Model Checking Knowledge and Linear Time: PSPACE Cases

Kai Engelhardt*, Peter Gammie, and Ron van der Meyden*

School of Computer Science and Engineering, The University of New South Wales, and National ICT Australia**, Sydney, NSW 2052, Australia,
{kaie|peteg|meyden}@cse.unsw.edu.au

Abstract. We present a general algorithm scheme for model checking logics of knowledge, common knowledge and linear time, based on simulations to a class of structures that capture the way that agents update their knowledge. We show that the scheme leads to PSPACE implementations of model checking the logic of knowledge and linear time in several special cases: perfect recall systems with a single agent or in which all communication is by synchronous broadcast, and systems in which knowledge is interpreted using either the agents' current observation only or its current observation and clock value. In all these results, common knowledge operators may be included in the language. Matching lower bounds are provided, and it is shown that although the complexity bound matches the PSPACE complexity of the linear time temporal logic LTL, as a function of the model size the problems considered have a higher complexity than LTL.

1 Introduction

The logic of knowledge [5] has been proposed as a formalism to express information theoretic properties in distributed and multi-agent systems, and has been shown to be useful for the analysis of distributed systems protocols [9], information flow security properties [10, 21, 24], as well as for problems such as diagnosis and recoverability [3, 4].

The semantics for knowledge operators can be defined in a variety of ways, depending on what information agents use when computing what they know. At one extreme (the "observational semantics") agents rely only on their current observation, at the other (the "synchronous perfect recall semantics") agents rely on the log of all their past observations. In between lies a "clock semantics" in which agents rely on their current observation plus a clock value. These

* Work was partially supported by ARC Discovery Grant RM02036.

** National ICT Australia is funded through the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian Research Council. The third author thanks the Computer Science Department, Courant Institute, New York University for their hospitality in hosting a sabbatical visit during which this work was conducted.

semantics have different motivations: the perfect recall semantics is most appropriate for security analyses and derivation of protocols that make optimal use of information; the other semantics are closer to system implementations.

A number of model checkers for the logic of knowledge have been recently been developed, which embody different choices of semantics for the knowledge operators and different types of expressiveness for the temporal dynamics. MCMAS [13] deals with the observational interpretation of knowledge and the branching time logic CTL. DEMO [27] deals with the dynamic logic based "update logic" [1], which handles what is in effect the perfect recall semantics for knowledge. The system MCK [7] covers a broad spectrum of definitions of knowledge (observational, clock, perfect recall), as well as dealing with both linear time and branching time temporal logic.

Where they deal with the perfect recall semantics for knowledge, these systems place severe constraints on the interaction between knowledge and temporal operators, for reasons of inherent complexity. The complexity of model checking the combination of the linear time temporal logic LTL with knowledge operators interpreted according to the perfect recall semantics has been studied by van der Meyden and Shilov [23], who show that this problem is decidable but with a non-elementary lower bound, and undecidable when operators for common knowledge (a type of fixpoint over knowledge operators) are added to the language. (Shilov et al [17, 18, 8] have also studied branching time versions of these results.)

However, as we show in this paper, this general result does not preclude the existence of special cases in which this model checking problem has lower complexity, even when common knowledge operators are included in the language. We identify a number of cases where the problem (including common knowledge) is solvable in PSPACE. These include systems with a single agent (discussed in Section 5.1) and systems in which all communication is by synchronous broadcast (treated in Section 5.2) The result concerning a single agent improves the nonelementary upper bound for the single agent case obtained from the algorithm of van der Meyden and Shilov.

Our approach to the proof of these results is by means of a general algorithm scheme (presented in Section 4) that relies upon the existence of a simulation from the (in effect, infinite) systems being checked to a finite structure that represents the way that agents update their knowledge in the system. In addition to the results about the perfect recall semantics, we show that this scheme can be used to obtain PSPACE complexity results for model checking the logic of knowledge and linear time for other interpretations for knowledge: in particular, we show that this complexity bound applies in the case of both the clock semantics and the observational semantics (see Section 6).

As the complexity of model checking the linear time temporal logic LTL alone is already PSPACE-complete, it may seem from these results that the extra expressiveness of the logic of knowledge in these cases comes at no extra cost. In fact, we show that there is a sense in which these model checking problems are harder than model checking LTL alone, by focussing on the complexity of model checking a fixed formula as a function of the size of the model. For LTL, this "model complexity" is linear-time for each formula [11]. We show that the model complexity can be as high as PSPACE-complete once the formula includes knowledge operators.

2 Basic Definitions

In this section we define the semantic framework with respect to which we study the model checking problem. The definitions closely follow [23], which dealt with model checking knowledge and linear time in multi-agent systems for a "perfect recall" interpretation of knowledge. We also define an alternate "clock" interpretation of knowledge, in which agents reason on the basis of their current observation and knowledge of the time.

Let *Prop* be a set of atomic propositional constants, $n > 0$ be a natural number, and let $\mathbb{A} = \{1, \ldots, n\}$ be a set of agents. We will be concerned with model checking a propositional multi-modal language for knowledge and linear time based on the set *Prop* of atomic propositional constants, with formulae generated by the modalities $\bigcirc$ (next), U (until), a knowledge operator K_i for each agent $i \in \mathbb{A}$, and a common knowledge operator C_G for each group of agents $G \subseteq \mathbb{A}$. 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 $\neg\varphi$, $\varphi \wedge \psi$, $\bigcirc\varphi$, $\varphi \,\mathsf{U}\, \psi$, $K_i\varphi$ and $C_G\varphi$ for each $i \in \mathbb{A}$ and group $G \subseteq \mathbb{A}$. We write $\mathcal{L}_{\{\bigcirc, \mathsf{U}, K_1, \ldots, K_n, C\}}$ 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, $\mathcal{L}_{\{\bigcirc, \mathsf{U}, K\}}$ refers to the sublanguage with just a single agent (in which case we may drop the subscript on the knowledge operator). As usual in temporal logic, we use the abbreviations $\Diamond\varphi$ for $\mathbf{true}\,\mathsf{U}\,\varphi$, and $\Box\varphi$ for $\neg\Diamond\neg\varphi$. The *knowledge depth* of a formula φ, denoted $\mathit{depth}(\varphi)$, is defined to be the maximal depth of nesting of K operators in φ. For example, $\mathit{depth}(K(p \wedge \neg Kq)) = 2$.

The semantics of this language is defined with respect to the following class of structures. Define an *interpreted environment (for* $\mathbb{A}$*)* to be a tuple E of the form $(S, I, \rightarrow, (O_i)_{i \in \mathbb{A}}, \pi, \alpha)$ where the components are as follows:

1. S is a set of *states* of the environment,
2. I is a subset of S, representing the possible *initial states*,
3. $\rightarrow \subseteq S^2$ is a *transition relation*,
4. for each $i \in \mathbb{A}$ the component $O_i : S \longrightarrow \mathcal{O}$, where $\mathcal{O}$ is a set of uninterpreted observations, is called the *observation function of agent* i,
5. $\pi : S \longrightarrow \mathcal{P}(\mathit{Prop})$ is an *interpretation*,
6. $\alpha \subseteq S$ is an *acceptance condition.*

Intuitively, an environment is a transition system where states encode values of local variables, messages in transit, failure of components, etc. For states s, s' the relation $s \rightarrow s'$ means that if the system is in state s, then at the next tick of the clock it could be in state s'. We call E finite whenever S is. 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 π maps a state s to the set of propositional constants in *Prop* that hold at s. The acceptance conditions are essentially Büchi conditions which model fairness requirements on evolutions of the environment.

A *path* p of E from a state s in S is a finite or infinite sequence of states $s_0 s_1 \ldots$ such that $s_0 = s$ and $s_j \rightarrow s_{j+1}$ for all j. We write $p(m)$ for s_m when m is an index of p. A path p is said to be *initialized* if $p(0) \in I$. We call an initialized finite path a *trace*. A path p is *fair* if it is infinite and $p(i) \in \alpha$ for infinitely many i.[1] We say that the acceptance condition of E is *trivial* if $\alpha = S$. We assume that environments satisfy the following well-formedness condition: for every state s, there exists a fair path with initial state s. A *run* of E is a fair, initialized path, and we write $r[0..m]$ for the trace that is the prefix of run r up to time m. Let *runs*(E) be the set of all runs of E. A *point* of E is a pair (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 approaches used in this paper model a *synchronous perfect-recall*, an *observational*, and a *clock* semantics of knowledge, each defined using a notion of local state. We define the *synchronous perfect recall local state of agent i at a point* (r, m) to be the sequence[2] $\{(r, m)\}_i^{\mathtt{pr}} = O_i(r[0..m])$. That is, the synchronous perfect recall 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. The *clock local state of agent i at a point* (r, m) is defined by $\{(r, m)\}_i^{\mathtt{clk}} = (m, O_i(r(m)))$. That is, in this definition, the agent's local state is taken to be the current time, together with the agent's current observation. Finally, the *observational local state of agent i at a point* (r, m) is $\{(r, m)\}_i^{\mathtt{obs}} = O_i(r(m))$. Effectively, an agent with this view of the world considers any reachable state giving the same observation to be possible. To distinguish these local state assignments, we define a *view* v to be one of the three possibilities `clk`, `obs`, and `pr`.

Given a view v, the corresponding local state assignment may be used to define for each agent i a relation $\overset{v}{\sim}_i$ of *indistinguishability* on points $(r, m), (r', m')$ of E, by $(r, m) \overset{v}{\sim}_i (r', m')$ if $\{(r, m)\}_i^v = \{(r', m')\}_i^v$. Intuitively, when $(r, m) \overset{v}{\sim}_i (r', m')$, agent i's local state according to the view v does not contain enough information for the agent to determine whether it is at one point or the other. Clearly, each $\overset{v}{\sim}_i$ is an equivalence relation. Both the synchronous perfect recall view and the clock view are "synchronous" in the sense that if $(r, m) \overset{v}{\sim}_i (r', m')$,

[1] Note that we do not assume that S is finite, but when so, this formulation is equivalent to the usual formulation of acceptance for Büchi automata: some $s \in \alpha$ occurs infinitely often.

[2] We adopt the convention that functions lift to sequences and sets in a pointwise fashion.

then we must have $m = m'$. Intuitively, this means that the agent "knows the time". The relations $\overset{v}{\sim}_i$ will be used to define the semantics of knowledge for individual agents. By $P_i^v(E, r, m)$ we denote the set

$$\{\ r'(m')\ \mid\ r' \in \mathit{runs}(E),\ m' \in \mathbb{N},\ \{(r', m')\}_i^v = \{(r', m)\}_i^v\ \}$$

of *possible states for agent* i at point (r, m).

To interpret the common knowledge operators, we use another relation. If $G \subseteq \mathbb{A}$ is a *group* of agents (i.e., two or more) then we define the relation $\overset{v}{\sim}_G$ on points to be the reflexive transitive closure of the union of all indistinguishability relations $\overset{v}{\sim}_i$ for $i \in G$, i.e., $\overset{v}{\sim}_G = (\bigcup_{i\in G} \overset{v}{\sim}_i)^*$.

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 with respect to a view v, denoted $E, (r, m) \models^v \varphi$, inductively on the structure of φ. The cases for the temporal fragment of the language are standard, and independent of v:

$E, (r, m) \models^v p$	if $p \in \pi(r(m))$, where $p \in \mathit{Prop}$,
$E, (r, m) \models^v \varphi_1 \wedge \varphi_2$	if $E, (r, m) \models^v \varphi_1$ and $E, (r, m) \models^v \varphi_2$,
$E, (r, m) \models^v \neg\varphi$	if not $E, (r, m) \models^v \varphi$,
$E, (r, m) \models^v \bigcirc\varphi$	if $E, (r, m+1) \models^v \varphi$,
$E, (r, m) \models^v \varphi_1 \mathsf{U} \varphi_2$	if there exists $m'' \geq m$ such that $E, (r, m'') \models^v \varphi_2$ and $E, (r, m') \models^v \varphi_1$ for all m' with $m \leq m' < m''$.

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

$E, (r, m) \models^v K_i\varphi$	if $E, (r', m') \models^v \varphi$ for all points (r', m') of E satisfying $(r', m') \overset{v}{\sim}_i (r, m)$
$E, (r, m) \models^v C_G\varphi$	if $E, (r', m') \models^v \varphi$ for all points (r', m') of E satisfying $(r', m') \overset{v}{\sim}_G (r, m)$

These definitions can be viewed as an instance of the "interpreted systems" framework for the semantics of the logic of knowledge proposed in [9]. 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 [5] for further motivation and background.

3 Main Results

We may now define the model checking problem we consider in this paper and state our main results.

Say that formula φ is *realized* in the environment E with respect to a view v, denoted $E \models^v \varphi$, if, for all runs r of E, we have $E, (r, 0) \models^v \varphi$. We are interested in the following problem, which we call the *realization problem* with respect to a view v: given an an environment E and a formula φ of a language $\mathcal{L}$, determine if φ is realized in E with respect to v.

The realization problem for the logic of knowledge and linear time has been studied by van der Meyden and Shilov [23], who show that for the perfect recall view, the problem is undecidable for the language $\mathcal{L}_{\{\bigcirc, \mathsf{U}, K_1,\ldots,K_n,C\}}$ when $n \geq 2$, and decidable for the language $\mathcal{L}_{\{\bigcirc, \mathsf{U}, K_1,\ldots,K_n\}}$, but with nonelementary complexity. More specifically, for $\mathcal{L}_{\{\bigcirc, \mathsf{U}, K_1,\ldots,K_n\}}$ their approach runs in space polynomial in $f(\mathit{depth}(\varphi), O(|E|))$, where the function f is defined by $f(0, m) = m$ and $f(k+1, m) = 2^{f(k,m)}$. It is also shown by van der Meyden and Shilov that there is a similar lower bound on the complexity when there is more than one agent.

Our main contribution in this paper is to develop a general algorithm scheme for model checking the logic of knowledge and time based on a notion of simulation of environments , and to show that this scheme yields improved complexity bounds in a number of special cases. The scheme itself is presented in Section 4, and parameterizes a procedure for model checking with respect to the observational view. In particular, this procedure yields the following result for the observational view.[3]

Theorem 1. *The problem of determining whether a given formula in the language $\mathcal{L}_{\{\bigcirc, \mathsf{U}, K_1,\ldots,K_n,C\}}$ is realized in a given environment E with respect to the observational view is decidable in PSPACE.*

By showing the existence of simulation from an enviroment representing the perfect recall semantics for a single agent to a suitable finite environment, we obtain the following result:

Theorem 2. *The problem of determining whether a given formula in $\mathcal{L}_{\{\bigcirc, \mathsf{U}, K\}}$ is realized in a given environment E with respect to the perfect recall view is in PSPACE.*

This shows that the complexity of the realization problem for formulae with a single agent with perfect recall is strictly lower than the general case, and significantly improves upon the complexity bound of van der Meyden and Shilov in this case.

By finding other suitable structures we may derive complexity bounds on several other cases of the realization problem, as stated in the following results. First, although with respect to the perfect recall view, the realization problem is non-elementary for $\mathcal{L}_{\{\bigcirc, \mathsf{U}, K_1,\ldots,K_n\}}$, there exist classes of environments with respect to which the problem has lower complexity, even if we add the common knowledge operators. In particular, this holds for *broadcast environments* [22].

[3] This result does not appear to have been previously stated in the literature, but we note that results of Vardi [28] on the problem of verifying that a concrete protocol implements a knowledge-based program are very closely related. Lomuscio and Raimondi have studied the complexity of model checking the combination of the logic of knowledge with the braching time logic CTL with respect to the observational semantics [12].

Intuitively, these are environments in which the only communication mechanism available to agents is to broadcast to *all* agents in the system. The formal definition will be given in section 5.2. For broadcast environments we show the following.

Theorem 3. *The problem of determining whether a given formula in the language $\mathcal{L}_{\{\bigcirc, \mathsf{U}, K_1,\ldots,K_n,C\}}$ is realized in a given broadcast environment E with respect to the perfect recall view is decidable in PSPACE.*

Realization for the clock view may also handled using the simulation technique and again the common knowledge operator may be included in the language.

Theorem 4. *The problem of determining whether a given formula in the language $\mathcal{L}_{\{\bigcirc, \mathsf{U}, K_1,\ldots,K_n,C\}}$ is realized in a given environment E with respect to the clock view is decidable in PSPACE.*

Note that the complexity of model checking linear time temporal logic (i.e. realization for the language $\mathcal{L}_{\{\bigcirc, \mathsf{U}\}}$) is PSPACE-complete [19]. Since $\mathcal{L}_{\{\bigcirc, \mathsf{U}\}}$ is a sublanguage of the languages in the above results, these results show that the above bounds are tight, in the sense that the problems are in fact PSPACE-complete.

That some of our complexity bounds are no more than the PSPACE complexity of the linear time temporal logic LTL may at first suggest that model checking these cases of the logic and knowledge and time could be as effective in practice as model checking LTL. However, a closer inspection indicates that it is not obvious that this will be case. The time complexity of LTL model checking a *fixed* formula is linear in the size of the model. The time complexity is exponential in the size of the formula. This exponential bound is not an impediment in practice since the formulas of interest tend to be small. The models, on the other hand, may be very large. We show that as a function of model size, the complexity of model checking fixed formulas of the logic of knowledge and time falling within our PSPACE cases can be be as high as PSPACE-hard (for $\mathcal{L}_{\{\bigcirc, \mathsf{U}, K\}}$ with respect to perfect recall) and at any level of the polynomial hierarchy for the clock view.[4]

Theorem 5. *There exists a formula φ of $\mathcal{L}_{\{\bigcirc, \mathsf{U}, K\}}$ such that the problem of deciding if φ is realized in a given environment E with respect to the perfect recall view is PSPACE-hard.*

In the case of the clock semantics, we may obtain the following lower bound.

Theorem 6. *For each level Π_k^p of the polynomial hierarchy, there exists a formula φ of $\mathcal{L}_{\{\bigcirc, \mathsf{U}, K_1,\ldots,K_n\}}$ such that the problem of deciding, given an environment E, whether $E \models^{\mathtt{clk}} \varphi$, is Π_k^p-hard.*

The proof involves a reduction from Π_k formulas of quantified boolean logic. Note that this implies PSPACE-hardness of the version of the problem in which the formula is given.

[4] For reasons of space most proofs have been relegated to the appendix.

4 An Algorithm Scheme

We approach the model checking problem in three stages. Firstly, given a finite environment E and a view v, we construct an infinite environment E^v that reduces the model checking problem with respect to v in E to one of model checking E^v with respect to $\mathtt{obs}$. Secondly, we introduce simulations between environments, which, together with the previous step, may enable the problem of model checking E with respect to v to be reduced to model checking with respect to $\mathtt{obs}$ in finite state environment E' (which is of exponential size in our applications). Finally, we combine alternating Turing machine techniques with standard Büchi automata techniques to obtain the general model checking procedure (which runs in PSPACE in our applications).

Let $E = (S, I, \to, (O_i)_{i\in\mathbb{A}}, \pi, \alpha)$ be a finite environment and let v be a view. Define E^v, the *v-environment for E*, to be $(S^v, I^v, \to^v, (O_i^v)_{i\in\mathbb{A}}, \pi^v, \alpha^v)$ where:

- $S^v = \mathit{runs}(E) \times \mathbb{N}$,
- $I^v = \mathit{runs}(E) \times \{0\}$,
- $(r, m) \to^v (r', m')$ if $r' = r$ and $m' = m + 1$,
- $O_i^v(r, m) = \{(r, m)\}_i^v$,
- $\pi^v(r, m) = \pi(r(m))$, and
- $(r, m) \in \alpha^v$ iff $r(m) \in \alpha$.

The following lemma states that the observational view on this (infinite) environment coincides with the view v on the original (finite) environment. Given a run r of E, write r^v for the run of E^v defined by $r^v(n) = (r, n)$ for all $n \in \mathbb{N}$.

Lemma 7. *Let $\varphi \in \mathcal{L}_{\{\bigcirc,\mathsf{U},K_1,\dots,K_n,C\}}$ and let (r, m) be a point of E. Then $E, (r, m) \models^v \varphi$ iff $E^v, (r^v, m) \models^{\mathsf{obs}} \varphi$.*

Since every run of E^v has the form r^v for some run of E, it follows that $E, \models^v \varphi$ iff $E^v \models^{\mathsf{obs}} \varphi$.

Let $\mathit{fpaths}(E)$ be the set of all fair paths of E. For $\rho \in \mathit{fpaths}(E)$ and $m \in \mathbb{N}$ let $\rho|_m$ be the fair path with $\rho|_m(j) = \rho(m + j)$, for $j \in \mathbb{N}$.

Observe that the semantics of $E, (r, n) \models^v \varphi$ refers only to the future of the points considered in unfolding the definition. To formalise this, consider the following alternate definition of a relation $E, \rho \models^* \varphi$, defined for all $\rho \in \mathit{fpaths}(E)$, not just the initialized ones:

$E, \rho \models^* p$	if $p \in \pi(\rho(0))$, where $p \in \mathit{Prop}$,
$E, \rho \models^* \varphi_1 \wedge \varphi_2$	if $E, \rho \models^* \varphi_1$ and $E, \rho \models^* \varphi_2$,
$E, \rho \models^* \neg\varphi$	if not $E, \rho \models^* \varphi$,
$E, \rho \models^* \bigcirc\varphi$	if $E, \rho|_1 \models^* \varphi$,
$E, \rho \models^* \varphi_1 \,\mathsf{U}\, \varphi_2$	if there exists $m'' \geq 0$ such that $E, \rho|_{m''} \models^* \varphi_2$ and $E, \rho|_{m'} \models^* \varphi_1$ for all m' with $0 \leq m' < m''$.
$E, \rho \models^* K_i\varphi$	if $E, \rho' \models^* \varphi$ for all $\rho' \in \mathit{fpaths}(E)$ with $O_i(\rho'(0)) = O_i(\rho(0))$
$E, \rho \models^* C_G\varphi$	if for all sequences of states $s_0, s_1, \dots, s_k$ such that (i) $s_0 = \rho(0)$, (ii) for all $j < k$ there exists an $i \in G$ such that $O_i(s_j) = O_i(s_{j+1})$, and (iii) for all $\rho' \in \mathit{fpaths}(E)$ with $\rho'(0) = s_k$, we have $E, \rho' \models^* \varphi$.

We write $E \models^* \varphi$ if $E, r \models^* \varphi$ for all runs r of E.

For an environment E, define a state to be reachable if it occurs in some run of E. Say that *observations in E preserve reachability* if for all states s, t of E and all agents i, if s is reachable and $O_i(s) = O_i(t)$ then t is reachable.[5]

Lemma 8. *If observations in E preserve reachability then $E, (r, m) \models^{\mathsf{obs}} \varphi$ iff $E, r|_m \models^* \varphi$.*

Next, we introduce a notion of simulation on environments (cf. [15]) in order to reduce the infinite state space of E^v to a finite one while preserving validity of formulae with respect to `obs`. For environments $E = (S, I, \rightarrow, (O_i)_{i \in \mathbb{A}}, \pi, \alpha)$ and $E' = (S', I', \rightarrow', (O'_i)_{i \in \mathbb{A}}, \pi', \alpha')$, a function $\sigma : S \longrightarrow S'$ is said to be a *simulation* from E to E' if the following hold:

1. $I' = \sigma(I)$,
2. if $s \rightarrow s'$ then $\sigma(s) \rightarrow' \sigma(s')$,
3. if $\sigma(s) \rightarrow' u$ then there exists $s' \in S$ such that $\sigma(s') = u$ and $s \rightarrow s'$,
4. if $O_i(s) = O_i(t)$ then $O'_i(\sigma(s)) = O'_i(\sigma(t))$,
5. if $O'_i(\sigma(s)) = O'_i(u)$ then there exists a state $t \in S$ such that $O_i(s) = O_i(t)$ and $\sigma(t) = u$.
6. $\pi' \circ \sigma = \pi$, and
7. $\sigma(s) \in \alpha'$ iff $s \in \alpha$.

Lemma 9. *Suppose that σ is a simulation from E to E'. Then*

1. *for all (initialised) $\rho \in fpaths(E)$, $\sigma(\rho)$ is a (initialised) fair path of E';*
2. *for all $\rho' \in fpaths(E')$ and (initial) states s of E, if $\sigma(s) = \rho'(0)$, then there exists a (initialised) $\rho \in fpaths(E)$ with $\rho(0) = s$ such that $\sigma(\rho) = \rho'$;*
3. *for all $\rho \in fpaths(E)$ we have $E, \rho \models^* \varphi$ iff $E', \sigma(\rho) \models^* \varphi$.*

Noting that all states of E^v are reachable, we obtain the following:

Corollary 10. *For all environments E and E', if there exists a simulation from E^v to E', then $E \models^v \varphi$ iff $E' \models^* \varphi$.*

This result provides the basic reduction that we use to obtain our complexity results. We now show that the relation $E' \models^* \varphi$ is decidable for finite environments E'. However, we will need to deal with the fact that the structure E' will be of size exponential in the size of E in our applications. For this reason, we express our decision procedure for $\models^*$ as an alternating computation [2], in which we guess and verify the components of E'.

We begin with a reduction to well-known techniques for LTL. Say that a formula is a *pure knowledge formula* if it is of the one of the forms $K_i\psi$ or $C_G\psi$, or their negation. Note that for formulas φ that are either atomic propositions

[5] We remark that it is always possible to ensure this by deleting the unreachable states from E, an operation that preserves satisfaction of formulas. However, this operation is undesirable in our applications since we will deal with exponential size structures, in which observations already preserve reachability.

or their negation, or pure knowledge formulas, we have that if $\rho(0) = \rho'(0)$, then $E, \rho \models^* \varphi$ iff $E, \rho' \models^* \varphi$. Thus, for such formulas φ, we may define $E, s \models^* \varphi$, where s is a state of E, to hold if $E, \rho \models^* \varphi$ for some (equivalently, every) path ρ with $\rho(0) = s$.

We may use this state-depence property to transform the $\mathcal{L}_{\{\bigcirc, \mathsf{U}, K_1,\ldots,K_n, C\}}$ model checking problem with respect to $\models^*$ into a problem of model checking $\mathcal{L}_{\{\bigcirc, \mathsf{U}\}}$, by replacing the pure knowledge subformulas by atomic propositions. Introduce a new atomic proposition $q_{K_i\psi}$ for each formula $K_i\psi$ and $q_{C_G\psi}$ for each formula $C_G\psi$. Let $\mathcal{L}^*_{\{\bigcirc, \mathsf{U}\}}$ be the language of temporal logic over the set of atomic propositions *Prop* together with these new atomic propositions. Given a formula φ of $\mathcal{L}_{\{\bigcirc, \mathsf{U}, K_1,\ldots,K_n, C\}}$ and an occurrence of a pure knowledge formula as a subformula of φ, say this occurrence is *maximal* if it does not lie within the scope of a knowledge or common knowledge operator. For example, in $(K_2 \bigcirc K_1 p) \vee K_1 p$, the maximal occurrences of knowledge subformulas are the occurrence of $K_2 \bigcirc K_1 p$ and the second (but not the first) occurrence of $K_1 p$. Define φ^* to be the formula of $\mathcal{L}^*_{\{\bigcirc, \mathsf{U}\}}$ obtained by replacing each maximal occurrence of a knowledge formula $K_i\psi$ by the proposition $q_{K_i\psi}$ and similarly for the maximal occurrences of $C_G\psi$. Thus, $((K_2 \bigcirc K_1 p) \vee K_1 p)^* = q_{K_2 \bigcirc K_1 p} \vee q_{K_1 p}$. Write $\mathit{Prop}_{\varphi^*}$ for the set of atomic propositions occuring in φ^* and $\mathit{KProp}_{\varphi^*}$ for the set of atomic propositions of the form $q_{K_i\psi}$ and $q_{C_G\psi}$ that occur that occur in φ^*.

Suppose we enrich the structure E by extending the valuation π so that $q_{K_i\psi} \in \pi(s)$ iff $E, s \models^* K_i\psi$ and $q_{C_G\psi} \in \pi(s)$ iff $E, s \models C_G\psi$. Call the resulting structure E^*. Then we have $E, \rho \models^* \varphi$ iff $E^*, \rho \models \varphi^*$. This turns the problem of model checking $\mathcal{L}_{\{\bigcirc, \mathsf{U}, K_1,\ldots,K_n, C\}}$ in E into the problem of model checking $\mathcal{L}^*_{\{\bigcirc, \mathsf{U}\}}$ in E^*. Of course, to apply this technique, we need to have the appropriate extension E^* of E. We may deal with this in an NPSPACE computation by *guessing* the extension E^*, iteratively verifying its correctness over larger and larger pure knowledge subformulas of φ (using LTL model checking techniques), and then model checking the formula φ^*. Since NPSPACE = PSPACE, this already yields a proof of Theorem 1.[6]

However, in our applications, we will not be interested in a given structure E, but in a structure E' of size exponential in the size of E. This means that the cost of guessing $(E')^*$ is exponential. We will handle this by guessing the extension not upfront, but on the fly, for each state of E' as it arises during the verification, and using and APTIME computation that incorporates a Büchi automaton emptiness check for the LTL parts of the verification.

Let $\mathcal{M}_{\varphi^*}$ be the nondeterministic Büchi automaton for the $\mathcal{L}^*_{\{\bigcirc, \mathsf{U}\}}$ formula φ^* over propositions $\mathit{Prop}_{\varphi^*}$, with states S_{φ^*}, initial states I_{φ^*}, transitions $\Rightarrow_a$ (where $a \in \mathcal{P}(\mathit{Prop}_{\varphi^*})$) and acceptance condition α_{φ^*}. We make use of the following properties of this automaton [29]: (1) The automaton is of size $O(2^{|\varphi^*|})$, where each state is of size $O(|\varphi|)$. (2) Deciding S_{φ^*}, I_{φ^*}, $\Rightarrow_a$, and α_{φ^*} can be done in ATIME($\log_2 |\varphi|$).

[6] The guess and verify technique discussed here is essentially that used in Vardi's results on verifying implementations of knowledge-based programs [28].

For a finite environment $E = (S, I, \rightarrow, (O_i)_{i \in \mathbb{A}}, \pi, \alpha)$, we define the product $E \times \mathcal{M}_{\varphi^*}$ (a transition system with Büchi acceptance condition) as follows.

- The transition system has states $\langle b, s, v \rangle$, where $b \in \mathcal{P}(\{0,1\})$, $s \in S$ and $v \in S_{\varphi^*}$. Intuitively, $0 \in b$ ($1 \in b$) represents that E (resp., $\mathcal{M}_{\varphi^*}$), has passed through an accepting state since the most recent accepting state of the product.
- The set of initial states consists of all $\langle \emptyset, s, v \rangle$ where $s \in I$ and $v \in I_{\varphi^*}$.
- There is a transition $\langle b, s, v \rangle \Rightarrow_k \langle b', s', v' \rangle$ for a set $k \subseteq \mathit{KProp}_{\varphi^*}$ when:
  - $s \rightarrow s'$,
  - $v \Rightarrow_{\pi(s) \cup k} v'$, and
  - $b' = b_0 \cup b_1 \cup b_2$, where if $b = \{0,1\}$ then $b_0 = \emptyset$, else $b_0 = b$; if $s \in \alpha$ then $b_1 = \{0\}$, else $b_1 = \emptyset$; and if $v \in \alpha_{\varphi^*}$ then $b_2 = \{1\}$, else $b_2 = \emptyset$;
- the automaton has as accepting states the states $\langle b, s, v \rangle$ with $b = \{0,1\}$.

Intuitively, this transition system represents running $\mathcal{M}_{\varphi^*}$ as a monitor on runs of E, with the values of the propositions $\mathit{KProp}_{\varphi^*}$ chosen arbitrarily. Thus, there exists a fair path $\rho = s_0 s_1 \ldots$ of E such that $E, \rho \models^* \varphi$ iff there exists an accepting run $\langle b_0, s_0, v_0 \rangle \Rightarrow_{k_0} \langle b_1, s_1, v_1 \rangle \Rightarrow_{k_1} \langle b_2, s_2, v_2 \rangle \Rightarrow_{k_2} \ldots$ of $E \times \mathcal{M}_{\neg\varphi^*}$ such that for all $j \geq 0$, we have $E, s_j \models^* k_j$. Applying the usual emptiness check for Büchi automata, such a path exists iff we can find a finite such sequence with $\langle b_l, s_l, v_l \rangle$ an accepting state and final element $\langle b_{l'}, s_{l'}, v_{l'} \rangle = \langle b_l, s_l, v_l \rangle$ for some $l' > l$, where both l and $l' - l$ are at most $|E \times \mathcal{M}_{\varphi^*}|$. Our decision procedure searches for such paths using a Savitch-style reachability procedure [16] in order to deal with the exponential size of the search-space.

For the verification that $E, s \models^* k$, it suffices to check, for each maximal knowledge subformula $K_i \psi$ of φ, that $q_{K_i \psi} \in k$ iff $O_i(s) = O_i(t)$ implies that for all fair paths $\rho = t_0 t_1 \ldots$ with $t_0 = t$, we have $E, \rho \models \varphi^*$. For this, we recursively apply the above ideas on $E \times \mathcal{M}_{\neg\psi^*}$. Since ψ is a strict subformula of φ, the recursion is well founded. A similar check is applied for the common knowledge subformulas.

We are now ready to present our general algorithm scheme as an alternating computation [2]. Suppose that we are given a finite environment E, for which it is known that there exists a simulation from E^v to a finite environment $E' = (S', I', \rightarrow', (O'_i)_{i \in \mathbb{A}}, \pi', \alpha')$. We assume that there is a representation of E' such that the states and other components of E' can be represented and verified within known space and alternating time complexity bounds. (That is, given E, the states of E' are representable as strings of length some known function of $|E|$, in such a way that we can decide whether such a string represents a state of E', whether $s \rightarrow' s'$ etc.with some known complexity bounds.) We define the following alternating procedure that searches for such runs by operating over the states $\langle b, s, v \rangle$ of the automata $E' \times \mathcal{M}_{\psi^*}$ for subformulas ψ of φ and their negations. For clarity, we write expressions referring to the components of E' (such as "choose $s \in I'$ and do X") which need to be expanded to expressions ("choose s and universally (1) verify $s \in I'$ and (2) do X") that use the verification routines assumed to exist.

```
VERIFY($E, \varphi$): Universally choose $s \in I'$ and call $\neg$FALSIFY($E, s, \varphi$)
FALSIFY($E, s, \psi$): Existentially choose $k \subseteq KProp_{\psi^*}$, an initial state $v$ of $\mathcal{M}_{\neg\psi^*}$,
    an accepting state $\langle b_0, s_0, v_0\rangle$ and a state $\langle b_1, s_1, v_1\rangle$ of $E' \times \mathcal{M}_{\neg\psi^*}$ such that
    $(b_0, s_0, v_0) \Rightarrow_k (b_1, s_1, v_1)$.
    Let $N = \lceil log_2 |states(E' \times \mathcal{M}_{\neg\psi^*})|\rceil$
    Universally call:
      1. REACH($E, (\emptyset, s, v), (b_0, s_0, v_0), N, \neg\psi$)
      2. CHECK($E, s_0, k, \psi$), and
      3. REACH($E, (b_1, s_1, v_1), (b_0, s_0, v_0), N, \neg\psi$)
CHECK($E, s, k, \psi$): Universally,
      – for each $p_{K_i\psi'}$ in $KProp_{\psi^*}$, if $p_{K_i\psi'} \in k$ then call KCHECK($E, s, K_i\psi'$) else
        call $\neg$KCHECK($E, s, K_i\psi'$)
      – for each $p_{C_G\psi'}$ in $KProp_{\psi^*}$, if $p_{C_G\psi'} \in k$ then call CKCHECK($E, s, C_G\psi'$)
        else call $\neg$CKCHECK($E, s, C_G\psi'$)
KCHECK($E, s, K_i\psi$): Universally, for each $s' \in S'$ where $O'_i(s) = O'_i(s')$, call
    $\neg$FALSIFY($E, s', \psi$)
CKCHECK($E, s, C_G\psi$): Universally, for each $s' \in S'$: (1) verify[7] there is a sequence
    $s = s_0, \ldots, s_k = s'$ with $k \leq |S'|$ and for each $j < k$ there is an $i \in G$ such
    that $O'_i(s_j) = O'_i(s_{j+1})$. (2) call $\neg$FALSIFY($E, s', \psi$)
REACH($E, (b_0, s_0, v_0), (b_1, s_1, v_1), N, \psi$): Accept if $(b_0, s_0, v_0) = (b_1, s_1, v_1)$.
    Otherwise if $N = 0$, existentially guess $k \subseteq KProp_{\psi^*}$ then
              universally verify that $(b_0, s_0, v_0) \Rightarrow_k (b_1, s_1, v_1)$ and CHECK($E, s_0, k, \psi$).
    If $N > 0$, existentially guess a state $(b_2, s_2, v_2)$ of $E \times \mathcal{M}_{\psi^*}$, then universally
    call:
              REACH($E, (b_0, s_0, v_0), (b_2, s_2, v_2), N - 1, \psi$) and
              REACH($E, (b_2, s_2, v_2), (b_1, s_1, v_1), N - 1, \psi$)
```

An analysis of the complexity of the algorithm scheme yields the following.

Theorem 11. *Let v be a view. Suppose that $\mathcal{C}$ is a class of environments such that for each environment $E \in \mathcal{C}$ there exists an environment E' with states that can be represented in space $f(|E|)$ and components that can be verified in ATIME($g(|E|)$), such that there is a simulation σ from E^v to E'. Then $\{ (E, \varphi) \in \mathcal{C} \times \mathcal{L}_{\{\bigcirc, \mathsf{U}, K_1, \ldots, K_n, C\}} \mid E \models^v \varphi \}$ is in ATIME($p(f(|E|), g(|E|), |\varphi|)$) for some polynomial p.*

We remark that since the procedure REACH() has an alternation before the recursive call, the number of alternations is also polynomial in $|E|$. Theorem 5 can be understood as asserting that this is inherently so.

In the following sections, we apply Theorem 11 to obtain complexity bounds for model checking the logic of knowledge and linear time in a number of cases. In each case, we identify an appropriate environment E' where the states can be represented and verified in polynomial space and time, respectively, hence the complexity of the alternating procedure is APTIME. By [2], this is equivalent

[7] In general, this may require another Savitch-style search. In fact, in our applications, $k \leq |S|$, i.e. the number of states of E, will suffice, so this is not necessary.

to PSPACE. The environments E' and the simulations we use are extensions (by the addition of transition relations $\rightarrow'$) of similar structures that have been used elsewhere in the literature [22] for another problem (existence of finite-state implementations of knowledge-based programs.)

5 Model Checking with Respect to Perfect Recall

In this section we consider several special cases of model checking with respect to perfect recall. The first restricts formulas to refer only to the knowledge of a single agent, and the latter concerns model checking the full language $\mathcal{L}_{\{\bigcirc, \mathsf{U}, K_1, \ldots, K_n, C\}}$ in restricted environments.

5.1 Formulas of $\mathcal{L}_{\{\bigcirc, \mathsf{U}, K\}}$

We first treat the case of model checking formulas of a single agent (agent 1) using the perfect recall view. (This case may also be applied to model checking formulas that refer only to a single agent's knowledge, simply by dropping the other agents' observation functions from the environment.)

In this setting it suffices to track the set of states the agent considers possible at each point in time. We define the environment $E' = (S', I', \rightarrow', O'_1, \pi', \alpha')$ by:

- $S' = \{ (s, P) \mid s \in S, P \subseteq S, s \in P \}$
- $I' = \{ (s, P_0(s)) \mid s \in I \}$ where $P_0(s) = \{ s' \in I \mid O_1(s) = O_1(s') \}$
- $(s, P) \rightarrow' (s', P')$ iff $s \rightarrow s'$ and $P' = \{ t' \mid t \in P, t \rightarrow t', O_1(t') = O_1(s') \}$, and
- $O'_1(s, P) = P$.

with simulation from E^{pr} given by $\sigma(r, m) = (r(m), P_1^{\mathrm{pr}}(E, r, m))$. Observe that a state of E' can be represented in space $O(\log_2 |S| + |S|)$. States of E' can be seen to be a special case (1-trees) of data structures previously used in [14] for model checking $\mathcal{L}_{\{K_1, \ldots, K_n\}}$. That σ is a simulation can be seen by arguments in that work. It is easy to check that observations preserve reachability. By Theorem 11 we conclude that this model checking problem can be decided in PSPACE, which completes the proof of Theorem 2.

5.2 Multi-agent Broadcast and $\mathcal{L}_{\{\bigcirc, \mathsf{U}, K_1, \ldots, K_n, C\}}$ with Perfect Recall Semantics

Broadcast environments [22, 25] model situations in which agents may maintain private information, but where the only means by which this information can be communicated is by synchronous simultaneous broadcast to all agents.

We give a definition of broadcast environments here that is slightly more abstract than previous formulations, which dealt with a notion of environment in which agents are equipped with actions that they may perform.

Formally, we define a broadcast environment to be an environment $E = (S, I, \rightarrow, (O_i)_{i \in \mathbb{A}}, \pi, \alpha)$ in which the states and observation functions and transition relation have a particular structure.

- The set S of states of E is a subset of $S_0 \times S_1 \times \ldots \times S_n$, where S_0 is a finite set S_0 of *shared states*, and for each agent i a set S_i of *private states.* If $s = \langle s_0, \ldots, s_n \rangle$ denotes a state, we write $\mathbf{p}_i(s)$ to denote agent i's private state s_i. For each agent i, define the binary function $\bowtie_i$ on S by $\langle s_0, s_1, \ldots s_n \rangle \bowtie_i \langle t_0, \ldots t_n \rangle = \langle s_0, s_1 \ldots s_{i-1}, t_i, s_{i+1}, \ldots s_n \rangle$. We require that S is closed under the functions $\bowtie_i$.
- The observation functions are given by $O_i(s) = (O_c(s_0), \mathbf{p}_i(s))$, where $O_c : S_0 \longrightarrow \mathcal{O}$ is a *common observation function.*
- The transition relation has the property that for each agent $i = 1 \ldots n$, if $O_i(s) = O_i(t)$ and $s \to s'$ and $t \to t'$ and $O_c(s') = O_c(t')$, then $s \to s' \bowtie_i t'$.
- The Büchi acceptance condition, used to model fairness constraints on the environment, is some arbitrary subset of S.

There is no constraint on the set of initial states. Intuitively, the common observation function models the information that is being broadcast, and the private states model private information that is being maintained by the agents. An agent's observation consists of the broadcast information and its private information. The condition on the transition function can be understood as saying that an agent's choice of update on its private state (1) may depend only on the current observation and the incoming common observation and (2) does not affect the update on the common state or any of the other agent's updates.

Paradigmatic examples of broadcast systems are card games such as bridge (where both bidding and playing of cards can be viewed as a broadcast) and systems composed of processes attached to bus, with all processes receiving every communication (as in "snoopy cache coherence protocols" [20]).

Note that every single agent system E is isomorphic to a broadcast system E'. For, if we represent a state s of E by a state $\langle s, O_1(s) \rangle$ of E', and view the first component as being the shared state and the second component as the private state of the single agent, and take $O_c(s) = O_1(s)$, then the constraint on the transition relation is trivially satisfied, because if $O_1(s) = O_1(t)$ then $(s, O_1(s)) \bowtie (t, O_1(t)) = (s, O_1(s))$.

For the broadcast, perfect recall case, we use the following environment E':

- S' is the set of elements $(s, f, t) \in I \times (I \longrightarrow \mathcal{P}(S)) \times S$ such that $t \in f(s)$,
- $(s, f, t) \in I'$ iff $s \in I$, for $s' \in I$ we have $f(s') = \{ \, t \in I \mid O_c(t) = O_c(s') \, \}$, and $t \in f(s)$,
- $(s, f, t) \to' (s, f', t')$ if $f'(s') = \{ \, u' \mid u \in f(s'), u \to u', O_c(u') = O_c(t') \, \}$ and $t \to t'$,
- $O'_i(s, f, t) = (O_i(s), f, O_i(t))$,

and the simulation σ given by $\sigma(r, m) = (r(0), f, r(m))$, where $f(s)$ is the set of states t such that there exists a trace $r'[0 \ldots m]$ of E with $O_c(r'[0..m]) = O_c(r[0..m])$ and $t = r'(m)$. Observations preserve reachability in this environment. The states of E' can be represented in size $O(\log_2 |S| + |I| \cdot |S| + \log_2 |S|) = O(|S|^2)$, and applying Theorem 11 shows once again that this model checking problem is in PSPACE, completing the proof of Theorem 3. Note also that in this structure, if $(s, f, t) \sim_G^{\mathcal{O}} (s', f', t')$, then it does so by means of a sequence of states all of which have second component f, and also $f' = f$. Thus a maximal path length of $|S|^2$ suffices in CKCHECK().

6 Formulas of $\mathcal{L}_{\{\bigcirc, \mathsf{U}, K_1,\ldots,K_n,C\}}$ for the Clock and Observational Views

In order to model check formulas with respect to the clock view, the image of a point (r, m) in the simulating environment E' needs to keep track of the set of states that are reachable in exactly m steps. We define E' by

- $S' = \{\ (s, P)\ |\ \ s \in S, P \subseteq S, s \in P\ \}$
- $I' = I \times \{I\}$,
- $(s, P) \rightarrow' (s', P')$ if $s \rightarrow s'$ and $P' = \{t' \mid t \in P, t \rightarrow t'\}$.
- $O'_i(s, P) = (O_i(s), P)$

The simulation is given by $\sigma(r, m) = (r(m), \{r'(m) \mid r' \in runs(E)\})$. Observations can be seen to preserve reachability. The states in the constructed environment can be represented in space $O(\log |S| + |S|)$. This problem is again in PSPACE by Theorem 11. This yields a proof of Theorem 4. In this structure, if $(s, P) \sim^{\mathcal{O}}_{G} (s', P')$, then it does so by means of a sequence of states all of which have second component P, and also $P' = P$. Thus a maximal path length of $|S|$ suffices in CKCHECK().

We can already decide realizability in a finite environment E with respect to the observational semantics by furnishing a standard LTL model checker with the equivalence classes induced by the observation function. To remain within our framework, it suffices to use an environment identical to E with simulation $\sigma(r, m) = r(m)$ from E^{obs} to E. Its states can be represented in size $O(\log |S|)$. However, in order for observations to preserve reachability in this case, we need to first remove unreachable states from the environment. Here also a maximal path length of $|S|$ suffices in CKCHECK().

7 Conclusion

We have shown that our general simulation-based scheme for model checking the logic of knowledge and linear time yields PSPACE complexity bounds in a number of interesting cases of the general problem (which has much higher complexity).

Our notion of simulation allows reductions on the temporal structure of environments, but we have not exploited this in our applications. It could be worth exploring this observation in practice. Experiments conducted by Fisler and Vardi [6] suggest that bisimulation reduction is of limited utility for temporal logic model checking, but arguments of van der Meyden and Zhang [26] suggest such reductions might be effective for the much larger search spaces produced when dealing with information flow properties.

The techniques are also applicable to show decidability for certain other classes of environments (with higher complexity bounds). We leave the details for elsewhere. We believe that the techniques we have developed can also be adapted to deal with the combination of branching time and the logic of knowledge: we leave this for future work.

Wozna et al have studied model checking a logic of knowledge and branching time in a real time systems modelled using timed automata [30]. Their semantics is close to our clock semantics, but we note that their until operator is bounded to a specific interval, so the closest appropriate comparison is to our language $\mathcal{L}_{\{\bigcirc,K_1,\dots,K_n,C\}}$. They give decidability but not complexity results, but study bounded model checking techniques for their logic.

References

[1] A. Baltag, L. S. Moss, and S. Solecki. The logic of public announcements, common knowledge, and private suspicions. In *TARK '98: Proceedings of the 7th conference on Theoretical aspects of rationality and knowledge*, pages 43–56, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[2] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, 1981.

[3] A. Cimatti, C. Pecheur, and R. Cavada. Formal verification of diagnosability via symbolic model checking. In G. Gottlob and T. Walsh, editors, *IJCAI*, pages 363–369. Morgan Kaufmann, 2003.

[4] A. Cimatti, C. Pecheur, and A. Lomuscio. Applications of model checking for multi-agent systems: Verification of diagnosability and recoverability. In L. Czaja, editor, *Proceedings of the International Workshop on Concurrency, Specification, and Programming (CS&P 2005).* Ruciane-Nida, 2005.

[5] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge.* MIT-Press, 1995.

[6] K. Fisler and M. Y. Vardi. Bisimulation and model checking. In *Conference on Correct Hardware Design and Verification Methods (CHARME'99)*, pages 338–341, 1999.

[7] P. Gammie and R. van der Meyden. MCK: Model checking the logic of knowledge. In R. Alur and D. Peled, editors, *CAV*, volume 3114 of *LNCS*, pages 479–483. Springer-Verlag, 2004. http://www.cse.unsw.edu.au/˜mck/.

[8] N. O. Garanina and N. V. Shilov. Well-structured model checking of multiagent systems. In *Sixth International Andrei Ershov Memorial Conference Perspectives of System Informatics*, LNCS. Springer-Verlag, 2006. to appear.

[9] J. Y. Halpern and Y. Moses. Knowledge and Common Knowledge in a Distributed Environment. *J. ACM*, 37(3), 1990.

[10] J. Y. Halpern and K. R. O'Neill. Anonymity and information hiding in multiagent systems. In *CSFW*, pages 75–88. IEEE Computer Society, 2003.

[11] O. Lichtenstein and A. Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In *POPL '85: Proceedings of the 12th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, pages 97–107, New York, NY, USA, 1985. ACM Press.

[12] A. Lomuscio and F. Raimondi. The complexity of model checking concurrent programs against CTLK specifications. In H. Nakashima, M. P. Wellman, G. Weiss, and P. Stone, editors, *AAMAS*, pages 548–550. ACM, 2006.

[13] A. Lomuscio and F. Raimondi. MCMAS: A model checker for multi-agent systems. In H. Hermanns and J. Palsberg, editors, *TACAS*, volume 3920 of *LNCS*, pages 450–454. Springer-Verlag, 2006.

[14] R. v. d. Meyden. Common knowledge and update in finite environments. *Information and Computation*, 140(2):115–157, Feb. 1998.

[15] D. M. R. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Theoretical Computer Science: 5th GI-Conference, Karlsruhe*, volume 104 of *LNCS*, pages 167–183. Springer-Verlag, Mar. 1981.

[16] W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *J. Comput. Syst. Sci.*, 4:177–192, 1970.

[17] N. V. Shilov and N. O. Garanina. Model checking knowledge and fixpoints. In Z. Ésik and A. Ingólfsdóttir, editors, *FICS*, volume NS-02-2 of *BRICS Notes Series*, pages 25–39. University of Aarhus, 2002.

[18] N. V. Shilov, N. O. Garanina, and K.-M. Choe. Update and abstraction in model checking of knowledge and branching time. *Fundamenta Informaticae*, 72(1–3):347–361, 2006.

[19] A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *J. ACM*, 32(3):733–749, 1985.

[20] P. Sweazey and A. J. Smith. A class of compatible cache consistency protocols and their support by the IEEE futurebus. In *ISCA '86: Proceedings of the 13th annual international symposium on Computer architecture*, pages 414–423, Los Alamitos, CA, USA, 1986. IEEE Computer Society Press.

[21] P. F. Syverson. Knowledge, belief, and semantics in the analysis of cryptographic protocols. *Journal of Computer Security*, 1(3–4):317–334, 1992.

[22] R. van der Meyden. Finite state implementations of knowledge-based programs. In V. Chandru and V. Vinay, editors, *FSTTCS*, volume 1180 of *LNCS*, pages 262–273. Springer-Verlag, 1996.

[23] R. van der Meyden and N. Shilov. Model checking knowledge and time in systems with perfect recall (extended abstract). In C. Pandu Rangan, V. Raman, and R. Ramanujam, editors, *Proceedings Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'99)*, volume 1738 of *LNCS*, pages 432–445. Springer-Verlag, Dec. 1999.

[24] R. van der Meyden and K. Su. Symbolic model checking the knowledge of the dining cryptographers. In *CSFW*, pages 280–. IEEE Computer Society, 2004.

[25] R. van der Meyden and T. Wilke. Synthesis of distributed systems from knowledge-based specifications. In M. Abadi and L. de Alfaro, editors, *CONCUR*, volume 3653 of *LNCS*, pages 562–576. Springer-Verlag, 2005.

[26] R. van der Meyden and C. Zhang. Algorithmic verification of noninterference properties. In *Proceedings Workshop on Views on Designing Complex Systems*, ENTCS, Bertinoro, Italy, Sept. 2006. to appear.

[27] J. van Eijck and S. Orzan. Modelling the epistemics of communication with functional programming. In M. van Eekelen, editor, *6th Symposium on Trends in Functional Programming, TFP 2005*, pages 44–59, 2005.

[28] M. Y. Vardi. Implementing knowledge-based programs. In Y. Shoham, editor, *TARK*, pages 15–30. Morgan Kaufmann, 1996.

[29] M. Y. Vardi and P. Wolper. Automata theoretic techniques for modal logics of programs (extended abstract). In *STOC*, pages 446–456. ACM, 1984.

[30] B. Wozna, A. Lomuscio, and W. Penczek. Bounded model checking for knowledge and real time. In *4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005)*, pages 165–172, Utrecht, July 25-29 2005. ACM.

A Proofs

We provide here proofs omitted in the body of the paper.

Proof (of Theorem 5). By reduction from the problem of deciding if, for a given nondeterministic finite state automaton A over an alphabet Σ, the language $L(A)$ is equal to the universal language Σ^*.

Let $A = (Q, q_0, \delta, F)$ be an NFA with states Q, initial state q_0, transition function $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ and final states F. We define an environment E_A that has two different types of runs: one corresponds to the generation of a sequence of inputs to A, the other corresponds to runs of A. We employ the special letter $\epsilon \notin \Sigma$ to handle the empty word in both types of runs. To ensure that E_A has a fair path starting at every state, we add the sink state $\bot \notin \Sigma$. Formally, the environment $E_A = (S, I, \to, O_1, \pi, \alpha)$ consists of:

- states $S = \Sigma \cup \{\epsilon, (\epsilon, q_0), \bot\} \cup \Sigma \times Q$,
- initial states $I = \{\epsilon, (\epsilon, q_0)\}$,
- transitions
  - $\epsilon \to l$ and $l \to l'$ for each $l, l' \in \Sigma$,
  - $(l, q) \to (l', q')$ for each $l \in \Sigma \cup \{\epsilon\}$, $q \in Q$, $l' \in \Sigma$ and $q' \in \delta(q, l')$,
  - $(l, q) \to \bot$ if $\delta(q, l) = \emptyset$,
  - $\bot \to \bot$,
- observation function $O_1(l) = l = O_1(l, q)$, for all $l \in \Sigma \cup \{\epsilon\}$ and $q \in Q$, and $O_1(\bot) = \bot$,
- interpretation π given by
  - $\pi(\bot) = \emptyset$,
  - $\pi(l) = \{\mathtt{in}\}$ for $l \in \Sigma \cup \{\epsilon\}$,
  - $\pi(l, q) = \{\mathtt{final}\}$ if $l \in \Sigma \cup \{\epsilon\}$ and $q \in F$ else $\pi(l, q) = \emptyset$, and
- trivial acceptance condition α.

Note that for a word $w \in \Sigma^*$ of length m, if $r[0 \ldots m] = \epsilon.w$ then $P_1^{\mathtt{pr}}(E_A, r, m) = \{l\} \cup \{\ (l, q) \mid q_0 \to^w q\ \}$, where $l = r(m)$. Since there is such a run r for every word $w \in \Sigma^*$, it follows that $E_A \models^{\mathtt{pr}} \Box(\mathtt{in} \Rightarrow \neg K \neg \mathtt{final})$ iff $L(A) = \Sigma^*$. □

Proof (of Theorem 6). Fix k, and consider Π_k^p quantified boolean formulas Φ of the form

$$\forall q_1^k \ldots q_n^k \exists q_1^{k-1} \ldots q_n^{k-1} \ldots (\forall/\exists) q_1^1 \ldots q_n^1 \, (\alpha)\,,$$

where α is a 3-CNF formula of propositional logic in the variables q_i^j. (Formulas with differing numbers of propositional variables in the quantifications can always be put into this form at polynomial cost $O(nk)$ symbols by padding with unused variables.)

We construct environments E corresponding to such formulas in which the transition relation is the disjoint union of cycles of the form $s_0 \to \ldots \to s_{N-1} \to s_0$. We call such a component of the transition relation a *cycle of length* N.

Such cycles are used to represent assignments to the truth values of the propositional variables q_i^j as follows. Let $p_1^1, \ldots, p_n^1, \ldots p_1^k \ldots p_n^k$ be the sequence of the first nk primes greater than 2. Then the largest number p_n^k in this sequence is known to be $O(nk(\log nk + \log\log nk)) = O(n^2)$. Let $N_i^j = \Pi_{1 \leq j' \leq j} p_i^{j'}$. Thus the largest of these numbers is $N_n^k = O(n^{2k})$.

We associate with each variable q_i^j several cycles, each of length N_i^j, with one such cycle for each positive or negative occurrence of q_i^j in α. Let $\alpha = \bigwedge_{c \in C} c$, where each $c = \{l_1^c, l_2^c, l_3^c\}$ is a set representing a disjuction of 3 literals. If q_i^j or $\neg q_i^j$ occurs in c, we include in E a cycle $x_0^{c,i,j} \rightarrow \ldots x_N^{c,i,j} \rightarrow x_0^{c,i,j}$ where $N = N_i^j - 1$. Note that occurrences of a variable in distinct clauses give rise to distinct cycles, i.e., if $c \neq c'$ then $x_l^{c,i,j} \neq x_m^{c',i,j}$, but these cycles have the same length. Of these states, the states $x_0^{c,i,j}$ are made initial. Thus, we have one initial state per cycle in the transition relation. The total number of states is $O(|\Phi| \cdot N_n^k) = O(|\Phi|^{2k+1})$.

We make all the states arising from the clause c mutually indistinguishable to agent 1, i.e., we define $O_1(x_l^{c,i,j}) = c$. The observation function for agent 2 is defined so as to make all states indistinguishable, i.e., $O_2(x) = \perp$ for all states x.

Let X^j be the set of states $x_l^{c,i,j}$, and call these the *level* l states. It follows that if $P_m = P(E, r, m)$ is the set of states possible at time m, then

$$P_m \cap X^j = \{x_l^{c,i,j} \mid c \in C,\ 1 \leq i \leq n,\ \{q_i^j, \neg q_i^j\} \cap c \neq \emptyset,\ l = m \bmod N_i^j\}.$$

Noting that the numbers N_i^j for fixed j are co-prime, we have that the sets $P_m \cap X^j$ cycle with period $\Pi_{i=1}^n N_i^j$. More precisely, we have the following properties:

P1. For each function $f : \mathbb{A} \longrightarrow \mathbb{N}$ such that $0 \leq f(i) < N_i^j$ for each $i \in \mathbb{A}$, there exists m such that $P_m \cap X^j = \{x_{f(i)}^{c,i,j} \mid c \in C, \{p_i^j, \neg p_i^j\} \cap c \neq \emptyset\}$.

P2. If c and c' are clauses with $\{q_i^j, \neg q_i^j\} \cap c \neq \emptyset$ and $\{q_i^j, \neg q_i^j\} \cap c' \neq \emptyset$, then for all $m \in \mathbb{N}$ and $0 \leq l < N_i^j$, we have $x_l^{c,i,j} \in P_m$ iff $x_l^{c',i,j} \in P_m$.

We now label the states with propositions as follows:

1. For each $j = 1 \ldots k$, there is a proposition $\texttt{level}_j$, which holds just at states of the form $x_l^{c,i,j}$ for some c, i, l.
2. For each level of quantification $j = 1 \ldots k$, there is a proposition $\texttt{passgt}_j$, which we assign to be true at all states $x_l^{c,i,j}$ if $j = 1$ and at states $x_l^{c,i,j}$ with $j > 1$ iff l is divisible by $N_i^j / p_i^j = N_i^{j-1}$. Thus, there are p_i^j such states on the cycle. Intuitively, $\texttt{passgt}_j$ holds at states that represent *possible* contributions to truth assignments to the level j variables: we treat proposition q_i^j as being possibly assigned a value of *true* at a state $x_l^{c,i,j}$ satisfying $\texttt{passgt}_j$ if l is even. Note that if we consider different clauses c, c', then, for states labelled $\texttt{passgt}_j$, property **P2** implies that at a given time m, all the assignments of truth value to q_i^j according to this rule are consistent.
   However, in the formula we construct, we will be interested not directly in the truth value assigned to a variable, but in whether this assignment causes a clause in which the variable occurs to be true. For this, we further label the states $x_l^{c,i,j}$ where $\texttt{passgt}_j$ holds with the proposition $\texttt{sat}_j$, provided either

l is even (so q_i^j is considered true) and q_i^j occurs positively in the clause c, or l is odd (so q_i^j is considered false) and q_i^j occurs negatively in the clause c. These are the only states in the cycle where $\texttt{sat}$ holds. Intuitively, this represents that the clause c is satisfied because of the choice of truth value for p_i^j.

We have said that truth of $\texttt{passgt}_j$ at a state indicates that the state represents a *possible* contribution to an assignment of truth values to a proposition at level j. In fact, not all such occurrences will be treated as yielding assignments, but only those at times such that *all* states in $P_m \cap X^j$ satisfy $\texttt{passgt}_j$. It can be seen that this is the case just when m is divisible by N_i^{j-1} for all $i = 1 \ldots n$, or equivalently (since the N_i^{j-1} are co-prime), when m is divisible by $\Pi_{i=1\ldots n} N_i^{j-1}$. Intuitively, this condition represents that m is a time instant from which an assignment of truth value for *all* the level j propositions q_i^j can be read off. We may capture the satisfaction of this condition by the formula

$$\texttt{Assgt}_j = K_2(\texttt{level}_j \Rightarrow \texttt{passgt}_j)$$

which expresses that all level j states at the given time instant satisfy $\texttt{passgt}$.

Suppose we are given an assignment $\pi : \mathbb{A} \longrightarrow \{0, 1\}$. Let $f : \mathbb{A} \longrightarrow \mathbb{N}$ be any function with $f(i) < N_i^j$ such that $f(i)$ is divisible by N_i^{j-1} and $f(i)$ is even iff $\pi(i) = 1$. Then by property **P1**, there exists m such that $P_m \cap X^j = \{x_{f(i)}^{c,i,j} \mid c \in C, \{p_i^j, \neg p_i^j\} \cap c \neq \emptyset\}$. Thus, the assignment to the level j variables at this instant of time is exactly π.

Moreover, all these possible asssignments occur within every interval of length $N_1^j \cdot \ldots N_n^j$. In particular, between any two times $m(N_1^{j-1} \cdot \ldots N_n^{j-1})$ and $(m+1)N_1^{j-1} \cdot \ldots N_n^{j-1}$, the combinations of level $j-1$ states cycle through all possibilities, so we have all possible assignments to the level $j-1$ variables represented between these successive level j assignments.

Instead of reading the value of a level j variable from the assignment at the current time, we read it from the next time that all the level j variables are assigned a value. This can be captured by the following formulas. First, define the expression $\texttt{all}_j(\varphi)$ as $\bigcirc[(\texttt{Assgt}_j \Rightarrow \varphi) \mathsf{U} (\texttt{Assgt}_{j+1})]$, which says that φ holds at all points corresponding to a j level assignment that precede the next level $j+1$ assignment. The dual of this is the expression $\texttt{some}_j(\varphi)$, defined as $\neg\texttt{all}_j(\neg\varphi)$, which says that φ holds at some point before the next level $j+1$ assignment.

Next, define $\texttt{Holds}$ as $\neg K_1 \neg\beta$, where

$$\beta = \bigvee_{j=1\ldots n} \texttt{next}(\texttt{Assgt}_j, \texttt{sat}_j)$$

where $\texttt{next}(\varphi, \psi)$ is the formula $\bigcirc((\neg\varphi) \mathsf{U} (\varphi \wedge \psi))$, which says that ψ holds at the next point (after the current) where φ holds. Note that, when $\texttt{Holds}$ is evaluated at point where the state is $x_l^{c,i,j}$, the definition of observability for agent 1 implies that we check β only at points where the state is of the form $x_{l'}^{c',i',j'}$ with $c' = c$. Thus, this formula corresponds to checking that some literal in c causes c to be satisfied, according to the "current assignment" to the variables, which is determined at each level by looking at the first time in the future that corresponds to a level j assignment.

We may then translate the given formula as

$$\Phi^* = \Box(\mathtt{Assgt}_k \Rightarrow \mathtt{some}_{k-1}\mathtt{all}_{k-2} \ldots (K_2\mathtt{Holds}))$$

Note that during the evaluation of this formula, because of the nesting structure for the occurrence of assignments down the levels, the successor assignment for each level j is preserved whenever an operator $\mathtt{all}_{j'}$ or $\mathtt{some}_{j'}$ with $j' < j$ moves the point of evaluation. Thus the successor assignments used in the evaluation of `Holds` are the same as those determined by the points of evaluation for these operators. The knowledge operator K_2 may move the point of evaluation from any point (r, n) to another point (r', n) at the same time. In particular, this operator captures quantification over all the clauses c in the given QBF formula. It follows that $E \models^{\mathtt{clk}} \Phi^*$ iff Φ is true. □

Proof (of Lemma 8). By induction on the construction of φ. The only non-trivial cases are those for the knowledge operators. We describe the argument for $K_i\varphi$, that for $C_G\varphi$ is similar. Suppose $E, (r, m) \models^{\mathsf{obs}} K_i\varphi$. Then for all points (r', m') of E with $O_i(r(m)) = O_i(r'(m'))$ we have $E, (r', m') \models^{\mathsf{obs}} \varphi$. We show that $E, r|_m) \models^* K_i\varphi$. For, let $O_i(r(m)) = O_i(\rho(0))$, where $\rho \in \mathit{fpaths}(E)$. Since observations preserve reachability, the state $\rho(0)$ is reachable, so there exists a sequence $s_0 \to s_1 \to \ldots s_{m'} = \rho(0)$ with $s_0 \in I$. Let r' be the sequence $s_0 \ldots s_{m'-1} \cdot \rho$. Then r' is a run of E and $(r, m) \overset{\mathsf{obs}}{\sim}_i (r', m')$. Hence $E, (r', m') \models^{\mathsf{obs}} \varphi$. By the inductive hypothesis, $E, r'|_{m'} \models^* \varphi$, i.e., $E, \rho \models^* \varphi$. Hence $E, r|_m \models^* K_i\varphi$.

Conversely, suppose $E, r|_m \models^* K_i\varphi$. Let (r', m') be a point with $O_i(r'(m')) = O_i(r(m))$. Then $r'|_{m'}$ is a fair path with $O_i(r'|_{m'}(0)) = O_i(r(m))$, so $E, r'|_{m'} \models^* \varphi$, hence $E, (r', m') \models^{\mathsf{obs}} \varphi$. This shows that $E, (r, m) \models^{\mathsf{obs}} K_i\varphi$. □

Proof (of Lemma 9). Let σ be a simulation from E to E'. Part (1) follows from points 1, 2, and 7. Part (2) follows from points 3, and 7.

For part (3), let $\rho \in \mathit{fpaths}(E)$. We proceed by induction on the construction of φ. The propositional case is immediate from 6. The temporal cases are straightforward.

For the knowledge case, assume $E, \rho \models^* K_i\psi$ and that $O'_i(\sigma(\rho(0))) = O'_i(\rho''(0))$ for some $\rho'' \in \mathit{fpaths}(E')$. By 3, there exists a state t of E such that $O_i(\rho(0)) = O_i(t)$ and $\sigma(t) = \rho''(0)$. By part (2), there exists a $\rho' \in \mathit{fpaths}(E)$ such that $\rho'(0) = t$ and $\sigma(\rho') = \rho''$. Thus $E, \rho' \models^* \psi$. By the induction hypothesis, $E', \rho'' \models^* \psi$. This shows that $E', \sigma(\rho) \models^* K_i\psi$.

Conversely, suppose $E', \sigma(\rho) \models^* K_i\psi$. Suppose $O_i(\rho(0)) = O_i(\rho'(0))$ where $\rho' \in \mathit{fpaths}(E)$. By part (1), $\sigma(\rho')$ is a fair path of E'. By 4, $O'_i(\sigma(\rho(0))) = O'_i(\sigma(\rho')(0))$. Thus $E', \sigma(\rho') \models^* \psi$. By the induction hypothesis, $E', \rho' \models^* \psi$. This shows that $E, \rho \models^* K_i\psi$.

The case for common knowledge follows by similar arguments. □

Proof. (of Theorem 11) Correctness of the alternating procedure is a straightforward combination of the correctness arguments for Büchi automaton emptiness checking, Savitch-style search and the definition of $\models^*$

For the complexity analysis, note that the number N used in $\texttt{FALSIFY}(E, s, \psi)$ is $O(f(|E|) + |\psi|)$. The routine $\texttt{FALSIFY}(E, s, \psi)$ generates a computation tree in which the longest branch is $O(f(|E|) + |\psi|)$ (for the existential choice) plus the maximum of $O(g(|E|))$ (for the verification of the guessed components) and the longest branch for $\texttt{REACH}(E, w, w', N, \psi)$.

Note that $\texttt{REACH}(E, w, w', n, \psi)$ calls $\texttt{CHECK}()$ only when $n = 0$, and each recursion before then adds time $O(f(|E|) + |\psi|)$ to construct the guess for the recursive call. Hence $\texttt{REACH}(E, w, w', N, \psi)$ runs in alternating time $O((f(|E|) + |\psi|)^2)$ plus the time required for the call to $\texttt{CHECK}(E, s, k, \psi)$ once $n = 0$. The largest cost in the latter is the calls to $\texttt{CKCHECK}(E, s, C_G\psi')$, which add another $O((f(|E|) + |\psi|)^2)$ alternating time steps before calling $\texttt{FALSIFY}(E, s, \psi')$, with ψ' of lower knowledge depth than ψ. Thus, if $T(E, h)$ is the alternating time required by $\texttt{FALSIFY}(E, s, \psi)$ for formulas ψ with $|\psi| \leq h$, we have the recurrence $T(E, h) = O((f(|E|)+h)^2+g(|E|))+T(E, h-1)$, hence $T(E, h) = O(h\cdot((f(|E|)+h)^2 + g(|E|)))$. This yields the result. □