Beyond Lamport's *Happened-Before*: On the Role of Time Bounds in Synchronous Systems

Ido Ben-Zvi[1] and Yoram Moses[2]

[1] Department of Computer Science, Technion
idobz@cs.technion.ac.il

[2] Department of Electrical Engineering, Technion
moses@ee.technion.ac.il

Abstract. Lamport's *Happened-before* relation is fundamental to coordinating actions in asynchronous systems. Its role is less dominant in synchronous systems, in which bounds are available on transmission times over channels. This paper initiates a study of the role that time bounds play in synchronous systems by focusing on two classes of problems: *Ordered Response*, in which a triggering event must be followed by a sequence of events (*"responses"*) performed in a prescribed temporal order, and *Simultaneous Response*, in which the responses must be performed simultaneously. In both cases, information about the triggering event must flow from its site of origin to the responding sites, and the responses must be timed as specified. A generalization of *happened- before* called *Syncausality*, is defined. A pattern of communication consisting of a syncausal chain coupled with an appropriate set of time bound guarantees gives rise to a communication structure called a *centipede.* Centipedes are a nontrivial generalization of message chains, and their existence is shown to be necessary in every execution of every protocol that solves ordered response. A variation on centipedes called *centibrooms* are shown to play an analogous role for Simultaneous Response: Every execution of a protocol for Simultaneous Response must contain a centibroom.

Keywords: synchronous message passing, bounded communication, ordered response, simultaneous response, causality, syncausality, knowledge, common knowledge, levels of knowledge, knowledge and time.

Dedicated to the memory of Amir Pnueli, a magnificent person and great inspiration.

1 Introduction

Many distributed systems applications need to react to spontaneous events, or ones initiated by their environment. Examples for such events are the activation of a fire alarm or smoke detector, a deposit or withdrawal from a bank account, or the identification of an interesting subject in a multi-camera surveillance system. In response to an external event, correct behavior of the system may require that an action, or more generally a set of actions, be performed. When multiple actions are performed, the temporal order in which the actions are performed is often important. Indeed, financial transactions will typically require a variety of actions involving testing various conditions and making related updates to be completed. To capture such situations, we define the following general problem:

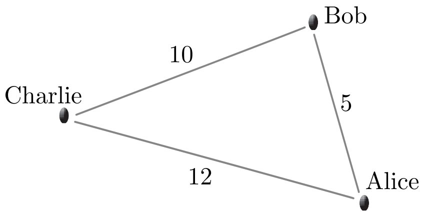
Definition 1 (Ordered Response [OrR]). *An instance of the* Ordered Response *problem is defined by a tuple* $\mathsf{OR} = \mathsf{OR}(e_{\mathtt{t}}, \alpha_1, \ldots, \alpha_k)$*, where* $e_{\mathtt{t}}$ *(the* triggering event*) is a spontaneous external input and* $\alpha_h = \langle a_h, i_h \rangle$ *where* a_h *is an action for process* i_h*.*[1] *A protocol solves* OR *if it guarantees that*
(1) if $e_{\mathtt{t}}$ *occurs, then process* i_h *will perform action* a_h*, for* $h = 1, \ldots, k$*;*
(2) α_h *will happen before* α_{h+1} *does, for all* $h < k$*; and finally*
(3) none of the actions a_h *will be performed in runs in which* $e_{\mathtt{t}}$ *does not occur.*

In the Ordered Response problem, α_h stands for the event of process i_h performing the action a_h. We shall denote the site of the triggering event $e_{\mathtt{t}}$ by i_0. Since the triggering event in $\mathsf{OR}(e_{\mathtt{t}}, \alpha_1, \ldots, \alpha_k)$ is a spontaneous event, information about the occurrence of $e_{\mathtt{t}}$ must flow from i_0 to each of the responding sites. Moreover, these sites must coordinate to perform the actions in the specified order. OR thus combines notification about $e_{\mathtt{t}}$ with a coordination problem. In asynchronous systems, both aspects of **OrR** are handled in a similar fashion, by generating message chains that ensure that Lamport's *happened-before* relation holds between $e_{\mathtt{t}}$ and α_1, and then between α_h to α_{h+1}, for all $h < k$. This paper studies **OrR** and related issues in *synchronous* systems, in which there are known bounds on message transmission, and processes share a global clock. Message chains play a somewhat different role in this setting. The following example illustrates some of the issues at play.

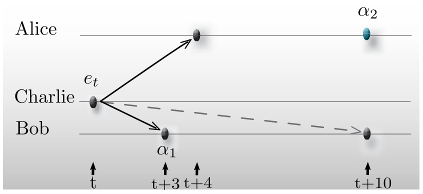
Example 1. Charlie's bank account is temporarily suspended due to credit problems. Should Charlie make a sufficient deposit at his local branch, Banker Bob at headquarters will re-activate the account. Alice holds a cheque from Charlie, but trying to cash it before the account is re-activated will result in her being fined by the bank rather than receiving payment. Alice, Bob and Charlie are connected by a communication network as depicted in Figure 1(a). In particular, messages from Charlie to Bob and Alice take up to 10 and 12 days to be delivered, respectively. We can view this as an instance of **OrR** in which the triggering event is a deposit by Charlie, and the responses are the account re-activation by Bob and, no sooner than Bob's action, Alice's cashing the cheque.

In a particular instance, depicted in Figure 1(b), Charlie makes a deposit at time t, and immediately broadcasts a message stating this to both Alice and Bob. The message reaches Bob in 4 days and Alice in 6. Bob immediately re-activates Charlie's account at $t + 4$. When can Alice deposit the cheque? The

[1] For simplicity, we assume that $e_{\mathtt{t}}$ happens at most once in any given execution, as do each one of the actions performed in response to it. The processes i_h need not be distinct, although it is natural and instructive to assume that adjacent processes in the sequence are distinct, so that $i_h \neq i_{h+1}$.



(a) The network of Example 1



(b) Cheque cleared at $t + 10$

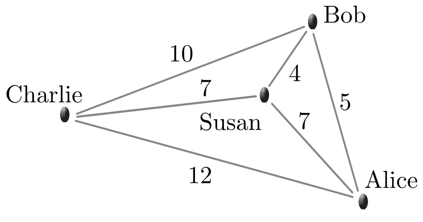
Fig. 1

cheque would be cashed successfully at any time after $t+4$. However, Alice only knows about Charlie's deposit at $t+6$. But even at that point, she must wait further. In the absence of additional information indicating when Bob actually received Charlie's message, she is only guaranteed that this will happen by time $t+10$. Knowing Bob's protocol, she can safely submit the cheque at or after time $t+10$, but not sooner. □

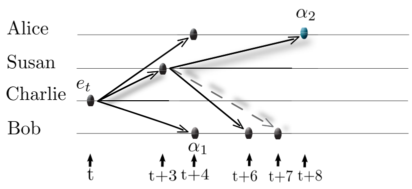
In this example, Alice acts after Bob does. While in the asynchronous setting she must obtain explicit notification that Bob acted, in the synchronous setting she can base her action on the information that Charlie sent Bob the message at time t, combined with the bound determining when this message will arrive, and her knowledge of Bob's protocol, which ensures action when Bob receives Charlie's message. Her action, which clearly depends on Bob's action having taken place, can be performed without an explicit message chain from Bob. This is no surprise. It is generally accepted that in the presence of bounds, Lamport's happened-before relation is not the sole factor in coordinating actions. Nevertheless, we know of no systematic treatment of how events can be coordinated in such synchronous systems. The purpose of this paper is to study the role that time bounds can play in coordination problems such as **OrR**. As we shall see, the set of communication structures that underly coordination in the synchronous case is considerably richer than it is in the asynchronous case.

Example 2. In a setting similar to Example 1, Susan is Bob's supervisor at the bank. The network is now as depicted in Figure 2(a). Suppose that Charlie broadcasts his deposit to all three, and that communication is delivered as in Figure 2(b). In this case Alice can, as before, submit her cheque at $t+10$. But she can do even better. Since she receives a message from Susan at $t+8$ that was sent at $t+3$, the bound on the (S, B) channel ensures her that Bob is be notified of Charlie's deposit by time $t+7$. The account will be solvent as of time $t+7$, and Alice can safely cash her cheque upon receiving Susan's message. □

In both examples, the timing of Alice's action depends on the time bounds. In Example 2, however, information that Alice receives in a message from Susan serves to update her knowledge about when Bob's action is performed, and enables her to perform her action earlier than she could before receiving it. This example illustrates the fact that the proper ordering of events can depend on a subtle interplay between the actual delivery times of messages, and the



(a) The network of Examples 2 and 3



(b) Susan speeds up Alice's response

Fig. 2

time bound guarantees. Notice that, for the purpose of properly ordering Alice's action, Susan plays a similar role in Example 2 to that played by Charlie in Example 1. Information about the triggering event is, in both cases, obtained from Charlie.

Intuitively, any solution to the **OrR** problem must ensure that particular knowledge is obtained following the occurrence of the triggering event. Since the response actions are performed if and only if $e_\mathtt{t}$ occurs, each of the responding processes must know that $e_\mathtt{t}$ occurred before performing any response action(s). If the triggering event $e_\mathtt{t}$ was unconditionally guaranteed to take place at some time t_0, then it would be trivial to coordinate an **OrR** response to $e_\mathtt{t}$ without need for any communication. However, being a spontaneous external input, $e_\mathtt{t}$ is not guaranteed to occur. Hence, information about its occurrence must flow from i_0 to the responders. The second aspect of **OrR** is the proper ordering of the responses. Before the response α_{h+1} can be performed, i_{h+1} must know that α_h has taken place. As the above examples suggest, this does not require explicit notification, and can be obtained by combining information about actual timing obtained through messages, with *a priori* bound information. But before α_h can take place, process i_h must know that all previous responses have occurred. As we shall show, ordered response is governed by a communication structure that we call a *centipede*, that generalizes the dynamics observed in Example 2.

Examples 1 and 2 illustrate how a process can come to know that communication directed at a remote site has arrived, based on transmission bound information. But bounds can be used in an additional fashion. Namely, if by time $t + b_{ij}$ process j receives no message sent by i at time t, then j can know that no such message was sent. Depending on i's protocol, this can provide j information about i's state at time t. Consider the following refinement of Example 2.

Example 3. In the network of Example 2 depicted in Figure 2(a), suppose that Susan sends Alice a message in every round as long as Susan has not heard from Charlie about an appropriate deposit. In this particular instance, Susan receives a message from Charlie at time $t+2$, at which point she stops sending her update messages. At time $t + 9$ Alice will be able to "time-out" on Susan's time $t + 2$ message. She then knows that Susan heard from Charlie at $t + 2$. Moreover, knowing that Susan relays information to Bob as before, Alice knows that Bob heard about the deposit no later than time $t + 5$. Hence, Alice can safely cash her cheque at time $t + 9$ rather than $t + 10$. $\square$

In Example 3 Alice learns of Charlie's deposit without receiving *any message whatsoever*. She clearly receives no message chain originating from Charlie. Nevertheless, it seems instructive to think of Susan as sending Alice a "silent message" at time $t + 2$, carrying relevant information, by *not* sending an actual message. This view will motivate an extension of Lamport's happened-before relation that we shall call *syncausality*, standing for "synchronous potential causality." A syncausal chain will then be a chain consisting of a sequence of messages and timeouts. As we will see, syncausality is a central element in informing processes about nondeterministic events such as spontaneous external inputs.

The main contributions of this paper are:

- The notion of syncausality is defined, generalizing Lamport's happened-before relation by adding timeout precedence.
- The Ordered Response (**OrR**) and Simultaneous Response (**SiR**) problems are defined, capturing a natural form of coordination in distributed systems. **OrR** is shown to require attaining nested knowledge, while Simultaneous Response requires obtaining common knowledge about the triggering event.
- Syncausality is shown to be a necessary condition for obtaining knowledge about nondeterministic events at a remote site.
- A notion of *Bound-based guarantees* is defined, corresponding to the causal guarantees that depend solely on transmission bounds. These guarantees account for changes in knowledge about remote sites that are not based solely on syncausality.
- *Centipedes*, a particular form of temporal communication structure, are defined. A centipede consists of a syncausal chain with "legs" that consist of bound-based guarantees. Centipedes are shown to be necessary in any run in which nested knowledge of an external input is attained. Consequently, every instance of **OrR** requires the construction of a centipede whose form depends on the instance of **OrR** being implemented. In a precise sense, centipedes are shown to be the analogue in synchronous systems to message chains through a given set of processes in asynchronous systems.
- *Centibrooms*, a slight variant of centipedes, are shown to be necessary for obtaining *common knowledge* (see [12]) of nondeterministic events in synchronous systems. This formally captures the fact that a single *pivotal event* is needed in order to obtain common knowledge in this setting.
- Finally, it is shown that every instance of **SiR** requires the construction of a centibroom whose structure depends on the instance of **SiR** being implemented.
- The technical results are obtained by way of a knowledge-based analysis. This is another illustration of the power of knowledge theory in the analysis of distributed system.

Related work: Explicit and implicit use of time bounds for coordination and improved efficiency is ubiquitous in distributed computing. An elegant example of its use is made by Hadzilacos and Halpern in [11]. That knowledge can be gained by way of timeouts when timing guarantees are available has been part of the folklore from decades. A tutorial by the second author suggests as a viable topic for future work performing an explicit analysis of the effect of timeouts on knowledge gain [20]. He also presents an example in which communication can be saved by using timeouts. However, [20] does not or suggest modifying Lamport causality to suit synchronous, and none of the new notions or technical results in the current paper were suggested in [20]. Krasucki and Ramanujam in [13] study of the interaction between knowledge and the ordering of events in a distributed system. They consider concurrency in a rather abstract setting, where they show that causality is related to the existence of particular partially ordered sets. They do not explicitly study the synchronous model, however, and do not explicitly consider synchronous time bounds on channels. Moses and Bloom [18] perform a knowledge-based analysis of clock synchronization in the presence of bounds on transmission times. They generalize Lamport's relation by defining a notion of timed causality $e \xrightarrow{\alpha} e'$ that corresponds to e taking place *at least* α time units before e'. It appears that '$\xrightarrow{\alpha}$' is a quantitative generalization of Lamport causality for the purpose of determining relative *timing* of events, while syncausality is a qualitative causality relation more suitable for studying knowledge gain and information flow. A similar notion appears in the work of Patt-Shamir and Rajsbaum [23]. Knowledge about knowledge touches on many fields, ranging from philosophy [15] and psychology [4], to linguistics [10,21], economics [1], AI [16], cryptography [5,24,9] and distributed systems [12,3,22]. In computer security, for example, it often becomes important to ensure that particular agents have access to particular information, and not know particular facts. Our analysis suggests tacit ways in which information can be transmitted in synchronous systems. One implication is that in order to guard against a particular form of knowledge gain, it is essential to deny the possibility of the appropriate centipede forming.

Organization: This paper is organized as follows. Section 2 presents a brief sketch of the model and the definitions used in the theorems and the proofs. Section 3 relates the Ordered Response problem to nested knowledge, and proves a knowledge gain theorem for two processes. Section 4 proceeds to define centipedes and state the Centipede Theorem for multi-process knowledge gain. In Section 5 we introduce the Simultaneous Response problem, review the definition of common knowledge, and relate the two. Centibrooms are defined, and are shown to be necessary in every instance of simultaneous response. Finally, Section 6 presents conclusions.

2 Background and Preliminary Definitions

As mentioned in Section 1, we focus on a simple synchronous setting with a global clock, in which processes that take steps at integer times, and bounds on transmission times over channels are given. We analyze knowledge in protocols that execute in such a setting by following the approach described in [8]. Namely, we separate the definition of the environment for which protocols are designed, formally called the *context*, from the actual protocol being executed in that context. Given a context γ and a protocol P designed to run in γ, there is a unique set $\mathcal{R} = \mathcal{R}(P, \gamma)$, of all runs of P in γ. This set is called a *system*, and we study how knowledge evolves in systems. The reason why it does not suffice to consider just one system—say the system consisting of all possible runs in γ—is because the protocol being executed plays an important role in determining what is known. Typically, the information inherent in receiving a particular message (or in not receiving one) depends on the protocol being used.

A fully detailed model is beyond the scope of this abstract. The crucial elements are the following.

– We assume that processes can receive external inputs from the outside world. These are determined in a genuinely nondeterministic fashion, and are not correlated with anything that comes before in the execution or with external inputs of other processes. Triggering events are always external events.
– The set of processes is denoted by $\mathbb{P}$. The network consists of the weighted channels graph over $\mathbb{P}$, in which the weights are the bounds b_{ij} for every channel (i, j). A copy of the (weighted) network, as well as the current global time, are part of every process' local state at all times.
– The scheduler, which we typically call the *environment*, is in charge of choosing the external inputs, and of determining message transmission times. The latter are also determined in a nondeterministic fashion, subject to the constraint that delivery satisfies the transmission bounds.
– Time is identified with the natural numbers, and each process is assumed to take a step at each time t. For simplicity, the processes follow deterministic protocols. Hence, a given protocol P for the processes and a given behavior of the environment completely determine the run.
– Events are sends, receives, external inputs and internal actions. All events in a run are distinct, and we denote a generic event by the letter e. For simplicity, events do not take time to be performed. At a given time point a process can perform an arbitrary finite set of actions.

We denote such a context by γ^{s}, and use $\mathcal{R}^{\mathsf{s}}$ to denote a system $\mathcal{R}(P, \gamma^{\mathsf{s}})$ consisting of the set of all runs of some protocol P in synchronous context γ^{s}. An *ND* (or *nondeterministic*) event is either (a) the arrival an external input, or (b) an early receive, i.e., a message delivery that occurs strictly before the transmission bound for its channel is met.

A *process-time node* (or simply *node*) is a pair (i, t), where i is a process and t is a time. Such a node represents an instant on i's timeline. While Lamport's happened-before relation is typically defined among events, we define *syncausality*, its generalization to synchronous systems, as a relation among process-time nodes. This choice allows us to avoid defining *non-receipt* events, for capturing timeouts and the expiration of time bounds. Since every event takes place at a particular node, working with nodes suffices. Formally, we proceed as follows.

Definition 2 (Syncausality). *Fix a run r. The* syncausality *relation $\rightsquigarrow$ over nodes of r is the smallest relation satisfying the following four conditions:*
1. If $t \leq t'$, then $(i, t) \rightsquigarrow (i, t')$;
2. If some message is sent at (i, t) and received at (j, t') then $(i, t) \rightsquigarrow (j, t')$;

3. *If i and j are connected by a channel with bound b_{ij} then $(i,t) \rightsquigarrow (j, t+b_{ij})$ for all t; and*
4. *If $(i,t) \rightsquigarrow (h,\hat{t})$ and $(h,\hat{t}) \rightsquigarrow (j,t')$, then $(i,t) \rightsquigarrow (j,t')$.*

Essentially, the first two clauses correspond to the local precedence and message precedence steps of Lamport's happened-before. Syncausality thus directly generalizes of happened-before. The third clause corresponds to *timeout precedence.*

2.1 Definition of Knowledge

We focus on a very simple logical language in which the set Φ of primitive propositions consists of propositions $\mathsf{occurred}(e)$ and $\mathtt{ND}(e)$, where e is an event. To obtain the logical language $\mathcal{L}$, we close Φ under propositional connectives and knowledge formulas. Thus, $\Phi \subset \mathcal{L}$, and if $\varphi \in \mathcal{L}$ and $i \in \mathbb{P}$, then $K_i\varphi \in \mathcal{L}$.[2] The formula $K_i\varphi$ is read *process i knows φ*. For ease of exposition, we assume that processes have *perfect recall* so that, intuitively, their local state at any time contains the full history of events that they have experienced. This assumption is needed only for the analysis of Response problems, and can be obtained by adding an auxiliary variable—only at the modeling stage and not the implementation—keeping track of the local history.

For defining the meaning of knowledge formulas, we follow the framework of [8]. The truth of formulas is evaluated with respect to a triple (R, r, t) consisting of a set of runs R, a run $r \in R$, and a time $t \in \mathbb{N}$, and we use $(R, r, t) \models \varphi$ to state that φ holds at time t in run r, with respect to R. Denoting by $r_i(t)$ process i's local state at time t in r, we inductively define

$(R, r, t) \models \mathsf{occurred}(e)$ if e occurs in r at a time $t' \leq t$;
$(R, r, t) \models \mathtt{ND}(e)$ if e is an ND event in r $\underline{\text{and}}$ $(R, r, t) \models \mathsf{occurred}(e)$; while
$(R, r, t) \models K_i\varphi$ if $(R, r', t') \models \varphi$ for every run r' satisfying $r_i(t) = r'_i(t')$.

By definition, $K_i\varphi$ is satisfied at a point (r, t) if φ holds at all points of R at which i has the same local state as at (r, t). Thus, given R, the local state determines what facts are true.

3 Knowledge and Ordered Response

We can now prove that particular nested knowledge is a precondition to action in **OrR**, formalizing and justifying the informal discussion in the introduction of how performing response actions in **OrR** requires processes to obtain nested knowledge.

Theorem 1. *Let $\mathsf{OR} = \mathsf{OR}(e_{\mathtt{t}}, \alpha_1, \ldots, \alpha_k)$ be an instance of* **OrR***, and assume that protocol P solves OR in γ^{s}. Let $r \in \mathcal{R}^{\mathsf{s}}$ be a run in which $e_{\mathtt{t}}$ occurs, let $1 \leq h \leq k$, and let t_h be the time at which i_h performs action a_h in r. Then*

$$(\mathcal{R}^{\mathsf{s}}, r, t_h) \models K_{i_h} K_{i_{h-1}} \cdots K_{i_1} \mathsf{occurred}(e_{\mathtt{t}}).$$

[2] This is a simplified language for ease of exposition. In Section 5 we extend it to allow for common knowledge.

Theorem 1 implies that for the last action in an ordered response to be performed, a nested knowledge formula stating that the last responder knows that the previous one knows... that the first responder knows $e_{\mathtt{t}}$. An analysis of when this property can hold will uncover the communication structure required in every run of an **OrR** protocol.

3.1 Causal Cones and Knowledge Gain

Having related the Ordered Response problem to knowledge, we now proceed to develop a theory that will provide the link that is still missing, relating knowledge and causality in synchronous systems. Lamport relates the happened-before relation to light cones in Minkowski space-time [14]. In the same vein, it is natural to consider past and future causal "cones" induced by syncausality. We define the *future causal cone* of a node α to be $\mathsf{fut}(r,\alpha) = \{\theta : \alpha \overset{r}{\rightsquigarrow} \theta\}$. Similarly, the *past causal cone* of α is $\mathsf{past}(r,\alpha) = \{\theta : \theta \overset{r}{\rightsquigarrow} \alpha\}$. Observe that the cones induced by syncausality in synchronous systems are significantly larger than the ones that follow just from Lamport's happened-before relation. Moreover, just as the future and past cones meet at the current point in space-time for light cones, we can show:

Lemma 1. *For all runs $r \in \mathcal{R}^{\mathsf{s}}$ and nodes α: $\mathsf{fut}(r,\alpha) \cap \mathsf{past}(r,\alpha) = \{\alpha\}$.*

The first step in relating syncausality to knowledge in synchronous systems comes from the observation that the events that occur in the past (syncausal) cone of a node completely determine the local state at the node. A proof by induction on all nodes (j,t') with $0 \le t' \le t$ shows:

Lemma 2. *Let $r, r' \in \mathcal{R}^{\mathsf{s}}$. If $\mathsf{past}(r,(i,t)) = \mathsf{past}(r',(i,t))$ and both runs agree on the initial states, external inputs, and early receives at all nodes of $\mathsf{past}(r,(i,t))$, then $r_i(t) = r'_i(t)$.*

Since the knowledge of a process in $\mathcal{R}^{\mathsf{s}}$ is determined by its local state, Lemma 2 implies that this knowledge depends only on the past causal cone. In the asynchronous setting, Chandy and Misra have shown that a process can know only about events in its past causal cone [3]. This is not the case in synchronous systems. It is true, however, for events known to be nondeterministic: Indeed, we can now state and prove using Lemma 2 the following knowledge gain theorem for two processes:

Theorem 2 (2-process Knowledge Gain). *Assume that e takes place at (i_0,t) in $r \in \mathcal{R}^{\mathsf{s}} = \mathcal{R}(P,\gamma^{\mathsf{s}})$. If $(\mathcal{R}^{\mathsf{s}}, r, t') \vDash K_{i_1}\mathtt{ND}(e)$ then $(i_0,t) \rightsquigarrow (i_1,t')$.*

The proof of Theorem 2 is obtained by constructing a run r' indistinguishable to i_1 at t' from r in which no ND events occur outside $\mathsf{past}(r',(i_1,t')) = \mathsf{past}(r,(i_1,t'))$. Theorem 2 captures a natural sense in which syncausality is a notion of potential causality for the synchronous model. Recall that every external input is, in particular, an ND event. Thus, Theorem 2 implies that knowledge about the occurrence of an external input requires a syncausal connection. Thus, by Theorem 1 it follows that in every run of an instance $\mathsf{OR} = \mathsf{OR}(e_\mathtt{t}, \alpha_1, \ldots, \alpha_k)$ there must be syncausal chains connecting the node (i_0, t) at which $e_\mathtt{t}$ to each of the nodes θ_h at which the responses α_h are performed.

4 Bound Guarantees and Centipedes

Our purpose in this section is to formalize and generalize the examples in the introduction, which dealt with a two-response instance of **OrR**, to obtain a structural requirement for arbitrary instances of **OrR**. Before we do so, let us revisit the issue of bounds on message transmission times. Recall that the bound b_{ij} provides a guarantee on the pace at which messages from i will reach j. We are sometimes interested in using these bounds to obtain upper bounds on message chains (or syncausal chains) that may involve a path of channels in the network, and not just one. It is natural, and will be useful, to define the *transmission distance* between processes h and k, which we denote by $D(h, k)$, to be the minimal distance between h and k in the weighted graph consisting of the communication network, in which edges (i, j) are assigned as weights the bounds b_{ij} on transmission times. In particular, $D(i, i) = 0$ for all $i \in \mathbb{P}$.

Intuitively, the examples in the Introduction demonstrated that without explicit information about actual communication deliveries, knowledge that a syncausal chain from an ND event has reached a destination node cannot be obtained before time equal to or exceeding the transmission distance from the source to destination has transpired. It is convenient to relate nodes by **bound guarantees** based on transmission distances as follows:

Definition 3. *We write* $(i, t) \dashrightarrow (j, t')$ *if* $t' \geq t + \mathsf{D}(i, j)$.

Observe that bound guarantees are independent of the speed at which messages *actually* arrive; they depend only on the weighted network topology. This is why it may be possible to know that a delivery has taken place based on the send event and the bound guarantees, without requiring further direct proof. As we shall see, if anyone knows at time t' that $(i, t) \rightsquigarrow (j, t')$ without obtaining syncausal information about early receives in the syncausal chain from (i, t) to (j, t'), then necessarily $(i, t) \dashrightarrow (j, t')$. Bound guarantees tie in with syncausal chains to determine the information structure that must underly solutions to ordered response. In fact, we now define a structure consisting of a careful combination of $\rightsquigarrow$ and $\dashrightarrow$ relations, that, in a precise sense, captures knowledge gain and ordering of events in synchronous systems.

Definition 4 (Centipede). *Let $r \in \mathcal{R}^\mathsf{s}$, let $i_h \in \mathbb{P}$ for $0 \leq h \leq k$ and let $t \leq t'$. A* centipede *for $\langle i_0, \ldots, i_k \rangle$ in the interval $(r, t..t')$ is a sequence of nodes $\theta_0 \rightsquigarrow \theta_1 \rightsquigarrow \cdots \rightsquigarrow \theta_k$ such that $\theta_0 = (i_0, t)$, $\theta_k = (i_k, t')$, and $\theta_h \dashrightarrow (i_h, t')$ holds for $h = 1, \ldots, k-1$.*

A centipede for $\langle i_0, \ldots, i_k \rangle$ in the interval $(r, t..t')$ is depicted in Figure 3. Intuitively, every node θ_{h+1} serves to ensure that its corresponding "leg" node (i_{h+1}, t') is causally affected, both by the origin node (i_0, t), and by the node θ_h.

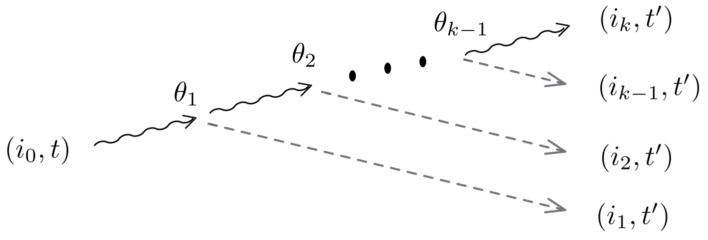


Fig. 3

The node θ_h, in turn, ensures the existence of a shorter centipede. We emphasize that the bound guarantee in $\theta_h \dashrightarrow (i_h, t')$ does not only (and will typically *not*) stand for causality based on unsent messages. Rather, it also stands for the fact that the information available at θ_h guarantees that i_h has enough time to learn by time t' about the part of the centipede to its left. This inductive structure underlies the close relationship the we will show between centipedes and knowledge gain. Observe that the shaded lines in Figure 2(b) outline an underlying 1-legged centipede.

We remark that, since both $\rightsquigarrow$ and $\dashrightarrow$ are reflexive, it is possible for adjacent θ_j's to coincide. Moreover, it is possible (in fact, probably quite common) for θ_h to occur at the same site i_h, with its "leg" (i_h, t_h) in many cases. Indeed, every simple (Lamport-style) message chain gives rise to a centipede of a simple form in which all internal nodes θ_h are co-located in this sense with their respective legs (i_h, t_h). It follows that a centipede is a natural, albeit nontrivial, generalization of a Lamport-causal chain. We can now show that a centipede is a necessary condition for knowledge gain in synchronous systems:

Theorem 3 (Centipede Theorem). *Let P be an arbitrary protocol, and let $r \in \mathcal{R}^{\mathsf{s}} = \mathcal{R}(P, \gamma^{\mathsf{s}})$. Moreover, assume that e is an ND event at (i_0, t) in r. If $(\mathcal{R}^{\mathsf{s}}, r, t') \models K_{i_k} K_{i_{k-1}} \cdots K_{i_1} \mathtt{ND}(e)$, then there is a centipede for $\langle i_0, \ldots, i_k \rangle$ in $(r, t..t')$.*

The proof of Theorem 3 is based on a nontrivial and subtle knowledge-based analysis of the interaction between bound guarantees and syncausality. Combining the Centipede Theorem with Theorem 1 we can obtain:

Corollary 1. *Let P be a protocol solving $\mathsf{OR} = \mathsf{OR}(e_{\mathtt{t}}, \alpha_1, \ldots, \alpha_k)$ in γ^{s}, and assume that $e_{\mathtt{t}}$ occurs at (i_0, t) in $r \in \mathcal{R}^{\mathsf{s}}$. If i_k performs a_k at time t' in r then there is a centipede for $\langle i_0, \ldots, i_k \rangle$ in $(r, t..t')$.*

5 Simultaneous Response and Centibrooms

In synchronous systems it is often desirable to perform actions simultaneously at different sites. A natural variant of **OrR** is the *Simultaneous Response* problem, defined as follows.

Definition 5 (Simultaneous Response [SiR]). *Let $e_{\mathtt{t}}$ be an external input. Then $\mathsf{SR} = \mathsf{SR}(e_{\mathtt{t}}, \alpha_1, \ldots, \alpha_k)$ defines an instance of the* Simultaneous Response *problem. A protocol solves SR if it guarantees that if the triggering event $e_{\mathtt{t}}$ occurs, then at some later point all actions $\alpha_1,\ldots,\alpha_k$ in the response set of SR will be performed simultaneously.*

As in the case of ordered response, we can obtain insight into the structure of simultaneous response via knowledge theory. The state of common knowledge has been shown to play an important role in agreements and in coordinating simultaneous actions [12,7,8]. Common knowledge, however, involves knowledge about knowledge for unbounded depths. To formally treat common knowledge, we extend our logical language $\mathcal{L}$ by adding the operators E_G (*everyone in G knows*) and C_G (*the processes in G have common knowledge that*) for every $G \subseteq \mathbb{P}$. Thus, if $\varphi \in \mathcal{L}$ then so are $E_G\varphi$ and $C_G\varphi$. We use $(E_G)^k$ as shorthand for nesting k levels of E_G. The definition of satisfaction for formulas is now extended by the following clauses:

$$(R, r, t) \vDash E_G\varphi \text{ if } (R, r, t) \vDash K_i\varphi \text{ for every } i \in G; \quad \text{and}$$
$$(R, r, t) \vDash C_G\varphi \text{ if } (R, r, t) \vDash (E_G)^k\varphi \text{ for every } k \geq 1.$$

In the terminology of [8], the responses $\alpha_1, \ldots, \alpha_k$ in an instance of **SiR** induce a *perfectly coordinated ensemble* of events in $\mathcal{R}^{\mathsf{s}}$. Using Proposition 11.2.2 of [8] we can conclude that common knowledge of the occurrence of triggering event $e_{\mathtt{t}}$ must hold before the responses can be performed in **SiR**:

Theorem 4. *Let $\mathsf{SR} = \mathsf{SR}(e_{\mathtt{t}}, \alpha_1, \ldots, \alpha_k)$ be an instance of* **SiR***, and assume that protocol P solves SR in γ^{s}. Moreover, $G = \{i_1, \ldots, i_k\}$ be the set of processes appearing the response set of SR. Finally, let $r \in \mathcal{R}^{\mathsf{s}}$ be a run in which $e_{\mathtt{t}}$ occurs. If the responses are performed at time t in r, then $(\mathcal{R}^{\mathsf{s}}, r, t) \vDash C_G\mathtt{ND}(e_{\mathtt{t}})$.*

As we now show, the Centipede Theorem can be extended to show an analogous result for common knowledge, with the centipede replaced by a simpler structure, defined as follows:

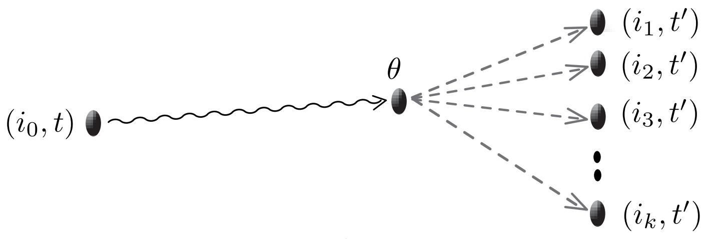


Fig. 4

Definition 6 (Centibroom). *Let $t \leq t'$ and $G \subseteq \mathbb{P}$. There is a* centibroom *$Br\langle i_0, G\rangle$ in $(r, t..t')$ if there is a node θ satisfying $(i_0, t) \rightsquigarrow \theta \dashrightarrow (i_h, t')$ for all $i_h \in G$.*

A pictorial depiction of a centibroom is given in Figure 4. The node θ is called the *pivot* of the centibroom. Observe that a pivot node embodies a "pivotal event" for the group G of processes: This pivot makes it possible, in principle, to guarantee that all members of G will know by time t' of the existence of this pivot for time t'.

Clearly, centibrooms are simpler structures than general centipedes. Notice, however, that a centibroom for $G = \{j_1, \ldots, j_\ell\}$ can be considered as a condensed representation of infinitely many centipedes, each of which can support knowledge gain of a particular formula. More concretely, we have the following.

Lemma 3. *Let $G \subseteq \mathbb{P}$, and let θ be a pivot node for $Br\langle i_0, G\rangle$ in $(r, t..t')$. Then for every sequence $\langle i_1, \ldots, i_k\rangle \in G^k$ of processes in G, the sequence $(i_0, t) \cdot \theta^k$ (where θ repeats k times) is a centipede for $\langle i_0, \ldots, i_k\rangle$ in $(r, t..t')$.*

Notice that Lemma 3 does not bound the value of k, nor does it restrict the possibility of repetitions in the sequence $\langle i_1, \ldots, i_k\rangle$ in question. Given the centipede Theorem, it seems natural to conjecture that a centibroom is a candidate to serve as the structure underlying common knowledge. We now show that this is indeed the case.

Theorem 5 (Common Knowledge Gain). *Let P be an arbitrary protocol, let $G \subseteq \mathbb{P}$, let $\mathcal{R}^{\mathsf{s}} = \mathcal{R}(P, \gamma^{\mathsf{s}})$, and let $r \in \mathcal{R}^{\mathsf{s}}$. Moreover, assume that e is an ND event at (i_0, t) in r. If $(\mathcal{R}^{\mathsf{s}}, r, t') \vDash C_G\texttt{ND}(e)$, then there is a centibroom $Br\langle i_0, G\rangle$ in $(r, t..t')$.*

The proof of Theorem 5 is based on the Centipede Theorem. Recall that $C_G\varphi$ implies arbitrarily deeply nested knowledge of φ. Every such nested knowledge formula implies the existence of a centipede. A nested knowledge formula is constructed whose centipede has sufficiently many nodes that at least one of them must be a pivot for G at t'. What results is the desired centibroom.

The pivot node in a centibroom embodies a "pivotal event" for the group G of processes. Theorem 5 shows that such a pivotal event is the *only* way common knowledge can arise in synchronous systems. This demonstrates that the nature of common knowledge is finitistic, despite its familiar definition being based on an infinite conjunction of facts. This phenomenon is consistent with the analysis of common knowledge in the work on fault-tolerance [6,19,17]. There, too, common knowledge arises at some time t' exactly if there is some property S of the correct nodes that ensures that all agents will know by time t' that the property S held in the run.

We remark that Theorem 5 relates to a familiar situation involving the evolution of knowledge in broadcasts. In a flooding protocol or a radio broadcast, for example, the contents being broadcast become common knowledge to a growing set of participants with time. Typically, after a time interval equivalent to the diameter of the system, the contents can become common knowledge to *all* processes in the system.

As in the case of Ordered Response, we can use Theorem 5 relate the *Simultaneous Response* problem **SiR** and centibrooms, as follows:

Corollary 2. *Let P be a protocol solving* $\mathsf{SR} = \mathsf{SR}(e_{\mathtt{t}}, \alpha_1, \ldots, \alpha_k)$ *in* γ^{s}*, and assume that* $\mathtt{e_t}$ *occurs at* (i_0, t) *in* $r \in \mathcal{R}^{\mathsf{s}}$*. If the response actions are performed at time* t' *in* r*, then there is a centibroom* $Br\langle i_0, G\rangle$ *in* $(r, t..t')$*.*

6 Conclusions

We have performed an analysis of causality in synchronous systems, where processes share a global clock and channels have bounded transmission times. We introduced the Ordered Response and Simultaneous Response problems, as natural coordination tasks in a distributed system. They involve natural forms of temporal and simultaneous coordination in any given context. While in asynchronous systems we have, via Lamport's relation, that causality requires message chains, causality has a richer structure in the synchronous setting. First of all, timeouts allow information flow via non-messages. This gives rise to syncausal chains, which consist of sequences of messages and timeouts. But syncausal chains do not tell the full story of causality in synchronous systems. We showed that guaranteeing that one event happens before another does not depend on a linear structure of information flow. Rather, the centipede structure, consisting of a restricted tree form combining syncausality and bound guarantees, is at the base of temporally ordering events under synchrony. Similarly, centibrooms are at the essence of simultaneous coordination in synchronous systems.

Our results all hold in particular in the case in which $b_{ij} = \infty$ for all channels, so that communication is asynchronous (although processes share the global clock and can move at each step). Because communication is asynchronous, bound guarantees are useless in this setting. Syncausality reduces to Lamport's happened-before, all possible centipedes collapse to message chains, and centibrooms do not exist. Thus, our results also apply to such contexts, reproving Chandy and Misra's Knowledge-gain theorem in a slightly more general setting. Asynchrony of communication alone suffices for this type of implosion.

Our theorems provide necessary conditions for information flow based on syncausality. How knowledge actually evolves in a system will depend on the particular protocol used. For example, a timeout at $t + b_{ij}$ is ineffective if the protocol would never have the "sender" i send a message at time t. Alice can learn by timing out on Susan's message in Example 3 only because, had Susan not obtained confirmation of Charlie's deposit, she would have sent a message at $t + 2$. The protocol used plays a crucial role in this knowledge transfer. As a first study of the role that protocols play in determining information flow in the synchronous contexts γ^{s}, we analyze the full-information protocol in a follow-on paper [2]. The necessary conditions in this paper's theorems are necessary and sufficient in that case. It follows that our characterization of coordination in terms syncausality, centipedes, and centibrooms is, in a precise sense, tight.

In future work we plan to study causality and coordination in a similar fashion for other models with synchronous features. Perhaps the most urgent would be a study of semi-synchronous systems in which clocks are private, and may drift over time. Some of our ideas regarding timeouts, syncausality and bound guarantees should have suitably modified analogues in such contexts. But additional issues may also be also be involved there, knowledge about the current level of synchronization, and about other processes' knowledge regarding clocks. Our work proves that outside the realm of asynchronous systems causality and the ordering of events involve more than Lamport's happened before relation. The theory of causality beyond asynchronous systems and Lamport's happened before promises to be rich and interesting.

Acknowledgments. We thank Danny Dolev, Idit Keidar and Ayelet Shiri for useful discussions that improved the presentation. Special thanks to Moshe Vardi for suggesting the term *centibroom.*

This research was supported in part by ISF grant 1339/05.

References

1. Aumann, R.J.: Agreeing to disagree. Annals of Statistics 4(6), 1236–1239 (1976)
2. Ben-Zvi, I., Moses, Y.: Sufficient conditions for knowledge gain and information flow in synchronous systems (2010) (in preparation)
3. Chandy, K.M., Misra, J.: How processes learn. Distributed Computing 1(1), 40–52 (1986)
4. Clark, H.H., Marshall, C.R.: Definite reference and mutual knowledge. In: Joshi, A.K., Webber, B.L., Sag, I.A. (eds.) Elements of Discourse Understanding. Cambridge University Press, Cambridge (1981)
5. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory 22(5), 644–654 (1976)
6. Dwork, C., Moses, Y.: Knowledge and common knowledge in a Byzantine environment: crash failures. Information and Computation 88(2), 156–186 (1990)
7. Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y.: Common knowledge revisited. In: Proc. 6th TARK, pp. 283–298. Morgan Kaufmann, San Francisco (1996)
8. Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y.: Reasoning about Knowledge. MIT Press, Cambridge (2003)
9. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM Journal on Computing 18(1), 186–208 (1989)
10. Grice, H.P.: Logic and conversation, pp. 41–58 (1975)
11. Hadzilacos, V., Halpern, J.Y.: Message-optimal protocols for byzantine agreement. Mathematical Systems Theory 26(1), 41–102 (1993)
12. Halpern, J.Y., Moses, Y.: Knowledge and common knowledge in a distributed environment. Journal of the ACM 37(3), 549–587 (1990)
13. Krasucki, P.J., Ramanujam, R.: Knowledge and the ordering of events in distributed systems (extended abstract). In: Proc. 5th TARK, pp. 267–283. Morgan Kaufmann, San Francisco (1994)
14. Lamport, L.: Time, clocks, and the ordering of events in a distributed system. Communications of the ACM 21(7), 558–565 (1978)
15. Lewis, D.: Convention, A Philosophical Study. Harvard University Press, Cambridge (1969)
16. McCarthy, J., Hayes, P.J.: Some philosophical problems from the standpoint of artificial intelligence. In: Michie, D. (ed.) Machine Intelligence, vol. 4, pp. 463–502. Edinburgh University Press, Edinburgh (1969)

17. Mizrahi, T., Moses, Y.: Continuous consensus via common knowledge. Distributed Computing 20(5), 305–321 (2008)
18. Moses, Y., Bloom, B.: Knowledge, timed precedence and clocks. In: Proc. 13th ACM Symp. on Principles of Distributed Computing, pp. 294–303 (1994)
19. Moses, Y., Tuttle, M.R.: Programming simultaneous actions using common knowledge. Algorithmica 3, 121–169 (1988)
20. Moses, Y.: Knowledge and communication: a tutorial. In: Proc. 4th TARK, pp. 1–14. Morgan Kaufmann, San Francisco (1992)
21. Parikh, R.: Finite and infinite dialogues. In: Moschovakis, Y.N. (ed.) Logic from Computer Science, MSRI Publication No. 21, pp. 481–497. Springer, Heidelberg (1991)
22. Parikh, R., Krasucki, P.: Levels of knowledge in distributed computing. Sādhanā 17(1), 167–191 (1992)
23. Patt-Shamir, B., Rajsbaum, S.: A theory of clock synchronization (extended abstract). In: Proc. 26th ACM STOC, pp. 810–819 (1994)
24. Rabin, M.O.: How to exchange secrets with oblivious transfer. Cryptology ePrint Archive, Report 2005/187 (2005), Originally written in (1981)