The Complexity of Reasoning About Knowledge and Time: Synchronous Systems

Joseph Y. Halpern
Moshe Y. Vardi

IBM Research
Almaden Research Center
650 Harry Rd.
San Jose, CA 95120-6099

Abstract: We study the propositional modal logic of knowledge and time for distributed systems. We consider a number of logics (ninety-six in all!), which vary according to the choice of language and the assumptions made on the underlying system. The major parameters in the language are whether there is a common knowledge operator, whether we reason about the knowledge of one or more than one processor, and whether our temporal operators are branching or linear. The assumptions on distributed systems that we consider are: whether or not processors forget, whether or not processors learn, whether or not time is synchronous, and whether or not there is a unique initial state in the system. We completely characterize the complexity of the validity problem for all the logics we consider. This paper focuses on synchronous systems. Our results here include a Π^1_1 upper bound for the language with common knowledge with respect to systems where processors do not forget, and a corresponding non-elementary-time result for the language without common knowledge. We also provide a complete axiomatization for the latter language.

1. Introduction[1]

It has been argued recently that knowledge is a useful concept for analyzing the behavior and interaction of processors in a distributed system [CM,DM,FI1,Hal,HF,HM1,LR,MT,PR, RK,Ros]. When analyzing a system in terms of knowledge, not only is the current state of knowledge of the processors in the system relevant, but also how that state of knowledge changes over time. A formal propositional logic of knowledge and time was first proposed by Sato [Sa]; others have since been proposed by Lehmann [Le1], Fagin *et al.* [FHV1], Parikh and Ramanajum [PR], and Ladner and Reif [LR]. Still others are implicit in all the other references cited above.

While Sato proved a nondeterministic exponential upper bound for his logic, Lehmann stated a theorem claiming a doubly exponential upper bound for his logic (which included common knowledge), and Ladner and Reif prove that one of their logics is undecidable. This apparent inconsistency is, of course, due to the fact that all these papers actually consider different logics. To add to the confusion, these papers use the same notation with different interpretations.

In this paper we try to bring some order to this confusion by categorizing logics for knowledge and time along two major dimensions: the language used and the assumptions made on the underlying distributed system. By varying these parameters, we end up with ninety-six logics. (Of course, they are not all of equal interest to distributed computing!) All of the logics considered in the papers mentioned above fit into our framework. Our major results involve completely characterizing the complexity of all these logics, showing how the subtle interplay of the parameters can have a tremendous impact on complexity.

The languages considered in the literature vary according to the modalities used for knowledge and time. As far as knowledge goes, the relevant issue is whether the language can talk about the knowledge of more than one agent, and whether we have a modal operator in the language for common knowledge (where common knowledge of a fact φ holds if everyone knows φ, everyone knows that everyone knows φ, etc.). For time, the question is whether we use *branching time* or *linear time* modalities (which essentially amounts to whether or not we can quantify over the possible executions of a program).

It is well known that if we consider either knowledge or time alone, the language used has a great impact on the complexity of the logic. As was shown by Halpern and Moses [HM2], the complexity of reasoning about knowledge for the notion of knowledge most appropriate for distributed systems (which satisfies the axioms of the modal logic S5), the validity problem for the logic is co-NP-complete if we can only reason about one agent or processor in the language, PSPACE-complete with two or more agents, and EXPTIME-complete if we add common knowledge to the language. If we consider time alone, the validity problem for the language with branching time modalities is EXPTIME-complete [EH1], while for the

---

1 The first two sections of this paper are essentially identical to the first two sections of [HV1]; we include them here for completeness. While in [HV1] we described in detail lower bound results, here we describe in detail some upper bound and completeness results. We plan to publish a more exhaustive acccount of our upper bound and completeness results in a future paper.

language with linear time modalities it is PSPACE complete [SC]. Not surprisingly, we find a similar phenomenon here; the complexity of reasoning about knowledge and time depends on the language used. What is perhaps more interesting is how the assumptions made on the underlying distributed system, which essentially place conditions on the interaction between knowledge and time, affect complexity.

The types of assumptions on the system that are typically made include whether or not processors *forget* (the assumption of no forgetting has also been called *unbounded memory* or *cumulative knowledge* in other papers [FHV2,HV,Mo]), whether or not processors can *learn*, whether or not there is a *unique initial state* in the system, and whether time is *synchronous* or *asynchronous*. We now explain each of these parameters in more detail, and motivate them in terms of distributed systems.

We first discuss the notion of knowledge in a distributed system. Although there have been many papers that consider this notion, they all have the same essential features. A distributed system is identified with a set of possible *runs* of the system, where a run is a complete history of the system's behavior over time. Thus, the run may include such things as each processor's initial state and its complete message history (i.e. the messages it has sent and received, in the order they were sent and received, time-stamped if the processors have local clocks). Formally, assume we have a system of m processors, each of which at any time is in some local state. This local state may encode such things as the processor's initial state, part or all of its message history, and the values of relevant variables. A run is a function from time (which, for simplicity, we assume is discrete and ranges over the natural numbers) to *global states* of the form $<l_1,...,l_m>$, where l_i is the local state of processor i.[2] Given a run r and a time n, we can think of the global state $r(n)$ as a "snapshot" describing the current state of the system. We can think of n as denoting the time on some external global clock (not necessarily observable by the processors). Following [HM1], we call such a pair (r,n) a *point*.

Processor i is said to *know* a fact φ (written $K_i\varphi$) at a given point if φ is true at all other points in which it is in the same state. Intuitively, a processor cannot distinguish two points if it is in the same state in both; thus, it knows φ if φ is true at all the points it cannot distinguish from the true state of affairs.[3] We say a processor *considers run r' possible* at point (r,n) if for some n', it cannot distinguish (r,n) from (r',n').

We say a processor *does not forget* if the set of runs the processor considers possible stays the same or decreases over time (intuitively, as a result of the processor getting more information). So if at some point (r,n) in run r processor i considers run r' possible, then

---

2 In a more general model we might augment the global state to include a component describing the *environment*, which intuitively consists of all the relevant features of the system not described by the processors' local states, such as messages in transit but not yet delivered, and so on (cf. [FHV2]). The environment component plays no role in the complexity analysis, so we omit it here.

3 This interpretation of knowledge is called a *state-based interpretation* in [HM1], and is essentially the interpretation used in [PR,HF,Ros,RK,FI1]. We will not consider the more general *epistemic interpretations* discussed in [HM1].

run r was indistinguishable from r' at all points in the past. Intuitively, a processor that cannot distinguish two runs that it could distinguish at an earlier time must have "forgotten" the information that allowed it to distinguish those runs. Note that no forgetting intuitively requires unbounded memory, so that a processor can store all the information it has received. Thus, the distinction between forgetting and no forgetting essentially corresponds to whether we view our processors as finite-state machines or Turing machines.

The dual notion to "no forgetting" is "no learning". A processor *does not learn* if the set of runs it considers possible stays the same or increases over time. More formally, if at some point (r,n) processor i considers run r' possible, then processor i will consider run r' possible at all times in the future (i.e. at all points (r,n') with $n' \geq n$). If processor i cannot distinguish two points $(r,0)$ and $(r',0)$ in a system with no learning and no forgetting, then i goes through the same sequence of states in both r and r', regardless of what messages i may receive. Such a system essentially corresponds to a *non-adaptive* algorithm. A processor does not modify its actions in response to signals from the outside world; in this precise sense, we can say that no learning takes place.

In some systems the assumption is made that each processor has a *unique initial state.* This means that there is a unique initial global state for the system (i.e. for all runs r and r', the global states $r(0)$ and $r'(0)$ are identical). The assumption of a unique initial state seems fairly innocuous. After all, we can always add a new initial state to every run and then let it develop as it did before. However, as we shall see, this assumption is not so innocuous when combined with the assumption of no learning.

In a *synchronous* system, we assume that a processor has access to a global clock that ticks at every instant of time and the clock reading is part of its state, so the processor always knows the time. Note that protocols that proceed in rounds can be viewed as running in synchronous systems.

An *interpreted system* is a pair (R,π) where R is a system and π is a truth assignment to the primitive propositions at every point of R. There is a straightforward way to extend π to all formulas (the details are discussed in the next section). For the rest of our discussion, it will be useful to have notation for different classes of interpreted systems and different languages. We use $\mathscr{C}$ to represent the class of all interpreted systems. We then use subscripts *nf*, *nl*, *uis*, *sync* to indicate restrictions to intepreted systems where, respectively, processors do not forget, processors do not learn, where there is a unique initial state, and where the system is synchronous. Thus, for example, $\mathscr{C}_{(nf,sync,uis)}$ represents the class of interpreted systems where processors do not forget, the system is synchronous, and there is a unique initial state.

We use the notations $CKL_{(m)}$, $CKB_{(m)}$, $KL_{(m)}$, and $KB_{(m)}$ to describe the languages we use. The L and the B tells us whether linear time or branching time modalities are used, the presence or absence of C in the name indicates whether or not common knowledge is included, and the subscript indicates the number of agents. Thus, $CKL_{(2)}$ is the language that uses linear time modalities and has modal operators K_1, K_2, and C for the knowledge of agent 1, agent 2, and common knowledge. (We describe the language and give its semantics in detail

in the next section.) Similarly, $KB_{(3)}$ is the language that uses branching time modalities and K_i, $i = 1,2,3$, but has no modal operator for common knowledge.

The logics that have been considered in other papers can now be classified as follows. Sato [Sa] and Lehmann [Le1] restrict attention to $\mathscr{C}_{(nf,sync)}$: synchronous systems where processors do not forget. Lehmann uses the languages $CKL_{(m)}$; Sato essentially does as well (although his language does not have explicit temporal operators). Halpern and Fagin [HF], Parikh and Ramanujam [PR], and the *tree logic of protocols* of Ladner and Reif [LR] also assume no forgetting, but do not require that time be synchronous, so the class of interpreted systems considered for these logics is $\mathscr{C}_{(nf)}$. On the other hand, these papers differ in the languages they consider: $CKL_{(m)}$ in [HF] and $KB_{(m)}$ in [PR] and [LR].[4] Ladner and Reif's *linear logic of protocols*, despite the name, also uses (a subset of) branching time, but restricts attention to the class of interpreted systems $\mathscr{C}_{(nf,nl,uis)}$, where processors neither forget nor learn, and there is a unique initial state. In the remaining papers that consider formal models of knowledge and time [CM,FI1,Ros,RK], the assumption of no forgetting is not imposed; all the interpreted systems in $\mathscr{C}$ are considered. However, in [Ros,RK] linear time is used, while [CM] and [FI1] implicitly use branching time, although neither of these latter two papers explicitly has temporal operators in their logics.

We do not discuss here which of these logics is most appropriate. Our feeling is that the choice should be guided by the application at hand (see [Pn,La2,EH2] for a discussion of these issues in the context of linear vs. branching time logics). Instead, we focus our attention on the complexity of the decision procedures for each of them.

At a high level, we can view our results as saying that assuming either no forgetting or no learning tends to make the complexity of reasoning about knowledge and time much worse. For example, if we have common knowledge in the language (and at least two agents, since common knowledge reduces to knowledge if we have only one agent), then the validity problem with respect to many classes of interpreted systems where processors do not forget or do not learn, such as $\mathscr{C}_{(nf)}$ and $\mathscr{C}_{(nl)}$, is wildly undecidable, in fact, Π^1_1-complete. (A precise definition of Π^1_1 appears in Section 3.) This means that there can be no complete axiomatization for these cases (since a complete axiomatization would imply that the set of valid formulas was r.e.).[5] On the other hand, for classes such as $\mathscr{C}$ or $\mathscr{C}_{(sync,uis)}$ where we do not make the assumption that processors do not learn or do not forget, the complexity of the validity problem for the language with common knowledge is (only!) EXPTIME-complete.

---

4 Actually, in [PR] there are also modal operators for what is called *implicit knowledge* in [HM1,HM2]. In addition, the branching time modalities used in [PR] and [LR] only give us a subset of the language $KB_{(m)}$ (a different subset in each of the papers). However, these differences have no impact on the complexity, so we do not focus on them further here. Also, the fact that the systems in [LR] are actually trees instead of sets of runs imposes another mild condition that we briefly discuss in the next section. Again, this difference has no impact on complexity.

5 As we remarked above, Lehmann claimed a doubly-exponential time decision procedure for his logic, which is $CKL_{(m)}$ interpreted over interpreted systems in $\mathscr{C}_{(nf,sync)}$. He also claimed a complete axiomatization [Le1]. Lehmann later retracted these claims, and only claimed these results for the one-agent case, without common knowledge [Le2]. Of course, our results show that the original claims were in fact incorrect.

A similar situation arises if we consider the language without common knowledge. Although the validity problem in the presence of no forgetting or no learning is in general decidable, it is non-elementary, while if we do not make the assumption that processors do not learn or do not forget, the validity problem is either PSPACE-complete or EXPTIME-complete (depending on whether we consider linear time or branching time).

There are some anomalous situations though, mainly those involving the combination of no learning and a unique initial state. For example, Ladner and Reif show that the validity problem for $KB_{(2)}$ is undecidable (even without common knowledge in the language) with respect to $\mathscr{C}_{(nf,nl,uis)}$. An easy extension of their proof shows it is actually Π^1_1-complete; these results also hold for the language $KL_{(2)}$. On the other hand, if we consider the class of interpreted systems $\mathscr{C}_{(nf,nl,sync,uis)}$, where we impose the additional condition of synchrony, the situation collapses. The validity problem for this logic is EXPSPACE-complete, even with common knowledge in the language! Intuitively, the reason is that the combination of these assumptions implies that no expressive power is gained by having common knowledge or more than one agent in the language.

Our results are summarized in the table below. The results given in the table are tight: the upper bounds match the lower bounds (to within constant factors). In order to explain the results for the languages $KL_{(m)}$ and $KB_{(m)}$, $m \geq 2$, in the first two rows of the table in a little more detail, we must introduce some notation. Let $ex(m,n)$ be defined inductively via $ex(0,n) = n$, $ex(m+1,n) = 2^{ex(m,n)}$ (so that, intuitively, $ex(m,n)$ is a stack of m 2's, with the top 2 having exponent n), let the *alternation depth* of φ, written $ad(\varphi)$, be the number of alternations of distinct knowledge modalities (K_i's) in φ, and let $|\varphi|$ be the length of φ when viewed as a string of symbols. The nonelementary time bound means that there is an algorithm for deciding if a formula φ is valid which runs in time $ex(1 + ad(\varphi), c|\varphi|)$, for some constant $c > 0$. Furthermore, any algorithm for deciding validity must run in time $ex(1 + ad(\varphi), d|\varphi|)$ for some constant $d > 0$ and infinitely many formulas φ. The explanation of the nonelementary space bound is analogous. Note that, by definition, for any formula φ of $KL_{(1)}$ or $KB_{(1)}$ we have $ad(\varphi) \leq 1$. Thus, the bounds for $KL_{(1)}/KB_{(1)}$ in the first two rows of the table are special cases of the bounds for $KL_{(m)}/KB_{(m)}$. In particular, Lehmann's doubly-exponential time upper bound for KL_1 is a special case of ours.

The difference between the nonelementary time bounds in the first row of the table, and the nonelementary space bounds in the second row of the table can roughly be explained by noting that allowing learning gives us the ability to encode alternation. More precisely, when we have no forgetting but allow learning, we can encode alternating Turing machines that run in space $ex(ad(\varphi), c|\varphi|)$ (which corresponds to time $ex(ad(\varphi)+1, c|\varphi|)$); once we impose the assumption of no learning, we can only encode deterministic Turing machines that run in space $ex(ad(\varphi), c|\varphi|)$.

Given the number of results, we concentrate in this paper on upper bounds and axiomatization in synchronous systems. For the lower bounds the reader is referred to [HV1] The rest of this paper is organized as follows. The next section describes the languages and the various kinds of interpreted systems discussed above in detail. The rest of the paper focuses on the class $\mathscr{C}_{(nf,synch)}$. In Section 3 we present our Π^1_1 upper bound results for the

	$CKL_{(m)}/CKB_{(m)}$, $m \geq 2$	$KL_{(m)}/KB_{(m)}$, $m \geq 2$	$KL_{(1)}/KB_{(1)}$
$\mathscr{C}_{(nf)}$, $\mathscr{C}_{(nf,sync)}$, $\mathscr{C}_{(nf,uis)}$, $\mathscr{C}_{(nf,sync,uis)}$	Π^1_1	nonelementary (time $ex(ad(\varphi)+1, c\|\varphi\|)$)	double-exponential time
$\mathscr{C}_{(nl)}$, $\mathscr{C}_{(nf,nl)}$, $\mathscr{C}_{(nf,nl,sync)}$, $\mathscr{C}_{(nl,sync)}$	Π^1_1	nonelementary (space $ex(ad(\varphi), c\|\varphi\|)$)	EXPSPACE
$\mathscr{C}_{(nf,nl,uis)}$	Π^1_1	Π^1_1	EXPSPACE
$\mathscr{C}_{(nl,uis)}$	co-r.e.	co-r.e.	EXPSPACE
$\mathscr{C}_{(nl,sync,uis)}$, $\mathscr{C}_{(nf,nl,sync,uis)}$	EXPSPACE	EXPSPACE	EXPSPACE
$\mathscr{C}$, $\mathscr{C}_{(sync)}$, $\mathscr{C}_{(sync,uis)}$, $\mathscr{C}_{(uis)}$	EXPTIME	PSPACE for $KL_{(m)}$, EXPTIME for $KB_{(m)}$	PSPACE for $KL_{(m)}$, EXPTIME for $KB_{(m)}$

Figure 1: The complexity of the validity problem for logics of knowledge and time

languages $CKL_{(m)}$ and $CKB_{(m)}$, $m \geq 2$, while in Section 4 we present the upper bound for $KL_{(m)}$, and in Section 5 we provide a complete axiomatization for $KL_{(m)}$.

2. The Formal Model: Language and Systems

The logics we are considering are all propositional. Thus, we start out with primitive propositions p, q, ... and we close the logics under negation and conjunction, so that if φ and ψ are formulas, so are $\sim\varphi$ and $\varphi \wedge \psi$. In addition, we close off under modalities for knowledge and time, as discussed below. As usual, we view *true* as an abbreviation for $\sim(p \wedge \sim p)$, $\varphi \vee \psi$ as an abbreviation for $\sim(\sim\varphi \wedge \sim\psi)$, and $\varphi \Rightarrow \psi$ as an abbreviation for $\sim\varphi \vee \psi$. We assume that $\wedge$ and $\vee$ bind more tightly than $\Rightarrow$, so that we write, for example, $\varphi \Rightarrow \psi \wedge \psi'$ rather than $\varphi \Rightarrow (\psi \wedge \psi')$.

If we have m agents (in distributed systems applications, this would mean a system with m processors), we add the modalities K_1, ..., K_m. Thus, if φ is a formula, so is $K_i\varphi$ (read "player i knows φ"). In some cases we also want to talk about common knowledge, so we add the modalities E and C into the language; $E\varphi$ says that everyone knows φ, while $C\varphi$ says φ is common knowledge.

The temporal modalities (sometimes called *operators* or *connectives*) that we use depend on whether we are considering linear time or branching time. In the linear time case, we have a unary operator $\circ$ and a binary operator U. Thus, if φ and ψ are formulas, then so are $\circ\varphi$ (read *nexttime* φ) and $\varphi U \psi$ (read φ *until* ψ). We view $\Diamond\varphi$ as an abbreviation for *true*$U\varphi$, while $\Box\varphi$ is an abbreviation for $\sim\Diamond\sim\varphi$. Intuitively, $\circ\varphi$ says that φ is true at the next point (one time unit later), $\varphi U \psi$ says that φ holds until ψ does, $\Diamond\varphi$ says that φ is eventually true (either in the present or at some point in the future), and $\Box\varphi$ says that φ is always true (in the present and at all points in the future). In the branching time case, we also have quantifiers over runs, so that if φ and ψ are formulas, so are $\forall\varphi U\psi$, $\exists\varphi U\psi$, $\forall\circ\varphi$, and $\exists\circ\varphi$. A formula

of the form $\forall \bigcirc \varphi$ is true at the point (r,n) if $\bigcirc\varphi$ is true at (r',n) for all runs r' extending (r,n), where the notion of *extending* will be made precise below. Similarly, $\exists \varphi U \psi$ is true at (r,n) if $\varphi U \psi$ is true at (r',n) for some run r' extending r. Again, we view $\forall\Diamond\varphi$ (resp. $\exists\Diamond\varphi$) as an abbreviation for $\forall true U \varphi$ (resp. $\exists true U \varphi$), and $\forall\Box\varphi$ (resp. $\exists\Diamond\varphi$) as an abbreviation for $\sim\exists\Diamond\sim\varphi$ (resp. $\sim\forall\Diamond\sim\varphi$). Thus, for example, $\forall\Diamond\varphi$ is true at the point (r,n) if φ is eventually true for all runs r' extending (r,n). It has been argued that a nexttime operator ($\bigcirc$) is inappropriate for reasoning about asynchronous systems (cf. [La1]); after all, the processors do not have access to an external clock in such systems, so it is not even clear that the notion of the ticking of such a clock makes sense. We remark that all our lower bounds also hold if the language does not have a nexttime operator.[6]

As we mentioned in the introduction, we take $|\varphi|$ to be the length of the formula φ viewed as a string of symbols, while in the languages without C and E (i.e. $KL_{(m)}$ and $KB_{(m)}$) we define $ad(\varphi)$ to be the greatest number of alternations of distinct K_i's along any branch in φ's parse tree. For example, $ad(K_1 \sim K_2 K_1 p) = 3$; temporal operators don't count, so that $ad(K_1 \Box K_1 p) = 1$. Note that $ad(\varphi) \leq |\varphi|$, and if φ is in $KL_{(1)}$ or $KB_{(1)}$, then $ad(\varphi) \leq 1$.

A *system* for m processors consists of a set R of runs, where each run $r \in R$ is a function from $\mathbb{N}$ to L^m, where L is some set of *local states*. Thus, $r(n)$ has the form $<l_1,...,l_m>$; such a tuple is called a *global state*. (Formally, we could view a system as a tuple (R, L, m), making the L and m explicit; we have chosen not to do so in order to simplify notation. The L and m should always be clear from context.) An *interpreted system M for m processors* is a tuple (R, π) where R is a system for m processors, and π maps every point $(r,n) \in R \times \mathbb{N}$ to a truth assignment $\pi(r,n)$ on the primitive propositions (so that $\pi(r,n)(p) \in \{\text{true}, \text{false}\}$ for each primitive proposition p).

We now give semantics to $CKL_{(m)}$ and $KL_{(m)}$. Given an interpreted system $M = (R, \pi)$, we write $(M,r,n) \models \varphi$ if the formula φ is true at (or *satisfied by*) the point (r,n) of interpreted system M. We define $\models$ inductively for formulas of $CKL_{(m)}$ (for $KL_{(m)}$ we just omit the clauses involving C and E). In order to give the semantics for formulas of the form $K_i\varphi$, we need to introduce one new notion. If $r(n) = <l_1,...,l_m>$, $r'(n') = <l'_1,...,l'_m>$, and $l_i = l'_i$, then we say that $r(n)$ and $r'(n')$ are *indistinguishable to processor i* and write $(r,n) \sim_i (r',n')$. Of course, $\sim_i$ is an equivalence relation on global states. $K_i\varphi$ will be defined to be true at (r,n) exactly if φ is true at all the points whose associated global state is indistinguishable to i from that of (r,n). We proceed as follows:

- $(M,r,n) \models p$ for a primitive proposition p iff $\pi(r,n)(p) = \text{true}$
- $(M,r,n) \models \varphi \wedge \psi$ iff $(M,r,n) \models \varphi$ and $(M,r,n) \models \psi$
- $(M,r,n) \models \sim\varphi$ iff $(M,r,n) \not\models \varphi$
- $(M,r,n) \models K_i\varphi$ iff $(M,r',n') \models \varphi$ for all (r',n') such that $(r,n) \sim_i (r',n')$

---

6 The G, F, and U operators of [PR] correspond to our $\forall\Box$, $\forall\Diamond$ and $\forall U$ respectively. Parikh and Ramanujam do not have a nexttime operator in their language. The $\Box$, $\Box^*$, $\Diamond$, and $\Diamond^*$ of [LR] correspond to our $\forall\bigcirc$, $\forall\Box$, $\exists\bigcirc$, and $\exists\Diamond$ respectively. Ladner and Reif have neither $\forall\Diamond$ nor an until operator. All our results are easily seen to hold for these restricted languages. We could, of course, also allow more complicated mixtures of modalities, such as $\forall\Box\Diamond\varphi$, as in the logics CTL* [EH2] or MPL [Ab]. Doing this seems to increase the complexity of the decision procedure by at least one exponential (cf. [VS]).

- $(M,r,n) \models E\varphi$ iff $(M,r',n') \models K_i\varphi$ for $i = 1,...,m$
- $(M,r,n) \models C\varphi$ iff $(M,r',n') \models E^k\varphi$, for $k = 1,2,...$ (where $E^1\varphi = E\varphi$ and $E^{k+1}\varphi = EE^k\varphi$)
- $(M,r,n) \models \bigcirc\varphi$ iff $(M,r,n+1) \models \varphi$
- $(M,r,n) \models \varphi U\psi$ iff there is some $n' \geq n$ such that $(M,r,n') \models \psi$, and for all n'' with $n \leq n'' < n'$, we have $(M,r,n'') \models \varphi$.

There is a graphical interpretation of the semantics of E^k and C which we shall find useful in the sequel. Fix an interpreted system M. We say a point (r',n') in M is *reachable* from a point (r,n) *in k steps* if there exist points $(r_0,n_0),...,(r_k,n_k)$ such that $(r,n) = (r_0,n_0)$, $(r',n') = (r_k,n_k)$, and for all $j = 0,...,k-1$ there exists i such that $(r_j,n_j) \sim_i (r_{j+1},n_{j+1})$. We say (r',n') is *reachable* from (r,n) if it is reachable in k steps for some k. It is easy to check that $(M,r,n) \models E^k\varphi$ iff $(M,r',n') \models \varphi$ for all points (r',n') reachable from (r,n) in k steps, and $(M,r,n) \models C\varphi$ iff $(M,r',n') \models \varphi$ for all points (r',n') reachable from (r,n).

We remark here that we could have presented the semantics in a slightly different way, more closely related to the standard Kripke semantics for knowledge (see, for example, [HM2]). Instead of associating to each point (r,n) the global state $r(n)$, we could view points as more abstract entities, without this additional structure. An interpreted system would now consist of a set of runs, a truth assignment π, and equivalence relations $\sim_1,...,\sim_m$ on the points. The semantics of formulas such as $K_i\varphi$ could be defined using these equivalence relations just as above. This approach was taken in an earlier version of this paper [HV] and is also taken by Lehmann [Le1]. The two definitions are equivalent in an obvious way: once we associate a global state to each point, we can use that to define an equivalence relation. Conversely, once we have an equivalence relation on the points, we can associate a global state with each point in such a way that two points are indistinguishable to i iff they are equivalent. We will use this observation in a number of our proofs below. We have chosen to use global states here in order to emphasize the intuitions coming from distributed systems. This choice also allows us to define branching time semantics in a natural way.

Given an interpreted system $M = (R,\pi)$, we say that $r' \in R$ *extends* the point $(r,n) \in R \times \mathbb{N}$ if $r'(n') = r(n')$ for all $n' \leq n$; i.e. if r and r' go through the same sequence of global states up to time n. With this definition, we can now give semantics to branching time formulas as follows:

- $(M,r,n) \models \exists\bigcirc\varphi$ iff $(M,r',n+1) \models \varphi$ for some run r' extending (r,n)
- $(M,r,n) \models \forall\bigcirc\varphi$ iff $(M,r',n+1) \models \varphi$ for all runs r' extending (r,n)
- $(M,r,n) \models \exists\varphi U\psi$ iff for some run r' extending (r,n) there exists some $n' \geq n$ such that $(M,r',n) \models \psi$, and for all n'' with $n \leq n'' < n'$, we have $(M,r',n'') \models \varphi$
- $(M,r,n) \models \forall\varphi U\psi$ iff for all runs r' extending (r,n) there exists some $n' \geq n$ such that $(M,r',n') \models \psi$, and for all n'' with $n \leq n'' < n'$, we have $(M,r',n'') \models \varphi$.[7]

---

7 The notion of branching time we have defined here differs slightly from that defined in [LR] and an earlier version of this paper [HV]. In these papers, the set of runs has a tree-like structure, which guarantees that the set is *limit closed*. As defined here, the set of runs is not necessarily limit closed, making it more like Abrahamson's MPL [Ab] than CTL (see [Em,EH2] for a detailed discussion of this issues). In our framework, we can say that a set R of runs is limit closed if, for all runs r, the fact that for all n there is a run $r_n \in R$ extending (r,n) implies that $r \in R$. By imposing the additional condition of limit closure on our classes of

As usual, we define a formula φ to be *valid with respect to a class $\mathcal{D}$ of interpreted systems* iff $(M,r,n) \models \varphi$ for all interpreted systems $M \in \mathcal{D}$, runs r in M, and times n. A formula φ is *satisfiable with respect to $\mathcal{D}$* iff for some $M \in \mathcal{D}$, r, and n we have $(M,r,n) \models \varphi$. It will often be more convenient for us to consider the satisfiability problem rather than the validity problem in proving lower bounds.

We now turn our attention to formally defining the classes of interpreted systems discussed in the introduction.

We say processor i *does not forget in* $M = (R,\pi)$ if all runs $r,r' \in R$ and times n,n',k, if $(r,n) \sim_i (r',n')$ and $k \leq n$, then there exists $k' \leq n'$ such that $(r,k) \sim_i (r',k')$. In order to motivate this definition, define *processor i's history at the point* (r,n) to be the sequence $l_0,\ldots,l_k$ of states that processor i takes on in run r up to time n, with consecutive repetitions omitted. For example, if from time 0 through 4 in run r processor i goes through the sequence l,l,l',l,l of states, its history at $(r,4)$ just l,l',l. Roughly speaking, processor i does not forget if it "remembers" its history. More precisely we have

Lemma 2.1. *Processor i does not forget in a system R iff for all runs $r,r' \in R$, if $(r,n) \sim_i (r',n')$ then processor i's history is the same at (r,n) and (r',n').*

Proof. The fact that remembering the history implies no forgetting is immediate from the definition. The converse can be proved by a straightforward induction on $n+n'$. ■

This lemma shows that no forgetting requires an unbounded number of local states in general, since processor i may have an infinite number of distinct histories in a given system. There is one more observation about systems where processors do not forget that we frequently use; this is captured in the following lemma.

Lemma 2.2. *If processor i does not forget in R and $(r,n) \sim_i (r,n')$, then $(r,n) \sim_i (r,n'')$ for all n'' with $n \leq n'' \leq n'$.*

Proof. We proceed by induction on n. Note that since $(r,n) \sim_i (r,n')$ and $n'' \leq n'$, by definition of no forgetting there must be some $k \leq n$ such that $(r,k) \sim_i (r,n'')$. If $n = 0$, we must have $k = n$. If $n > 0$, then if $k = n$ we are done, while if $k < n$, by the induction hypothesis (where k plays the role of n, n plays the role of n'', and n'' plays the role of n'), it follows that $(r,n) \sim_i (r,k)$, and by transitivity we get $(r,n) \sim_i (r,n'')$. ■

A system where processor i does not forget is shown in Figure 2, where the vertical lines denote runs (with time 0 at the top) and all points that i cannot distinguish are enclosed in the same region.

In a system where processor i *does not learn*, we have the opposite situation: If $(r,n) \sim_i (r',n')$, then for all $k \geq n$ there must be some $k' \geq n'$ such that $(r,k) \sim_i (r',k')$. A system where processor i does not forget and does not learn is shown in Figure 3. With no learning, the equivalence relations do not refine. Note how i goes through the same sequence of states in all runs it cannot distinguish (modulo *stuttering*, i.e. the same state repeating at consecutive

---

runs, we get precisely the classes considered in [LR]. This condition has no impact on the complexity of the decision procedure, although it does slightly affect the axioms for the logic. In practice we would not want to impose this condition since it is easier to consider issues of fairness without it.

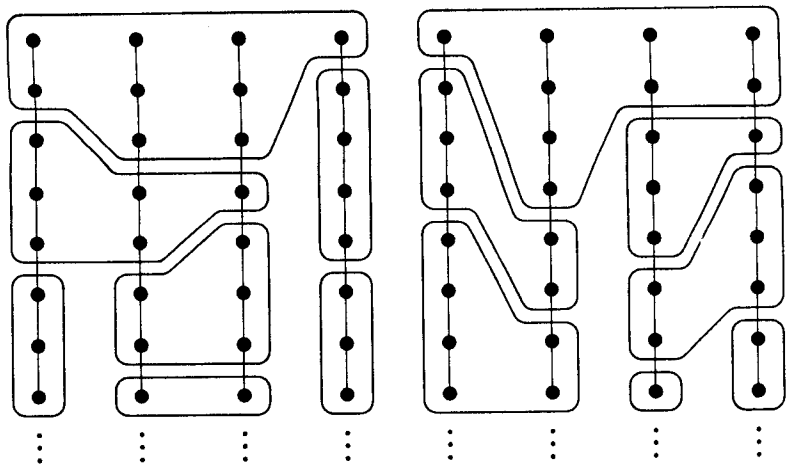

Figure 2: A system where processor i does not forget.

points). (We remark that if we consider no learning but allow forgetting, the situation is slightly more complicated. If processor i cannot distinguish $(r,0)$ and $(r',0)$, then there may be a set S of states such that i is in every state of S infinitely often in both runs r and r', but it does not go through the states in the same sequence in r and r'.)

In a *synchronous* system, we assume that every processor has access to a global clock that ticks at every instant of time, and the clock reading is part of its state. Thus, in a synchronous system, each processor always "knows" the time. More formally, we say a time is synchronous in R if for all processors i and all runs r,r', if $(r,n) \sim_i (r',n')$, then $n = n'$. We remark that in a previous version of this paper [HV], we took a slightly weaker definition: we required that for all runs r, if $(r,n) \sim_i (r,n')$ then $n = n'$. Let us call a system that satisfies that latter condition *weakly synchronous*. Note that the definition of weakly synchronous only considers one run r rather than two runs r and r'. It is easy to show (by induction on n) that the two definitions are equivalent for systems where processors do not forget. However, in general

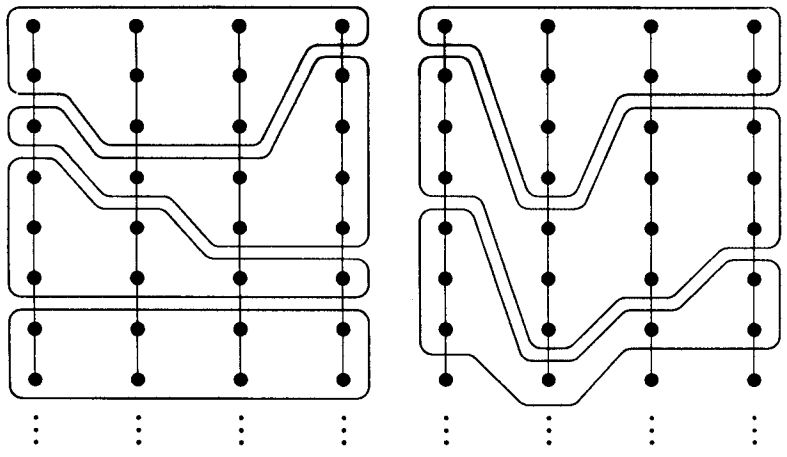

Figure 3: A system where processor i does not forget and does not learn.

they are different. (We remark that the notion of weak synchrony is important in some of our proofs.) Observe that in a synchronous system where $(r,n) \sim_i (r',n)$, an easy induction on n shows that if i does not forget and $n > 0$, then $(r,n-1) \sim_i (r',n-1)$, while if i does not learn, then $(r,n+1) \sim_i (r',n+1)$.

Finally, we say that a system R has a *unique initial state* if for all runs $r,r' \in R$, we have $r(0) = r'(0)$. Thus, if R is a system with a unique initial state, then we have $(r,0) \sim_i (r',0)$ for all runs r,r' in R and all processors i.

We say that $M = (R,\pi)$ is an interpreted system where processors do not forget (resp. processors do not learn, time is synchronous, there is a unique initial state) exactly if R is a system with that property. As we mentioned in the introduction, we use the notation $\mathscr{C}$ to represent the class of all interpreted systems, and add the subscripts *nf*, *nl*, *sync*, and *uis* to denote particular subclasses of $\mathscr{C}$.

3. Upper bounds for $CKL_{(m)}$ and $CKB_{(m)}$

In this section we show that the validity problem for $CKL_{(m)}$ and $CKB_{(m)}$ any of the classes of structures that we consider is in Π^1_1. We begin with a brief review of the notions of Π^1_1 and its dual Σ^1_1. Further details can be found in [Rog] or any other standard textbook of recursive function theory.

Formulas of *second-order arithmetic with set variables* consist of formulas of first-order arithmetic, augmented with expressions of the form $x \in X$, where x is a number variable and X is a set variable, together with quantification over set variables and number variables. Second-order arithmetic with set variables is a very powerful language. For example, the following (true) sentence of the language expresses the law of mathematical induction over $\mathbb{N}$:

(1) $\forall X(0 \in X \wedge \forall x((x \in X \Rightarrow x+1 \in X) \Rightarrow \forall x(x \in X)))$

A Π^1_1-sentence (resp. Σ^1_1-sentence) of second-order arithmetic with set variables is one of the form $\forall X_1 \ldots \forall X_n \varphi$ (resp. $\exists X_1 \ldots \exists X_n \varphi$), where φ is a formula of second-order arithmetic with set variables that has no quantification over set variables. A set A of natural numbers is in Π^1_1 (resp. Σ^1_1) exactly if there is a Π^1_1-sentence (resp. Σ^1_1-sentence) $\psi(x)$ with one free number variable x and no free set variables such that $a \in A$ iff $\psi(a)$ is true. Π^1_1-hardness and Π^1_1-completeness are defined in the obvious way (the reduction is via recursive functions). It is well-known that Π^1_1-complete sets are not recursively enumerable (cf. [Rog]). In particular, it follows from the fact that the validity problem for both $CKL_{(m)}$ and $CKB_{(m)}$, $m \geq 2$, is Π^1_1-complete that there can be no complete (recursive) axiomatization for these languages.

Theorem 3.1. *The validity problem for $CKL_{(m)}$ and $CKB_{(m)}$, $m \geq 2$, in $\mathscr{C}_{(nf,sync)}$ is in Π^1_1.*

Proof. It suffices to show that the satisfiability problem is in Σ^1_1. We give the argument for $CKL_{(m)}$, and then comment on the modifications required to deal with $CKB_{(m)}$. The argument is the same for both the synchronous and asynchronous cases. The first step is to show that a formula in $CKL_{(m)}$ is satisfiable iff it is satisfiable in a countable system (i.e., a system with only countably many runs).

Clearly if a formula is satisfiable in a countable system it is satisfiable. For the converse, suppose φ is satisfied in some interpreted system $M = (R,\pi)$. Thus, we have $(M,r,n) \models \varphi$ for

some run r in R. We define a sequence $R_0, R_1, \ldots$ of countable sets of runs in R such that $R_0 = \{r\}$, and for all $j > 0$ and every formula of the form $\sim K_i\psi$ such that $(M, r', n') \models \sim K_i\psi$ for some $r' \in \bigcup_{j' < j} R_{j'}$, there is some $r'' \in R_j$ and some n'' such that $(r', n') \sim_i (r'', n'')$ and $(M, r'', n'') \models \sim\psi$. It is obvious that we can indeed define such a sequence inductively. Let $R' = \bigcup_j R_j$ and let $M' = (R', \pi')$, where π' is the restriction of π to $(R' \times I\!N)$. By construction, M' is a countable system.

Moreover, we can show that for all $r' \in R'$, all $n' \in I\!N$, and all formulas ψ, we have $(M', , r', n') \models \psi$ iff $(M, r', n') \models \psi$. The proof is by induction on the number of occurrences of C in φ, with a subinduction on the structure of φ. If ψ is of the form $K_i\psi'$, then if $(M, r', n') \models K_i\psi'$, we must have $(M, r'', n'') \models \psi'$ for all (r'', n'') such that $(r', n') \sim_i (r'', n'')$. Since R' is a subset of R, we easily get $(M', r', n') \models K_i\psi'$ from the induction hypothesis. And if $(M, r', n') \not\models K_i\psi'$, then $(M, r', n') \models \sim K_i\psi'$. Since $r' \in R'$, we must have $r' \in R_j$ for some j. By construction, there is some $r'' \in R_{j+1}$ and some n'' such that $(r', n') \sim_i (r'', n'')$ and $(M, r'', n'') \models \sim\psi'$. Again, using the induction hypothesis, we have $(M', r'', n'') \models \sim\psi'$. Thus, by definition, $(M', r', n') \not\models K_i\psi'$. If ψ is of the form $E^i\psi'$, note that $(M, r'', n'') \models E\psi'$ iff $(M, r'', n'') \models K_i\psi'$ for $i = 1, \ldots, m$, so this case quickly reduces to the last one. Finally, note that $(M, r', n') \models C\psi'$ iff $(M, r', n') \models E^k\psi'$ for $k = 1, 2, \ldots$. Since $E^i\psi'$ has one less occurrence of C than $C\psi'$, the case of $C\psi'$ also easily follows using the inductive hypothesis. This completes the induction proof. Since, by construction, $r \in R'$, we have $(M', r, n) \models \varphi$. Thus, M' is a countable model of φ, as desired.

We now sketch how to encode the satisfiability of an $CKL_{(m)}$ formula φ in a Σ^1_1-formula. Suppose φ has k subformulas $\varphi_1, \ldots, \varphi_k$, where $\varphi = \varphi_k$. Note that a formula has only finitely many subformulas. Given a countable system $M = (R, \pi)$, we can assume without loss of generality that R is just $I\!N$. Thus, a point (r, n) can be encoded by the number 2^r3^n. We can also assume without loss of generality that π assigns truth values only to the primitive propositions that appear in φ (otherwise we can just ignore the truth values π assigns to the other primitive propositions). Thus, we can easily encode π as a set Y of numbers. We can also assume that the equivalence relation $\sim_i$ is also encoded as a set Z_i of numbers. We then use the sets $X_1, \ldots, X_k$ to encode the sets of points where the formulas $\varphi_1, \ldots, \varphi_k$ are true. We can then write down some obvious consistency conditions that these sets must satisfy. For example, if φ_j is of the form $\varphi_{j_1} \wedge \varphi_{j_2}$, then X_j must be the intersection of X_{j_1} and X_{j_2}. Similarly, if φ_j is of the form $K_i\varphi_{j_1}$, then X_j consists of all the (encodings of) points (r, n) such that for all (encodings of) points (r', n) with $(r', n) \sim_i (r, n)$, we have $(r', n) \in X_{j_1}$. Note that we have to "consult" the set Z_i to check if $(r', n) \sim_i (r, n)$. The fact that φ is true at some point in M is simply the statement that X_k is nonempty. Finally, we have to write down conditions on the Z_i's that will force the system to have the right properties (such as no forgetting or no learning). By existentially quantifying appropriately over the sets $X_1, \ldots, X_k, Y, Z_1, \ldots, Z_m$, we end up with a Σ^1_1 formula that is true iff φ is satisfiable at some point in a countable system iff, by the proof given above, φ is satisfiable.

The proof in the case of $CKB_{(m)}$ is identical, except that in the proof that there always exists a countable model for a satisfiable formula, we also have to add runs to R' ensure that formulas of the form $\exists\bigcirc\psi$ and $\exists\psi U\psi'$ are satisfied in M'. We leave the straightforward details to the reader. ■

We note that essentially the same proof provides Π^1_1 upper bounds for satisfiability of $CKL_{(m)}$ and $CKB_{(m)}$ with respect to all classes of structures discussed earlier.

4. Upper bounds for $KL_{(m)}$ and $KB_{(m)}$

If we do not reason about common knowledge, then, at least in the synchronous case, the logic becomes decidable, although non-elementary. We focus here on $KL_{(m)}$.

To understand the upper bound, consider again the proof of the Π^1_1 lower bounds in [HV1]. In that proof we used C to play the role of $\Box_r$, i.e., to talk about things that happen arbitrarily far to the right. Without common knowledge, the logic cannot talk about things that happen arbitrarily far to the right; roughly speaking, a formula φ can only talk about things that happens within distance $ad(\varphi)$. Thus, to check if a formula φ is satisfiable at a point $(r,0)$, it suffices to consider only a "vertical cylinder" of radius $ad(\varphi)$ around run r. We now formalize this intuition.

We start by defining the *closure* of φ. A formula ψ that is not of the form $\sim\psi'$ is called *positive*. The closure of φ, denoted $cl(\varphi)$, is the least set of formulas such that:

- All subformulas of φ are in $cl(\varphi)$.
- If $\psi \in cl(\varphi)$ and ψ is positive, then $\sim\psi \in cl(\varphi)$.
- If $\psi U \xi \in cl(\varphi)$, then $\circ(\psi U \xi) \in cl(\varphi)$.
- If $\sim\circ\psi cl(\varphi)$ and ψ is positive, then then $\circ\sim\psi \in cl(\varphi)$.
- If $\sim\circ\sim\psi cl(\varphi)$, then then $\circ\psi \in cl(\varphi)$.

It is easy to check (by induction on the structure of φ) that $|cl(\varphi)| \leq 6|\varphi|$.

A subset a of $cl(\varphi)$ is an *atom* of φ if the following holds:

- If $\psi \in cl(\varphi)$ and ψ is positive, then $\psi \in a$ iff $\sim\psi \notin a$.
- If $\psi \wedge \xi \in cl(\varphi)$, then $\psi \wedge \xi \in a$ iff $\psi \in a$ and $\xi \in a$.
- If $\sim\circ\psi \in cl(\varphi)$ and ψ is positive, then $\sim\circ\psi \in a$ iff $\circ\sim\psi \in a$.
- If $\sim\circ\sim\psi \in cl(\varphi)$, then $\sim\circ\sim\psi \in a$ iff $\circ\psi \in a$.
- If $\psi U \xi \in cl(\varphi)$, then $\psi U \xi \in a$ iff either $\xi \in a$ or both $\psi \in a$ and $\circ(\psi U \xi) \in a$.
- If $K_i\psi \in a$, then $\psi \in a$.

Let $at(\varphi)$ be the set of all atoms of φ.

We consider finite trees whose nodes labelled by atoms of φ and whose edges are labelled by the agents mentioned in the formula φ; without loss of generality, we will assume that the set of agents is $\{1,...,m\}$. We are only interested in trees of height at most $ad(\varphi)$. Intuitively, these trees represent the set of points in a horizontal "layer" that a formula of depth at most $ad(\varphi)$ can talk about. Given a tree T and a node t in T, we use $atom(t)$ to denote the atom labelling the node t and $tree(t)$ to denote the subtree of T rooted at t. We often identify a node t in a tree T with the tree $tree(t)$. (However, note that even if t_1 and t_2 are distinct nodes in a tree T, we may have $tree(t_1) = tree(t_2)$.) An edge labelled by j is called a j edge. If there is a j edge from s to t in a tree, then we say that t is a j-child of s and s is the j-parent of t. A tree T is *lean* if for all nodes s in T, if t and t' are distinct j-children of s, $1 \leq j \leq m$, then $tree(t)$ and $tree(t')$ are distinct. A tree T is *compact* if no two adjacent edges on a branch have the same label, i.e. if t_1, t_2, t_3 are nodes of T and t_2 is a

j-child of t_1, then t_3 cannot be a j-child of t_2. A tree T is *full* if whenever a node t is of height k, then all children of t are of height $k-1$. *Standard* trees are lean, compact, and full. A *k-tree* T is a standard tree of height k.

Proposition 4.1. *There are at most* $ex(1+k, 7|\varphi|+k)$ *distinct k-trees.*

Proof. We proceed by induction on k to show that there are at most $ex(1+k, 4|\varphi|+k)$ distinct lean trees of height k. Since the standard trees of height k are a subset of the lean trees, the result follows.

Let N_k be the number of distinct lean trees of height k. Since a lean tree of height 0 is a node labelled by an atom, and there are at most $2^{6|\varphi|}$ atoms, clearly $N_0 \leq 2^{6|\varphi|}$. For $k>0$, observe that a lean tree of height k consists of a root labelled by an atom, together with j edges to lean trees of height $k-1$, for $1 \leq j \leq m$. Since the j edges must lead to distinct trees of height $k-1$, we can completely characterize a lean tree of height k by the atom labelling the root and the subset of lean trees of height $k-1$ that the j edges lead to, for $1 \leq j \leq m$. Thus, there are at most $2^{6|\varphi|}2^{mN_{k-1}}$ lean trees of height k. Since $6|\varphi| \leq N_{k-1}$ and $m \leq |\varphi|-1$, it follows that $N_k \leq 2^{|\varphi|N_{k-1}}$. Thus, $N_1 \leq 2^{|\varphi|2^{6|\varphi|}} \leq 2^{2^{7|\varphi|}}$, and for $k \geq 1$, an easy computation shows that $N^k \leq 2^{|\varphi|ex(k,7|\varphi|+(k-1))} \leq ex(1+k, 7|\varphi|+k)$. ■

Corollary 4.2. *There are at most* $ex(1+ad(\varphi), 8|\varphi|)$ *standard trees of height* $\leq ad(\varphi)$.

Let T be a tree and let t_1, t_2 be nodes in T. We say that t_1 and t_2 are *j-relatives* if either $t_1 = t_2$, t_1 is a j-child of t_2, t_2 is a j-child of t_1, or there is a node t_3 that is the j-parent of both t_1 and t_2. In general, j-relativeness is not an equivalence relation, but it is easy to see that it is an equivalence relation for compact trees.

A tree T is a *knowledge tree* if it satisfies

- If t and t' are nodes of T that are j-relatives then $K_j\psi \in atom(t)$ iff $K_j\psi \in atom(t')$.
- If t is an internal node of T and $\sim K_j\psi \in atom(t)$, then there is a node t' of T that is a j-relative of t and $\sim\psi \in atom(t')$.

Note that in the second clause we do *not* require that $\sim K_j\psi$ formulas at the leaves of T are taken care of. The idea is that knowledge trees of height k will satisfy formulas of depth k. Thus, the leave have only to satisfy formulas of zero depth.

We now define the notion of *potential successors*. Let S and T be trees, and let s be a node of S. We say that T is a potential successor of (S, s) if

- for $\circ\psi \in cl(\varphi)$, we have that $\circ\psi \in atom(s)$ iff $\psi \in atom(T)$, and
- if T' is a j-child of T, then there is some s' in S such that s' is a j-relative of s and T' is a potential successor of (S, s').

Note that potential succession is a well-defined notion, since the height of T' is smaller than the height of T.

The following lemma is proven by easy induction.

Lemma 4.3. *Let S and T be standard trees, let s be a node of S, and let t_1 be a node of T. If T is a potential successor of (S, s), then there is a node s_1 of S such that t_1 is a potential successor of (S, s_1). Furthermore, if t_2 is a j-relative of t_1, then there is a node s_2 of S such that t_2 is a potential successor of (S, s_2) and s_2 is a j-relative of s_1.* ■

In general a system in $\mathscr{C}_{(nf,sync)}$ is a "grid-like" structure, where a view knowlegde edges as horizontal edges and temporal edges as vertical edges. Indeed, in [HV1] this property of the structures is used to obtain the undecidability result for $CKL_{(m)}$. The crux of the decidability proof here is that when considering satisfiability of $KL_{(m)}$ formulas we can describe structures by trees. These trees are called *Hintikka trees*. We now proceed with a formal definition.

A Hintikka tree for φ is an infinite tree τ rooted at x_0 such that an element x of τ (called *vertex*) is labelled by a standard knowledge tree, denoted $lab(x)$, and the following conditions are satisfied.

(H1) Let x be a vertex of τ, let $S = lab(x)$, and let X be the set of successors of x. Then there is a bijection σ_x from the nodes of S to X such that the following holds: Let s be a node of S, where s is a k-tree, let $\sigma_x(s) = y$, and let $lab(y) = t$, where t is a k'-tree. Then t is a potential successor of (S, s) and $k' \geq k$.

(H2) $lab(x_0)$ is a k-tree for for some $k \geq ad(\varphi)$ and $\varphi \in atom(lab(x_0))$.

(H3) Let x be a vertex of τ, and $lab(x) = S$. Then $\psi U \xi \in atom(S)$ iff there is a sequence of vertices $x_1, \ldots, x_l$ such that $x = x_1$, $x_i = \sigma_{x_i}(lab(x_{i-1}))$ and $lab(x_i) = S_i$ for $1 < i \leq l$, $\xi \in atom(S_l)$, and $\psi \in atom(S_i)$ for $1 \leq i < l$.

Intuitively, every vertex in τ consists of a description of a horizontal slice of the structure center at some point. The slice has to be big enough to satisfy formulas at that point. In particular, to satisfy φ we need in (H2) a knowledge tree of height at least $ad(\varphi)$. In (H1), s takes care of formulas of depth k, which explains why t has to be of at least height k.

The connection between Hintikaa trees and structures is established by the following lemma.

Lemma 4.4. *Let φ be a formula of $KL_{(m)}$. The the following are equivalent:*

1. *φ is satisfiable in $\mathscr{C}_{(nf,sync)}$.*
2. *φ has a Hintikka tree.*
3. *φ has a Hintikka tree whose vertices are labeled by k-trees for $k \leq ad(\varphi)$.*

Proof. For the sake of the proof we take the alternative view of interpreted systems, where instead of viewing runs as functions from $I\!N$ to a set of local states, we view runs as abstract objects and we specify the indistinguishability relations explicitly. That is, an interpreted system is system is a tuple $(R, \sim_1, \ldots, \sim_m, \pi)$, where $\sim_i$ is an equivalence relation on $R \times I\!N$ and π is a mapping from $R \times I\!N$ to truth assignments. We have already observed earlier that this notion is equivalent to our standard notion.

We first prove that if φ is satisfiable, then it has a Hintikka tree whose nodes are labeled by k-trees for $k \leq ad(\varphi)$. Let $M = (R, \sim_1, \ldots, \sim_m, \pi)$ be an interpreted system. With every point (r, i) we associate an atom, denoted $atom(r, i)$, which is the set of formulas $\{\psi : \psi \in cl(\varphi) \text{ and } (M, r, i) \models \psi\}$. It is easy to see that $atom(r, i)$ is indeed an atom. We aslo associate nonstandard trees with the points of M. More precisely, with every point (r, i) and $k \geq 0$ we associate a nonstandard tree, denoted k-$nst(r, i)$. These trees will be lean and full, but not compact. The idea is that k-$nst(r, i)$ describes a "circle" of radius k around (r, i). The construction is by induction on k. For the basis of the induction, let 0-$nst(r, i)$ consist of a

single node labeled by $atom(r,i)$. Suppose now that $k\text{-}nst(r,i)$ has been defined for all r and i. The root of $(k+1)\text{-}nst(r,i)$ is labeled by $atom(r,i)$. The j-children of $(k+1)\text{-}nst(r,i)$ are all the *distinct* trees of the form $k\text{-}nst(r',i)$, where $(r,i)\sim_j(r',i)$ (the emphasis on distinctness guarantees leanness). This completes the construction. Note that if t is a node of height l in $(k+1)\text{-}nst(r,i)$, then there is a run r' such that $t = l\text{-}nst(r',i)$. Two distinct nodes can, however, be associated with the same run r'.

We now prune these trees to get standard trees. More precisely, if t_1, t_2, and t_3 are nodes of $k\text{-}nst(r,i)$, t_1 is the j-parent of t_2, and t_2 is the j-parent of t_3, then we delete t_3 (and the nodes below it). We continue until we end up with $k-tree(r,i)$.

We prove that $k\text{-}tree(r,i)$ is a knowledge tree. Let t_1 and t_2 be nodes of $k\text{-}tree(r,i)$, where t_1 is of height k_1 and t_2 is of height k_2. Then there are runs r_1 and r_2 such that t_1 is obtained from $k_1\text{-}tree(r_1,i)$ by pruning and t_2 is obtained from $k_2\text{-}tree(r_2,i)$ by pruning. If t_1 and t_2 are j-relatives, then $(r_1,i)\sim_j(r_2,i)$. Thus, $K_j\psi \in atom(t_1)$ iff $(M,r_1,i) \models K_j\psi$ iff $(M,r_2,i) \models K_j\psi$ iff $K_j\psi \in atom(t_2)$. If $k_1 \geq 1$ and $\sim K_j\psi \in atom(t_1)$, then $(M,r_1,i) \models \sim K_j\psi$. Thus, for some r_3 such that $(r,i)\sim_j(r_3,i)$, we have that $(M,r_3,i) \models \sim\psi$. If t_1 is a j-child of some node t, then $k_1\text{-}tree(r_3,i)$ with its j-children deleted is also a j-child of t. Otherwise $(k_1-1)\text{-}tree(r_3,i)$ with its j-children deleted is a j-child of t_1. In either case, t_1 has a j-relative t_3 such that $\sim\psi \in atom(t_3)$.

We now show that $k\text{-}tree(r,i)$ is a potential successor of $(l\text{-}tree(r,i-1), l\text{-}tree(r,i-1))$, when $k \leq l$. The proof is by induction on k. For $k=0$, the claim follows from the definition of $atom(r,i)$. Assume that the claim has been proven for $k-1$. The first condition in the definition of potential successor again follows from the definition of $atom(r,i)$. So it remains to prove that the second condition holds. Suppose that T' is a j-child of $k\text{-}tree(r,i)$. Then there is a run r' such that $(r',i)\sim_j(r,i)$ and T' is $(k-1)\text{-}tree(r',i)$ with its j-children deleted. By the induction hypothesis, $(k-1)\text{-}tree(r',i)$ is a potential successor of $((l-1)\text{-}tree(r',i-1),(l-1)\text{-}tree(r',i-1))$. But $(l-1)\text{-}tree(r',i-1)$ with its j-children deleted is a j-child of $l\text{-}tree(r,i-1)$. The claim follows.

We now define a Hintikka tree τ by induction on the depth. With every vertex x of τ we associate a point $point(x)$ of M and we let $lab(x) = k\text{-}tree(point(x))$ for some k. Let $r_0 \in R$ be such that $(M,r_0,0) \models \varphi$. We let $point(x_0) = (r_0,0)$ and $lab(x_0) = ad(\varphi)\text{-}tree(r,0)$. Clearly, $\varphi \in atom(lab(x_0))$. Thus, (H2) is satisfied.

Suppose that all the vertices of τ up to level i and their labels have been defined. Let x be a vertex, with $point(x) = (r,i)$ and $lab(x) = k\text{-}tree(r,i)$. Consider a node t of $lab(x)$. If t is the root, then we add a successor x' of x with $point(x') = (r,i+1)$, $lab(x') = k\text{-}tree(r,i+1)$, and $\sigma_x(t) = x'$. If t is a j-child, then there is some $r' \in R$ such that t is $(k-l)\text{-}tree(r',i)$ with its j-children deleted, for some $l<k$. We add a successor x'' of x with $point(x'') = (r',i+1)$, $lab(x'') = (k-l)\text{-}tree(r',i+1)$, and $\sigma_x(t) = x''$.

We have to show that $(k-l)\text{-}tree(r',i+1)$ is a potential successor of $(k\text{-}tree(r,i),t)$. We know that $(k-l)\text{-}tree(r',i+1)$ is a potential succssor of $((k-l)\text{-}tree(r',i),(k-l)\text{-}tree(r',i))$. Thus, if t was a root, then we are done. It remains to deal with the case that t is a j-child. Let s be a child of $(k-l)\text{-}tree(r',i+1)$. We know that $(k-l)\text{-}tree(r',i)$ has a child s' such that s is a potential successor of (s',s'). Thus, if s is not a j-child, then s' is a child of t and s is a potential successor of $(k\text{-}tree(r,i),s')$. If, on the other hand, s is a j-child, then s' is a

j-child of t's j-parent, so again s is a potential successor of $(k\text{-}tree(r,i),s')$. This shows that (H1) is satisfied.

Finally, it remains to verify that (H3) is satisfied. Let x be a vertex of τ. Suppose that $\psi U\xi \in atom(lab(x))$. Then $(M,r,i) \models \psi U\xi$. Therefore, there is some $i' \geq i$ such that $(M,r,i') \models \xi$, and for all i'' with $i \leq i'' < i'$, we have $(M,r,i'') \models \psi$. Define a sequence $x_1, x_2, \ldots$ of vertices of τ as as follows: $x_1 = x$, and $x_{j+1} = \sigma_{x_j}(lab(x_j))$, $j \geq 1$. We know that $point(x_j) = (r, i+j-1)$. Thus, there is an l such that $\xi \in atom(lab(x_l))$, and $\psi \in atom(lab(x_i))$ for $1 \leq i < l$. Conversely, suppose that there is a sequence of vertices $x_1, \ldots, x_l$ such that $x = x_1$, $x_j = \sigma_{x_j}(lab(x_{j-1}))$ for $1 < j \leq l$, $\xi \in atom(lab(x_l))$, and $\psi \in atom(lab(x_i))$ for $1 \leq j < l$. We then have that $point(x_j) = (r, i+j-1)$, so $(M,r,i+l-1) \models \xi$ and $(M,r,i+j-1) \models \psi$ for $1 \leq j < l$. It follows that $(M,r,i) \models \psi U\xi$, so $\psi U\xi \in atom(lab(x))$.

We have shown that if φ is satisfiable then it has a Hintikka tree. We now show that if φ has a Hintikka tree τ, then it is satisfiable. With every vertex x of τ we associate a run r_x. The set R of runs is the set of runs r_x for all the vertices x in τ. To define the equivalence relations $\sim_j$, $1 \leq j \leq m$, we need some additional machinery.

With every vertex x and $i \geq 0$, we associate a vertex x_i and a node $t_{x,i}$ of $lab(x_i)$ such that $t_{x,i}$ is a potential successor of $(lab(x_{i-1}), t_{x,i-1})$. The construction is by backward and forward induction.

Suppose that x is at level n of τ. We first deal with the case $0 \leq i \leq n-1$. Let $x_0, \ldots, x_{n-1}$ be the path leading to x. That is, $x_{n-1} = x$, and x_j is the successor of x_{j-1} for $1 \leq j \leq n-1$. For $i = n-1$, let $t_{x,n-1} = lab(x_{n-1})$. Assume by induction that $t_{x,i}$ is defined. By (H1), there is a node s of $lab(x_{i-1})$ such that $lab(x_i)$ is a potential successor of $(lab(x_{i-1}), s)$. It follows by Lemma 4.3 that that there is a node $t_{x,i-1}$ of $lab(x_{i-1})$ such that $t_{x,i}$ is a potential successor of $(lab(x_{i-1}), t_{x,i-1})$.

We now deal with the case $i \geq n-1$. Recall that $t_{x,n-1} = lab(x_{n-1})$. Suppose that x_i and $t_{x,i}$ have been defined. By (H1), there is a successor x_{i+1} of x_i such that $lab(x_{i+1})$ is a potential successor of $(lab(x_i), lab(x_i))$. Thus, let $t_{x,i+1} = lab(x_{i+1})$.

We can now define the equivalence relations $\sim_j$ for $1 \leq j \leq m$. We let $(r_x, i) \sim_j (r_y, i)$ if $x_i = y_i$, and $t_{x,i}$ and $t_{y,i}$ are j-relatives. It is easy to see that $\sim_j$ is an equivalence relation. Since we are trying to construct a synchronous system, we have to show that $(r_x, i) \sim_j (r_y, i)$ entails $(r_x, i-1) \sim_j (r_y, i-1)$.

Suppose that $(r_x, i) \sim_j (r_y, i)$. Then $x_i = y_i$ and $t_{x,i}$ and $t_{y,i}$, which are nodes of $lab(x_i)$, are j-relatives. But x_{i-1} is the parent of x_i and y_{i-1} is the parent of y_i. It follows that $x_{i-1} = y_{i-1}$. Furthermore, by Lemma 4.3, $t_{x,i-1}$ and $t_{y,i-1}$ are j-relatives.

Finally, we associate a truth assignment $\pi(r_x, i)$ with every point (r_x, i) in the following way. Then $\pi(r_x, i)(p) = true$ iff $p \in atom(t_{x,i})$.

Let M be $(R, \sim_1, \ldots, \sim_m, \pi)$. We claim that $(M, r_{x_0}, 0) \models \varphi$. To prove this, we would like to prove that $(M, r_x, i) \models \psi$ iff $\psi \in atom(t_{x,i})$ for $\psi \in cl(\varphi)$. Unfortunately, this is not the case, since "deep" knowledge formulas need "deep" knowledge trees to satisfy them. Thus, we attempt to prove a weaker statement: if $t_{x,i}$ is a k-tree and $ad(\psi) \leq k$, then $(M, r_x, i) \models \psi$ iff $\psi \in atom(t_{x,i})$ (the earlier claim then would follow by condition (H2)). Unfortunately, even

this claim does not hold, and we need to weaken it further. Let $\psi \in cl(\varphi)$ such that $ad(\psi) \leq k+1$ and, furthermore, if $K_j\xi$ is a subformua of ψ such that $ad(K_j\psi) = k+1$, then $t_{x,i}$ is a j-child. We prove that $(M, r_x, i) \models \psi$ iff $\psi \in atom(t_{x,i})$. The earlier claim then follows by condition (H2).

The proof is by induction on the structure of the formulas. For primitive propositions the claim follows from the definition of π. For the Boolean connectives the claim follows from the definition of atoms. Consider now a formula $\circ\psi$. We know that $t_{x,i+1}$ is a potential successor of $(lab(x_i), t_{x,i})$. Thus, $(M, r_x, i) \models \circ\psi$ iff $(M, r_x, i+1) \models \psi$ iff (by the induction hypothesis) $\psi \in atom(t_{x,i+1})$ iff (by the definition of potential successor) $\circ\psi \in atom(t_{x,i})$.

Consider now a formula $\psi U \xi$. Now $(M, r_x, i) \models \psi U \xi$ iff there is some $i' \geq i$ such that $(M, r_x, i') \models \xi$, and for all i'' with $i \leq i'' < i'$, we have $(M, r_x, i'') \models \psi$. Thus, $(M, r_x, i) \models \psi U \xi$ iff (by the induction hypothesis) there is some $i' \geq i$ such that $\xi \in atom(t_{x,i'})$, and for all i'' with $i \leq i'' < i'$, we have $\psi \in atom(t_{x,i''})$. Assume $(M, r_x, i) \models \psi U \xi$. Then it follows by the definition of potential successors and atoms that $\psi U \xi \in atom(t_{x,i})$. Assume, on the other hand that $\psi U \xi \in atom(t_{x,i})$. Suppose that x is at level n of τ. There are two cases to consider. If $i \geq n$, then $t_{x,i} = lab(x_i)$. Thus, by (H3), $(M, r_x, i) \models \psi U \xi$. If, on the other hand, $i < n$, then one can prove by induction that either there is some i, $i \leq i' \leq n$, such that $\xi \in atom(t_{x,i'})$, and for all i'' with $i \leq i'' < i'$, we have $\psi \in atom(t_{x,i''})$, or $\psi \in atom(t_{x,i})$ for all i'' with $i \leq i'' \leq n$ and $\psi U \xi \in atom(lab(x))$. It follows, either immediately in the first case or by applying (H3) in the second case, that $(M, r_x, i) \models \psi U \xi$.

It remains to deal with knowledge formulas. Let $K_j\psi \in cl(\varphi)$. Suppose that $K_j\psi \in atom(t_{x,i})$. Let $(r_x, i) \sim_j (r_y, i)$. It follows from the definition of $\sim_j$ and from the definition of knowledge trees that $K_j\psi \in atom(t_{y,i})$, so we have $\psi \in atom(t_{y,i})$. There are now two cases to consider. If $t_{x,i}$ is a j-child, then it is possible that $ad(K_j\psi) = k+1$. But in that case, since $(r_x, i) \sim_j (r_y, i)$, we have that $t_{y,i}$ is of height $k+1$ or k. If, on the other hand $t_{x,i}$ is not a j-child, then $t_{y,i}$ is of height k or $k-1$. But in that case $ad(K_j\psi) \leq k$. Thus, in either case it follows, by the induction hypothesis, that $(M, r_y, i) \models \psi$. Since this is true for all y such that $(r_x, i) \sim_j (r_y, i)$, it follows that $(M, r_x, i) \models K_j\psi$.

Suppose now that $\sim K_j\psi \in atom(t_{x,i})$. Then $t_{x,i}$ has a j-relative t', where t' is a k'-tree, such that $\sim\psi \in atom(t')$. Let $y = \sigma_{x_i}(t')$. It is easy to see that $y_i = x_i$ and $t_{y,i} = t'$. Thus, $(r_x, i) \sim_j (r_y, i)$. As before, either $k' \geq k$ or $k' \geq k-1$, $ad(K_j\psi) \leq k$, and t' is a j-child. In either case, the induction hypothesis applies and we have $(M, r_x, i) \models \sim\psi$, so $(M, r_x, i) \models \sim K_j\psi$. This completes the proof. ■

We can now use Lemma 4.4 to derive an upper bound for satisfiability of $KL_{(m)}$.

Theorem 4.5. *There exist a positive constant c such that the problem of deciding the satisfiability of a formula φ of $KL_{(m)}$ in $\mathscr{C}_{(nf,synch)}$ can be solved in time $O(ex(1 + ad(\varphi), c|\varphi|))$.*

Proof. By Proposition 4.1, if the vertices in a Hintikka tree for φ are labeled by trees of height bounded by $ad(|\varphi|)$, then the fan-out of the tree is bounded by $ex(ad(\varphi), d|\varphi|)$ for some positive constant c. To obtain a decision procedure we use the automata-theoretic technique described in [VW]. We construct a Büchi automaton that recognize Hintikka trees for φ, where the labels are trees of height bounded by $ad(|\varphi|)$. The automaton consists of two parts: the *local* automaton and the *eventuality* automaton. The local automaton, whose

states are k-trees, checks that conditions (H1) and (H2) are satisfied. Intuitively, the local automaton is a recursive algorithm that starts at the root of the tree, checks that the root satisfies (H1), checks that the root has successors so that (H2) is satisfied, and then applies itself recrusively to the children of the root (without, of course, checking again for (H1)). The eventuality automaton, whose states are sets of formulas in $cl(\varphi)$, checks that conditions (H3) is satisfied. Intuitively, the eventuality automaton is also a recursively descending algorithm that carries with it a set of formulas of the form $\psi U \xi$ and check that they are eventually satisfied. The reader is referred to [VW, Section 3] for more delatils. There is a positive constant c such that the size of this automaton is bounded by $ex(1 + ad(\varphi), c \mid \varphi \mid)$. Now φ is satisfiable iff the automaton accepts some infinite tree. This can be checked in time polynomial in the size of the automaton [Ra] whence the bound in the theorem. ■

5. Complete Axiomatizations

In the literature can be found complete axiomatizations for reasoning about knowledge ([Hi,HM2]), common knowledge ([Mi,Le,HM2]), linear time ([GPSS]), and branching time ([EH2]). These are reviewed below.

Axioms for propositional reasoning:

PA1. All instances of propositional tautologies

PR1. $\dfrac{p,\ p \Rightarrow q}{q}$ (modus ponens)

Axioms for knowledge

KA1. $K_i\varphi \wedge K_i(\varphi \Rightarrow \psi) \Rightarrow K_i\psi$
KA2. $K_i\varphi \Rightarrow \varphi$
KA3. $K_i\varphi \Rightarrow K_iK_i\varphi$
KA4. $\sim K_i\varphi \Rightarrow K_i \sim K_i\varphi$

KR1. $\dfrac{\varphi}{K_i\varphi}$

Axioms for common knowledge (with m knowers)

CA1. $E\varphi \equiv K_1\varphi \wedge \ldots \wedge K_m\varphi$
CA2. $C\varphi \Rightarrow \varphi \wedge EC\varphi$
CA3. $C\varphi \wedge C(\varphi \Rightarrow \psi) \Rightarrow C\psi$
CA4. $C(\varphi \Rightarrow E\varphi) \Rightarrow (\varphi \Rightarrow C\varphi)$

CR1. $\dfrac{\varphi}{C\varphi}$

Axioms for linear time

LA1. $\circ\sim\varphi \equiv \sim\circ\varphi$
LA2. $\circ\varphi \wedge \circ(\varphi \Rightarrow \psi) \Rightarrow \circ\psi$
LA3. $\varphi U \psi \equiv \psi \vee (\varphi \wedge \circ(\varphi U \psi))$

LR1. $\dfrac{\varphi}{\bigcirc\varphi}$

LR2. $\dfrac{\varphi' \Rightarrow \sim\psi \wedge \bigcirc\varphi'}{\varphi' \Rightarrow \sim(\varphi U \psi)}$

Axioms for branching time

BA1. $\forall\bigcirc\varphi \equiv \sim\exists\bigcirc\sim\varphi$
BA2. $\forall\bigcirc\varphi \wedge \forall\bigcirc(\varphi \Rightarrow \psi) \Rightarrow \forall\bigcirc\psi$
BA3. $\forall\bigcirc\varphi \Rightarrow \exists\bigcirc\varphi$
BA4. $\forall(\varphi U \psi) \equiv \psi \vee (\varphi \wedge \forall\bigcirc\forall(\varphi U \psi))$
BA5. $\exists(\varphi U \psi) \equiv \psi \vee (\varphi \wedge \exists\bigcirc\exists(\varphi U \psi))$

BR1. $\dfrac{\varphi}{\forall\bigcirc\varphi}$

BR2. $\dfrac{\varphi' \Rightarrow (\sim\psi \wedge \exists\bigcirc\varphi')}{\varphi' \Rightarrow \sim\forall(\varphi U \psi)}$

BR3. $\dfrac{\varphi' \Rightarrow (\sim\psi \wedge \forall\bigcirc(\varphi' \vee \sim\exists(\varphi U \psi)))}{\varphi' \Rightarrow \sim\exists(\varphi U \psi)}$

It is easy to check that all these axioms are sound for all the classes of structures we study here. For system with no forgetting, it follows from Π^1_1 lower bound in [HV1] that no axiomatization exists once we allow common knowledge into the language. Lehman does give some axioms that are sound for the synchronous case though [Le1]. We can show that by adding one of his axioms, we get a complete axiomatization for $KL_{(m)}$ in the synchronous case. Adding a similar axiom leads to a complete axiomatization for $KB_{(m)}$ in the synchronous case. Consider the following axioms, which intuitively describes the fact that the set of runs is always decreasing over time, so there is "no forgetting".

Axioms for the interaction between knowledge and time[8]

LKA1. $K_i\bigcirc\varphi \Rightarrow \bigcirc K_i\varphi$
BKA1. $K_i\forall\bigcirc\varphi \Rightarrow \forall\bigcirc K_i\varphi$

It is easy to check that LKA1 (resp. BKA1) is sound for $KL_{(m)}$ (resp. $KB_{(m)}$) in $\mathscr{C}_{(nf,sync)}$, and, as we now show, it is the only extra axiom required for completeness in this case. (Note, as pointed out in [LR], it is *not* sound in the asynchronous case.)

We focus here on $KL_{(m)}$. We want to prove that every formula that is *consistent* (with the axioms for propositional reasoning, knowledge, linear time, and interaction between knowledge and time) is satisfiable. The main idea of the completeness proof is to use the fact that the axioms for knowledge (together with the axioms for propositional reasoning) are complete for knowledge, and the axioms for linear time (together with the axioms for propositional reasoning) are complete for linear time. To prove that the axioms are complete we have to show that if a formula is consistent, then it is satisfiable. We shall do that by showing that

---

8 It is interesting to note that the contrapositives of LKA1 and BKA1, i.e., $\bigcirc K_i\varphi \Rightarrow K_i\bigcirc\varphi$ and $\forall\bigcirc K_i\varphi \Rightarrow K_i\forall\bigcirc\varphi$, respectively, captures the interaction between knowledge and time in synchronous systems with no learning.

if a formula is consistent, then it has a Hintikka tree. We construct the tree by alternately considering formulas of $KL_{(m)}$ as knowledge formulas and as temporal formulas.

We now formalize the idea that a formula of $KL_{(m)}$ can be treated both as a knowledge formula and as temporal formulas. time. We first give semantics to *pure knowledge formulas*, i.e., formulas that are constructed from the primitive propositions by Boolean connectives and knowledge modalities. A *knowledge structure* $M_K = (W, \sim_1, \ldots, \sim_m, \pi)$, where W is a set of states, each $\sim_i$ is an equivalence relation on W, and $\pi(w)$ is a truth assigment on primitive propositions for each $w \in W$. We write $(M_K, w) \models \varphi$ if the pure knowledge formula φ is satisfied by the state w of M_K. We define $\models$ inductively:

- $(M_K, w) \models p$ for a primitive proposition p if $\pi(w)(p) = \text{true}$.
- $(M_K, w) \models \varphi \wedge \psi$ if $(M_K, w) \models \varphi$ and $(M_K, w) \models \psi$
- $(M_K, w) \models \sim\varphi$ if $(M_K, w) \not\models \varphi$
- $(M_K, w) \models K_i\varphi$ if $(M_K, w') \models \varphi$ for all w' such that $w \sim_i w'$.

We write $M_K \models \varphi$ if $(M_K, w) \models \varphi$ for all $w \in W$.

We now give semantics to *pure temporal formulas*, i.e., formulas that are constructed from the primitive propositions by Boolean and temporal connectives. A *linear temporal structure* M_L is simply a sequence of truth assignments. That is, we view the natural numbers as states, and $M_L(i)$ is a truth assignment on primitive propositions. We write $(M_L, i) \models \varphi$ if the pure linear temporal formula φ is satisfied by the state i of M_L. We define $\models$ inductively:

- $(M_L, i) \models p$ for a primitive proposition p if $M_L(i)(p) = \text{true}$.
- $(M_L, i) \models \varphi \wedge \psi$ if $(M_L, i) \models \varphi$ and $(M_L, i) \models \psi$
- $(M_L, i) \models \sim\varphi$ if $(M_L, i) \not\models \varphi$
- $(M_L, i) \models \varphi U \psi$ if there is some $i' \geq i$ such that $(M_L, i') \models \psi$ and for all i'', $i \leq i'' < i$, we have that $(M_L, i'') \models \varphi$.

We write $M_L \models \varphi$ if $(M_L, i) \models \varphi$ for all $i \geq 0$.

Consider now formulas of $LK_{(m)}$. A *general truth assignment* is a function from formulas to truth values. A *temporal formula* is either a primitive proposition, a formula of the form $\bigcirc\psi$, or a formula of the form $\psi U \xi$. A *temporal truth assignment* is a truth assigment on temporal formulas. A temporal truth assignment *for* φ is a truth assigment on the temporal subformulas of φ. A *pseudo* knowledge structure for φ $M_K = (W, \sim_1, \ldots, \sim_m, \pi')$ for a formula φ of $LK_{(m)}$ is like a knowledge structure, where π' assigns to every state a temporal truth assignment for φ.

We can view φ as a pure knowledge formula by treating each temporal subformula of φ as a primitive proposition. Thus, we can talk about satisfaction of arbitrary $KL_{(m)}$ formulas in M_K. We say that M_K is a *pseudo knowledge model* for φ if for some $w \in W$ we have that $(M_K, w) \models \varphi$, and for all $w \in W$ the set $\{\psi : (M_L, w) \models \psi\}$ is consistent.

Lemma 5.1. *Let φ be a formula of $LK_{(m)}$. If φ is consistent then φ has a pseudo knowledge model.*

Proof. The proof is a modification of the completeness proof for pure knowledge formulas. The following is proven is [HM2]: Let φ be a pure knowledge formula. If φ is consistent with the axioms and inference rules for propositional reasoning and knowledge, then φ is

satisfiable.

The proof has the following structure: Assume that that φ is consistent with the axioms and inference rules for propositional reasoning and knowledge. The proof then proceeds to construct a structure M, where every state in M is a consistent set of formulas, and every consistent set of formulas is a state in M. Furthermore if a formula is a member of a state then it is satisfied by that state. Since φ is consistent, there is a state that contains φ, and consequently satisfies φ.

Here we assume that φ is consistent with the axioms and inference rules of propositional reasoning, knowledge, linear time, and the interaction between knowledge and time. To adapt the proofs to our need, we use in the constructions only sets of formulas that are consistent in that stronger sense. With every set X of formulas we associate a temporal truth assignment σ_X such that $\sigma_X(\psi) = \text{true}$ iff $\psi \in X$. The argument in [HM2] shows that we get a pseudo knowledge model for φ. ■

So far we have treated formulas of $LK_{(m)}$ as pure knowledge formulas. We now describe the dual approach, whereby these formulas are treated as pure temporal formulas.

A *knowledge formula* is either a primitive proposition or a formula of the form $K_i\psi$. A knowledge truth assignment is a truth assignment on knowledge formulas. A knowledge truth assignment *for* φ is a truth assigment on the knowledge subformulas of φ. A *pseudo* (linear) temporal structure M_L for φ is simply a sequence of knowledge truth assignments on φ

We can view φ as a pure temporal formula by treating each knowledge subformula of φ as a primitive proposition. Thus, we can talk about satisfaction of arbitrary $KL_{(m)}$ formulas in M_L. We say that M_K is a *pseudo temporal model* for φ if for some $i \geq 0$ we have $(M_L, 0) \models \varphi$ and for all $j \geq 0$ the set $\{\psi \in cl(\varphi) : (M_L, i) \models \psi\}$ is consistent.

Lemma 5.2. *Let φ be a formula of $LK_{(m)}$. If φ is consistent then φ has a pseudo temporal model.*

Proof. The proof is a modification of the completeness proof for pure linear temporal formulas. The following is proven is [GPSS]: Let φ be a pure temporal formula. If φ is consistent with the axioms and inference rules for propositional reasoning and linear time, then φ is satisfiable.

The proof has the following structure: Assume that that φ is consistent with the axioms and inference rules for propositional reasoning and linear time. The proof then proceeds to construct a structure M, where every state in M is a consistent set of formulas in $cl(\varphi)$. Furthermore if a formula is a member of a state then it is satisfied by that state. Since φ is consistent, there is a state that contains φ, and consequently satisfies φ.

Here we assume that φ is consistent with the axioms and inference rules of propositional reasoning, knowledge, linear time, and the interaction between knowledge and time. To adapt the proofs to our need, we use in the constructions only sets of formulas that are consistent in that stronger sense. With every set X of formulas we associate a knowledge truth assignment σ_X such that $\sigma_X(\psi) = \text{true}$ iff $\psi \in X$. The argument in [GPSS] shows that we get a knowledge pseudo model for φ. ■

The basic idea of the completeness proof is to show that if φ is a consistent formula, then it has a Hintikka tree. To construct the tree we use Lemma 5.1 and Lemma 5.2. We first

consider φ as a knowledge formula, and using Lemma 5.1 we construct a knowledge tree. Now we consider the atoms labeling the nodes of the tree as temporal formulas, and using Lemma 5.2 we construct pseudo temporal models, which we "hook" to these nodes. The construction continues by alternately considering formulas as knowledge formulas and temporal formulas.

It turns out that its is fairly easy to ensure that (H2) and (H3) are satisfied, but it is quite hard to ensure that (H1) is satisfied. This is because we have to ensure that that there is a relation of potential succession between the knowledge trees labeling successive vertices. This relation depends on *all* atoms labeling nodes in these knowledge trees. What we need is to reduce this to some condition that depends only on the roots of the knowledge trees.

To reduce potential succession to a relation between roots of knowledge trees, we need to define the notion of the *extended closure* of φ, denoted $ecl(\varphi)$. We define a sequence of "closures", $cl^0(\varphi), cl^1(\varphi), \ldots$, such that $cl^0(\varphi) = cl(\varphi)$, and $ecl(\varphi) = cl^{ad(\varphi)}(\varphi)$. The definition is by induction:

1. $cl^i(\varphi) \subseteq cl^{i+1}(\varphi)$.
2. If $\{\bigcirc\psi_1, \ldots, \bigcirc\psi_k\} \subseteq cl^i(\varphi)$, then $\sim\bigcirc K_j(\psi_1 \vee \ldots \vee \psi_k\} \in cl^{i+1}(\varphi)$.
3. $cl^{i+1}(\varphi)$ is closed under subformulas.

The intuition behind the definition of the extended closure will be explained shortly. Note that the size of $ecl(\varphi)$ is nonelementary in the length of φ. Note also that $cl^i(\varphi)$ may contain formulas of alternation depth $ad(\varphi) + i$. Thus, $ecl(\varphi)$ may contain formulas of alternation depth $2ad(\varphi)$.

We define an *extended atom* of φ to be a subset of $ecl(\varphi)$ that satisfies all the clauses in the definitions of atoms, where $ecl(\varphi)$ is substituted for $cl(\varphi)$. We also define *extended trees* to be trees labeled by extended atoms. We denote the label of an extended tree T by *eatom*(T). We denote by *atom*(T) the atom *eatom*(T) $\cap\, cl(\varphi)$. To every extended tree T, there corresponds a tree T, obtained by considering the atoms of nodes rather than the extended atoms (that is, we use bold face letters to denote extended trees and italics letters to denote the corresponding trees).

As said earlier we want to reduce potential succession to a condition that depends only on the roots of the involved knowledge trees. To accomplish that we need to make sure that the roots of the involved knowledge trees contain sufficient information about the rest of the tree. This will be achieved by *extended knowledge trees*, where an extended knowledge tree T is an extended tree that satisfies all the considions of knowledge trees and also the following additional condition:

- Let t be a node of T, and let $\{\psi_1, \ldots, \psi_k\} \subseteq ecl(\varphi)$ be a set of formulas such that (a) $\bigcirc K_j(\psi_1 \vee \ldots \vee \psi_k) \in ecl(\varphi)$, and (b) if t' is a j-relative of t, then $\bigcirc\psi_i \in atom(t)$ for some $1 \leq i \leq k$. Then $\bigcirc K_j(\psi_1 \vee \ldots \vee \psi_k) \in eatom(t)$.

Thus, in an extended knowledge tree every nodes keeps information about its relatives. In particular, the root of the tree keeps information about the whole tree. The defintion of extended closure was designed in such a way that the root has enough information to enable

us to reduce potential succession to a relation between roots. Note that if T is a standard extended knowledge tree, then T is a standard knowledge tree.

Let S be an extended standard knowledge tree, and let s be a node of S. Let T be an extended standard knowledge tree of height $l \leq ad(\varphi)$. We say that T is an *apparent successor* of $(\mathbf{S}, s)$, if for all $\bigcirc\psi \in ecl(\varphi)$ such that $ad(\psi) \leq ad(\varphi) + l$, if $\bigcirc\psi \in eatom(s)$ then $\psi \in eatom(\mathrm{T})$.

Note that apparent succession is a relation between roots of extended knowledge trees. The next lemma gives us the reduction of potential succession to apparent succession.

Lemma 5.3. *Let* S *be an extended standard knowledge tree, and let* s *be a node of* S. *Let* T *be an extended standard knowledge tree of height* $l \leq ad(\varphi)$. *Let* S *and* T *be the corresponding standard knowledge trees. If* T *is an apparent successor of* $(\mathbf{S}, s)$, *then* T *is a potential successor of* (S, s).

Proof. The proof is by induction on the height of T.

Assume first that $l = 0$. Let $\bigcirc\psi \in cl(\varphi)$. Since $ad(\psi) \leq ad(\varphi)$, if $\bigcirc\psi \in atom(s)$, then by the definition of apparent successors, we have $\psi \in atom(T)$. If $\sim\bigcirc\psi \in atom(s)$, then $\sim\bigcirc\psi \in atom(s)$, and consequently, $\bigcirc\sim\psi \in atom(s)$. It follows that $\sim\psi \in atom(T)$, so $\psi \notin atom(T)$. Thus, T is a potential successor of (S, s).

We assume that the claim has been shown when the height of T is l, and we consider T of height $l+1$. The argument that for $\bigcirc\psi \in cl(\varphi)$ we have that $\bigcirc\psi \in atom(s)$ iff $\psi \in atom(T)$ is as before. Let T' be a j-child of T. We claim that there is some j-relative s' of s, and T' is an apparent successor of $(\mathbf{S}, s')$. If the claim is indeed true, then we can apply the induction hypothesis to complete the proof. So it remains to prove the claim.

Suppose that the claim is not true. Then for every s' that is a j-relative of s, there is a formula $\bigcirc\psi \in ecl(\varphi)$ such that $ad(\psi) \leq ad(\varphi) + l - 1$, $\bigcirc\psi \in eatom(s')$ but $\psi \notin eatom(\mathrm{T}')$. We say ψ *separates* s' from T'. Let $\psi_1, \ldots, \psi_k$ be the formulas that separate the j-relatives of s from T'. Note that the alternation depth of all these formulas is bounded by $ad(\varphi) + l - 1$. Thus, by the definition of extended knowledge trees, $\bigcirc K_j(\psi_1 \vee \ldots \vee \psi_k) \in eatom(s)$, and therefore $K_j(\psi_1 \vee \ldots \vee \psi_k) \in eatom(\mathrm{T})$, It follows that $\psi_1 \vee \ldots \vee \psi_k \in eatom(\mathrm{T}')$, which implies that $\psi_i \in eatom(\mathrm{T}')$ for some $1 \leq i \leq k$. But $\psi_1, \ldots, \psi_k$ are the formulas that separate the j-relatives of s from T', so by definition $\psi_i \notin eatom(\mathrm{T}')$ for $1 \leq i \leq k$ - contradiction. This completes the proof. ■

We say that an extended knowledge tree T is consistent if for all nodes t of T we have that $eatom(\mathrm{T})$ is consistent. We are almost ready for the construction of Hintikka trees for consistent formulas. The last technical result we need is a guarantee for the existence of consistent extended knowledge tree.

Lemma 5.4. *Let* φ *be a formula of* $LK_{(m)}$ *and* $l \geq 0$. *If* φ *is consistent then there is a consistent standard extended knowledge tree* T *of depth* l *such that* $\varphi \in eatom(\mathrm{T})$.

Proof. By Lemma 5.1, since φ is consistent, it has a pseudo knowledge model M. With every state of M we associate an extended standard knowledge tree as in the proof of Lemma 4.4 That is, we first label points by extended atoms, then we associate nonstandard trees, and finally we prune them to get standard knowledge trees. We have to show that these are extended knowledge trees, i.e., that the additional clause in the definition of extended knowledge trees is satisfied.

A node t of the tree corresponds a a state w in M. The j-relatives of t corresponds the the state that are related to w via $\sim_j$. Let $\{\psi_1, \ldots, \psi_k\}$ be a set of formula such that if t' is a j-relative of t then $\circ\psi_i \in atom(t)$ for some $1 \leq i \leq k$. Then, by the knowledge axioms, $K_j\circ(\psi_1 \vee \ldots \vee \psi_k)$ is in w. By the axiom for interaction of knowledge and time we have that $\circ K_j \bigvee(\psi_1 \vee \ldots \vee \psi_k)$ is also in w.

Finally, there is a state w in M that satisfies φ. Let φ be the consistent extended standard knowledge tree T of depth l associated with w. Since w satisfies φ, we have that $\varphi \in eatom(\mathrm{T})$. ■

We are now ready for our main result.

Theorem 5.5. *The axioms for propositional reasoning, knowledge, linear time (resp. branching time) together with LKA1 (resp. BKA1) give a sound and complete axiomatization for $KL_{(m)}$ (resp. $KB_{(m)}$) in $\mathscr{C}_{(nf,sync)}$.*

Proof. Assume that a formula φ is consistent. We prove that φ is satisfiable by constructing a Hintikka tree for φ. In this construction it is convenient to assume that the tree is ordered, and, in particular, for every vertex x there is a *leftmost* successor x'. A path $x_i, x_{i+1}, \ldots$ in the tree is a leftmost path if x_{j+1} is the leftmost successor of x_j for all $j \geq i$. We assume that for every extended tree T there is a standard enumeration $t_0, t_1, \ldots$ of the nodes of T, where t_0 is T. We assume without loss of generality that if τ is a Hintikka tree x is a vertex of τ, and t_i is the i-th node of $lab(x)$, then $\sigma_x(t_i)$ is the i-th successor of x. In particular, $\sigma_x(lab(x))$ is the leftmost successor of x. Thus, the path that satisfies the formulas of the form $\psi U \xi$ in $atom(lab(x))$ is a leftmost path.

We now describe the construction of a Hintikka tree τ for φ. We first construct an *extended Hintikka tree* τ', which is labeled by extended standard knowledge trees. We then replace every extended knowledge tree by its associated knowledge tree, to obtain a Hintikka tree.

By Lemma 5.4, there is a consistent extended standard knowledge tree T of depth $ad(\varphi)$ such that $\varphi \in atom(\mathrm{T})$. We let T be the label of the root x_0 of τ. Consider now all the nodes $t_0, \ldots, t_k$ of T, where t_0 is T. Since T is consistent, $eatom(t_i)$ is consistent for each i. By Lemma 5.2, there is a pseudo temporal model $M_{L,i}$ of $eatom(t_i)$. We use $M_{L,i}$ to label the path $x_1, x_2, \ldots$, where x_1 is the i-th successor of x_0, and x_{j+1} is the leftmost successor of x_j, for $j \geq 1$. (In particular, $M_{L,0}$ will be used to label the leftmost path starting at x_0.)

Consider the consistent set of formulas $\alpha_j = \{\psi \in ecl(\varphi) : (M_{L,i}, j) \models \psi\}$. By Lemma 5.4, there is a consistent standard knowledge tree T_j of depth $ad(\varphi)$ such that $\alpha_j = atom(\mathrm{T})$. We let T_j be the label of x_j.

In the inductive stage of the construction we have a partially labelled extended Hintikka tree, with the property that if x is labelled, then the leftmost path starting at x is labelled. If x is labelled, and its i-th successor is not labelled, then we use the pseudo temporal model $M_{L,i}$ of $eatom(t_i)$ to label the i-th successor of x and the leftmost path starting at the i-th successor.

Assume that have labeled all the nodes by extended standard knowledge trees. We now replace every extended knowledge tree by its associated knowledge tree, to obtain a Hintikka

tree. It is easy to check that the resulting tree τ satisfies conditions (H2) and (H3); by Lemma 5.3 it also satisfies (H1) ■

Acknowledgements

We would like to thank Martin Abadi, Karl Abrahamson, Ron Fagin, and Ed Wimmers for their comments on a previous draft of this paper.

References

[Ab] K.R. Abrahamson, *Decidability and expressiveness of logics of processes,* Ph.D. Thesis, University of Washington Technical Report #80-08-01, 1980.

[CM] M. Chandy and J. Misra, How processes learn, *Distributed Computing* **1**:1, 1986, pp. 40-52.

[DM] C. Dwork and Y. Moses, Knowledge and common knowledge in a Byzantine environment I: crash failures, *Theoretical Aspects of Reasoning About Knowledge: Proceedings of the 1986 Conference* (ed. J.Y. Halpern), Morgan Kaufmann, 1986, pp. 149-169.

[Em] E. A. Emerson, Alternative semantics for temporal logics, *Theoretical Comp. Sci.* **26**, 1983, pp. 121-130.

[EH1] E.A. Emerson and J.Y. Halpern, Decision procedures and expressiveness in the temporal logic of branching time, *J. Computer and Systems Science*, **30**:1, 1985, pp. 1-24.

[EH2] E.A. Emerson and J.Y. Halpern, "Sometimes" and "not never" revisited: on branching vs. linear time, *J. ACM*, 33:1, 1986, pp. 151-178.

[FHV1] R. Fagin, J.Y. Halpern, and M.Y. Vardi, A model-theoretic analysis of knowledge, *Proceedings of the 25th Annual IEEE Symposium on Foundations of Computer Science*, 1984, pp. 268-278.

[FHV2] R. Fagin, J.Y. Halpern, and M.Y. Vardi, What can machines know? On the epistemic properties of machines, *Proceedings of AAAI-86*, 1986, pp. 428-434.

[FI1] M.J. Fischer and N. Immerman, Foundations of knowledge for distributed systems, *Theoretical Aspects of Reasoning About Knowledge: Proceedings of the 1986 Conference* (ed. J.Y. Halpern), Morgan Kaufmann, 1986, pp. 171-185.

[GPSS] D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi, On the temporal analysis of fairness, *Proceedings of the 7th ACM Symposium on Principles of Programming Languages*, 1980, pp. 163-173.

[Hal] J.Y. Halpern, Using reasoning about knowledge to analyze distributed systems, *Annual Review of Computer Science, Vol. 2*, Annual Reviews Inc., 1987, to appear.

[HF] J.Y. Halpern and R. Fagin, A formal model of knowledge, action, and communication in distributed systems: preliminary report, *Proceedings of the 4th ACM Symposium on Principles of Distributed Computing*, 1985, pp. 224-236.

[HM1] J.Y. Halpern and Y.O. Moses, Knowledge and common knowledge in a distributed environment, in *Proceedings of the 3rd ACM Symposium on Principles of Distributed Computing*, 1984, pp. 50-61; revised report appears as IBM Research Report RJ 4421, Jan. 1986.

[HM2] J. Y. Halpern and Y.O. Moses, A guide to the modal logics of knowledge and belief: preliminary report, *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, 1985, pp. 480-490.

[HV] J.Y. Halpern and M.Y. Vardi, The complexity of reasoning about knowledge and time: extended abstract, *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, 1986, pp. 304-315.

[HV1] J.Y. Halpern and M.Y. Vardi, *The complexity of reasoning about knowledge and time, I - lower bounds*, IBM Research Report RJ5764, August 1987.

[LR] R. Ladner and J.H. Reif, The logic of distributed protocols, *Theoretical Aspects of Reasoning About Knowledge: Proceedings of the 1986 Conference* (ed. J.Y. Halpern), Morgan Kaufmann, 1986, pp. 207-221.

[La1] L. Lamport, What good is temporal logic?, *Information Processing 83* (ed. R.E.A. Mason), Elsevier Publishers, 1983, pp. 657-668.

[La2] L. Lamport, "Sometime" is sometimes "not never": on the temporal logic of programs, *Proceedings of the 7th ACM Symposium on Principles of Programming Languages*, 1980, pp. 174-185.

[Le1] D.J. Lehmann, Knowledge, common knowledge, and related puzzles, *Proceedings of the 3rd ACM Symposium on Principles of Distributed Computing*, 1984, pp. 62-67.

[Le2] D.J. Lehmann, Talk given at the 3rd ACM Symposium on Principles of Distributed Computing, August, 1984.

[Mi] P. Milgrom, An axiomatic characterization of common knowledge, *Econometrica*, 49:1, 1981, pp. 219-222.

[Mo] R.C. Moore, Reasoning about knowledge and action, in *Formal Theories of the Commonsense World* (ed. J. Hobbs and R.C. Moore), Ablex Publishing Corp., Norwood, New Jersey, 1985.

[MT] Y. Moses and M. Tuttle, Programming simultaneous actions using common knowledge, *Proceedings of the 27th Annual IEEE Symposium on Foundations of Computer Science*, 1986, pp. 208-221.

[PR] R. Parikh and R. Ramanujam, Distributed processing and the logic of knowledge, *Proceedings of the Workshop on Logics of Programs* (ed. R. Parikh), Springer-Verlag, Lecture Notes in Computer Science, vol. 193, 1985, pp. 256-268.

[Pn] A. Pnueli, Linear and branching structures in the semantics and logics of reactive systems, *Proceedings of the 12th International Colloquium on Automata, Languages and Programming*, Springer-Verlag, Lecture Notes in Computer Science - v. 194, 1985, pp. 15-32.

[Ra] M.O. Rabin, Weakly definable relations and special automata. *Proceedings of the Symposium on Mathematical Logic and Foundations of Set Theory* (Y. Bar-Hilel, ed.), North-Holland, 1970, pp. 1-23.

[Rog] H. Rogers, Jr., *Theory of Recursive Functions and Effective Computability*, McGraw-Hill, 1967.

[Ros] S. Rosenschein, Formal theories of knowledge in AI and robotics, *New Generation Computing* 3, 1985, pp. 345-357.

[RK] S. Rosenschein and L. Kaelbling, The synthesis of digital machines with provable epistemic properties, *Theoretical Aspects of Reasoning About Knowledge: Proceedings of the 1986 Conference* (ed. J.Y. Halpern), Morgan Kaufmann, 1986, pp. 83-97.

[Sa] M. Sato, A study of Kripke-style methods of some modal logics by Gentzen's sequential method, *Publications Research Institute for Mathematical Sciences, Kyoto University*, 13:2, 1977.

[SC] A.P. Sistla and E.M. Clarke, The complexity of propositional linear temporal logics, *J. ACM*, 32:3, 1985, pp. 733-749.

[VS] M.Y. Vardi and L.J. Stockmeyer, Improved upper and lower bounds for modal logics of programs, *Proceedings of the 17th ACM Symposium of Computing*, 1985, pp. 240-251.

[VW] M.Y. Vardi and P.L. Wolper, Automata-theoretic techniques for modal logics of programs, *J. Computer and System Sciences* 32(1986), pp. 183-221.