Symbolic Model Checking Algorithms for Temporal-Epistemic Logic

Xiaowei Huang and Ron van der Meyden

School of Computer Science and Engineering,
University of New South Wales, Australia

Abstract. The paper presents ordered binary decision diagram based symbolic model checking algorithms for fragments of a temporal-epistemic logic based on the temporal logic CTL* with operators for the knowledge and common knowledge of multiple agents. The knowledge operators are interpreted with respect to several distinct semantics: observational, clock, synchronous perfect recall and asynchronous perfect recall.

1 Introduction

Model checking [5] is an approach to systems verification based on the use of algorithms that check whether a specification, expressed as a formula of some modal logic, holds in a particular model of that logic, which represents the system to be verified. *Symbolic* model checking refers to an approach that works with symbolic representations of both the model and the specification formula. In particular, a prominent set of these algorithms use Ordered Binary Decision Diagrams [3] (or OBDD's).

The OBDD approach to symbolic model checking was first developed for branching time temporal logics [4], but has been extended to richer types of modal logic. In this paper, we present OBDD-based symbolic model checking algorithms for temporal-epistemic logics [7]. Epistemic logic provides operators that describe what an agent *knows*. Epistemic logic has been applied within computer science to distributed algorithms [7], computer security [17], game theory [9], and multi-agent systems in AI [14].

A range of different interpretations of knowledge can be given, depending on the amount of memory one associates with an agent. In this paper, we consider four distinct memory assumptions, each associated with a different semantics for the knowledge operators: the *observational* semantics captures what an agent knows based on just its current observation, the *clock* semantics captures what an agent knows based on its current observation, plus the current value of the clock, the *synchronous perfect recall* semantics assumes that an agent is aware of each clock tick, and remembers all its observations, the *asynchronous perfect recall* semantics assumes that the system operates asynchronously, with agents aware of the passing of time only when their observations change, but that agents remember the sequence of distinct observations they have made. We deal with a branching-time temporal epistemic logic CTL*K$_n$ that combines operators for linear time, temporal branching, and operators for the knowledge of multiple agents. We describe algorithms that are tailored for each of these different semantics. In general, the richer the semantics, the more restrictive is the fragment of the specification

language on which it is decidable, so most of the algorithms operate only on fragments of the full logic.

The model checker MCK[1] implements all the algorithms described here. Many of the main ideas were already implemented in the 2003 release [8], but have never been documented in the literature. Our purpose in this paper is to close this gap, and to describe the versions implemented in MCK 0.5.0 (Nov 2010), which enlarges the coverage of some of the algorithms and includes algorithms for perfect recall not available in earlier releases.

The structure of the paper is as follows. In Section 2 we describe the logic we work with and the model checking problems based on the four possible interpretations of knowledge. Section 3 reviews OBDD's and operations on this symbolic representation that are used in our algorithms. Sections 4-7 consider algorithms for the four different semantics. Section 8 concludes with pointers to applications.

2 The Temporal-Epistemic Model Checking Problem

We work with a logic $\mathtt{CTL^*K}_n$ that combines the branching time logic CTL* with operators from the logic of knowledge. Its syntax is presented as follows:

$$\phi \;=\; p \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid A\phi \mid X\phi \mid \phi_1 U \phi_2 \mid K_i\phi \mid C_G\phi \;,$$

where p is an element of the set Prop of atomic propositions and i is an element of the set of agents $Agt = \{1, \ldots, n\}$ and G is a nonempty subset of Agt. Other common operators may be defined from the others, in particular $\phi \Rightarrow \psi = \neg\phi \vee \psi$, and $E\phi = \neg A\neg\phi$, and $F\phi = \mathbb{T}\, U\, \phi$, and $G\phi = \neg F\neg\phi$. We use $\mathbb{T}$ and $\mathbb{F}$ to denote the truth values True and False, respectively.

Semantics of the logic can be given using *interpreted systems* [7]. Let S be a set, which we call the set of environment states. A *run* over environment states S is a function $r : \mathbf{N} \to S \times L_1 \times \ldots \times L_n$, where each L_i is some set, called the set of *local states of agent i*. These local states are used to concretely represent the information on the basis of which agent i computes its knowledge. Given a run r, agent i and time m, we write $r_i(m)$ for the $i+1$-st component (in L_i) of $r(m)$, and $r_e(m)$ for the first component (in S). An *interpreted system* over environment states S is a tuple $\mathcal{I} = (\mathcal{R}, \pi)$, where $\mathcal{R}$ is a set of runs over environment states S, and $\pi : S \to \mathcal{P}(Prop)$ is an interpretation function.

A *point* of $\mathcal{I}$ is a pair (r, m) where $r \in \mathcal{R}$ and $m \in \mathbf{N}$. We define the semantics of $\mathtt{CTL^*K}_n$ by means of a relation $\mathcal{I}, (r, m) \models \phi$, where $\mathcal{I}$ is an intepreted system, (r, m) is a point of $\mathcal{I}$, and ϕ is a formula. This relation is defined by:

- $\mathcal{I}, (r, m) \models p$ if $p \in \pi(r(m))$,
- $\mathcal{I}, (r, m) \models \neg\phi$ if not $\mathcal{I}, (r, m) \models \phi$
- $\mathcal{I}, (r, m) \models \phi \wedge \psi$ if $\mathcal{I}, (r, m) \models \phi$ and $\mathcal{I}, (r, m) \models \psi$
- $\mathcal{I}, (r, m) \models A\phi$ if for all runs $r' \in \mathcal{R}$ with $r'(k) = r(k)$ for all $k = 0 \ldots m$, we have $\mathcal{I}, (r', m) \models \phi$,
- $\mathcal{I}, (r, m) \models X\phi$ if $\mathcal{I}, (r, m + 1) \models \phi$.

[1] `http://www.cse.unsw.edu.au/~mck`

- $\mathcal{I}, (r, m) \models \phi \, U \, \psi$ if for some $m' \geq m$, $\mathcal{I}, (r, m') \models \psi$ and $\mathcal{I}, (r, m'') \models \phi$ for all m'' with $m \leq m'' < m'$
- $\mathcal{I}, (r, m) \models K_i \phi$ if for all points (r', m') of $\mathcal{I}$ such that $r_i(m) = r'_i(m')$ we have $\mathcal{I}, (r', m') \models \phi$
- $\mathcal{I}, (r, m) \models C_G \phi$ if for all sequences $(r, m) = (r^0, m_0) \sim_{i_0} \ldots \sim_{i_{k-1}} (r^k, m_k) = (r', m')$ such that $i_l \in G$ for all $l = 0 \ldots k-1$, we have $\mathcal{I}, (r', m') \models \phi$.

While they give a clean and coherent semantics to the logic, interpreted systems are not suitable as inputs for a model checking program, since they are infinite structures. Instead, we use the following notion. A (finite) *environment* is a tuple $M = (S, I, \rightarrow, \{O_i\}_{i=1\ldots n}, \pi, \chi)$ where S is a (finite) set of states, $I \subseteq S$ is the set of initial states, $\rightarrow \subseteq S \times S$ is a serial temporal transition relation, each $O_i : S \rightarrow O$ is an observation function for agent $i \in Agt$, $\pi : S \rightarrow \mathcal{P}(Prop)$ is a propositional interpretation, and $\chi \subseteq \mathcal{P}(S) \setminus \{\emptyset\}$ is a *generalized Büchi fairness condition*. The environment M can also be regarded as a generalized Büchi automaton with χ the set of acceptance sets. We define a *fullpath* from a state s to be is an infinite sequence of states $\rho = s_0 s_1 ...$ such that $s_0 = s$ and $s_i \rightarrow s_{i+1}$ for all $i \geq 0$. We say a fullpath is *initialised* if $s_0 \in I$. The fairness condition places an additional constraint on fullpaths. A fullpath $s_0 s_1 \ldots$ is said to be *fair* if for all $Q \in \chi$, there exists a state $s \in Q$ such that $s = s_i$ for infinitely many i.

We may construct several different interpreted systems for each environment, depending on a *view* that is used to define the local states associated to agents. We consider the following views: the observational view `obs`, the clock view `clk`, the synchronous perfect recall view `spr` and the asynchronous perfect recall view `apr`. Given a fair fullpath ρ and a view $\mathcal{V}$, we may construct a run $\rho^{\mathcal{V}}$ by defining the components at each time m as follows. In all cases, the environment state $\rho_e^{\mathcal{V}}(m) = \rho(m)$. The local state of agent i at time m for each of the different views is given by:

- $\rho_i^{\texttt{obs}}(m) = O_i(\rho(m))$: here agent i's local state at time m is its current observation;
- $\rho_i^{\texttt{clk}}(m) = (m, O_i(\rho(m)))$: this models that the agent's local state is its current observation, plus the clock value;
- $\rho_i^{\texttt{spr}}(m) = O_i(\rho(0)) \ldots O_i(\rho(m))$: representing that the agent remembers all its observations;
- $\rho_i^{\texttt{apr}}(m) = \mathit{reduce}(\rho_i^{\texttt{spr}}(m))$, where the function *reduce* reduces a sequence of observations to a shorter sequence by replacing maximal subsequences of consecutively repeated copies of an observation by a single copy of that observation (for example, $\mathit{reduce}(xxyyxxx) = xyx$): this represents that the agent has perfect recall, but operates asynchronously, and is aware of the passing of time only when its observation changes.

Given an environment M over states S, and a view $\mathcal{V}$, we may construct an interpreted system $\mathcal{I}^{\mathcal{V}}(M) = (\mathcal{R}^{\mathcal{V}}, \pi)$ over global states S. The interpretation π is the same as in M, and the set of runs $\mathcal{R}^{\mathcal{V}}$ is defined to be the set of runs $\rho^{\mathcal{V}}$ where ρ is a fair initialised fullpath of M.

A formula ϕ of $\texttt{CTL}^*\texttt{K}_n$ is said to hold in M with respect to a view $\mathcal{V}$, written $M \models^{\mathcal{V}} \phi$, if $\mathcal{I}^{\mathcal{V}}(M), (r, 0) \models \phi$ for all $r \in \mathcal{R}^{\mathcal{V}}$. The model checking problem is then to determine, given an enviroment M, view $\mathcal{V}$ and formula ϕ, whether $M \models^{\mathcal{V}} \phi$.

3 Ordered Binary Decision Diagrams

Ordered Binary Decision Diagrams (OBDD's) [3] are a symbolic representation of boolean functions that can, in practice, be quite compact and that support efficient computation of operations that combine boolean functions. Let V be a set of variables. A *V-assignment* is a function $s : V \to \{0,1\}$. Write $Assgts(V)$ for the set of all V-assignments, and $s[v \mapsto x]$ for the function that is identical to s except that it takes value x on input v. A *V-indexed boolean function* is a mapping $f : Assgts(V) \to \{0,1\}$. Note that such functions are able to represent sets $X \subseteq Assgts(V)$ by their characteristic functions f_X, mapping s to 1 just in case $s \in X$. One way to represent such a function f is using a binary tree of height n, with each level corresponding to one of the variables in V, and leaves labelled from $\{0,1\}$. This tree can in turn be thought of as a finite state automaton on alphabet $\{0,1\}$. OBDD's more compactly represent such a function as a dag of height n, with binary branching, by applying the usual finite state automaton minimization algorithm. Given this minimal representation, it is moreover possible to compute (in practice, often in reasonable time) the following operations.

- The boolean operations $\wedge, \neg$, defined pointwise on functions. E.g., if $f, g : Assgts(V) \to \{0,1\}$, then $f \wedge g : Assgts(V) \to \{0,1\}$ is defined by $(f \wedge g)(s) = f(s) \wedge g(s)$, and $\neg f : Assgts(V) \to \{0,1\}$ is defined by $(\neg f)(s) = \neg f(s)$.
- Boolean quantification $\exists, \forall$, e.g., if $f : Assgts(V) \to \{0,1\}$ and $v \in V$ then $\exists v(f) : Assgts(V \setminus \{v\}) \to \{0,1\}$ maps $s \in Assgts(V \setminus \{v\})$ to $f(s[v \mapsto 0]) \vee f(s[v \mapsto 1])$.
- variable substitution: if $f : Assgts(V) \to \{0,1\}$ and $U \subseteq V$ and U' are sets with $U' \cap (V \setminus U) = \emptyset$, and $\sigma : U \to U'$ is a bijection, then $f_\sigma : Assgts((V \setminus U) \cup U') \to \{0,1\}$ maps $s : Assgts((V \setminus U) \cup U')$ to $f(s_\sigma)$, where $f(s_\sigma)(v)$ is $s(v)$ when $v \in V \setminus U$ and $s(\sigma(v))$ when $v \in U'$.

These operations on the minimal OBDD representations are used in all the algorithms we discuss in this paper.

Each of the algorithms relies on an OBDD representation of the environment M in the input to the model checking problem. This environment is represented as follows:

1. Assuming $S \subseteq Assgts(Prop)$, the set of states S can be represented by its characteristic function $f_S : Assgts(Prop) \to \{0,1\}$. Similarly, OBDD representations of the characteristic functions f_I, f_Q are used to represent the set I of initial states, and the fairness conditions $Q \in \chi$, respectively.
2. For the transition relation $\to$, we use the set of "primed" versions of the state variables $Prop$, defined by $Prop' = \{v' \mid v \in Prop\}$, and the function $f_\to : Assgts(Prop \cup Prop') \to \{0,1\}$ such that if $s \in Assgts(Prop)$ and $s' \in Assgts(Prop')$, then $f_\to(s \cup s') = 1$ iff $s \to s'$.
3. The observation functions O_i are associated with indistinguishability relations $\sim_i$ on states, defined by $s \sim_i s'$ if $O_i(s) = O_i(s')$. These can be represented as a boolean functions $f_{\sim_i} : Assgts(Prop \cup Prop') \to \{0,1\}$ such that if $s \in Assgts(Prop)$ and $s' \in Assgts(Prop')$, then $f_{\sim_i}(s \cup s') = 1$ iff $s \sim_i s'$.

Using these basic OBDD representations of various sets and relations, the algorithms below compute other sets and relations, also represented as OBDDs. For clarity,

we generally focus below on the set level descriptions rather than on how the OBDD representations are computed. We give just one example here: suppose that we define the $reach(U, \rightarrow) = \{t \in S \mid \exists s \in U(s \rightarrow^* t)\}$, i.e., the set of states reachable via $\rightarrow$ from a state in U. The OBDD representation of this can be computed the first element g_j of the sequence defined inductively by $g_0 = f_U$, and $g_{i+1} = (\exists Prop(g_i \wedge f_{\rightarrow}))[Prop' \mapsto Prop]$ such that $g_{j+1} = g_j$. Note that here $\exists Prop$ abbreviates the quantifier sequence $\exists v_1 \ldots \exists v_k$, where $Prop = \{v_1..v_k\}$, and $[Prop' \mapsto Prop]$ is the OBDD operation of substituting each $v' \in Prop'$ with the corresponding variable $v \in Prop$.

4 Model Checking for the Observational View

Symbolic model checking for the observational view for the full language $\mathtt{CTL}^*\mathrm{K}_n$ can be handled by means of straightforward extensions of known algorithms for OBDD-based model checking of branching time logics, and has been implemented in a number of model checkers. MCK 0.1.0 (2003) separately implemented model checking for (observational view) epistemic extensions of the branching time logic CTL and the linear time logic LTL. MCMAS [12] supports the combination of CTL, epistemic logic, Alternating Time Logic, and deontic modalities. MCTK [15] supports the combination of CTL* and epistemic logic, which is now also supported in MCK 0.5.0. We just highlight here a few points that may not be clear in the literature and that are relevant to what follows.

First, when dealing with epistemic operators, it is important to take into account fair reachability. We say that a state $s \in S$ is *fair* if it is the initial state of some fair fullpath, otherwise the state is *unfair*. A state s is *reachable* if there exists a sequence $s_0 \rightarrow s_1 \rightarrow \ldots s_k = w$ where $s_0 \in I$. A state is fair and reachable iff it occurs in some run. Note that some reachable states may be unfair.

OBDD-based model checking for CTL is based on recursively computing an OBDD representation of the set S_ϕ of all states (reachable or unreachable) at which ϕ holds, and testing whether $I \subseteq S_\phi$. When dealing with knowledge operators, the appropriate clause of the recursion is based on the equation

$$S_{K_i\phi} = \{s \in S \mid \forall s' \in S_{fair} \cap S_{reach}\ (O_i(s) = O_i(s') \Rightarrow s \in S_\phi)\}\ .$$

Note that the quantification over s' needs to be restricted to the set of fair and reachable states, since these are the only states that occur in the interpreted system $\mathcal{I}(M)$, so that agents, implicitly, know that only fair and reachable states are possible. OBDD representations of S_{fair}, S_{reach} can be computed using techniques described in [5]. A similar issue arises in the fixpoint used to compute $S_{C_G\phi}$.

Second, the incorporation of fairness constraints in the algorithms for the above grammar is not explicitly documented in the literature, but it is a straightforward application of ideas from [5]. (The combination of CTL, knowledge operators, and fairness constraints is first handled following methods for CTL together with the above clause, and then extended to the combination of CTL*, knowledge operators, and fairness constraints, by means an extension of the translation from LTL to CTL.)

5 Model Checking for the Clock View

Model checking the clock view is more complex than the observational view, because the observability of the clock means that agent knowledge needs to take into account the set of states possible at a given time, which can take $2^{|S|}$ possible values. Complexity analysis conducted in [6, 10], shows that model checking the language $\mathtt{CTL}^*\mathtt{K}_n$ with respect to the clock view is decidable in PSPACE, based on an input representation in which all states are given in the input, rather than symbolically represented. Several fragments are discussed in these papers, but these have complexity at some level Π_k^p of the polynomial hierarchy. (Model Checking based on symbolic inputs is likely to be more complex. [11] proves a PSPACE-complete complexity for model checking CTL based on symbolic inputs, compared to PTIME for explicitly represented systems. This result is generalized in [13] to show that model checking on CTLK_n^{obs} based on symbolic inputs is also PSPACE-complete.) Model checking LTL is also PSPACE-complete, but there is the essential difference that most of its complexity derives from the size of the (typically small) formula, whereas for $\mathtt{CTL}^*\mathtt{K}_n$ there is also a PSPACE-complete dependence in the (generally much larger) size of the model. In spite of this complexity, several smaller fragments exist that can be practicably handled using OBDD-based algorithms.

5.1 MCK Algorithm spec_clk_xn

This algorithm works for formulas of the form $X^d\varphi$, where φ is a formula containing only epistemic operators, i.e. given by the grammar

$$\phi \ := \ p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid K_i\phi \mid C_G\phi \ ,$$

where $p \in Prop$, $i \in Agt$ and $G \subseteq Agt$. The idea of the algorithm is to first compute the set S_d of all states possible at time d, and then model check ϕ as if this set were the set of states of Kripke structure in which the indistinguishability relations are defined using observational indistinguishability. The set S_d can be computed recursively by $S_0 = I \cap S_{fair}$, and $S_{m+1} = \{s \in S \mid \exists t \in S_m : t \rightarrow s, s \in S_{fair}\}$.

Theorem 1. *$M \models^{\mathtt{clk}} X^d\varphi$ iff $M_d \models^{\mathtt{obs}} \varphi$, where $M_d = (S_d, S_d, \rightarrow, \{O_i\}_{i\in Agt}, \pi, \chi)$.*

The RHS of this result can be computed by the OBDD-based algorithms for the observational semantics.

5.2 MCK Algorithm spec_clk_ctl_nested

The algorithm of Section 5.1 does not allow temporal operators to be nested inside epistemic operators. A related set-based algorithm works for the more general fragment of the logic, defined by the following grammar:

$$\phi = p \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid AX\phi \mid K_i\phi \mid C_G\phi \ .$$

It proceeds by computing the set of states S_ϕ^Z at which the formula ϕ (from the above grammar) would be true if we were to take Z as the set of initial states of the environment. Assuming $Z \subseteq S_{fair}$, this set can be computed recursively as follows:

$S^Z_v = Z \cap \{s \in S \mid v \in \pi(s)\}$
$S^Z_{\neg\varphi} = Z \setminus S^Z_\varphi \qquad S^Z_{\varphi_1 \wedge \varphi_2} = S^Z_{\varphi_1} \cap S^Z_{\varphi_2}$
$S^Z_{AX\varphi} = \{s \in Z \mid \forall s' \in S(s \rightarrow s' \wedge s' \in S_{fair} \Rightarrow s' \in S^{Z'}_\varphi)\}$, where $Z' = \{s' \in S \mid \exists s \in Z(s \rightarrow s', s' \in S_{fair})\}$
$S^Z_{K_i\varphi} = \{s \in Z \mid \forall s' \in Z(s \sim_i s' \Rightarrow s' \in S^Z_\varphi)\}$
$S^Z_{C_G\varphi}$ is the first element U_i of the (decreasing) sequence $U_0, U_1, \ldots$ such that $U_i = U_{i+1}$, where $U_0 = Z$ and $U_{j+1} = \{s \in Z \mid \forall i \in G \forall s' \in Z(s \sim_i s' \Rightarrow s' \in S^Z_\varphi \cap U_j)\}$.

OBDD representations of these sets can be straightforwardly computed using the OBDD operations. The algorithm then proceeds by using the RHS of the following result.

Theorem 2. $M \models^{\mathtt{clk}} \phi$ *iff* $I \cap S_{fair} \subseteq S^{I \cap S_{fair}}_\phi$.

5.3 MCK Algorithm spec_clk_nested

The previous algorithm requires that the branching operator A and the next-time operator X occur only in the combination AX. This constraint can be relaxed by means of an approach that works with a boolean function of finite state sequences rather than with sets of states. The grammar in this case is

$$\phi = p \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid A\phi \mid X\phi \mid K_i\phi \mid C_G\phi \ .$$

A run prefix of M is a sequence of states $t = s_0 \ldots s_n$ that can be extended to a run of M. (Note that this implies all states in the sequence are fair.) Given formula ϕ, let d be the maximal depth of nesting of the next-time operator X in ϕ. Write $\mathcal{R}_d(M)$ for the set of run prefixes of length $d + 1$.

For subformulas φ of ϕ, let $f^x_\varphi : \mathcal{R}_d(M) \rightarrow \{0, 1\}$ be a boolean function such that, intuitively, $f^x_\varphi(t)$ means that formula φ holds at time x in run prefix t, where $0 \leq x \leq d$. The value $f^x_\varphi(t)$, can be computed recursively as follows:

$f^x_v(t) = t(x)(v)$
$f^x_{\neg\varphi}(t) = \neg f^x_\varphi(t) \qquad f^x_{\varphi_1 \wedge \varphi_2}(t) = f^x_{\varphi_1}(t) \wedge f^x_{\varphi_2}(t)$
$f^x_{X\varphi}(t) = f^{x+1}_\varphi(t)$
$f^x_{K_i\varphi}(t) = \forall t' \in \mathcal{R}_d(M)\ (O_i(t(x)) = O_i(t'(x)) \Rightarrow f^x_\varphi(t'))$
$f^x_{A\varphi}(t) = \forall t' \in \mathcal{R}_d(M)\ (t(x) = t'(x) \Rightarrow f^x_\varphi(t'))$
(remark: it is not necessary for t, t' to be identical up to time x here)
$f^x_{C_G\varphi}(t) = g_k(t)$, where g_k is the first element of the sequence defined by $g_0 = \lambda t(f^x_\phi(t))$ and $g_{j+1} = \lambda t(\forall t' \in \mathcal{R}_d(M)(\forall i \in G(O_i(t(x)) = O_i(t'(x)) \Rightarrow g_j(t'))))$, with $g_k = g_{k+1}$

We then have the following characterization:

Theorem 3. $M \models^{\mathtt{clk}} \phi$ *iff* $\forall t \in \mathcal{R}_d(M)(f^0_\phi(t))$.

The functions f^x_φ can be straightforwardly encoded as OBDD's based on a set of $(d + 1)|Prop|$ variables that represent t, and the calculation of the RHS of the theorem done using OBDD operations. Compared to the algorithm of the previous section, which works with characteristic functions of state sets, requiring only $Prop$ variables, for formulas in the domain of both, the algorithm of this section can be expected to consume more time and space.

6 Model Checking Synchronous Perfect Recall

For the full language $\mathtt{CTL}^*\mathtt{K}_n$, model checking with respect to the perfect recall interpretation is undecidable [16]. However, OBDD-based algorithms can be developed for various fragments.

6.1 MCK Algorithm spec_spr_xn

One special case of perfect recall model checking that can be handled effectively in practice is formulas of the form $X^d\varphi$, where φ is a formula in which the only modal operator is K_i, for some *fixed* agent i. (That is, the formula talks about the knowledge of just one agent.) The following symbolic model checking algorithm for this case was first described in [17]. The idea is to work with boolean functions $\mathtt{T}_\varphi : O^{d+1} \times S \to \{0,1\}$, where O is the set of values of the observation function O_i in M, and, intuitively, $\mathtt{T}_\varphi(o_0...o_d, s) = 1$ just in case for all run prefixes t of length $d+1$ with final state s, extendable to a run r with $r_i(d) = o_0 \dots o_d$, we have $\mathcal{I}(M), (r,d) \models^{\mathrm{spr}} \varphi$. The function $\mathtt{T}_\varphi$ can be recursively computed as follows. We first compute a function $\mathtt{P}(o_0 \dots o_d, s)$ that is true just in case it is possible to reach state s in such a way that agent i makes observation sequence $o_0 \dots o_d$, by $\mathtt{P}(o_0, s) = (s \in I) \wedge (O_i(s) = o_0)$, and

$$\mathtt{P}(o_0...o_{m+1}, s) = \exists t \in S\,(\mathtt{P}(o_1...o_m, t) \wedge t \to s \wedge O_i(s) = o_{m+1})\ .$$

The general case is then

$\mathtt{T}_p(o_0...o_d, s) = (p \in \pi(s))$
$\mathtt{T}_{\varphi_1 \wedge \varphi_2}(o_0...o_d, s) = \mathtt{T}_{\varphi_1}(o_0...o_d, s) \wedge \mathtt{T}_{\varphi_2}(o_1...o_d, s)$
$\mathtt{T}_{\neg\varphi}(o_0...o_d, s) = \neg\mathtt{T}_\varphi(o_0...o_d, s)$
$\mathtt{T}_{K_i\varphi}(o_0...o_d, s) = \forall t \in S\,(\mathtt{P}(o_0...o_d, t) \wedge t \in S_{fair} \Rightarrow \mathtt{T}_\varphi(o_0...o_d, t))$

We then have the following characterization:

Theorem 4. *$M \models^{\mathrm{spr}} X^d\varphi$ is equivalent to the truth of $\forall o_0....o_d, s \in S\,(\mathtt{P}(o_0...o_d, s) \wedge s \in S_{fair} \Rightarrow \mathtt{T}_\varphi(o_0...o_d, s))$.*

The functions $\mathtt{P}$, $\mathtt{T}_\varphi$ and the RHS of the theorem can be computed using OBDDs.

6.2 MCK Algorithm spec_spr_nested

The restriction to a single agent's knowledge in the algorithm of the previous section can be overcome using an approach similar to that in Section 5.3. This allows us to model check formulas from the grammar

$$\phi = p \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid A\phi \mid X\phi \mid K_i\phi \mid C_G\phi\ .$$

with respect to the perfect recall interpretation of knowledge. Again, we work with functions $f^x_\varphi : \mathcal{R}_d(M) \to \{0,1\}$, where d is the depth of nesting of X in ϕ, such that encoding that the subformula φ of ϕ holds at time x of runs with prefix $t \in \mathcal{R}_d(M)$. These values can be computed recursively by:

$f^x_v(t) = t(x)(v)$
$f^x_{\neg\varphi}(t) = \neg f^x_\varphi(t)$ $\qquad$ $f^x_{\varphi_1\wedge\varphi_2}(t) = f^x_{\varphi_1}(t) \wedge f^x_{\varphi_2}(t)$
$f^x_{X\varphi}(t) = f^{x+1}_\varphi(t)$
$f^x_{A\varphi}(t) = \forall t' \in \mathcal{R}_d(M)\ (t =_x t' \Rightarrow f^x_\varphi(t'))$,
where $t =_x t'$ is $\bigwedge^x_{y=0}\ t(y) = t'(y)$, (compared to the clock case, note that here we need to check the history, since this affects knowledge)
$f^x_{K_i\varphi}(t) = \forall t' \in \mathcal{R}_d(M)\ (t \sim^x_i t' \Rightarrow f^x_\varphi(t'))$,
where $t \sim^x_i t'$ is $\bigwedge^x_{j=0}\ O_i(t(j)) = O_i(t'(j))$,
$f^x_{C_G\varphi}(t) = g_k(t)$, where g_k is the first element of the sequence defined by $g_0 = \lambda t(f^x_\varphi(t))$ and $g_{j+1} = \lambda t(\forall t' \in \mathcal{R}_d(M)(\forall i \in G(t \sim^x_i t' \Rightarrow g_j(t'))))$, with $g_k = g_{k+1}$.

The following result characterizes perfect recall model checking using these values:

Theorem 5. *$M \models^{\mathtt{spr}} \phi$ is equivalent to $\forall t \in \mathcal{R}_d(M)\ (f^0_\phi(t))$.*

6.3 MCK Algorithm spec_spr_g

This algorithm deals with formulae of the form $G(\varphi)$, where φ is formula in which the only modal operator is K_i, for a single agent i. Call a set of states $U \subseteq S$ an `spr`-*knowledge set* for agent i if $U = \{r'_e(m) \mid \exists r' \in \mathcal{R}^{\mathtt{spr}}(M)(r_i(m) = r'_i(m))\}$, where (r, m) is a point of $\mathcal{I}^{\mathtt{spr}}(M)$. The algorithm works by enumerating all `spr`-knowledge sets U for agent i, and checking whether the formula φ holds at all points in the Kripke structure $(U, U \times U, \pi)$, in which the equivalence relation used to interpret K_i is the universal relation $U \times U$ on U.

To obtain OBDD representations of the knowledge sets U, we proceed by an iterative procedure. Let O be the set of all values of O_i. Define the relation $U \to U'$ for sets $U, U' \subseteq S$ to hold if for some $o \in O$ we have $U' = \{t \in S_{fair} \mid \exists s \in U(s \to t) \wedge O_i(t) = o\}$. Given an OBDD representation of U, we can easily obtain an OBDD representation of U'.

At time 0, we have knowledge sets $\mathcal{U}_0 = \{I \cap \{s \in S \mid O_i(s) = o\} \cap S_{fair} \mid o \in O\}$. Let $\mathcal{U}^\infty$ be the smallest set containing $\mathcal{U}_0$ and closed under relation $\to$ on subsets of S. The set $\mathcal{U}^\infty$ is finite, and may be constructed using a depth or breadth first search from $\mathcal{U}_0$.

Theorem 6. *$M \models^{\mathtt{spr}} G(\varphi)$ iff $(U, U \times U, \pi) \models \varphi$ for all $U \in \mathcal{U}^\infty$.*

This gives an algorithm that operates symbolically for each U, represents the set $\mathcal{U}^\infty$ by listing all its elements. Since this set is potentially of size $2^{|S|}$, the algorithm is unlikely to be practical when M has a large number of `spr`-knowledge sets.

7 Model Checking Asynchronous Perfect Recall

Temporal-epistemic specifications are concerned with how agents' knowledge changes over time. Under the synchronous perfect recall and clock semantics, each transition causes a change in the agents' state of knowledge, so that $X^k K_i\phi$ says that in agent i's k-th state of information in each run, $K_i\phi$ holds. No such correspondence between time

and states of information holds under asynchronous semantics for knowledge, where in k steps, the agent may go through any number between 1 and k different states of information.

In order to reinstate such a correspondence, it is convenient to introduce a new operator X_i, such that, intuitively, $X_i\phi$ says that if there exists a later time where agent i's local state differs from its current local state, then while the agent is in that next local state, the formula ϕ holds. More precisely, the semantics of this operator is given by $\mathcal{I}, (r, m) \models X_i\phi$ if either $r_i(m) = r_i(m')$ for all $m' \geq m$, or, for the least $m' > m$ such that $r_i(m) \neq r_i(m')$, we have $\mathcal{I}, (r, k) \models \phi$ for all $k \geq m'$ such $r_i(m') = r_i(m'+1) = \ldots = r_i(k)$. Thus, the specification $X_i^d\phi$, evaluated at time 0, states that at all points where agent i has made exactly $d + 1$ observations, ϕ holds.

7.1 MCK Algorithm spec_apr_xn

An algorithm resembling that of Section 6.1 can be developed for specifications of the form $X_i^d\varphi$, where the only modal operator in φ is the operator K_i. We again use functions $\mathtt{P}_k : O^k \times S \to \{0, 1\}$ for $k = 1 \ldots d+1$ such that, intuitively, $P_k(o_0 \ldots o_{k-1}, s) = 1$ just in case there exists a (not necessarily fair) run r and a time m where $r_i(m) = o_0 \ldots o_{k-1}$ and $r_e(m) = s$, and a function $\mathtt{T}_\varphi : O^{d+1} \times S \to \{0, 1\}$, with $\mathtt{T}_\varphi(o_o...o_d, s) = 1$ just in case for all runs r and times m with $r_i(m) = o_0 \ldots o_d$ and $r_e(m) = s$, we have $\mathcal{I}(M), (r, m) \models^{\mathtt{apr}} \varphi$. (Note that because of asynchrony, we are no longer assured that $m = d + 1$, as we had in the synchronous case: m may be arbitrarily large.) Let $\to_=$ and $\to_{\neq}$ be the relations on states defined by $s \to_= t$ if $s \to t$ and $O_i(s) = O_i(t)$, and $s \to_{\neq} t$ if $s \to t$ and $O_i(s) \neq O_i(t)$. The function $\mathtt{P}(o_1...o_d, s)$ can be computed as follows.

1. $\mathtt{P}_1(o_0, s) = s \in \mathit{reach}(I, \to_=) \wedge (O_i(s) = o_0)$,
2. $\mathtt{P}_{k+1}(o_1...o_{k+1}, s) = \exists t, t' \in S\,(\mathtt{P}(o_1...o_k, t) \wedge t \to_{\neq} t' \wedge t' \to_=^* s \wedge O_i(s) = o_{k+1})$

The function $\mathtt{T}_\varphi(o_1...o_d, s)$ can be computed recursively by

1. $\mathtt{T}_p(o_0...o_d, s) = (p \in \pi(s))$
2. $\mathtt{T}_{\varphi_1 \wedge \varphi_2}(o_0...o_d, s) = \mathtt{T}_{\varphi_1}(o_0...o_d, s) \wedge \mathtt{T}_{\varphi_2}(o_0...o_d, s)$
3. $\mathtt{T}_{\neg\varphi}(o_0...o_d, s) = \neg\mathtt{T}_\varphi(o_0...o_d, s)$
4. $\mathtt{T}_{K_i\varphi}(o_0...o_d, s) = \forall t \in S\,(\mathtt{P}(o_0...o_d, t) \wedge t \in S_{fair} \Rightarrow \mathtt{T}_\varphi(o_0...o_d, t))$

OBDD representations of all of these functions can be computed, and the algorithm then uses the characterization in the following result:

Theorem 7. *$M \models^{\mathtt{apr}} X_i^d\varphi$ is equivalent to truth of $\forall o_0....o_d \in O\,\forall s \in S$ $(\mathtt{P}(o_0...o_d, s) \wedge s \in S_{fair} \Rightarrow \mathtt{T}_\varphi(o_0...o_d, s))$.*

7.2 MCK Algorithm spec_apr_g

It is also possible to adapt the algorithm of Section 6.3 to the asynchronous perfect recall semantics. The class of formulas is the same, i.e., formulas of the form $G(\varphi)$, where φ is a formula in which the only modal operator is K_i, for a single agent i. The approach of the algorithm is again to deal with knowledge sets for agent i, in this case,

defined as sets of the form $\{r'_e(m') \,|r' \in \mathcal{R}^{\mathtt{apr}}(M),\ m' \in \mathbf{N},\ r_i(m) = r'(m')\}$ where (r, m) is a point of $\mathcal{I}^{\mathtt{apr}}(M)$. Note that in the case of the asynchronous perfect recall semantics, the times m, m' may differ, with no bounds on m'. As before, we represent knowledge sets as OBDDs, and enumerate them by a depth or breadth first search. The difference is that we now need a fixpoint computation to construct a knowledge set.

Define the relation $U \rightarrow^{\mathtt{apr}} U'$ for sets $U, U' \subseteq S$ to hold if for some $o \in O$ we have $U' = \{t' \in S_{fair} \mid \exists s \in U \exists t \in S(s \rightarrow_{\neq} t) \wedge (t \rightarrow^*_= t') \wedge O_i(t') = o\}$. Given an OBDD representation of U, we can easily obtain an OBDD representation of U', using a fixpoint computation to deal with $\rightarrow^*_=$.

At time 0, we have the collection of knowledge sets

$$\mathcal{U}_0 = \{\ \{t \in S_{fair} \,|\exists s \in I(s \rightarrow^*_= t \wedge O_i(t) = o)\} \mid o \in O_i(I)\}\ .$$

Let $\mathcal{U}^\infty$ be the smallest set containing $\mathcal{U}_0$ and closed under relation $\rightarrow^{\mathtt{apr}}$ on subsets of S. The set $\mathcal{U}^\infty$ is finite, and may be constructed using a depth or breadth first search from $\mathcal{U}_0$.

Theorem 8. *$M \models^{\mathtt{apr}} G(\varphi)$ iff $(U, U \times U, \pi) \models \varphi$ for all $U \in \mathcal{U}^\infty$.*

8 Conclusion

Due to space limitations, a high level of dependence of performance on the application example, and incomparability of the semantics and range of applicability of the algorithms, we omit performance results. However, the algorithms, particularly those for synchronous and asynchronous perfect recall, have successfully been used in a range of nontrivial application studies, e.g. [17, 2, 1], that report performance data.

References

1. Omar I. Al-Bataineh and Ron van der Meyden. Abstraction for epistemic model checking of dining cryptographers-based protocols. *CoRR*, abs/1010.2287, 2010.
2. K. Baukus and R. van der Meyden. A knowledge based analysis of cache coherence. In *ICFEM*, volume 3308 of *LNCS*, pages 99–114. Springer, 2004.
3. R.E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, 1986.
4. J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and L.J. Hwang. Symbolic model checking: 10^{20} states and beyond. In *LICS*. IEEE, 1990.
5. E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. The MIT Press, 1999.
6. K. Engelhardt, P. Gammie, and R. van der Meyden. Model checking knowledge and linear time: PSPACE cases. In *LFCS*, pages 195–211, 2007.
7. R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning About Knowledge*. The MIT Press, 1995.
8. P. Gammie and R. van der Meyden. Mck: Model checking the logic of knowledge. In *CAV*, pages 479–483, 2004.
9. J.Y. Halpern and Y. Moses. Characterizing solution concepts in games using knowledge-based programs. In *IJCAI*, pages 1300–1307, 2007.

10. X. Huang and R. van der Meyden. The complexity of epistemic model checking: Clock semantics and branching time. In *ECAI*, pages 559–554, 2010.
11. O. Kupferman, M. Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. *J. ACM*, 47(2):312–360, 2000.
12. A. Lomuscio, H. Qu, and F. Raimondi. MCMAS: A model checker for the verification of multi-agent systems. In *TACAS*, 2006.
13. A. Lomuscio and F. Raimondi. The complexity of model checking concurrent programs against ctlk specifications. In *AAMAS*, pages 548–550. ACM, 2006.
14. Y. Shoham and K. Leyton-Brown. *Muti-agent Systems: Algorithmic, game-theoretic and logical foundations*. Cambridge University Press, 2009.
15. K. Su, A. Sattar, and X. Luo. Model checking temporal logics of knowledge via OBDDs. *The Computer Journal*, 50(4):403–420, 2007.
16. R. van der Meyden and N. Shilov. Model checking knowledge and time in systems with perfect recall. In *Conference on Foundations of Software Technology and Theoretical Computer Science, LNCS*, volume 1738, pages 432–445, 1999.
17. R. van der Meyden and K. Su. Symbolic model checking the knowledge of the dining cryptographers. In *CSFW*, 2004.