K_m \phi$. Therefore, there is $(e_1, g) \sim_c (e, f)$ such that $(e_1, g) \models \text{ACCEPT}_c \wedge \neg \diamond K_m \phi$. Therefore, there is e_2 extending (e_1, g) such that $(e_2, h) \models \neg K_m \phi$, for all $h \geq g$. Therefore, by the antecedent, $(e_2, h) \models \neg \text{AWARD}_m^c$, for all $h \geq g$. Therefore, for some $i \geq g$, $(e_2, i) \models \text{REJECT}_m^c$ (because the system is process-failure-free and nonblocking). But $(e_1, g) \models \text{ACCEPT}_c$, and ACCEPT_c is stable, so $(e_2, i) \models \text{ACCEPT}_c \wedge \text{REJECT}_m^c$, violating **Decision Harmony**. \square

For use in the next lemma, we call part 1 of Lemma 33 the Award Knowledge Rule 33, and we call part 2 the Accept Knowledge Rule 33. Lemma 34 shows that, in a process-failure-free, nonblocking C-system, in order to accept, c must have arbitrarily interleaved knowledge about m 's knowledge of c 's knowledge of the dependency. Similarly, to award to c , m must have arbitrarily interleaved knowledge of c 's knowledge of the dependency.

¹³In Section 8, we develop the knowledge requirements in *general* nonblocking C-systems. Note that Hadzilacos (1990) assumes process-failure-free systems in proving his results on the impossibility of nonblocking atomic commitment protocols; he does not give general knowledge requirements.

Lemma 34 For any process-failure-free, nonblocking C-system M_{PFF} and $c \in \mathcal{C}$,

- $M_{PFF} \models ACCEPT_c \supset (K_c \diamond K_m \diamond)^j K_c DEPEND_m^x$, for all $j \geq 0$.
- $M_{PFF} \models AWARD_m^c \supset K_m \diamond (K_c \diamond K_m \diamond)^i K_c DEPEND_m^x$,
for any $c \in \mathcal{C}$ and for all $i \geq 1$.

Proof: We show the first; the second follows similarly. We prove this by induction on j .

Base case: $j = 0$. The claim that $M_{PFF} \models ACCEPT_c \supset K_c DEPEND_m^x$ follows from the locality of $ACCEPT_c$ and Lemma 14.

Inductive hypothesis: $j > 0$. Assume the inductive hypothesis holds for $j - 1$. Therefore, $M_{PFF} \models ACCEPT_c \supset (K_c \diamond K_m \diamond)^{j-1} K_c DEPEND_m^x$. We now claim that $M_{PFF} \models ACCEPT_c \supset K_c \diamond K_m \diamond (K_c \diamond K_m \diamond)^{j-1} K_c DEPEND_m^x$. This holds immediately from (a) an application of the Accept Knowledge Rule 33 on $(K_c \diamond K_m \diamond)^{j-1} K_c DEPEND_m^x$, and (b) an application of the Award Knowledge Rule 33 on the result of application (a). \square

From the existence of a dependency message chain (Lemma 19), c must receive a message to come to know $DEPEND_m^x$.

Lemma 35 In any C-system M subject to communication failures, for $c \in \mathcal{C}$, wff $K_c DEPEND_m^x$ is c -receive-dependent. \square

Lemma 36 shows that c must receive a message to attain certain useful levels of interleaved knowledge, namely, $(K_c \diamond K_m \diamond)^j K_c DEPEND_m^x$ and m must receive a message for $K_m \diamond (K_c \diamond K_m \diamond)^i K_c DEPEND_m^x$ to hold.

Lemma 36 In any target C-system M ,

1. $(K_c \diamond K_m \diamond)^j K_c DEPEND_m^x$ is c -receive-dependent and initially false, for all $j \geq 0$.
2. $K_m \diamond (K_c \diamond K_m \diamond)^i K_c DEPEND_m^x$ is m -receive-dependent and initially false, for all $i \geq 1$.

Proof: We prove the first, by induction on the knowledge nesting level j (the second follows analogously by induction on i).

Base case: $j = 0$. That $K_c \diamond DEPEND_m^x$ is c -receive-dependent is shown in Lemma 35. $K_c DEPEND_m^x$ is initially false because $DEPEND_m^x$ is so (by **No Predetermined Dependencies**).

Inductive step: $j > 0$. Assume the inductive hypothesis holds for $j - 1$. Therefore, $(K_c \diamond K_m \diamond)^{j-1} K_c DEPEND_m^x$ is c -receive-dependent and initially false. We now claim that the same holds for $K_c \diamond K_m \diamond (K_c \diamond K_m \diamond)^{j-1} K_c DEPEND_m^x$. $(K_c \diamond K_m \diamond)^{j-1} K_c DEPEND_m^x$ is local to c , by introspection. Therefore, by hypothesis and Lemma 17,

$K_m \diamond (K_c \diamond K_m \diamond)^{j-1} K_c \text{DEPEND}_m^x$ is m -receive-dependent. By hypothesis, $(K_c \diamond K_m \diamond)^{j-1} K_c \text{DEPEND}_m^x$ is c -receive-dependent. Therefore, by Lemma 13 and Lemma 10, $\diamond (K_c \diamond K_m \diamond)^{j-1} K_c \text{DEPEND}_m^x \equiv (K_c \diamond K_m \diamond)^{j-1} K_c \text{DEPEND}_m^x$, which by hypothesis is initially false. Therefore, $K_m \diamond (K_c \diamond K_m \diamond)^{j-1} K_c \text{DEPEND}_m^x$ is initially false.

$K_m \diamond (K_c \diamond K_m \diamond)^{j-1} K_c \text{DEPEND}_m^x$ is also local to m . Therefore, by Lemma 17, $K_c \diamond K_m \diamond (K_c \diamond K_m \diamond)^{j-1} K_c \text{DEPEND}_m^x$ is c -receive-dependent. Further, by m -receive-dependency, Lemma 13, and Lemma 10, $\diamond K_m \diamond (K_c \diamond K_m \diamond)^{j-1} K_c \text{DEPEND}_m^x \equiv K_m \diamond (K_c \diamond K_m \diamond)^{j-1} K_c \text{DEPEND}_m^x$, which is initially false. Therefore, $K_c \diamond K_m \diamond (K_c \diamond K_m \diamond)^{j-1} K_c \text{DEPEND}_m^x$ is initially false. \square

Lemma 37 characterizes the sequence of consecutive message chains required to establish the knowledge levels in Lemma 36.

Lemma 37 In any target C-system M ,

1. for all $j \geq 0$, if, for some $(e, f) \in \text{Pts}(\mathcal{E})$ and $c \in \mathcal{C}$, $(e, f) \models (K_c \diamond K_m \diamond)^j K_c \text{DEPEND}_m^x$, then there is a sequence of consecutive message chains $m \xrightarrow{+} c(\xrightarrow{+} m \xrightarrow{+} c)^j$ in $(e, 0)$ to (e, f) .
2. for all $i \geq 0$, if, for some $(e, f) \in \text{Pts}(\mathcal{E})$ and $c \in \mathcal{C}$, $(e, f) \models K_m \diamond (K_c \diamond K_m \diamond)^i K_c \text{DEPEND}_m^x$, then there is a sequence of consecutive message chains $m \xrightarrow{+} c(\xrightarrow{+} m \xrightarrow{+} c)^i \xrightarrow{+} m$ in $(e, 0)$ to (e, f) .

Proof: We prove the first, by induction on the length of the sequence of chains, j (the second follows analogously by induction on i).

Base case: $j = 0$. The claim is that, for any $(e, f) \in \text{Pts}(\mathcal{E})$, if $(e, f) \models K_c \text{DEPEND}_m^x$, then there is a DEPEND_m^x -message chain from m to c in interval $(e, 0)$ to (e, f) . This holds by Lemma 19.

Inductive step: $j > 0$. Assume the inductive hypothesis holds for $j - 1$. Now we claim that $K_c \diamond K_m \diamond (K_c \diamond K_m \diamond)^{j-1} K_c \text{DEPEND}_m^x$ requires a sequence of message chains $m \xrightarrow{+} c(\xrightarrow{+} m \xrightarrow{+} c)^j$. That is, if $(e, f) \models K_c \diamond K_m \diamond (K_c \diamond K_m \diamond)^{j-1} K_c \text{DEPEND}_m^x$, then there is a sequence of consecutive message chains $m \xrightarrow{+} c(\xrightarrow{+} m \xrightarrow{+} c)^j$ in interval $(e, 0)$ to (e, f) . From the inductive hypothesis, we can assert the existence of the chain sequence $m \xrightarrow{+} c(\xrightarrow{+} m \xrightarrow{+} c)^{j-1}$, required to establish $(K_c \diamond K_m \diamond)^{j-1} K_c \text{DEPEND}_m^x$ in $(e, 0)$ to (e, f) .

Consider $K_m \diamond (K_c \diamond K_m \diamond)^{j-1} K_c \text{DEPEND}_m^x$. By introspection, $(K_c \diamond K_m \diamond)^{j-1} K_c \text{DEPEND}_m^x$ is local to c . By Lemma 36, $(K_c \diamond K_m \diamond)^{j-1} K_c \text{DEPEND}_m^x$ is c -receive-dependent in M . Therefore, by Lemma 13, $(K_c \diamond K_m \diamond)^{j-1} K_c \text{DEPEND}_m^x$ is nontrivial in M . Therefore, by Lemma 10, $M \models (K_c \diamond K_m \diamond)^{j-1} K_c \text{DEPEND}_m^x \equiv \diamond (K_c \diamond K_m \diamond)^{j-1} K_c \text{DEPEND}_m^x$. By Lemma 36, $(K_c \diamond K_m \diamond)^{j-1} K_c \text{DEPEND}_m^x$ is initially false in M . Therefore,

$\diamond(K_c \diamond K_m \diamond)^{j-1} K_c \text{DEPEND}_m^x$ is initially false in \mathbf{M} , nontrivial in \mathbf{M} , and local to c . Therefore, by the Message Chain Theorem, a $(K_c \diamond K_m \diamond)^{j-1} K_c \text{DEPEND}_m^x$ -message chain $c \xrightarrow{+} m$ is required in $(e, 0)$ to (e, f) to establish $K_m \diamond (K_c \diamond K_m \diamond)^{j-1} K_c \text{DEPEND}_m^x$.

Now this new chain must strictly follow $m \xrightarrow{+} c(\xrightarrow{+} m \xrightarrow{+} c)^{j-1}$, because $(K_c \diamond K_m \diamond)^{j-1} K_c \text{DEPEND}_m^x$ must hold at the start of the new $c \xrightarrow{+} m$, and $(K_c \diamond K_m \diamond)^{j-1} K_c \text{DEPEND}_m^x$ cannot hold any earlier than the end of $m \xrightarrow{+} c(\xrightarrow{+} m \xrightarrow{+} c)^{j-1}$. From this, we conclude the existence of $m \xrightarrow{+} c(\xrightarrow{+} m \xrightarrow{+} c)^{j-1} \xrightarrow{+} m$. By similar reasoning, there is a $K_m \diamond (K_c \diamond K_m \diamond)^{j-1} K_c \text{DEPEND}_m^x$ -message chain, to establish $K_c \diamond K_m \diamond (K_c \diamond K_m \diamond)^{j-1} K_c \text{DEPEND}_m^x$. We can conclude the existence of $m \xrightarrow{+} c(\xrightarrow{+} m \xrightarrow{+} c)^{j-1} \xrightarrow{+} m \xrightarrow{+} c$, or $m \xrightarrow{+} c(\xrightarrow{+} m \xrightarrow{+} c)^j$. \square

Theorem 38 There is no nonblocking C-system \mathbf{M} which is either subject to permanent communication failures or weakly terminating and subject to communication failures.

Proof: Bwoc, fix any nonblocking target C-system \mathbf{M} . By Lemma 32, there is a process-failure-free, nonblocking target C-system \mathbf{M}_{PFF} . From Lemmas 34 and 37, we may conclude that, in order for c to decide to accept, there must be an infinitely long sequence of message chains from c to m and back in \mathbf{M}_{PFF} . That is, if there is some $(e, f) \in \text{Pts}(\mathcal{E})$ such that $(e, f) \models \text{ACCEPT}_c$, then there must be such an infinitely long sequence, which cannot occur in a finite portion of execution e . Therefore, c may never accept in \mathbf{M}_{PFF} .

Accept decisions must be possible in \mathbf{M}_{PFF} . Therefore, \mathbf{M}_{PFF} is not a C-system, a contradiction. \square

8 Knowledge Levels in Nonblocking Commitment Systems

In Section 7.3, we derived knowledge levels for process-failure-free, nonblocking commitment systems. We now determine knowledge levels required for decision in general nonblocking systems, which may admit process failures. As one might expect, the knowledge levels are more complex in these systems than in those of Section 7.3.

We have shown (in Lemma 14) that a contractor c which has accepted knows it is a codependent; now it is also true that c also knows that eventually everyone in the dependency set will fail or know that it is a codependent. For $P \subseteq \Pi$, we abbreviate by $\mathbf{F}_P \phi$ the formula $\bigwedge_{p \in P} (K_p \phi \vee \text{FAILED}_p)$; that is, every process in P knows ϕ or is failed [Hadz87]. Now, for each $x \in \mathcal{D}_c$, $\mathbf{M} \models [\text{ACCEPT}_c \wedge \text{DEPEND}_m^x] \supset K_c \diamond \mathbf{F}_x \text{DEPEND}_m^x$. Further, an accepting contractor must know that eventually each codependent will fail or will eventually know that each codependent will fail or will eventually know that it is a codependent: $\mathbf{M} \models [\text{ACCEPT}_c \wedge \text{DEPEND}_m^x] \supset K_c \diamond [\mathbf{F}_x \diamond [\mathbf{F}_x \text{DEPEND}_m^x]]$; and so on. More generally, the following holds, for all $i \in \mathbb{N}$:

$$\mathbf{M} \models [\text{ACCEPT}_c \wedge \text{DEPEND}_m^x] \supset K_c \sigma_i \quad (i), \text{ for } i \geq 1$$

where $\sigma_0 = \text{DEPEND}_m^x$, and $\sigma_{i+1} = \Diamond \mathbf{F}_x \sigma_i$, for $i \geq 0$.

Now we show that each member of this collection is valid in a nonblocking C-system.

Lemma 39 For any nonblocking C-system \mathbf{M} , $c \in \mathcal{C}$, $x \in \mathcal{D}_c$,

$\mathbf{M} \models [\text{ACCEPT}_c \wedge \text{DEPEND}_m^x] \supset K_c \sigma_i$, for all $i \geq 1$.

Proof: By induction on i .

Base case: $i = 1$. The claim is $\mathbf{M} \models [\text{ACCEPT}_c \wedge \text{DEPEND}_m^x] \supset K_c \Diamond \mathbf{F}_x \text{DEPEND}_m^x$, for all $c \in \mathcal{C}$, $x \in \mathcal{D}_c$. Assume bwoc that there is $(e, f) \in \text{Pts}(\mathcal{E})$ such that

$(e, f) \models \text{ACCEPT}_c \wedge \text{DEPEND}_m^x \wedge \neg K_c \Diamond \mathbf{F}_x \text{DEPEND}_m^x$. Then there is $(e_1, g) \sim_c (e, f)$ such that $(e_1, g) \models \neg \Diamond \mathbf{F}_x \text{DEPEND}_m^x \wedge \text{ACCEPT}_c$. Therefore, in some execution e_2 extending (e_1, g) , for some $d \in x$, $(e_2, h) \models \neg (\text{FAILED}_d \vee K_d \text{DEPEND}_m^x)$, for all $h \geq g$. Therefore, there is some $i \geq g$ such that $(e_2, i) \models \text{REFUSE}_d$ (because the system is nonblocking, and \mathbf{M} supports **Dependent Acceptance** and **Exclusive Dependencies**), but $(e_2, i) \models \text{REFUSE}_d \wedge \text{ACCEPT}_c$, violating **Decision Harmony**.

Inductive step: Assume the hypothesis is true for $i - 1$, so $\mathbf{M} \models [\text{ACCEPT}_c \wedge \text{DEPEND}_m^x] \supset K_c \sigma_{i-1}$; we show it for i . The claim is $\mathbf{M} \models [\text{ACCEPT}_c \wedge \text{DEPEND}_m^x] \supset K_c \sigma_i$, or $\mathbf{M} \models [\text{ACCEPT}_c \wedge \text{DEPEND}_m^x] \supset K_c \Diamond \mathbf{F}_x \sigma_{i-1}$, or $\mathbf{M} \models \text{ACCEPT}_c \wedge \text{DEPEND}_m^x \supset K_c \Diamond \wedge_{d \in x} (K_d \sigma_{i-1} \vee \text{FAILED}_d)$.

Bwoc, assume for some (e, f) that

$(e, f) \models \text{ACCEPT}_c \wedge \text{DEPEND}_m^x \wedge \neg K_c \Diamond \wedge_{d \in x} (K_d \sigma_{i-1} \vee \text{FAILED}_d)$. Then there is $(e_1, g) \sim_c (e, f)$ such that $(e_1, g) \models \text{ACCEPT}_c \wedge \text{DEPEND}_m^x \wedge \neg \Diamond \wedge_{d \in x} (K_d \sigma_{i-1} \vee \text{FAILED}_d)$.

Therefore, without loss of generality, for every $h \geq g$,

$(e_1, h) \models \text{ACCEPT}_c \wedge \text{DEPEND}_m^x \wedge \neg \wedge_{d \in x} (K_d \sigma_{i-1} \vee \text{FAILED}_d)$. There is thus a $d \in x$ such that $(e_1, h) \models \text{ACCEPT}_c \wedge \text{DEPEND}_m^x \wedge \neg (K_d \sigma_{i-1} \vee \text{FAILED}_d)$ for all $h \geq g$. Because this

is a nonblocking system, d must decide in e_1 ; in order not to violate **Decision Harmony**, it must accept. Therefore, there is $i \geq g$ such that

$(e_1, i) \models \text{ACCEPT}_c \wedge \text{DEPEND}_m^x \wedge \text{ACCEPT}_d \wedge \neg K_d \sigma_{i-1}$. violating the inductive hypothesis. \square

Nonblocking C-systems have a special interprocess knowledge property, illustrated above, which we can characterize as follows. Essentially, a deciding process p must know that eventually any process which must decide consistently with p must eventually fail or know enough to decide! In particular (assume we have a specific $x \in \mathcal{D}_c$),

1. if an accepting contractor must know ϕ , then the awarding manager must know that eventually each codependent will know ϕ or will fail.

if $\mathbf{M} \models \wedge_{c \in \mathcal{C}} (\text{ACCEPT}_c \supset K_c \phi)$

then $\mathbf{M} \models [\text{AWARD}_m^c \wedge \text{DEPEND}_m^x] \supset K_m \Diamond \mathbf{F}_x \phi$.

2. if an accepting contractor c must know ϕ , then c must know that eventually each codependent will fail or know ϕ .

if $M \models \bigwedge_{c \in \mathcal{C}} (ACCEPT_c \supset K_c \phi)$
 then $M \models [ACCEPT_c \wedge DEPEND_m^x] \supset K_c \diamond F_x \phi$.

3. if a deciding manager must know ϕ to award to c , then an accepting c must know that eventually the manager will fail or know ϕ .

if $M \models (AWARD_m^c \supset K_m \phi)$
 then $M \models ACCEPT_c \supset K_c \diamond F_{\{m\}} \phi$.

For example, let $\phi = DEPEND_m^x$; item 2 yields the base case of Lemma 39.

For rejection and refusal, we have (assume we have a specific $x \in \mathcal{D}_c$)

4. if $M \models \bigwedge_{c \in \mathcal{C}} (REFUSE_c \supset K_c \phi)$
 then $M \models [REJECT_m^c \wedge DEPEND_m^x] \supset K_m \diamond F_x \phi$.

5. if $M \models \bigwedge_{c \in \mathcal{C}} (REFUSE_c \supset K_c \phi)$
 then $M \models [REFUSE_c \wedge DEPEND_m^x] \supset K_c \diamond F_x \phi$.

6. if $M \models (REJECT_m^c \supset K_m \phi)$
 then $M \models REFUSE_c \supset K_c \diamond F_{\{m\}} \phi$.

We call items 1 through 6 the *Rule of Codependent Knowledge Necessitation*. Lemmas 33 and 34 prescribe $ACCEPT_c$ and $AWARD_m^c$ knowledge levels for process-failure-free nonblocking C-systems; those results use special cases of the Rule of Codependent Knowledge Necessitation.

The need to achieve the knowledge levels required by the Rule of Codependent Knowledge Necessitation seems to provide a daunting challenge to the protocol designer. After all, the need to achieve the similar knowledge levels of Lemma 34 led to the nonblocking impossibility results of Theorem 38, yet the three-phase atomic commitment protocol of Skeen (1982) is nonblocking in systems free of communication failures. We will now discuss informally how one might achieve these knowledge levels in a protocol. (That a *particular* protocol actually achieves these levels must be proved formally with respect to that protocol.) Assume that communication is failure-free and that a process q will receive any message sent to it (as long as q does not fail). Then process p knows, upon sending message \underline{m} , that q will eventually receive \underline{m} (as long as q does not fail).

Consider now that m has selected dependency set x . If m sends a “ $DEPEND_m^x$ ” message to each member of x announcing the dependency set, then, under our assumptions, each member of x will (either fail or) receive the message and therefore know the dependency set. After sending all $|x|$ messages, then, $K_m \diamond F_x DEPEND_m^x$ holds. For $AWARD_m^c$ to hold for each $c \in x$, $K_m \diamond F_x \diamond F_x DEPEND_m^x$ must hold (by the Rule of Codependent Knowledge Necessitation). If our protocol is so designed that m will now send a “ $K_m \diamond F_x DEPEND_m^x$ ” message to each member of x , then (assuming for the moment that m does not fail) $K_m \diamond F_x \diamond F_x DEPEND_m^x$ holds, even before sending the messages. Now, for $AWARD_m^c$ to hold, $K_m \diamond F_x \diamond F_{\{m\}} \diamond F_x DEPEND_m^x$ must also hold. Note, however, that when each $c \in x$ receives “ $K_m \diamond F_x DEPEND_m^x$ ”, then

$K_c K_m \diamond F_x \text{DEPEND}_m^x$ holds. Therefore, $K_m \diamond F_x \diamond F_{\{m\}} \diamond F_x \text{DEPEND}_m^x$ holds at the same time that $K_m \diamond F_x \diamond F_x \text{DEPEND}_m^x$ holds, which is before m even sends “ $K_m \diamond F_x \text{DEPEND}_m^x$ ” (1). Continuing along this line of reasoning, we can conclude that m may achieve its AWARD_m^c knowledge by first sending the $|x|$ “ DEPEND_m^x ” messages and then being committed by the protocol to send the $|x|$ “ $K_m \diamond F_x \text{DEPEND}_m^x$ ” messages (unless m fails).

Let us consider how c gains its accepting knowledge. $K_c \text{DEPEND}_m^x$ holds when c receives “ DEPEND_m^x ” from m . For ACCEPT_c to hold, the Rule of Codependent Knowledge Necessitation tells us, $K_c \diamond F_{\{m\}} \text{DEPEND}_m^x$ must hold; the stronger $K_c K_m \text{DEPEND}_m^x$ holds, however, when $K_c \text{DEPEND}_m^x$ does. Upon receiving “ $K_m \diamond F_x \text{DEPEND}_m^x$ ”, the required knowledge levels $K_c \diamond F_x \text{DEPEND}_m^x$ and $K_c \diamond F_{\{m\}} \diamond F_x \text{DEPEND}_m^x$ both hold. Based on our discussion above (see (1)), $K_c \diamond F_{\{m\}} \diamond F_x \diamond F_x \text{DEPEND}_m^x$ also holds. Again, continuing this line of reasoning, we may conclude that c achieves its ACCEPT_c knowledge by receiving “ DEPEND_m^x ” and then “ $K_m \diamond F_x \text{DEPEND}_m^x$ ”.

Suppose m instead fails before sending all “ $K_m \text{DEPEND}_m^x$ ” messages. Then the termination technique used by the “live” processes must guarantee that each “live” member of x will receive “ $\diamond F_x \text{DEPEND}_m^x$ ”. If the termination method is such, then again the informal reasoning above suggests that the requisite knowledge levels are attained.

The point is that, if our protocol is designed properly, then the processes can achieve the required infinitely nested knowledge levels in a small number of messages. This is the idea used in the nonblocking, three-phase atomic commitment protocol [Skee82, BeHG87], in which a PRECOMMIT message serves as a “ DEPEND_m^x ” message, and the COMMIT message serves as “ $K_m \diamond F_x \text{DEPEND}_m^x$ ”.

In a discussion of atomic commitment (under permanent process failures), Dwork and Skeen (1983) give a nonblocking protocol which uses $2(n-1)$ nonnull messages (excluding the contract announcements) in the failure-free case, which number they claim as a lower bound for nonblocking atomic commitment. Based on our discussion above, one might expect a protocol to require at least $3(n-1)$ messages. Dwork and Skeen achieve their lower bound in a strictly synchronous protocol which avoids $(n-1)$ messages (the set of messages that one might use, as suggested in the preceding paragraphs, to establish $K_c \diamond F_c \text{DEPEND}_m^c$, for all $c \in \mathcal{C}$) by associating with the achievement of $K_c \diamond F_c \text{DEPEND}_m^c$ (actually $K_c F_c \text{DEPEND}_m^c$) a null message receipt in a particular *step* in the protocol, for each $c \in \mathcal{C}$. In other words, the synchrony and absence of messages yields the required knowledge states. The protocol as presented is nonintuitive and difficult to understand. We found the protocol much easier to understand *once we associated the requisite levels of knowledge with each process at the appropriate step*. [Maze89] casts atomic commitment as negotiated commitment and shows how knowledge-theoretic results for nonblocking commitment behaviour correspond to results of Skeen (1982).

9 Summary

We specified negotiated commitment systems and derived results on properties of these systems, using a knowledge-theoretic approach. We determined both ① *what* knowledge states a process needs to commit to an outcome and ② *how* to attain the required knowledge, that is, system behaviour, in terms of message passing and local computation, required to attain these knowledge states. Using the knowledge results and the Message Chain Theorem, we gave several impossibility results: a message lower bound, impossibility of independent recovery, impossibility of termination under certain conditions, and impossibility of nonblocking behaviour under certain conditions. The result showing the impossibility of a weakly terminating commitment system under process failures and recovery and bounded communication time is new; the other impossibility results used some new intermediate results, extended some previously known results, and further demonstrated the utility of reasoning about knowledge in analyzing distributed problems. The impossibility results have a common form: first, determine the knowledge required for commitment in the given setting; second, show that certain kinds of message chains are required to achieve that knowledge; and third, show that the message chains (and therefore the knowledge) are unattainable.

The reader should take away from this discussion the following general points about reasoning about knowledge: ① from a specification of a distributed problem, one can derive knowledge requirements for solving the problem; ② from the knowledge requirements and system characteristics, one can derive message chain requirements; and, ③ using the knowledge and communication requirements, one can prove impossibility results, derive underlying protocol structure, and design protocols. Further, one uses high-level concepts that support the formalization of some common kinds of informal, intuitive reasoning.

This paper contains a significant exploration of reasoning about knowledge to analyze distributed problems. We and others assert that reasoning about knowledge offers insight into the nature of distributed computation, under various system assumptions. This approach also offers a tool which supports the analysis of problem specifications to yield useful insights into solutions. Often, one must appeal to intuition in formulating an approach to a solution to a problem, or a proof to a theorem, but the translation of that intuition into a precise form, to be manipulated and analyzed, can be difficult. Knowledge theory gives us a formal and direct way to articulate, clarify, and verify our informal intuition about the extent to which the local state of a process reflects important global states of the system. The use of knowledge theory to analyze distributed problems is relatively new. By applying the abstraction of knowledge to different kinds of problems, researchers are finding other “knowledge concepts” to be useful, such as probabilistic knowledge [FaHa88] or resource-bounded knowledge [Mose88, HaMT88]. We encourage the use of reasoning about knowledge for examining other distributed problems.

A Closure Properties of Systems of Executions

The proof of the Message Chain Theorem requires three closure properties. Given a set of executions \mathcal{E} , points $(e', g), (e, f) \in \mathbf{Pts}(\mathcal{E})$, and process set $P \subseteq \Pi$,

$\mathbf{replace}((e', g), P, (e, f))$ is the set of executions $e'' \in \mathcal{E}$ which extend (e', g) such that

- ① in e'' , each member of \bar{P} executes the same event at $(e'', g + 1)$ as it did at $(e', g + 1)$,
- ② in e'' , the event executed by each member of P at $(e', g + 1)$ is *replaced* by the event it executed at $(e, f + 1)$, and
- ③ the messages from P to Π are the messages not sent or received in e' between g and $g + 1$, plus any messages received in e' at $g + 1$ by $p \in P$ but not in e by p at $f + 1$, plus any messages newly sent by P ; and the messages from \bar{P} to Π are whatever was in e' at $g + 1$, plus whatever P received in e' at $g + 1$, minus whatever P received in e at $f + 1$ or was lost in transit to P in e at $f + 1$ [Hadz90].

The progress-closure properties we require here are these:¹⁴

S1 (Nonreceive Progress): The ability of a process to perform a nonRECV event depends on the process's behaviour only.

Let $(e, f), (e', g) \in \mathbf{Pts}(\mathcal{E})$ and $p \in \Pi$ be such that

- $(e, f) \sim_p (e', g)$,
- $\text{RECV}(\underline{m}, q) \not\sqsubseteq (e, f + 1, p)$, for any message $\underline{m}, q \in \Pi$

Then $\mathbf{replace}((e', g), \{p\}, (e, f)) \neq \emptyset$.

S λ (Null Receive Progress): A process' ability to receive a null message from another process depends on the state of the former process, the messages sent to the recipient by the latter process, and the behaviour of the communication system.

Let $e, e' \in \mathcal{E}$, $g \in N$, and $q, p \in \Pi$ be such that

- $(e, g) \sim_p (e', g)$,
- $e'(g, \mathcal{N})[\{q\}, \{p\}] \subseteq e(g, \mathcal{N})[\{q\}, \{p\}]$,
(There are no different messages in the communication system from q to p up to time g in e' than in e), and
- $\text{RECV}(\lambda, q) \sqsubset (e, g + 1, p)$.

Then $\mathbf{replace}((e', g), \{p\}, (e, g)) \neq \emptyset$.

S3 (Available Receive Progress): Once a process p has sent a message to another process q , q 's ability to receive the message cannot depend on p 's subsequent behaviour; q 's ability to receive the message does depend on the communication system.

¹⁴These closure properties correspond to a natural set of assumptions on the protocols which "produce" the behaviours in the execution sets [Maze89].

Let $e, e' \in \mathcal{E}$, $f, g \in N$, and $q, p \in \Pi$ be such that

- $(e, g) \sim_p (e', g)$
- $\text{RECV}(\underline{m}, q) \sqsubset (e, g + 1, p)$,
- $\langle q, \underline{m}, p, f \rangle \in e'(g, \mathcal{N})$ (the message is still available in e' at g , as in e)
- let $\eta = \{\langle q, \underline{m}, p, i \rangle \mid \langle q, \underline{m}, p, i \rangle \in e'(g, \mathcal{N}) \text{ and } 0 \leq i \leq f\}$;
 $\eta \subseteq e(g, \mathcal{N})[\{q\}, \{p\}]$ (The set of messages available at (e', g) , sent before $f + 1$, and from q to p are a subset of those available at (e, g) .)

Then $\text{replace}((e', g), \{p\}, (e, g)) \neq \emptyset$.

B Independent Recovery and Termination

We show a connection between a system supporting independent recovery and a system which is weakly terminating, k -transit-bounded, and subject to process failures and recovery. First, we weaken the definition of a C-system which supports independent recovery. In a system which supports *weak* independent recovery, a failed process *may* recover and decide in some extension without receiving a nonnull message.

Definition 40 In a C-system \mathbf{M} which supports *weak independent recovery*, \mathbf{M} is subject to process failures and recovery, and for all $(e, f) \in \text{Pts}(\mathcal{E})$, $f > 0$,

- if, for $c \in \mathcal{C}$, $\text{FAIL} \dashv e(f - 1, c)$ and $\text{FAIL} \not\vdash e(f, c)$,
then there is (e_1, g) extending (e, f) such that $(e_1, g) \models \text{ACCEPT}_c \vee \text{REFUSE}_c$
and $\text{RECV}(\underline{m}, p) \not\sqsubset e_1(g, c) - e_1(f - 1, c)$
for all messages \underline{m} and $p \in \Pi \setminus \{c\}$; and
- if $\text{FAIL} \dashv e(f - 1, m)$ and $\text{FAIL} \not\vdash e(f, m)$,
then there is (e_1, g) extending (e, f) such that $(e_1, g) \models \bigwedge_{c \in \mathcal{C}} (\text{AWARD}_m^c \vee \text{REJECT}_m^c)$
and $\text{RECV}(\underline{m}, p) \not\sqsubset e_1(g, m) - e_1(f - 1, m)$
for all messages \underline{m} and $p \in \Pi \setminus \{m\}$. \square

Therefore, any C-system which supports independent recovery also supports weak independent recovery.

Now, the proof of the impossibility of independent recovery becomes the proof of the impossibility of weak independent recovery, by the following changes to the proof of Claim 1 in the proof of Lemma 23: first, delete the line marked “(*)”; second, in the line marked “(**)”, change the phrase “independent recovery, there is $k \geq i$ ” to “weak independent recovery, there is (e_5, k) extending (e_5, i) ”. Therefore, we have

Proposition 41 There is no C-system which supports weak independent recovery. \square

Proposition 42 Any C-system M which is subject to process failures and recovery, k -transit bounded, and weakly terminating also supports weak independent recovery.

Proof: By Lemma 22, all processes are nonfailed and decided in any terminating point in M . Further, by Theorem 12, any point in M has a terminating extension in which no nonnull message is received. Pick any point $(e, f) \in \text{Pts}(\mathcal{E})$ such that $\text{Failed}(e, f) \neq \emptyset$. There is a terminating extension (e_1, g) of (e, f) in which no nonnull messages are received and all processes are nonfailed and decided. $\text{Failed}(e, f) = \text{Failed}(e_1, f)$. Therefore, for every process $p \in \text{Failed}(e_1, f)$, there is a time after f in e_1 such that p recovers (permanently) and p decides: for all $p \in \text{Failed}(e_1, f)$, there is h such that $f \leq h < g$ and $(e_1, h) \models \text{FAILED}_p$ and $(e_1, h') \models \neg \text{FAILED}_p$, for $h' > h$ and

(if $p = c$) $(e_1, g) \models (\text{ACCEPT}_c \vee \text{REFUSE}_c)$ and $\text{RCV}(\underline{m}, q) \notin e_1(g, p) - e_1(f - 1, p)$, or

(if $p = m$) $(e_1, g) \models \bigwedge_{c \in \mathcal{C}} (\text{AWARD}_m^c \vee \text{REJECT}_m^c)$ and $\text{RCV}(\underline{m}, q) \notin e_1(g, p) - e_1(f - 1, p)$.

Therefore, all p may recover independently. \square

An alternate proof to Theorem 30 is now the following: Fix any weakly terminating C-system M which is k -transit bounded and subject to process failures and recovery; by Proposition 42, M supports weak independent recovery, but Proposition 41 shows that no such system exists, a contradiction.

Acknowledgements

Many discussions with Vassos Hadzilacos greatly influenced the work reported here. Discussions with Shaike Artsy, Joe Halpern, Manolis Koubarakis, Yoram Moses, Jonathan Rose, Ken Sevcik, Mark Tuttle, Vic Vyssotsky, and Dave Wilkes helped us to clarify some important issues. This research was partially supported by the Natural Sciences and Engineering Research Council of Canada, under grant A3356, and by fellowships from Massey College, Trinity College, and the School of Graduate Studies at the University of Toronto.

References

- [BeHG87] P. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Database Systems*, Reading MA: Addison-Wesley Publishing Company, 1987.
- [Bürg89] U. Bürger. "A Flexible Two-Phase Commit Protocol." *Computer Networks and ISDN Systems*, **17**, 1989, 175-85.
- [ChLa85] K.M. Chandy and L. Lamport. "Distributed Snapshots: Determining Global States of Distributed Systems." *ACM Trans. on Computer Systems*, **3**, 1 (February 1985), 63-75.

- [ChMi86] K.M. Chandy and J. Misra. "How Processes Learn." *Distributed Computing*, 1, 1 (1986), 40-52.
- [FaHa88] R. Fagin and J.Y. Halpern. "Reasoning About Knowledge and Probability: Preliminary Report." *Proc. Conf. on Theoretical Aspects of Reasoning About Knowledge*, 6-9 March 1988, Asilomar CA, 277-93.
- [Gray79] J.N. Gray. "Notes on Data Base Operating Systems." in *Operating Systems: An Advanced Course*, Ed. R. Bayer, R.M. Graham, and G. Seegmüller, Berlin: Springer-Verlag, 1979, 393-481.
- [Hadz87] V. Hadzilacos. "A Knowledge Theoretic Analysis of Atomic Commitment Protocols (Preliminary Report)." *Proc. ACM Symp. Principles of Database Systems*, 1987, 129-34.
- [Hadz88] V. Hadzilacos. Private communication.
- [Hadz89] V. Hadzilacos. "On the Relationship Between the Atomic Commitment Problem and Consensus Problems." *Proc. Workshop on Fault-Tolerant Distributed Computing* (March 1986, Asilomar CA), Eds. B. Simons and A. Spector, Springer-Verlag, 1989.
- [Hadz90] V. Hadzilacos. "A Knowledge Theoretic Analysis of Atomic Commitment." Submitted for publication.
- [HaFa89] J.Y. Halpern and R. Fagin. "Modelling knowledge and action in distributed systems." *Distributed Computing*, 3, 4 (1989), 159-77.
- [Halp87] J.Y. Halpern. "Using Reasoning About Knowledge to Analyze Distributed Systems." in *Annual Review of Computer Science*, II, Ed. J.F. Traub, Annual Reviews, Inc., 1987, 37-68.
- [HaMo90] J. Halpern and Y. Moses. "Knowledge and Common Knowledge in a Distributed Environment." *Journal ACM*, 37, 3 (July 1990), 549-87.
- [HaMT88] J.Y. Halpern, Y. Moses, and M.R. Tuttle. "A Knowledge-Based Analysis of Zero Knowledge (Preliminary Report)." *Proc. Symp. Theory of Computing*, 2-4 May 1988, Chicago IL, 132-47.
- [HaZu89] J.Y. Halpern and L.D. Zuck. *A Little Knowledge Goes A Long Way: Simple Knowledge-based Derivations and Correctness Proofs for a Family of Protocols*, Revised Research Report RJ5857, IBM Research Laboratory, Almaden CA, 1989.
- [KoTo88] R. Koo and S. Toueg. "Effects of Message Loss on the Termination of Distributed Protocols." *Information Processing Letters*, 27, 4 (1988), 181-88.
- [Lamp80] L. Lamport. "'Sometime' Is Sometimes 'Not Never' (On the Temporal Logic of Programs)." *Seventh ACM Symposium on Principles of Programming Languages*, 28-30 January 1980, Las Vegas, 174-85.
- [Lamp81] B. Lampson. "Atomic Transactions." in *Distributed Systems — Architecture and Implementation*, B.W. Lampson, M. Paul, and H.J. Siegart, Eds. New York NY: Springer-Verlag, 1981. (ISBN 0-387-12116-1)
- [Maze89] M.S. Mazer. *A Knowledge-Theoretic Account of Negotiated Commitment*, Ph.D. Thesis, Department of Computer Science, University of Toronto, 1989 (available as Technical Report CSRI-237, Computer Systems Research Institute, University of Toronto, 1990).

- [Maze90] M.S. Mazer. "Communication Requirements for Knowledge Gain in Unpredictable Distributed Systems," in preparation. (A preliminary version appeared as "A Link Between Knowledge and Communication in Faulty Distributed Systems (Preliminary Report)." *Proc. Third Conference on Theoretical Aspects of Reasoning About Knowledge*, 4-7 March 1990, Asilomar CA, 289-304.)
- [Mose88] Y.O. Moses. "Resource-bounded Knowledge (Extended Abstract)." *Proc. Conf. Theoretical Aspects of Reasoning About Knowledge*, 6-9 March 1988, Asilomar CA, 261-76.
- [MoTu88] Y.O. Moses and M. Tuttle. "Programming Simultaneous Actions Using Common Knowledge." *Algorithmica*, **3** (1988), 121-69.
- [RoKa86] S. Rosenschein and L. Kaelbling. "The Synthesis of Digital Machines With Provable Epistemic Properties." *Proc. Conf. on Theoretical Aspects of Reasoning About Knowledge*, 19-22 March 1986, Monterey CA, 83-98.
- [Skee82] M.D. Skeen. *Crash Recovery in a Distributed Database System*. Ph.D. Thesis, Department of Electrical Engineering and Computer Science, University of California, Berkeley CA, 1982.
- [SkSt83] D. Skeen and M. Stonebraker. "A Formal Model of Crash Recovery in a Distributed System." *IEEE Transactions on Software Engineering*, **SE-9**, 3 (May 1983), 219-28.
- [Tutt89] M.R. Tuttle. *Knowledge and Distributed Computation*. Ph.D. Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge MA, 1989.

