Model Checking Knowledge and Time in Systems with Perfect Recall (Extended Abstract)*

R. van der Meyden[1] and N. V. Shilov[2]

[1] School of Computer Science and Engineering,
University of New South Wales, Sydney 2052, Australia.
meyden@cse.unsw.edu.au
[2] Institute of Informatics Systems, Novosibirsk
6, Lavrent'ev av., Novosibirsk, 630090, Russia
shilov@iis.nsk.su

Abstract. This paper studies model checking for the modal logic of knowledge and linear time in distributed systems with perfect recall. It is shown that this problem (1) is undecidable for a language with operators for *until* and *common knowledge*, (2) is PSPACE-complete for a language with *common knowledge* but without *until*, (3) has non-elementary upper and lower bounds for a language with *until* but without *common knowledge*. Model checking *bounded knowledge depth* formulae of the last of these languages is considered in greater detail, and an automata-theoretic decision procedure is developed for this problem, that yields a more precise complexity characterization.

1 Introduction

Modal logics have been found to be convenient formalisms for reasoning about distributed systems [MP91], in large part because such logics enable automated verification by model checking of specifications [CGP99]. This involves constructing a model of the system to be verified, and then testing that this model satisfies a formula specifying the system. Frequently, the model of the system is finite state, but model checking of infinite state systems is an emerging area of research.

Epistemic logic, or the *logic of knowledge* [HM90,FHMV95], is a recent addition to the family of modal logics that have been applied to reasoning about distributed systems. This logic allows one to express that an agent in the system *knows* (has the information) that some fact holds. This expressiveness is particularly useful for reasoning about distributed systems with unreliable components or communication media. In such settings, information arises in subtle ways, and it can be difficult to

* To appear, Proc. Conf. on Foundations of Software Technology and Theoretical Computer Science, Madras, Dec 1999. Copyright Springer-Verlag. Work supported by an Australian Research Council Large Grant, done while the authors were employed in the School of Computing Sciences, University of Technology, Sydney. The first author acknowledges the hospitality of the Department of Computer Science, Utrecht University while revising this paper.

express the precise conditions under which an agent has certain knowledge. On the other hand, the behavior of agents is often a simple function of their state of knowledge. Examples of knowledge-level analysis of systems illustrating this claim are given in [FHMV95].

A topic of interest for logics of knowledge is the extent to which they (like other modal logics) allow for automated analysis of designs and specifications. Combinations of temporal and epistemic logics are especially significant, since a frequent concern in applications is how knowledge changes over time. A number of papers have studied the problem of model checking logics of knowledge and time in *finite* state systems [HV91,FHMV95,Var96]. However, much of the literature on applications of logics of knowledge assumes that agents have *perfect recall*, i.e., remember all their past states, and this results in infinite state systems. Model checking of the logic of knowledge with respect to the perfect recall semantics has been considered by van der Meyden [Mey98], but this work deals with a language that does not include temporal operators.

In the present paper, we study model checking a combined logic of knowledge and linear time in *synchronous* systems with perfect recall. Like van der Meyden [Mey98], we assume that agents operate in a finite state environment, but we extend this framework to allow Büchi fairness constraints. Since the perfect recall assumption generates an infinite Kripke structure from the environment, the problem we study is an example of infinite state model checking.

Formal definitions of synchronous systems with perfect recall and the model checking problem are presented in Section 2. While model checking a logic with operators for knowledge and *common knowledge* is decidable [Mey98], the addition of the linear time temporal operators *next* and *until* makes the problem undecidable (Section 3). However, decidability is retained for two fragments of this extended language: the fragments in which we (1) omit the *until* operator (this case is PSPACE complete) or (2) omit the *common knowledge* operator (this case is non-elementary). The latter result may be obtained by means of reductions to and from various powerful logics already known to be decidable (weak S1S and Chain Logic with an equal level predicate [Tho92]). However, we also present (in Section 4) an alternative proof of this latter result, using novel automata-theoretic constructions, that provides a more informative complexity characterization. Section 5 discusses related work and topics for further research.

2 Basic Definitions

This section further develops the definition of environments of [Mey98] by adding fairness constraints, and defines the model checking problem we study.

Let *Prop* be a set of atomic propositional constants, $n \geq 0$ be a natural number and $\mathcal{O}$ be a set. Define a *finite interpreted environment for n agents* to be a tuple E of the form $\langle S, I, T, O, \pi, \alpha \rangle$ where the components are as follows:

1. S is a finite set of *states* of the environment,
2. I is a subset of S, representing the possible *initial states*,
3. $T \subseteq S^2$ is a *transition relation*,
4. O is a tuple $(O_1, \ldots, O_n)$ of functions, where for each $i \in \{1 \,..\, n\}$ the component $O_i : S \to \mathcal{O}$ is called the *observation function of agent i*,
5. $\pi : S \to \{0,1\}^{Prop}$ is an *interpretation*,
6. $\alpha \subseteq S$ is an *acceptance condition*.

Intuitively, an environment is a finite-state transition system where states encode values of local variables, messages in transit, failure of components, etc. For states s, s' the relation $s T s'$ means that if the system is in state s, then at the next tick of the clock it could be in state s'. If s is a state and i an agent then $O_i(s)$ represents the observation agent i makes when the system is in state s, i.e., the information about the state that is accessible to the agent. The interpretation maps each state to an assignment of truth values to the atomic propositional constants in *Prop*. The acceptance conditions are standard Büchi conditions which are used to model fairness requirements on evolutions of the environment.

A *trace* of an environment E is a finite sequence of states $s_0 s_1 \dots s_m$ such that $s_0 \in I$ and $s_j \; T \; s_{j+1}$ for all $j < m$. A *run* of an environment E is an infinite sequence $r : \mathbb{N} \to S$ of states of E such that every finite prefix of r is a trace of E and there exists a state $s \in \alpha$ that occurs infinitely often in r. We say that the acceptance condition of E is *trivial* if $\alpha = S$. A *point* of E is a tuple (r, m), where r is a run of E and m a natural number. Intuitively, a point identifies a particular instant of time along the history described by the run.

Individual runs of an environment provide sufficient structure for the interpretation of formulae of linear temporal logic. To interpret formulae involving knowledge, we use the agents' observations to determine the points they consider possible. There are many ways one could do this. The particular approach used in this paper models a *synchronous perfect-recall* semantics of knowledge. Given a run r of an environment for n agents with observation functions $O_1, \dots, O_n$, we define the *local state of agent i at time* $m \geq 0$ to be the sequence $r_i(m) = O_i(r(0)) \dots O_i(r(m))$. That is, the local state of an agent at a point in a run consists of a complete record of the observations the agent has made up to that point. These local states may be used to define for each agent i a relation $\sim_i$ of *indistinguishability* on points $(r, m), (r', m')$ of E, by $(r, m) \sim_i (r', m')$ if $r_i(m) = r'_i(m')$. Intuitively, when $(r, m) \sim_i (r', m')$, agent i has failed to receive enough information at these points to determine whether it is in one situation or the other. Clearly, each $\sim_i$ is an equivalence relation. The use of the term "synchronous" above reflects the fact that if $(r, m) \sim_i (r', m')$, then we must have $m = m'$. The relations $\sim_i$ will be used to define the semantics of knowledge for individual agents. We will also consider an operator for common knowledge, a kind of group knowledge, for which we use another relation. If $G \subseteq \{1 \,.\,.\, n\}$ is a *group* of agents (i.e., two or more) then we define the relation $\sim_G$ on points to be the reflexive transitive closure of the union of all indistinguishability relations $\sim_i$ for $i \in G$, i.e., $\sim_G = (\bigcup_{i \in G} \sim_i)^*$.

We will be concerned with model checking a propositional multi-modal language for knowledge and linear time based on a set *Prop* of atomic propositional constants, with formulae generated by the modalities $\bigcirc$ (next), $\mathcal{U}$ (until), a knowledge operator K_i for each agent $i \in \{1 \,.\,.\, n\}$, and a common knowledge operator C_G for each group of agents $G \subseteq \{1 \,.\,.\, n\}$. Formulae of the language are defined as follows: each atomic propositional constant $p \in \mathit{Prop}$ is a formula, and if φ and ψ are formulae, then so are $\neg\varphi$, $\varphi \wedge \psi$, $\bigcirc\varphi$, $\varphi \, \mathcal{U} \, \psi$, $K_i\varphi$ and $C_G\varphi$ for each $i \in \{1 \,.\,.\, n\}$ and group $G \subseteq \{1 \,.\,.\, n\}$. We write $\mathcal{L}_{\{\bigcirc, \mathcal{U}, K_1, \dots, K_n, C\}}$ for the set of formulae. We will refer to sublanguages of this language by a similar expression that lists the operators generating the language. For example, $\mathcal{L}_{\{K_1, \dots, K_n, C\}}$ refers to the language of the logic of knowledge (without time). As usual, we use the abbreviations $\Diamond\varphi$ for $\mathbf{true} \, \mathcal{U} \, \varphi$, and $\Box\varphi$ for $\neg\Diamond\neg\varphi$.

The semantics of this language is defined as follows. Suppose we are given an environment E with interpretation π. We define satisfaction of a formula φ at a point (r, m) of a run of E, denoted $E, (r, m) \models \varphi$, inductively on the structure of φ. The cases for the temporal fragment of the language are standard:

$$E, (r, m) \models p \qquad \text{if } \pi(r(m))(p) = 1, \text{ where } p \in \mathit{Prop},$$

$$
\begin{array}{ll}
E,(r,m) \models \varphi_1 \wedge \varphi_2 & \text{if } E,(r,m) \models \varphi_1 \text{ and } E,(r,m) \models \varphi_2, \\
E,(r,m) \models \neg\varphi & \text{if not } E,(r,m) \models \varphi, \\
E,(r,m) \models \bigcirc\varphi & \text{if } E,(r,m+1) \models \varphi, \\
E,(r,m) \models \varphi_1 \,\mathcal{U}\, \varphi_2 & \text{if there exists } m'' \geq m \text{ such that } E,(r,m'') \models \varphi_2 \text{ and} \\
& E,(r,m') \models \varphi_1 \text{ for all } m' \text{ with } m \leq m' < m''.
\end{array}
$$

The semantics of the knowledge and common knowledge operators is defined by

$$
E,(r,m) \models K_i\varphi \text{ if } E,(r',m') \models \varphi \text{ for all points } (r',m') \text{ of } E \text{ satisfying } (r',m') \sim_i (r,m)
$$

$$
E,(r,m) \models C_G\varphi \text{ if } E,(r',m') \models \varphi \text{ for all points } (r',m') \text{ of } E \text{ satisfying } (r',m') \sim_G (r,m)
$$

This definition can be viewed as an instance of the general framework for the semantics of knowledge proposed in [HM90]. Intuitively, an agent knows a formula to be true if this formula holds at all points that the agent is unable to distinguish from the actual point. Common knowledge may be understood as follows. For G a group of agents, define the operator E_G, read "everyone in G knows" by $E_G\varphi \equiv \bigwedge_{i \in G} K_i\varphi$. Then $C_G\varphi$ is equivalent to the infinite conjunction of the formulae $E_G^k\varphi$ for $k \geq 1$. That is, φ is common knowledge if everyone knows φ, everyone knows that everyone knows φ, etc. We refer the reader to [HM90,FHMV95] for further motivation and background.

We may now define the model checking problem we consider in this paper. Say that a formula φ is *realized* in the environment E if for all runs r of E, we have $E,(r,0) \models \varphi$. We are interested in the following problem, which we call the *realization problem*: given an an environment E and a formula φ of a language $\mathcal{L}$, determine if φ is realized in E. We will consider this problem with respect to several sublanguages of $\mathcal{L}_{\{\bigcirc,\mathcal{U},K_1,\ldots,K_n,C\}}$.

3 Complexity Bounds

We now present a number of results on the complexity of the realization problem for various fragments of the language, and briefly sketch their proofs. First, we consider the most expressive language $\mathcal{L}_{\{\bigcirc,\mathcal{U},K_1,\ldots,K_n,C\}}$, containing all the modal operators we have defined. Here the outcome of our investigation is negative:

Theorem 1. *There exist a class of finite environments for two agents with trivial acceptance conditions and a formula of $\mathcal{L}_{\{\bigcirc,\mathcal{U},K_1,\ldots,K_n,C\}}$ such that it is undecidable whether the formula is realized in a given environment of this class.*

That is, even a restricted case of the realization problem for $\mathcal{L}_{\{\bigcirc,\mathcal{U},K_1,\ldots,K_n,C\}}$ is undecidable. The proof of Theorem 1 employs ideas from [Mey98] concerning *model checking at a trace* for the language $\mathcal{L}_{\{K_1,\ldots,K_n,C\}}$. Stated in terms of our current terms and notations, this problem is to determine, *given an environment E with a trivial acceptance condition, a trace t of E and a formula φ of $\mathcal{L}_{\{K_1,\ldots,K_n,C\}}$, whether $E,(r,|t|) \models \varphi$ for all runs r of E extending t.*[1] This model checking problem was studied in [Mey98] for both the synchronous and an *asynchronous* perfect recall semantics of knowledge. The following two results are proved in [Mey98].[2]

[1] It can be shown that if r and r' are two runs extending t and $\varphi \in \mathcal{L}_{\{K_1,\ldots,K_n,C\}}$ then $E,(r,|t|) \models \varphi$ iff $E,(r',|t|) \models \varphi$.

[2] In both these results $\{1,2\}$ is a set of agents while p is a propositional constant.

Theorem 2. *With respect to the* synchronous *perfect recall semantics, the problem of model checking at a trace is in PSPACE for the language* $\mathcal{L}_{\{K_1,\ldots,K_n,C\}}$. *It is PSPACE hard for the fixed formula* $C_{\{1,2\}}p$.

Theorem 3. *There exists an environment for two agents such that with respect to the* asynchronous *perfect recall semantics, the problem of model checking the fixed formula* $C_{\{1,2\}}p$ *at a given trace of the environment is undecidable.*

The proof of the lower bound in Theorem 2 involved showing that the synchronous semantics can simulate PSPACE computations, with Turing machine configurations represented as traces and the step relation on configurations represented by the composition $\sim_1 \circ \sim_2$. The common knowledge operator then allows us to represent the transitive closure of the step relation, enabling a formula to refer to the result of a PSPACE computation. The proof of Theorem 3 used a similar representation of Turing machine computations, but first uses asynchrony to "guess" the amount of space required by the computation. To prove Theorem 1 we reuse this approach to representation of Turing machine computations. However, instead of asynchrony, we now use the temporal operators to refer to a sufficiently long configuration. As before, we then describe the outcome of the computation starting at that configuration using the common knowledge operator.

In the language $\mathcal{L}_{\{\bigcirc,\mathcal{U},K_1,\ldots,K_n,C\}}$ we have two operators, the until operator and the common knowledge operator, whose semantics allows an arbitrary reach through two orthogonal dimensions in our semantic structures. In other contexts, these operators are individually tractable, e.g., the validity problem for both the logic of knowledge and common knowledge $\mathcal{L}_{\{K_1,\ldots,K_n,C\}}$ and temporal logic $\mathcal{L}_{\{\bigcirc,\mathcal{U}\}}$ are known to be decidable. It therefore makes sense to study the result of eliminating one of these operators from our language. For the language obtained by excluding the until operator we have:

Theorem 4. *The realization problem for* $\mathcal{L}_{\{\bigcirc,K_1,\ldots,K_n,C\}}$ *is PSPACE complete.*

The proof of Theorem 4 is similar to the proof in [Mey98] of Theorem 2, and exploits the fact, for checking realization of a formula φ, instead of the infinite set of runs we can confine our attention to the finite set of traces of length at most $|\varphi|$, which is, intuitively, the furthest that the temporal operators can reach. This set of traces has exponentially many elements, but since each trace is of linear size we may do model checking within polynomial space using techniques of [Mey98].

For the language without common knowledge, some new techniques are required. Here we obtain the following.

Theorem 5. *The realization problem for* $\mathcal{L}_{\{\bigcirc,\mathcal{U},K_1,\ldots,K_n\}}$ *is decidable, with non-elementary upper and lower bounds.*

The proof for both the upper and the lower bound can be obtained by reductions from variants of SnS, the Monadic Second Order Logic of n Successors [Tho92], which is interpreted over the infinite tree $\{1 \mathinner{..} n\}^*$. The proof of the lower bound in Theorem 5 is by a reduction from WS1S, a version of S1S in which the second order quantifiers are restricted to range over finite sets. It is known [Mey74] that WS1S is decidable with lower bound non-elementary in the size of the formula. The upper bound result can be established by a translation to the problem of checking the validity of a formula of *Chain Logic with the Equal Level predicate* [Tho92] (or CLE) on tree structures. The logic CLE is an extension of a restriction of SnS. Chain Logic (CL) is obtained from SnS by restricting the interpretation of the second order quantifiers to sets that are chains, i.e., are totally ordered by the prefix relation. The logic CLE is obtained by adding to this restriction of SnS the equal level predicate, defined on $u, v \in \{1..n\}^*$ by $E(u, v)$ if $|u| = |v|$. This approach to the upper bound for the realization problem for $\mathcal{L}_{\{\bigcirc,\mathcal{U},K_1,\ldots,K_n\}}$ from this proof is rather indirect, however, as decidability of CLE is proved by Thomas [Tho92] using a translation to S1S. The logic S1S in turn is known to be decidable using automata theoretic arguments [Büc60]. Thus, we have gone from automata (in the definition of environments) to logic, and back to automata. In the following section we present a proof of Theorem 5 that is based directly on automata theoretic constructions, and which yields a more informative complexity characterization.

4 An algorithm for bounded knowledge depth formulae

Our more informative characterization of the complexity of realization for the language $\mathcal{L}_{\{\bigcirc,\mathcal{U},K_1,\ldots,K_n\}}$ is cast in terms of the *knowledge depth* of formulae, i.e., the maximal depth of nesting of knowledge operators in a formula. For example, $depth(K_1(\bigcirc K_2(q \wedge K_2 r))) = 3$. Our approach to the decidability result will exploit k-*trees*, a data structure that has previously been used in the literature [Mey98] to represent depth k formulae of $\mathcal{L}_{\{K_1,\ldots,K_n\}}$ holding at a point of an environment. We show in this section that k-trees also encode enough information to interpret formulae of $\mathcal{L}_{\{\bigcirc,\mathcal{U},K_1,\ldots,K_n\}}$ with knowledge depth at most k. Throughout this section we assume a fixed finite environment E. We assume without loss of generality that every trace of E can be extended to a run of E.[3]

4.1 Trees

Intuitively, a k-tree, for $k \geq 0$, is a type of finite tree of height k in which vertices are labelled by states of the environment and edges are labelled by agents. It is convenient to represent these trees as follows.[4] For numbers $k \geq 0$ we define by mutual recursion the set $\mathcal{T}_k$ of k-*trees over* E, and the set $\mathcal{F}_k$ of *forests of* k-*trees over* E. Define $\mathcal{T}_0$ to be the set of tuples of the form $\langle s, \emptyset, \ldots, \emptyset \rangle$ where s is a state of E and the number of copies of the empty set $\emptyset$ is equal to the number of agents n. Once $\mathcal{T}_k$ has been defined, let $\mathcal{F}_k$ be the set of all subsets of $\mathcal{T}_k$. Now, define $\mathcal{T}_{k+1}$ to be the set of all tuples of the form $\langle s, U_1, \ldots, U_n \rangle$, where s is a state and U_i is in $\mathcal{F}_k$ for each $i \in \{1..n\}$. We denote $\bigcup_{k \geq 0} \mathcal{T}_k$ by $\mathcal{T}$.

Intuitively, in a tuple $\langle s, U_1, \ldots, U_n \rangle$, the state s represents the actual state of the environment, and for each $i \in \{1..n\}$ the set U_i represents the knowledge of agent i. Identifying a 0-tree $\langle s, \emptyset, \ldots, \emptyset \rangle$ with the state s, note that each component U_i in a 1-tree is simply a set of states: intuitively, those states agent i considers possible. For

---

[3] An environment not satisfying this condition can easily by modified (without changing its realization properties) by eliminating states that do not belong to any run.

[4] The definitions we give here are (for reasons of clarity and space) a slight simplification of those in [Mey98], which add some complications to enable k-trees to be used to interpret formulae of *alternation depth* at most k, a slightly larger class than the class of formulae of knowledge depth at most k.

higher k, the set U_i represents agent i's knowledge both about the universe and other agents' knowledge, up to depth k.

The elements of $\mathcal{T}_k$ correspond in an obvious way to trees of height k, with edges labelled by agents and nodes labelled by states. If $w = \langle s, U_1, \ldots, U_n \rangle$ we define $root(w)$ to be the state s. If w' is an element of U_i, then we say that w' is an *i-child* of w. When $w \in \mathcal{T}_0$ the labelled tree corresponding to w consists of just the root, labelled $root(w)$. The tree corresponding to $w \in \mathcal{T}_{k+1}$ has root labelled with $root(w)$, and for each i-child $w' \in \mathcal{T}_k$ of w there is an i-labelled edge from the root to a vertex at which the labelled subtree is that corresponding to w'. The following result characterises C_k, the number of k-trees over E.

Lemma 1. *Let $k \geq 0$ be a natural number and E be a finite environment for n agents with l states. Then C_k is not greater than $\exp(n \times l, k)/n$, where $\exp(a, b)$ is the function defined by $\exp(a, 0) = a$ and $\exp(a, b+1) = a2^{\exp(a,b)}$.*

For each $k \geq 0$ we may associate with each point (r, m) of E a k-tree $F_k(r, m)$, which captures some of the structure of the indistinguishability relations of the environment around that point. We proceed inductively. For $k = 0$ we define $F_0(r, m) = \langle r(m), \emptyset, \ldots, \emptyset \rangle$. For $k > 0$ we define $F_k(r, m) = \langle r(m), U_1, \ldots, U_m \rangle$, where for each agent i we have U_i equal to the set of $k-1$-trees $F_{k-1}(r', m')$ where (r', m') is a point of E with $(r', m') \sim_i (r, m)$.

For each point (r, m) of E let $\tau(r, m)$ be the trace $r(0) \ldots r(m)$. It is not difficult to see that for all $k \geq 0$ and for all points (r, m) and (r', m') with $\tau(r, m) = \tau(r', m')$ we have $F_k(r, m) = F_k(r', m')$. Thus, we may also view F_k as a function mapping traces of E to k-trees, and write $F_k(\tau)$ where τ is a trace of E. Note that we exploit the fact that every trace may be extended to a run here.

We now recall from [Mey98] some functions that may be used to update k-trees. These functions were used in [Mey98] to provide an algorithm for the problem of model checking at a trace (see above). We will use these functions below to define a sequence of Büchi automata for the realization problem. Let S, T and $\mathcal{O}$ be the set of states, the transition relation, and the set of observations of the environment E, respectively. We define for each number $k \geq 0$ the function $G_k : \mathcal{T}_k \times S \to \mathcal{T}_k$. The definition of G_k will be by mutual recursion with the functions $H_{k,i} : \mathcal{F}_k \times \mathcal{O} \to \mathcal{F}_k$, where $i \in \{1 \,..\, n\}$ and $k \geq 0$. Intuitively, if agent i's state of knowledge (to depth k) is represented by the the set of k-trees U, then $H_{k,i}(U, o)$ represents the agent's revised state of knowledge after it makes the observation $o \in \mathcal{O}$. We define $G_0(w, s) = \langle s, \emptyset, \ldots, \emptyset \rangle$. Once G_k has been defined, we define for each $i \in \{1 \,..\, n\}$ the function $H_{k,i}$ by taking $H_{k,i}(U, o)$ to be the set of k-trees $G_k(w, s)$ where $w \in U$ and $O_i(s) = o$ and $root(w)Ts$, i.e., there exists a transition of E from $root(w)$ to s. Using the functions $H_{k,i}$ we may now define G_{k+1} by setting $G_{k+1}(\langle s, U_1, \ldots, U_n \rangle, s')$ to be $\langle s', H_{k,1}[U_1, O_1(s')], \ldots, H_{k,n}[U_n, O_n(s')] \rangle$.

For our definitions (which are a slight variant of those in [Mey98]), we may establish the following theorem, essentially the same as a result proved in [Mey98].

Theorem 6. *For each $k \geq 0$, and for every finite trace $\tau \cdot s$ of E with final state s and prefix τ, we have the incremental update property $F_k(\tau \cdot s) = G_k(F_k(\tau), s)$.*

The definition of the function F_k above is not effective, ranging over the possibly infinite set of runs of E. In the case of traces of length 0 is it easily seen how to make it effective, obtaining functions that represent agents' knowledge in the *initial* states of the environment as a k-tree. We inductively define mappings $f_k : I \to \mathcal{T}_k$ for $k \geq 0$. In the base case, we put $f_0(s) = \langle s, \emptyset, \ldots, \emptyset \rangle$. For $k \geq 0$ we define $f_{k+1}(s)$ to be the $k+1$-tree with root s and an i-child $f_k(s')$ for each initial state s' with $O_i(s') = O_i(s)$.

Lemma 2. *For all runs r of E we have $F_k(r, 0) = f_k(r(0))$.*

Note that, here again, we rely on the fact that every trace (of length 0) can be extended to a run.

4.2 An automaton theoretic characterization

We now give an automata-theoretic characterization of realization that forms the basis for the algorithm discussed below.

We begin by defining a type of Büchi automata [Büc60]. The specific variety of automata we need are tuples of the form $A = \langle S, I, T, \alpha \rangle$, where S is a finite set of (control) states, $I \subseteq S$ is the set of initial states of the automaton, $T \subseteq S^2$ is a a transition relation, and $\alpha \subseteq S$ is its acceptance condition.[5] An *execution* of A is an infinite sequence $e : \mathbb{N} \to S$ of states of S such that for all $m \geq 0$ we have $e(m)Te(m+1)$. An execution e is said to be *properly initialised* if $e(0) \in I$. An execution is said to be *fair* if some state in α occurs infinitely often in the execution. A fair, properly initialised execution is called *accepting*. We also call accepting executions *runs*. The language accepted by the automaton A is the set $\mathcal{L}(A)$ of all runs of A.

Given the environment E fixed above, we now define an infinite sequence of Büchi automata $A_0(E), \ldots, A_k(E), \ldots$. Each automaton $A_k(E)$ defines a language consisting of infinite sequences of k-trees over that environment. These automata will be crucial to the algorithm we develop.

Let E be $\langle S, I, T, O, \pi, \alpha \rangle$ and $k \geq 0$. Define $A_k(E) = \langle S_k, I_k, T_k, \alpha_k \rangle$ to be the Büchi automaton with

1. S_k equal to the set $\mathcal{T}_k$ of k-trees over E,
2. initial states I_k equal to the set of k-trees $f_k(s)$ where $s \in I$,
3. transition relation T_k defined by wT_kw' when there exists state $s \in S$ such that $root(w)Ts$ and $w' = G_k(w, s)$,
4. acceptance condition α_k defined by $\alpha_k = \{w \in S_k : root(w) \in \alpha\}$.

Since $A_k(E)$ is a Büchi automaton on infinite words, the notions of an execution, a fair execution, a properly intialised execution and a run of $A_k(E)$ are meaningful. We define a *projection* operation $Proj_k$ mapping runs r of the automata $A_k(E)$ to infinite sequences of states of E, by $Proj_k(r)(m) = root(r(m))$. Conversely, there exists a *lift* operation $Lift_k$ mapping runs r of the environment E to sequences of k-trees, defined by $Lift_k(r)(m) = F_k(r(0) \ldots r(m))$. The proof of the following is straightforward from the definitions, Theorem 6 and Lemma 2:

Lemma 3. *For each $k \geq 0$ the mappings $Proj_k$ and $Lift_k$ are inverse functions; $Proj_k$ maps the set of runs of $A_k(E)$ onto the set of runs of E while $Lift_k$ maps the set of runs of E onto the set of runs of $A_k(E)$.*

We now show that the automata $A_k(E)$ adequately capture the depth k formulae of $\mathcal{L}_{\{\bigcirc, \mathcal{U}, K_1, \ldots, K_n\}}$ holding at points of E. To do so, we define for each k a relation $\models_k$ between points in executions of $A_k(E)$ and formulae of knowledge depth $\leq k$. The definition of $\models_k$ is by induction on k, as follows. For the basic propositions and the temporal operators, the definition is much like the standard semantics. Thus, for an execution e of $A_k(E)$ and $m \geq 0$,

[5] These are slightly less general than usual, in that the input alphabet and control states coincide, and the transition function has a specific form that is derived from the transition relation.

$$
\begin{array}{ll}
E,(e,m) \models_k p & \text{if } \pi(\mathit{root}(e(m)))(p) = 1, \text{ where } p \in \mathit{Prop}, \\
E,(e,m) \models_k \varphi_1 \wedge \varphi_2 & \text{if } E,(e,m) \models_k \varphi_1 \text{ and } E,(e,m) \models_k \varphi_2, \\
E,(e,m) \models_k \neg\varphi & \text{if not } E,(e,m) \models_k \varphi, \\
E,(e,m) \models_k \bigcirc\varphi & \text{if } E,(e,m+1) \models_k \varphi, \\
E,(e,m) \models_k \varphi_1 \,\mathcal{U}\, \varphi_2 & \text{if there exists } m'' \geq m \text{ such that } E,(e,m'') \models_k \varphi_2 \text{ and} \\
& E,(e,m') \models_k \varphi_1 \text{ for all } m' \text{ with } m \leq m' < m''.
\end{array}
$$

The interesting case concerns the knowledge operators, where we make use of the fact that we are dealing with a sequence of k-trees. It is convenient to first define $\models_k$ not on points, but on *k-trees* w, for formulae $K_i\varphi$ of $\mathcal{L}_{\{\bigcirc,\mathcal{U},K_1,\ldots,K_n\}}$ of knowledge depth at most k, by

$E, w \models_k K_i\varphi$ if for all $k-1$-trees w' that are i-children of w, and for all fair executions e of $A_{k-1}(E)$ such that $e(0) = w'$, we have $E,(e,0) \models_{k-1} \varphi$.

Note that we consider fair executions e rather than runs in this definition because w' is not necessarily an initial state of $A_{k-1}(E)$. We then define $\models_k$ on points by

$E,(e,m) \models_k K_i\varphi$ if $E, e(m) \models_k K_i\varphi$.

The following result establishes a connection between the relations $\models_k$ and the semantics of $\mathcal{L}_{\{\bigcirc,\mathcal{U},K_1,\ldots,K_n\}}$ in an environment E.

Lemma 4. *For every natural number $k \geq 0$, every formula φ in $\mathcal{L}_{\{\bigcirc,\mathcal{U},K_1,\ldots,K_n\}}$ of knowledge depth at most k, for every environment E, every run r of E and $m \geq 0$ we have $E,(r,m) \models \varphi$ iff $E,(\mathit{Lift}_k(r),m) \models_k \varphi$.*

This equivalence forms the basis for our decision procedure for realization.

4.3 The algorithm

We are now in a position to present the algorithm for the realization problem for $\mathcal{L}_{\{\bigcirc,\mathcal{U},K_1,\ldots,K_n\}}$. Let us first note that in the special case where $\mathit{depth}(\varphi) = 0$, i.e., formulae not containing the knowledge operators, testing realization amounts to a problem of temporal logic to which well-known techniques may straightforwardly be applied (see below). To generalize to formulae of greater depth, we show how to decide the relation $E, w \models_k K_i\varphi$. We achieve this by factoring formulae into their temporal and knowledge components. To represent the temporal components, define a *context* to be just like a formula of $\mathcal{L}_{\{\bigcirc,\mathcal{U},K_1,\ldots,K_n\}}$, but with additional propositional variables from a special separate countable set Var. If β is a context then we denote by $\mathit{Var}(\beta)$ the set of all variables which occur in β. A *pure temporal context* is a context not containing any occurrences of the knowledge operators.

We separate the temporal and knowledge aspects of formulae by means of the following way of "exploding" a formula. Define a *puff* to be a finite sequence of pairs $(\mathit{set}_0, \mathit{map}_0)\ldots(\mathit{set}_m, \mathit{map}_m)$ where the set_j are finite sets of pure temporal contexts, such that $\mathit{Var}(\mathit{set}_0) = \emptyset$, and for $j \neq j'$ the sets $\mathit{Var}(\mathit{set}_j)$ and $\mathit{Var}(\mathit{set}_{j'})$ are disjoint, and the map_j are mappings $\mathit{map}_j : \mathit{Var}(\mathit{set}_j) \rightarrow \{K_1,\ldots,K_n\} \times \mathit{set}_{j-1}$. (Thus, $\mathit{map}_0 = \emptyset$.) The result $\varphi \cdot \mathit{map}_j$ of applying a mapping map_j to a context φ is defined to be the context obtained by simultaneously substituting for each occurrence in φ of a variable $x \in \mathit{Var}(\mathit{set}_j)$ the formula $K_i\psi$ such that $\mathit{map}_j(x) = (K_i,\psi)$. A puff is said to be a *complete separation* of a formula φ if set_m contains a single context β, such that $\varphi = \beta \cdot \mathit{map}_m \cdot \ldots \cdot \mathit{map}_1$. Let *cep* be a function from formulae to puffs such that $\mathit{cep}(vp)$ is a complete separation of φ. We call the unique pure temporal context β in the top level set_m of a complete separation of φ the *temporal skeleton* of φ.

Example 1. The puff

$$\begin{array}{ll} set_0 = \{p\} & map_0 = \emptyset \\ set_1 = \{\bigcirc x,\ \Box x\} & map_1 : x \mapsto (K_2, p) \\ set_2 = \{y\,\mathcal{U}\,z\} & map_2 : y \mapsto (K_1, \bigcirc x) \\ & map_2 : z \mapsto (K_2, \Box x) \end{array}$$

is a complete separation of the formula $(K_1 \bigcirc K_2 p)\mathcal{U}(K_2 \Box K_2 p)$. The temporal skeleton of this formula is $y\,\mathcal{U}\,z$.

A *valuation* is a partial mapping $v : Var \rightarrow \mathcal{P}(\mathcal{T})$. This associates with each propositional variable the set of trees at which it is true. For each valuation v one can extend the relation $\models_k$ on formulae to a relation $\models_k^v$ on temporal contexts by simply adding the superscript v to $\models_k$ throughout the clauses above, and by adding the following clause:

- $E, (e, m) \models_k^v x$, where $x \in Var$ is a variable, iff $e(m) \in v(x)$.

A valuation v is said to be *consistent* with a puff $(set_0, map_0) \ldots (set_m, map_m)$ if for each $j \in \{1..m\}$, every variable $x \in Var(set_j)$ and every $w \in \mathcal{T}_j$, if $map_j(x) = (K_i, \beta)$ then $w \in v(x)$ iff $E, w \models_j^v K_i\beta$.

Lemma 5. *Suppose φ is a formula of $\mathcal{L}_{\{\bigcirc, \mathcal{U}, K_1, \ldots, K_n\}}$ of knowledge depth k. Let β be the temporal skeleton of φ. Suppose that v is a valuation consistent with $cep(\varphi)$. Then for all fair executions e of $A_k(E)$ and for all $m \geq 0$, we have $E, (e, m) \models_k \varphi$ iff $E, (e, m) \models_k^v \beta$.*

By Lemma 4, to determine $E, (r, m) \models \varphi$ for formulae of depth φ it suffices to decide $E, (Lift_k(r), m) \models_k \varphi$. The effect of Lemma 5 is to reduce the complicated recursion through k-trees required to evaluate the latter to the problem $E, (Lift(r), m) \models_k^v \beta$, whose determination involves only temporal steps.

Thus, we obtain the following approach to deciding realization: (1) represent a formula as a puff, (2) construct a consistent valuation for the puff, (3) evaluate the temporal skeleton of the formula with respect to this valuation and (4) check that the skeleton is valid for all initial states. This approach is formalized in the algorithm in Figure 1. By construction and in accordance with the definition of consistency, the assertion "v is consistent with $(set_0, map_0) \ldots (set_j, map_j)$" is an invariant of the loop of the algorithm. Combining this with the results above we conclude that the algorithm is correct. As presented, the algorithm is not yet fully operational: we still need to show how it is possible to compute $[\beta]$ at steps 2(a) and 3. This can be done in space polynomial in the size of $A_k(E)$ and β using known techniques [SC85].

Theorem 7. *The problem of determining if a formula φ of $\mathcal{L}_{\{\bigcirc, \mathcal{U}, K_1, \ldots, K_n\}}$ is realized in an environment E is decidable in space polynomial in $|\varphi| \cdot \exp(depth(\varphi), O(|E|))$.*

5 Conclusion

It is interesting to note that for each of the languages we have considered, the complexity we have obtained for the realization problem is the same as the complexity obtained by Halpern and Vardi [HV88,HV89] for the validity problem. While there are

INPUT: a finite environment E and a formula φ
OUTPUT: Y if φ is realized in E, N otherwise.

PROCEDURE:

1. Let $k := depth(\varphi)$ and suppose $cep(\varphi) = (set_0, map_0) \dots (set_k, map_k)$.
   Let $v := \emptyset$.
2. For $j := 0$ to $k-1$ do:
   (a) For all $\beta \in set_j$, let $[\beta] :=$
   $\{ w \in \mathcal{T}_j : (e,0) \models_j^v \beta$ for every fair execution e of $A_j(E)$ with $e(0) = w \}$.
   (b) For all $x \in Var(set_{j+1})$, if $map_{j+1}(x) = (K_i, \beta)$,
   let $[x] := \{w \in \mathcal{T}_{j+1} : w' \in [\beta]$ for all i-children w' of $w\}$.
   (c) Let $v := v \cup (\bigcup_{x \in Var(set_{j+1})} \{(x,[x])\})$.
3. For temporal skeleton β of the formula φ let $[\beta] :=$
   $\{ w \in \mathcal{T}_k : (e,0) \models_k^v \beta$ for every fair execution e of $A_k(E)$ with $e(0) = w \}$.
4. If $w \in [\beta]$ for all $w \in \mathcal{I}_k$ then output(Y) else output(N).

Fig. 1. An algorithm for realization

some commonalities in the proof techniques used, we are not aware of any straightforward reductions between the two problems.

The problem we have studied here, of checking whether a formula is *realized* in a given environment, is closely related to a problem studied by van der Meyden and Vardi [MV98]. This work also concerns the synchronous perfect recall semantics. They deal with a notion of environment in which agents are able to choose their actions based on their local state, and the choice of action determines the state transitions. They consider the *realizability* question, of whether it is possible to decide the existence of (and if so, construct) a protocol (a function from local state to actions) for an agent such that running this protocol in a given environment generates a system realizing a given specification in the logic of knowledge and time. By contrast with our results in this paper, however, realizability is decidable only in the single agent case, even for environments with trivial acceptance condition.

Model checking a logic of knowledge and time has also been studied by Clarke et al. [CJM98] in the context of verification of cryptographic protocols. Their work assumes a semantics of knowledge very different from ours, based on explicit computation rather than the information theoretic notion we have studied. An interesting topic for further research is the applicability of our results, or adaptations thereof, to verification of cryptographic protocols.

Several dimensions of generalisation of our results are worth considering. In particular, it would be desirable to know if our results generalise to the case of languages with past time or branching time operators — this generalisation is able to express the problem of checking that a given finite state protocol implements a given *knowledge based program* [FHMV95,FHMV97] in a given environment, in such a way that agents operate as if they had perfect recall. A knowledge based program is a type of specification that describes how an agent's actions relate to its state of knowledge. Verification of knowledge based programs for finite state definitions of knowledge has been shown decidable by Vardi [Var96]. The perfect recall case remains to be addressed, although the linear time case we have presented is already able to yield this result in the special case of *deterministic* knowledge-based programs [MV98]. Also of interest is the complexity of realization with respect to other natural definitions of knowledge, such as the *asynchronous* perfect recall semantics.

References

[Büc60] J.R. Büchi. On a decision method in restricted second order arithmetic. In *Proc. Internat. Congr. on Logic, Methodology and Philosophy of Science*, pages 1–11, Stanford, CA, 1960. Stanford Univ. Press.

[CGP99] E.M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, Cambridge, MA, 1999.

[CJM98] E. Clarke, S. Jha, and W. Marrero. A machine checkable logic of knowledge for specifying security properties of electronic commerce protocols. In *LICS Workshop on Formal Methods and Security Protocols*, 1998.

[FHMV95] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.

[FHMV97] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. Knowledge-based programs. *Distributed Computing*, 10(4):199–225, 1997.

[HM90] J. Y. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37(3):549–587, 1990.

[HV88] J. Y. Halpern and M. Y. Vardi. The complexity of reasoning about knowledge and time: synchronous systems. Research Report RJ 6097, IBM, 1988.

[HV89] J. Y. Halpern and M. Y. Vardi. The complexity of reasoning about knowledge and time, I: lower bounds. *Journal of Computer and Systems Science*, 38(1):195–237, 1989.

[HV91] J. Y. Halpern and M. Y. Vardi. Model checking vs. theorem proving: a manifesto. In V. Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation (Papers in Honor of John McCarthy)*, pages 151–176. Academic Press, San Diego, Calif., 1991.

[Mey74] A. R. Meyer. The inherent complexity of theories of ordered sets. In *Proc. of the Int. Congr. of Mathematics*, volume 2, pages 477–482, Vancouver, 1974. Canadian Mathematical Congress.

[Mey98] R. van der Meyden. Common knowledge and update in finite environments. *Information and Computation*, 140(2):115–157, 1998.

[MP91] Z. Manna and A. Pnueli. *The Temporal logic of Reactive and Concurrent Systems*. Springer-Verlag, Berlin, 1991.

[MV98] R. van der Meyden and M. Y. Vardi. Synthesis from knowledge-based specifications. In *Proc. CONCUR'98, 9th International Conf. on Concurrency Theory*, pages 34–49. Springer LNCS No. 1466, 1998.

[SC85] A. P. Sistla and E. M. Clark. The complexity of propositional linear temporal logic. *Journal of the ACM*, 32(3):733–749, 1985.

[Tho92] W. Thomas. Infinite trees and automaton-definable relations over ω-words. *Theoretical Computer Science*, 103:143–159, 1992.

[Var96] M. Y. Vardi. Implementing knowledge-based programs. In *Proc. of the Conf. on Theoretical Aspects of Rationality and Knowledge*, pages 15–30, San Mateo, CA, 1996. Morgan Kaufmann.