Model Checking Temporal Logics of Knowledge in Distributed Systems

Kaile Su
Department of Computer Science, Zhongshan University
Guangzhou 510275, P.R. China
isskls@zsu.edu.cn

Abstract

Model checking is a promising approach to automatic verification, which has concentrated on specification expressed in temporal logic. Comparatively little attention has been given to temporal logics of knowledge, although such logics have been proven to be very useful in the specifications of protocols for distributed systems. In this paper, we address ourselves to the model checking problem for a temporal logic of knowledge (Halpern and Vardi's logic of CKL_n). Based on the semantics of *interpreted systems* with *local propositions*, we develop an approach to symbolic CKL_n model checking via OBDDs. In our approach to model checking specifications involving agents' knowledge, the knowledge modalities are eliminated via quantifiers over agents' non-observable variables.

Introduction

Model checking is most widely understood as a technique for automatically verifying that finite state systems satisfy formal specifications. The success of model checking in mainstream computer science has led to a recent growth of interest in the use of the technology in fields of AI such as planning and multiagent systems. However, the formal specifications for finite state systems are most commonly expressed as formulae of temporal logics such as LTL (linear temporal logic) in the case of SPIN (Holzmann 1997) and FORSPEC (Vardi 2001) and CTL in the case of SMV (McMillan 1993), while the specifications for multiagent systems involve agents' knowledge, belief and other notions of agents' mental states. In this paper, we address ourselves to the model checking problem for a temporal logic of knowledge (Halpern and Vardi's logic of CKL_n).

The application of model checking within the context of the logic of knowledge was first mooted by (Halpern & Vardi 1991). A number of algorithms for model checking epistemic specifications and the computational complexity of the related problems were studied in (van der Meyden 1998). However, they did not investigate "practical" model checking for knowledge and time.

(Rao & Georgeff 1993) investigated the model checking problem for situated reasoning systems, but they did not consider S5 logics of knowledge and they did not implement any of the techniques they developed. (Benerecetti, Giunchiglia, & Serafini 1999; Benerecetti & Giunchiglia 2000) developed techniques for some temporal modal logics, but these logics have an unusual (non-Kripke) semantics.

(van der Meyden & Su 2004) took a promising first step towards model checking of anonymity properties in formulas involving knowledge. Nevertheless, they took the assumptions that agents are of perfect recall and considered only a small class of epistemic formulas without any nest of epistemic modalities.

(Hoek & Wooldridge 2002) developed an approach to reduce CKL_n model checking to linear temporal logic (LTL) (Pnueli 1977) model checking. However, the verification process of their approach still requires an input from a human verifier (to obtain the so-called local propositions when reducing the CKL_n specification to LTL). A "direct" implementation of CKL_n model checking would thus be desirable.

Our approach presents a methodology for symbolic CKL_n model checking, based on the semantics of *interpreted systems* with *local propositions* (Engelhardt, van der Meyden, & Moses 1998), which leads to a "direct" implementation of CKL_n model checking. Moreover, by the results presented, we can provide via OBDD (Bryant 1986) an approach to symbolic verifying CTL^*, the combination of LTL and CTL (branching temporal logic). This is interesting because LTL and CTL have been well studied and implemented efficiently into a number of tools (Clark, Grumberg, & Peled 2000; Holzmann 1997) and the community of model checking expects such a tool that can verify specifications in full CTL^* efficiently.

The present paper follows similar lines to (Hoek & Wooldridge 2002), which is based on the idea of local propositions as described in (Engelhardt, van der Meyden, & Moses 1998; Engelhardt, van der Meyden, & Su 2002). The main advantages of the present paper over (Hoek & Wooldridge 2002) are:

1. We explicitly introduce the notion of finite-state program with n-agents (which is a symbolic representation of the well-known interpreted systems) and present some interesting results on the theoretical foundations of (Hoek & Wooldridge 2002).
2. In order to determine whether $K_i\varphi$ holds at some point

of an interpreted system, Hoek and Wooldridge (Hoek & Wooldridge 2002) attempt to find an i-local proposition ψ which is equivalent to $K_i\varphi$ at that point; whereas, we try to get an i-local proposition ψ which is equivalent to $K_i\varphi$ at any point (see Remark 11).

The structure of the paper is as follows. In the next section, we shortly introduce the well-known interpreted system (Fagin *et al.* 1995) and a temporal logic of knowledge, Halpern and Vardi's CKL_n (Halpern & Vardi 1989). Then, we define a class of interpreted systems that are generated by finite-state programs with n-agents. The most exciting result is to show how to use OBDDs to implement symbolic CKL_n model checking, based on those interpreted systems generated by finite-state programs with n-agents.

Knowledge in an Interpreted System with Local Variables

In this section, we define the semantic framework within which we study the model checking of specifications in the logic of knowledge. First, we introduce interpreted systems (Fagin *et al.* 1995) and a temporal logic of knowledge CKL_n (Halpern and Vardi's CKL_n (Halpern & Vardi 1989)). Then, we present the notion of a finite-state program with n-agents, a finite-state transition representation for those interpreted systems with local variables

Interpreted systems

The systems we are modelling are composed of multiple agents, each of which is in some state at any point of time. We refer to this as the agent's *local* state, in order to distinguish it from the system's state, the *global* state. Without loss of too much generality, we make the system's state a tuple $(s_1, \cdots, s_n)$, where s_i is agent i's state.

Let L_i be a set of possible local states for agent i, for $i = i, \cdots, n$. We take $G \subseteq L_1 \times \cdots \times L_n$ to be the set of *reachable global* states of the system. A *run* over G is a function from the time domain–the natural numbers in our case–to G. Thus, a run over G can be identified with a sequence of global states in G. We refer to a pair (r, m) consisting of a run r and time m as a *point*. We denote the i'th component of the tuple $r(m)$ by $r_i(m)$. Thus, $r_i(m)$ is the local state of agent i in run r at "time" m.

The idea of the interpreted system semantics is that a run represents one possible computation of a system and a system may have a number of possible runs, so we say a *system* is a set of runs.

Assume that we have a set Φ of primitive propositions, which we can think of as describing basic facts about the system. An *interpreted system* $\mathcal{I}$ consists of a pair $(\mathcal{R}, \pi)$, where $\mathcal{R}$ is a set of runs over a set of global states and π is a valuation function, which gives the set of primitive propositions true at each point in $\mathcal{R}$ (Fagin *et al.* 1995).

To define knowledge in interpreted systems, we associate with every agent i, an equivalence relation $\sim_i$ over the set of points (Fagin *et al.* 1995): $(r, u) \sim_i (r', v)$ iff $r_i(u) = r'_i(v)$.

If $(r, u) \sim_i (r', v)$, then we say that (r, u) and (r', v) are indistinguishable to agent i, or, alternatively, that agent i carries exactly the same information in (r, u) and (r', v).

To give a semantics to the "common knowledge" among a group Γ of agents, two further relations, $\sim_\Gamma^E$ and $\sim_\Gamma^C$, are introduced (Fagin *et al.* 1995). We define the relation $\sim_\Gamma^E$ as $\bigcup_{i\in\Gamma} \sim_i$ and the relation $\sim_\Gamma^C$ as the transitive closure of $\sim_\Gamma^E$.

Notice that a system as a set of infinite runs seems not well suited to model checking directly as it is generally applied to the finite state systems. In fact, we can represent an interpreted system as a *finite-state program* (G, G_0, R, V), where G_0 is a set of initial states, R is a total "next time" relation, and V associates each state with a truth assignment function. A set of infinite runs is then obtained by "unwinding" the relation R starting from initial states in G_0.

Semantics

Given a set Φ of primitive propositions, we use $Prop$ to denote the set of all propositional formulas over Φ.

The *linear temporal logic* LTL (Manna & Pnueli 1995) is propositional logic augmented by the future-time connectives $\bigcirc$ (next) and $\mathbf{U}$ (until). The other future-time connectives $\Diamond$ (sometime or eventually) and $\Box$ (always) can be introduced as abbreviations.

The language of CKL_n is the language of propositional temporal logic augmented by a modal operator K_i for each agent i, and common knowledge operators C_Γ, where Γ is a group of agents. The semantics of CKL_n is given via the satisfaction relation "$\models_{CKL_n}$". Given an interpreted system $\mathcal{I} = (\mathcal{R}, \pi)$ and a point (r, u) in $\mathcal{I}$, we define $(\mathcal{I}, r, u) \models \psi$ by the induction on the structure ψ. The only nontrivial cases are when ψ is of the forms $K_i\varphi$, $C_\Gamma\varphi$, $\bigcirc\varphi$ and $\varphi\mathbf{U}\varphi'$.

- $(\mathcal{I}, r, u) \models_{CKL_n} K_i\varphi$ iff $(\mathcal{I}, r', v) \models_{CKL_n} \varphi$ for all (r', v) such that $(r, u) \sim_i (r', v)$.
- $(\mathcal{I}, r, u) \models_{CKL_n} C_\Gamma\varphi$ iff $(\mathcal{I}, r', v) \models_{CKL_n} \varphi$ for all (r', v) such that $(r, u) \sim_\Gamma^C (r', v)$.
- $(\mathcal{I}, r, u) \models_{CKL_n} \bigcirc\varphi$ iff $(\mathcal{I}, r, (u + 1)) \models_{CKL_n} \varphi$
- $(\mathcal{I}, r, u) \models_{CKL_n} \varphi\mathbf{U}\varphi'$ iff $(\mathcal{I}, r, u') \models_{CKL_n} \varphi'$ for some $u' \geq u$ and $(\mathcal{I}, r, u'') \models_{CKL_n} \varphi$ for all u'' with $u \leq u'' < u'$.

We say that φ is valid in $\mathcal{I}$, denoted by $\mathcal{I} \models_{CKL_n} \varphi$, if $(\mathcal{I}, r, u) \models_{CKL_n} \varphi$ for every point (r, u) in $\mathcal{I}$. We also write $(\mathcal{I}, r, u) \models_{LTL} \varphi$ for $(\mathcal{I}, r, u) \models_{CKL_n} \varphi$ when φ is an LTL formula. For a propositional formula φ, we use $\models \varphi$ to express that φ is a valid formula or tautology.

Finite-state program with n agents

A *finite-state program with n agents* is a finite-state program associated with a set O_i of observable variables for each agent i. To get a symbolic representation of a finite-state program with n agents, we present a symbolic representation of a finite-state program (G, G_0, R, V) in what follows.

1. We use a tuple of boolean variables $\mathbf{x} = \{x_1, \cdots, x_k\}$ and encode a state as an assignment for $\mathbf{x}$, or a subset of $\mathbf{x}$. (For convenience, we sometimes do not distinguish a set and its characteristic function.) Thus, G_0 and any set of states can be represented as a propositional formula over $\mathbf{x}$.

2. Further, we use another tuple of boolean variables $\mathbf{x}' = \{x'_1, \cdots, x'_k\}$ and represent the "next time" relation R between two states as a propositional formula τ over $\mathbf{x} \cup \mathbf{x}'$. In other words, for two assignments s and s' for $\mathbf{x}$, sRs' holds iff $\tau(\mathbf{x}, \mathbf{x}')$ is satisfied by the assignment $s \cup N(s')$, where $N(s')$ denotes $\{x'_j \mid x_j \in s' \text{ and } 0 < j \leq k\}$.

3. We assume that for each s, $V(s)$ equals s, that is, for each variable x_j $(1 \leq j \leq k)$, $V(s)(x_j) = 1$ iff $s(x_j) = 1$.

Omitting the component V, we represent the finite-state program (G, G_0, R, V) just as $(\mathbf{x}, \theta(\mathbf{x}), \tau(\mathbf{x}, \mathbf{x}'))$.

Hence, we formally define a (symbolic) *finite-state program with n agents* as a tuple $\mathcal{P} = (\mathbf{x}, \theta(\mathbf{x}), \tau(\mathbf{x}, \mathbf{x}'), O_1, \cdots, O_n)$, where

1. $\mathbf{x}$ is a set of system variables;
2. θ is a boolean formula over $\mathbf{x}$, called the *initial condition*;
3. τ is a boolean formula over $\mathbf{x} \cup \mathbf{x}'$, called the *transition relation*; and
4. for each i, $O_i \subseteq \mathbf{x}$, containing agent i's *local variables*, or *observable variables*.

Given a state s, we define agent i's local state at state s to be $s \cap O_i$. For convenience, we denote $(s \cap O_1, \cdots, s \cap O_n)$ by $g(s)$. We associate with $\mathcal{P}$ the interpreted system $\mathcal{I}_\mathcal{P} = (\mathcal{R}, \pi)$, where $\mathcal{R}$ is a set of those runs r satisfying that

1. for each m, $r(m)$ is of the form $g(s) = (s \cap O_1, \cdots, s \cap O_n)$ where s is a state in $\mathcal{P}$ and the assignment $\pi(s)$ is the same as s;
2. $r(0)$ is $g(s)$ for some assignment s that satisfies θ;
3. for each natural number m, if $r(m) = g(s)$ and $r(m+1) = g(s')$ for some assignments s and s' for $\mathbf{x}$, then $s \cup N(s')$ is an assignment satisfying $\tau(\mathbf{x}, \mathbf{x}')$.

The interpreted system $\mathcal{I}_\mathcal{P}$ is called *the generated interpreted system of $\mathcal{P}$*.

For convenience, we fix throughout this paper $\mathcal{P} = (\mathbf{x}, \theta(\mathbf{x}), \tau(\mathbf{x}, \mathbf{x}'), O_1, \cdots, O_n)$ to be a finite-state program with n agents.

Local propositions

We now introduce the notion of a *local* proposition (Engelhardt, van der Meyden, & Moses 1998). An i-local proposition is a formula whose interpretation is the same in each of the points in each equivalence class induced by the $\sim_i$ relation. Formally, given an interpreted system $\mathcal{I}$ and an agent i, a formula φ is *i-local* iff for each point (r, u) in $\mathcal{I}$, if $(\mathcal{I}, r, u) \models_{CKL_n} \varphi$, then $(\mathcal{I}, r', u') \models_{CKL_n} \varphi$ for all points (r', u') such that $(r, u) \sim_i (r', u')$. Further, for a set $\Gamma \subseteq \{1, \cdots, n\}$, we say a formula φ is *Γ-local* if φ is *i-local* for each $i \in \Gamma$.

The *model checking* problem for CKL_n we are concerned is the problem of determining whether, given an interpreted system $\mathcal{I} = (\mathcal{R}, \pi)$ and a formula φ, the formula φ is true in the initial state of every run in $\mathcal{R}$. More concisely, given an interpreted system $\mathcal{I}$ and a formula φ, we say that $\mathcal{I}$ realizes φ, denoted by $mc_{CKL_n}(\mathcal{I}, \varphi)$, if for all runs r in $\mathcal{I}$, we have $(\mathcal{I}, r, 0) \models_{CKL_n} \varphi$.

If φ is an LTL formula in the above definition, we also write $mc_{LTL}(\mathcal{I}, \varphi)$ to stand for $mc_{CKL_n}(\mathcal{I}, \varphi)$. We use $l_i(\mathcal{I}_\mathcal{P}, r, u)$ to denote the above formula $(\bigwedge_{x \in r_i(u)} x \wedge \bigwedge_{x \in (O_i - r_i(u))} \neg x)$.

Proposition 1 *A formula φ is i-local in the generated interpreted system $\mathcal{I}_\mathcal{P}$ iff there is a propositional formula ψ containing only variables over O_i such that $mc_{CKL_n}(\mathcal{I}_\mathcal{P}, \Box(\varphi \Leftrightarrow \psi))$.*

Proposition 2 *Let Γ be a set of agents. Then, a formula φ is Γ-local in the generated interpreted system $\mathcal{I}_\mathcal{P}$ iff for each agent i in Γ, there is a propositional formula ψ_i containing only variables over O_i such that $mc_{CKL_n}(\mathcal{I}_\mathcal{P}, \Box(\varphi \Leftrightarrow \psi_i))$.*

We omit the proofs of the two propositions above, which present both necessary and sufficient conditions for i-locality and Γ-locality, respectively, whereas Proposition 1 and 2 in (Hoek & Wooldridge 2002) give only sufficient conditions.

Reachable global states

Let ξ be an operator from the set of boolean formulas over $\mathbf{x}$ to the set of boolean formulas over $\mathbf{x}$. We say ψ is a *fixed point* of ξ, if $\models \xi(\psi) \Leftrightarrow \psi$. We say a ψ_0 is a *greatest fixed point* of ξ, if ψ_0 is a fixed point of ξ and for every fixed point ψ of ξ, we have that $\models \psi \Rightarrow \psi_0$. Clearly, any two greatest fixed points are logically equivalent to each other. Thus, we denote a greatest fixed point of ξ by $\mathbf{gfp}Z\xi(Z)$. Similarly, We say a ψ_0 is a *least fixed point* of ξ, if ψ_0 is a fixed point of ξ and for every fixed point ψ of ξ, we have that $\models \psi_0 \Rightarrow \psi$. A least fixed point of ξ is denoted by $\mathbf{lfp}Z\xi(Z)$. We say ξ is *monotonic*, if for every two formulas ψ_1 and ψ_2 such that $\models \psi_1 \Rightarrow \psi_2$, we have $\models \xi(\psi_1) \Rightarrow \xi(\psi_2)$. For a finite set $\mathbf{x}$ of boolean formulas if ξ is monotonic, then there exist a least fixed point and a greatest fixed point (Tarski 1955).

As usual, for a set of boolean variables $\mathbf{v} = \{v_1, \cdots, v_m\}$, $\exists \mathbf{v}\varphi$ $(\forall \mathbf{v}\varphi)$ stands for $\exists v_1 \cdots \exists v_m \varphi$ $(\forall v_1 \cdots \forall v_m \varphi)$, and $\psi(\frac{\mathbf{x}'}{\mathbf{x}})$ is the result of renaming variables in $\mathbf{x}'$ by those in $\mathbf{x}$ respectively.

Let

$$G(\mathcal{P}) = \mathbf{lfp}Z\left[\theta(\mathbf{x}) \vee (\exists \mathbf{x}(Z \vee \tau(\mathbf{x}, \mathbf{x}'))) \left(\frac{\mathbf{x}'}{\mathbf{x}}\right)\right].$$

The following lemma says that the (quantified) boolean formula $G(\mathcal{P})$ expresses the set of *reachable global states* .

Lemma 3 *The following holds:*

1. $\mathcal{I}_\mathcal{P} \models_{CKL_n} G(\mathcal{P})$.
2. *For a boolean formula φ, $\mathcal{I}_\mathcal{P} \models_{CKL_n} \varphi$ iff $\models G(\mathcal{P}) \Rightarrow \varphi$.*

Symbolic Model Checking CKL_n

The intuition of our approach to symbolic model checking CKL_n is to replace a formula of the form $K_i\varphi$ by some i-local formula ψ. There are two cases depending on whether φ is a pure propositional formula or an LTL formula containing modalities $\mathbf{U}$ or $\bigcirc$.

Model checking knowledge of state properties

First, we consider the case that φ does not contain temporal modalities, this is, φ represents a state property.

Proposition 4 *Let φ be a formula that does not containing any temporal modalities. Then*

$$\mathcal{I}_{\mathcal{P}} \models_{CKL_n} K_i\varphi \Leftrightarrow \forall(\mathbf{x} - O_i)(G(\mathcal{P}) \Rightarrow \varphi).$$

Proof: The conclusion of the proposition follows by Proposition 1 and Lemma 3. ∎

Proposition 5 *Let φ be a formula that does not containing any temporal modalities, Γ a set of agents, and Λ an operator such that*

$$\Lambda(Z) = \bigwedge_{i\in\Gamma} \forall(\mathbf{x} - O_i)(G(\mathcal{P}) \Rightarrow Z).$$

Then

$$\mathcal{I}_{\mathcal{P}} \models_{CKL_n} C_\Gamma\varphi \Leftrightarrow \mathbf{gfp}\ Z(G(\mathcal{P}) \wedge \varphi \wedge \Lambda(Z)).$$

Proof: Omitted for limited space. ∎

By Proposition 4 and 5, when we do the task of model checking CKL_n formula, we can replace formulas of the form $K_i\varphi$ ($C_\Gamma\varphi$) by some i-local (Γ-local) proposition, where φ does not containing any temporal modalities.

Model checking knowledge of temporal properties

Now, we deal with the case that φ may contain some temporal modalities. We use the idea of the so-called tableau construction as descried in (Lichtenstein & Pnueli 1985) and (E.M. Clarke & Hamaguchi 1994). For a formula, ψ, we write $\psi \in \varphi$ to denote that ψ is a sub-formula of (possibly equals to) φ. Formula ψ is called *principally temporal* if its main operator is temporal operator, i.e., ψ is of the form $\bigcirc\alpha$ or $\alpha\mathbf{U}\beta$.

Given a formula φ, we define a finite-state program $\mathcal{P}_\varphi = (\mathbf{x}_\varphi, \theta_\varphi, \tau_\varphi, O_1, \cdots, O_n)$ as follows.

System variables: The set $\mathbf{x}_\varphi$ of system variables of $\mathcal{P}_\varphi$ consists of $\mathbf{x}$ plus a set of auxiliary boolean variables

$$X_\varphi : \{x_\psi \mid \psi \text{ is a principally temporal sub-formula of } \varphi\}.$$

The auxiliary variable x_ψ is intended to be true in a state of a computation iff the temporal formula ψ holds at the state.

For convenience, we define a function χ which maps every sub-formula of φ into a boolean formula over $\mathbf{x} \cup X_\varphi$.

$$\chi(\psi) = \begin{cases} \psi & \text{for } \psi \text{ a variable in } \mathbf{x} \\ \neg\chi(\alpha) & \text{for } \psi = \neg\alpha \\ \chi(\alpha) \wedge \chi(\beta) & \text{for } \psi = \alpha \wedge \beta \\ x_\psi & \text{for principally temporal } \psi \end{cases}$$

Let X'_φ be the primed version of X_φ. For a formula ψ over $\mathbf{x} \cup X_\varphi$, we use $\chi'(\psi)$ to denote the formula $\psi(\frac{\mathbf{x}\cup X_\varphi}{\mathbf{x}'\cup X'_\varphi})$, i.e., the primed version of ψ.

Initial condition: The initial condition of $\mathcal{P}_\varphi$ is the same as for $\mathcal{I}_{\mathcal{P}}$.

Transition relation: The transition relation τ_φ of $\mathcal{P}_\varphi$ is the transition relation τ plus

$$\bigwedge_{\bigcirc\psi\in\varphi} (x_{\bigcirc\psi} \Leftrightarrow \chi'(\psi)) \wedge \bigwedge_{\alpha\mathbf{U}\beta\in\varphi} (x_{\alpha\mathbf{U}\beta} \Leftrightarrow (\chi(\beta) \vee (\chi(\alpha) \wedge x'_{\alpha\mathbf{U}\beta})))$$

For convenience, we introduce now some more notations from the CTL logic (Clark, Grumberg, & Peled 2000). Let $\mathbf{EX}$ be the operator such that for a boolean formula ψ over $\mathbf{x} \cup X_\varphi$,

$$\mathbf{EX}\psi(\mathbf{x}, X_\varphi) = \exists(\mathbf{x}' \cup X'_\varphi)(\psi(\mathbf{x}', X'_\varphi) \wedge \tau_\varphi).$$

In other words, the set of those states satisfying $\mathbf{EX}\psi(\mathbf{x}, X_\varphi)$ is the image of the set of those states satisfying ψ under the transition relation τ_φ.

The operators $\mathbf{EF}$ and $\mathbf{EU}$ are defined by the least fixed point of some monotonic operators: $\mathbf{EF}f = \mathbf{lfp}\ Z(f \vee \mathbf{EX}Z)$, and $\mathbf{EU}(f, g) = \mathbf{lfp}\ Z(g \vee (f \wedge \mathbf{EX}Z))$.

Let $\mathcal{J}_\varphi$ be the set of all formulas $\neg x_{\alpha\mathbf{U}\beta} \vee \chi(\beta)$, where $\alpha\mathbf{U}\beta$ is a sub-formula of φ. To give the knowledge of agent i at some state, we consider the following fairness constraints:

C^1_φ: There is a computational path for which each formula in $\mathcal{J}_\varphi$ holds infinitely often.

C^2_φ: There is a finite computational path such that each formula in $\mathcal{J}_\varphi$ holds at the last state of the computational path, and the last state does not have any next state in the system $\mathcal{I}_{\mathcal{P}}$.

Clearly, if C^1_φ holds with $\mathcal{J}_\varphi \neq \emptyset$, then there is a computational path which is infinitely long. If C^2_φ holds, then there is a finite computational path at which the last state does not have a next state in the system $\mathcal{I}_{\mathcal{P}}$.

We suppose that $\mathcal{J}_\varphi$ is not an empty set. This assumption does not lose any generality because we can put $\mathbf{true}$ in $\mathcal{J}_\varphi$.

The constrain C^2_φ can be expressed as $\mathbf{EF}(End(\mathcal{P}) \wedge \bigwedge_{\psi\in\mathcal{J}_\varphi} \psi)$ in the standard CTL logic, where $End(\mathcal{P})$ is the formula related to the set of dead states in the system $\mathcal{I}_{\mathcal{P}}$, it can be represented as $\neg\exists\mathbf{x}'\tau(\mathbf{x}, \mathbf{x}')$.

The constrain C^1_φ can be defined as:

$$C^1_\varphi = \mathbf{gfp}\ Z[\bigwedge_{J\in\mathcal{J}_\varphi} \mathbf{EX}(\mathbf{EU}(\mathbf{true}, Z \wedge J))].$$

It is not difficult to see that a state satisfies the condition C^1_φ iff the state is at some run where each $J \in \mathcal{J}_\varphi$ holds for infinite times along the run (Clark, Grumberg, & Peled 2000).

We say a run r_φ in $\mathcal{I}_{\mathcal{P}_\varphi}$ is *fair*, if either r_φ is infinitely long and each J in $\mathcal{J}_\varphi$ is satisfied by infinitely many states at r_φ, or there is a state, say s_{end}, such that $s_{end} \cap \mathbf{x}$ has no successor in $\mathcal{I}_{\mathcal{P}}$ and each $J \in \mathcal{J}_\varphi$ is satisfied by s_{end}. It follows the following assertion.

Lemma 6 *Let φ be an LTL formula, s_φ a state of $\mathcal{I}_{\mathcal{P}_\varphi}$. Then, s_φ satisfies $C^1_\varphi \vee C^2_\varphi$ iff s_φ is at some fair run of $\mathcal{I}_{\mathcal{P}_\varphi}$.*

Lemma 7 *Let φ be an LTL formula. Then for each run r in $\mathcal{I}_{\mathcal{P}}$, there is a fair run r_φ in $\mathcal{I}_{\mathcal{P}_\varphi}$ such that for every natural number u and for every subformula ψ of φ,*

1. $r(u) = r_\varphi(u) \cap \mathbf{x}$,
2. $(\mathcal{I}_\mathcal{P}, r, u) \models_{LTL} \psi$ *iff* $\chi(\psi)$ *is satisfied by* $r_\varphi(u)$.

Proof: Let r be a run in $\mathcal{I}_\mathcal{P}$. We define a fair run r_φ in $\mathcal{I}_{\mathcal{P}_\varphi}$ as follows. For each point $r(u)$, let $r_\varphi(u)$ be the variable set

$$r(u) \cup \left\{ x_\alpha \;\middle|\; \begin{array}{l} \alpha \text{ is principally temporal subformula of } \varphi \\ \text{and } (\mathcal{I}_\mathcal{P}, r, u) \models_{LTL} \alpha \end{array} \right\}$$

It follows immediately that $r(u) = r_\varphi(u) \cap \mathbf{x}$ and, for a principally temporal subformula ψ of φ, we have that $(\mathcal{I}_\mathcal{P}, r, u) \models_{LTL} \psi$ iff $\chi(\psi)$ is satisfied by $r_\varphi(u)$. For other subformula ψ of φ, we can prove the above assertion holds by induction on ψ. It is also easy to see that r_φ is a run in $\mathcal{I}_{\mathcal{P}_\varphi}$ and r_φ is fair. ∎

Lemma 8 *Let φ be as in Lemma 7. Then, for each fair run r_φ in $\mathcal{I}_{\mathcal{P}_\varphi}$, there is a run r in $\mathcal{I}_\mathcal{P}$ such that for every natural number u and for every subformula ψ of φ,*

1. $r(u) = r_\varphi(u) \cap \mathbf{x}$,
2. $(\mathcal{I}_\mathcal{P}, r, u) \models_{LTL} \psi$ *iff* $\chi(\psi)$ *is satisfied by* $r_\varphi(u)$.

Proof: Let r_φ be a fair run in $\mathcal{I}_{\mathcal{P}_\varphi}$. We define a run r in $\mathcal{I}_\mathcal{P}$ simply by $r(u) = r_\varphi(u) \cap \mathbf{x}$ for each state $r_\varphi(u)$.

We assume that r_φ is infinitely long because the other case, where r_φ is finite, can be dealt with in the same way.

Given a subformula ψ of φ, we show, by induction on the structure of ψ, the claim that $(\mathcal{I}_\mathcal{P}, r, u) \models_{LTL} \psi$ iff $\chi(\psi)$ is satisfied by $r_\varphi(u)$. The conclusion of the lemma follows by the above claim. ∎

We now extend the logic CKL_n by introducing two path quantifiers **A** and **E**. The resulting language is denoted by $ECKL_n$. For a finite-state program $\mathcal{P}$ with n agents, a run r in $\mathcal{I}_\mathcal{P}$, a formula ψ, and a natural number u, we have $(\mathcal{I}_\mathcal{P}, r, u) \models_{CKL_n} \mathbf{E}\psi$ iff there is a run r' such that, for some natural number v, $r(u) = r'(v)$ and $(\mathcal{I}_\mathcal{P}, r', v) \models_{CKL_n} \psi$. We define $\mathbf{A}\psi$ as $\neg\mathbf{E}\neg\psi$.

Clearly, if we remove knowledge modalities from $ECKL_n$, we get the well-known logic CTL^*. The following proposition presents a methodology of implementing symbolic verifying CTL^* via OBDDs.

Proposition 9 *Let φ be an LTL formula. Then*

$$\mathcal{I}_\mathcal{P} \models_{CKL_n} \mathbf{E}\varphi \Leftrightarrow \exists X_\varphi((C^1_\varphi \vee C^2_\varphi) \wedge \chi(\varphi)).$$

Proof: ($\Rightarrow$) Assume that $(\mathcal{I}_\mathcal{P}, r, u) \models_{CKL_n} \mathbf{E}\varphi$. There is a run r' and a natural number v such that $r'(v) = r(u)$ and $(\mathcal{I}_\mathcal{P}, r', v) \models_{CKL_n} \varphi$. By Lemma 7, there is a fair run r_φ in $\mathcal{I}_{\mathcal{P}_\varphi}$, such that $r_\varphi(v)$ satisfies $\chi(\varphi)$ iff $(\mathcal{I}_\mathcal{P}, r, v) \models_{CKL_n} \varphi$. Thus, $r_\varphi(v)$ satisfies $\chi(\varphi)$. Moreover, because r_φ is a fair run, every state at run r_φ must satisfy $C^1_\varphi \vee C^2_\varphi$. So, $r_\varphi(v)$ satisfies $(C^1_\varphi \vee C^2_\varphi) \wedge \chi(\varphi)$. Because $r(u) = r'(v) = r_\varphi(v) \cap \mathbf{x}$, we have that $r(u) \cup (r_\varphi(v) \cap X_\varphi) = r_\varphi(v)$ satisfies $(C^1_\varphi \vee C^2_\varphi) \wedge \chi(\varphi)$. This is, $(\mathcal{I}_\mathcal{P}, r, u) \models_{CKL_n} \exists X_\varphi((C^1_\varphi \vee C^2_\varphi) \wedge \chi(\varphi))$.

($\Leftarrow$) Suppose that $(\mathcal{I}_\mathcal{P}, r, u) \models_{CKL_n} \exists X_\varphi((C^1_\varphi \vee C^2_\varphi) \wedge \chi(\varphi))$. Then, $r(u)$ satisfies $\exists X_\varphi((C^1_\varphi \vee C^2_\varphi) \wedge \chi(\varphi))$, and there is a state s_φ in $\mathcal{I}_{\mathcal{P}_\varphi}$ such that $r(u) = s_\varphi \cap \mathbf{x}$, and s_φ satisfies $((C^1_\varphi \vee C^2_\varphi) \wedge \chi(\varphi))$. By the fact that s_φ satisfies $C^1_\varphi \vee C^2_\varphi$ and Lemma 6, we get that s_φ is at some fair run r_φ in $\mathcal{I}_{\mathcal{P}_\varphi}$, and there is a natural number v such that $s_\varphi = r_\varphi(v)$. By Lemma 8, there is a run r' in $\mathcal{I}_\mathcal{P}$ such that $r'(v) = r_\varphi(v) \cap \mathbf{x}$ and $(\mathcal{I}_\mathcal{P}, r', v) \models_{CKL_n} \varphi$ iff $r_\varphi(v)$ satisfies $\chi(\varphi)$. Recalling $r_\varphi(v) = s_\varphi$ and $r(u) = s_\varphi \cap \mathbf{x}$, we have $r'(v) = r(u)$. Moreover, by the fact that $r_\varphi(v)$ satisfies $\chi(\varphi)$, we have that $(\mathcal{I}_\mathcal{P}, r', v) \models_{CKL_n} \varphi$. Hence, $(\mathcal{I}_\mathcal{P}, r, u) \models_{CKL_n} \mathbf{E}\varphi$. ∎

Now follow the main results in this section.

Proposition 10 *Let φ be an LTL formula. Then, the following formula*

$$K_i\varphi \Leftrightarrow \forall(X_\varphi \cup \mathbf{x} - O_i)((C^1_\varphi \vee C^2_\varphi) \wedge G(\mathcal{P}) \Rightarrow \chi(\varphi))$$

is valid in $\mathcal{I}_\mathcal{P}$.

Proof: We first notice that the formula $K_i\varphi \Leftrightarrow K_i\mathbf{A}\varphi$ is valid in $\mathcal{I}_\mathcal{P}$. Proposition 9 says that the formula $\mathbf{A}\varphi \Leftrightarrow \forall X_\varphi((C^1_\varphi \vee C^2_\varphi) \Rightarrow \chi(\varphi))$ is valid. Hence,

$$\mathcal{I}_\mathcal{P} \models_{CKL_n} K_i\varphi \Leftrightarrow K_i(\forall X_\varphi((C^1_\varphi \vee C^2_\varphi) \Rightarrow \chi(\varphi))).$$

By Proposition 4, the following formula

$$K_i\varphi \Leftrightarrow \forall(\mathbf{x} - O_i)(G(\mathcal{P}) \Rightarrow \forall X_\varphi((C^1_\varphi \vee C^2_\varphi) \Rightarrow \chi(\varphi)))$$

must be valid in $\mathcal{I}_\mathcal{P}$. Because variables in X_φ do not appear in $G(\mathcal{P})$, the formula

$$K_i\varphi \Leftrightarrow \forall(X_\varphi \cup \mathbf{x} - O_i)((C^1_\varphi \vee C^2_\varphi) \wedge G(\mathcal{P}) \Rightarrow \chi(\varphi))$$

is thus valid in $\mathcal{I}_\mathcal{P}$. ∎

Remark 11 In order to determine whether $K_i\varphi$ holds at some point of an interpreted system, Hoek and Wooldridge (Hoek & Wooldridge 2002) attempt to find an i-local proposition ψ such that $K_i\varphi$ holds iff ψ holds at that point. However, how to get such an i-local proposition ψ was not presented in (Hoek & Wooldridge 2002). In addition, the local-proposition formula ψ may depend on the point at which we check $K_i\varphi$ (see Proposition 5 in (Hoek & Wooldridge 2002)). Thus, when faced with the problem of determining whether some point satisfies a formula α with a subformula of the form $K_i\varphi$, we could not reduce the problem to determining whether the point satisfies the formula $\alpha(\frac{K_i\varphi}{\psi})$ (which results from α by replacing $K_i\varphi$ with ψ.) The main advantage of Proposition 10 over Hoek and Wooldridge's results is that the i-local proposition ψ is given out (i.e. $\forall(X_\varphi \cup \mathbf{x} - O_i)((C^1_\varphi \vee C^2_\varphi) \wedge G(\mathcal{P}) \Rightarrow \chi(\varphi))$) and the proposition ψ does not depend on the point (r, u).

We also remark that Proposition 10 provides a reduction of CKL_n to LTL, while Proposition 9 gives a method of model checking LTL formulas. The complexity of our reduction of CKL_n to LTL is $PSPACE$-complete. Nevertheless, because C^1_φ, C^2_φ and quantifications of boolean functions can be dealt with in any $OBDD$ package, the reduction of CKL_n to LTL and the LTL model checking method can be based on OBDDs. Thus, the CKL_n model checking algorithm via Proposition 10 and 9 might be practically implementable.

As for model checking common knowledge of temporal properties, we can see the following proposition holds.

Proposition 12 *Let φ be a formula that may contain some temporal modalities, Λ an operator such that*

$$\Lambda(Z) = \bigwedge_{i\in\Gamma} \forall(\mathbf{x} - O_i)(G(\mathcal{P}) \Rightarrow Z).$$

Then, the following formula is valid in $\mathcal{I}_{\mathcal{P}}$:

$$C_\Gamma\varphi \Leftrightarrow \mathbf{gfp}\ Z[G(\mathcal{P}) \wedge \forall X_\varphi((C^1_\varphi \vee C^2_\varphi) \Rightarrow \chi(\varphi)) \wedge \Lambda(Z)]$$

Proof: By Proposition 9 and Proposition 5. ∎

Conclusions

In this paper, we have considered the model checking problem for Halpern and Vardi's well-known temporal epistemic logic CKL_n. We have introduced the notion of a finite state program with n agents, which can be thought of as a symbolic representation of interpreted systems. We have developed an approach to symbolic CKL_n model checking, using OBDDs. In our approach to model checking specifications involving agents' knowledge, the knowledge modalities are eliminated via quantifiers over agents' non-observable variables. As a by-product, we have presented a methodology of implementing symbolic verifying CTL^* via OBDDs.

We are currently working on an implementation of a CKL_n model checker based on the results in this paper, via CUDD library developed by Fabio Somenzi at Colorado University. We have founded the prototype of the model checking system and finished the kernel part of it. Because of limited space, we do not include experimental results here. As for future work, we are interested in providing automated support for the analysis of knowledge in distributed system protocols and game theoretic examples, and the verification and compilation of knowledge-based programs (Fagin *et al.* 1995).

Acknowledgement

Thanks to Yanyan Xu, Qingliang Chen and the AAAI reviewers for their valuable comments. This work was supported by the National Science Foundation of China under grants 60073056 and 60273062.

References

Benerecetti, M., and Giunchiglia, F. 2000. Model checking security protocols using a logic of belief. In Graf, S., and Schwartzbach, M., eds., *Proc. 6th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2000)*, 519–534.

Benerecetti, M.; Giunchiglia, F.; and Serafini, L. 1999. A model checking algorithm for multi-agent systems. In Muller, J.; Singh, M.; and Rao, A., eds., *Intelligent Agents V*, volume LNAI Vol. 1555. Berlin: Springer-Verlag.

Bryant, R. 1986. Graph-based algorithms for boolean function manipulation. *IEEE Transaction on Computers* (C-35(8)).

Clark, E.; Grumberg, O.; and Peled, D. 2000. *Model Checking*. Cambridge, MA: The MIT Press.

E.M. Clarke, O. G., and Hamaguchi, K. 1994. Another look at LTL model checking. In *Proc. 6th Conference on Computer Aided Verification*, 415–427. Springer LNCS Vol. 818.

Engelhardt, K.; van der Meyden, R.; and Moses, Y. 1998. Knowledge and the logic of local propositions. In *Theoretical Aspects of Rationality and Knowledge, Proc. of TARK 1998*.

Engelhardt, K.; van der Meyden, R.; and Su, K. 2002. Modal logics with a hierarchy of local propositional quantifiers (preliminary version). In *Advance in Modal Logic 2002 (AiML)*, 63–76.

Fagin, R.; Halpern, J.; Moses, Y.; and Vardi, M. 1995. *Reasoning about knowledge*. Cambridge, MA: MIT Press.

Halpern, J., and Vardi, M. 1989. The complexity of reasoning about knowledge and time, I: Lower bounds. *Journal of Computer and System Sciences* 38(1):195–237.

Halpern, J., and Vardi, M. 1991. Model checking vs. theorem proving: A manifesto. In *Proc. 2nd Int. Conf. on Principles of Knowledge Representation and Reasoning*, 325–334.

Hoek, W. v. d., and Wooldridge, M. 2002. Model checking knowledge and time. In *9th Workshop on SPIN (Model Checking Software)*.

Holzmann, G. 1997. The spin model checker. *IEEE Transaction on Software Engineering* 23:279–295.

Lichtenstein, O., and Pnueli, A. 1985. Checking that finite-state concurrent programs satisfy their linear apecification. In *Proc. 12th ACM Symp. of Prog. Lang.*, 97–107.

Manna, Z., and Pnueli, A. 1995. *Temporal Verification of Reactive Systems*. Berlin, Germany: Springer-verlag.

McMillan, K. 1993. *Symbolic Model Checking*. Boston: Kluwer Academic Publisher.

Pnueli, A. 1977. The temporal logic of programs. In *Proc. 18th IEEE Symposium on Foundations of Computer Science*, 46–57.

Rao, A., and Georgeff, M. 1993. A model theoretic approach to the verification of situated reasoning systems. In *Proc. 13th International Joint Conference on Artificial Intelligence*, 318–324.

Tarski, A. 1955. A lattice-theoretical fixpoint theorem and its applications. *Pacific J. Math.* 5:285–309.

van der Meyden, R., and Su, K. 2004. Symbolic model checking the knowledge of the dining cryptographers. In *Proc. of 17th IEEE Computer Security Foundation Workshop*.

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

Vardi, M. 2001. Branching vs. linear time:. In Margaria, T., and Yi, W., eds., *Proc. 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2001)*, 1–22.