Model Checking Epistemic and Probabilistic Properties of Multi-agent Systems

Wei Wan[1], Jamal Bentahar[2], and Abdessamad Ben Hamza[2]

[1] Department of Electrical and Computer Engineering, Concordia Univeristy
[2] Concordia Institute for Information Systems Engineering, Concordia University
w_wan@encs.concordia.ca, {bentahar,hamza}@ciise.concordia.ca

Abstract. Model checking, a formal automatic verification method, has been widely used in multi-agent systems to verify specifications that contain qualitative properties (e.g safety and liveliness) and quantitative properties. Decision making processes based on inherent knowledge are necessary for agents to act appropriately, particularly in uncertain settings. In order to check epistemic (i.e knowledge) and measurable properties in multi-agent systems, we propose a new logic $PCTLK$, which uses probabilistic, epistemic, and temporal modal operators. We exploit Discrete-Time Markov Chains (DTMC), in which we are able to represent measurable properties with probability, to model uncertainty in multi-agent systems. We extend the formalism of interpreted systems by adding probabilistic features to suit DTMC models and to present the model checking algorithm for our logic. At the end of this paper, we simulate our algorithm using an example of online shopping.

Keywords: Probabilistic model checking, Discrete-Time Markov Chains (DTMC), Epistemic logic, Probabilistic logic, Interpreted systems.

1 Introduction

Multi-agent systems are comprised of a set of intelligent agents interacting with each other. In such systems, reasoning with uncertainty is an important feature. Model checking, a well-designed formal technique for verifying if models satisfy specific properties, is widely used to verify the epistemic properties of these systems [9,10,13]. In the past two years, researchers in [3,4,7] have proposed methods to model and check multi-agent systems using Markov chains, which are stochastic processes behaving as transition systems extended by probabilistic choices among successor states.

The reason why probabilistic model checking is gaining more and more interest is that classical model checking techniques focus only on the absolute guarantee of correctness and some assumptions are made world that systems run ideally. However, in practice concrete scenarios are characterized by a degree of uncertainty. For example, in multi-agent systems, because of the agents' autonomous reactivity, their actions are based on observing the environment changes, but very often agents cannot observe the complete environment. Therefore, agents

K.G. Mehrotra et al. (Eds.): IEA/AIE 2011, Part II, LNAI 6704, pp. 68–78, 2011.

must make estimations through the observable world as a part of their decision-making processes. Considering probabilistic aspects when modeling the system allows providing such estimations by assessing the likelihood of different events.

Two categories of properties exist [1]: immeasurable *qualitative properties* and measurable *quantitative properties*. For qualitative properties, Boolean variables (true or false) can be used to express whether a good event will happen or a bad event will never happen. However, sometimes the likelihood of certain events need to be considered. For example, we may want to know "after ordering, what percentage of items will be successfully delivered". In this case, qualitative properties cannot provide the precise probability required. Quantitative properties are also necessary to accurately express the system requirements, and probabilistic model checking seems to be appropriate.

Checking quantitative properties for knowledge-based multi-agent systems is the main motivation of this paper. Because the majority of existing multi-agent logics only consider certain knowledge, modeling and verifying uncertain knowledge still need more investigation. Uncertain knowledge can be represented using probabilities and the multi-agent system can be modeled as a probabilistic Kripke-like model. In this paper, we define a new logic called probabilistic temporal logic of knowledge $PCTLK$, that extends $PCTL$, the probabilistic computation tree logic proposed in [6], by the K operator of knowledge. We also extend the classical interpreted systems used to model multi-agent systems [5] by adding probabilistic transition functions. This extension allows us to model interpreted systems as Discrete-Time Markov Chains (DTMC), which are transition systems with probability distributions. Two reasons are behind the use of DTMC; first, DTMC is the underlying formal model of $PCTL$, and second, DTMC are widely used to model systems with uncertain information. In order to verify properties expressed with $PCTLK$, we put forward the model checking algorithm for this new logic.

The structure of this paper is organized as follows. In Section 2, we explain how we extend the normal interpreted systems to be modeled using DTMC. In Section 3, we present the syntax and semantics of the $PCTLK$ logic. The model checking problem of $PCTLK$ is described in Section 4. We provide an example, customer online shopping, to simulate the model checking algorithm in Section 5. Finally, we summarize our work and list the future work in Section 6.

2 Interpreted Systems for $DTMC$

Probability is used in the design and analysis of an agent to measure the likelihood that some specific events will occur. There are several methods of systems modeling to express the probability attributes. One of the most popular operational probabilistic models is Markov chain [1], which is a transition system with probability distributions over the transitions. DTMC is one of three Markov chain models, in which a system is in a given state at each "step", with the state changing randomly between steps.

In Markov chains, the process can only be in a finite set of states. Over a set of atomic propositions AP, a model of DTMC can be expressed as a tuple $\{S, P, I_{init}, L\}$, where:

- S is a nonempty and finite set of states.
- $P : S \times S \to [0, 1]$ is the transition probability function, such that for every state $s \in S$, we have $\sum_{s' \in S} P(s, s') = 1$.
- $I_{init} : S \to [0, 1]$ is the initial distribution such that for all states $s \in S$, $\sum_{s \in S} I_{init}(s) = 1$.
- $L : S \to 2^{AP}$ is a state labeling function.

For mathematics treatment purposes, the initial distribution I_{init} can be viewed as an ordered list of row vector $(I_{init}(s))_{s \in S}$, in which the value of every row represents the initial probability from all states in the model. The transition probability function $P : S \times S \to [0, 1]$ is represented by the matrix $(P(s, t))_{s,t \in S}$. The probabilities of state s to its successors t are shown on the row of the matrix, while the probabilities of entering state s from start t are shown on the column of the matrix.

According to Fagin et al in [5], in order to model knowledge, we need a framework based on a number of other possible states of affairs besides the true state of affairs. The formalism of interpreted systems, which provides a well-defined semantics to reason about time and knowledge in multi-agent systems, is frequently used in epistemic model checkers, such as MCMAS [11] and MCK[13]. We extend Fagin et al.'s interpreted systems with probability attributes. We then introduce our probabilistic model checking algorithm.

Let $A = \{1, \ldots, n\}$ be a set of n agents in the system. Every agent $i \in A$ is associated with its local state set L_i, and possible actions set Act_i. Besides L_i and Act_i, there are also a set L_e and a set Act_e for the environment, a special agent for providing global variables and actions that all agents are able to access. Therefore, for the system, a set of global states $G \subseteq L_1 \times \ldots \times L_n \times L_e$ is the set of all possible tuples $(l_1, \ldots, l_n, l_e)$, and each tuple represents a computational state for the whole system. For each agent i, we also use a set of protocols $P_i : L_i \to 2^{Act_i}$ assigning a list of enabled actions to each local state. Associated with the environment is a protocol $P_e : L_e \to 2^{Act_e}$ that represents the functioning behavior of the environment agent. The probabilistic transition function T for the system can be defined as $T : G \times Act \times P_{Act} \times G \to [0, 1]$, where Act is the set of joint actions $Act \subseteq Act_1 \times \ldots \times Act_n \times Act_e$ that are performed by all the agents and environment respectively. Each agent is associated with a local probabilistic transition function $t_i \subseteq T$. P_{Act} is a probability function of the set of joint actions. For every global state $g \in G$, with transition to state $g' \in G$, $\sum_{g' \in G} P_{Act}(g, g') = 1$. P_{Act_i} is the local probability function for each agent i. Given a global initial distribution I_{init}, for all states $s \in G$, $\sum I_{init}(s) = 1$. Given a set of atomic propositions AP and an interpretation $V \subseteq G \times AP$, an interpreted system over probabilistic transition function is a tuple: $IS =< (L_i, Act_i, t_i, P_{Act_i}, P_i)_{i \in A}, I_{init}, V >$

Now, we need to associate DTMC, which is a Kripke-like structure extended with probabilities, to our extended interpreted system in order to evaluate

epistemic probabilistic properties. The resulting model is $M_{IS} = (W, P_t, I'_{init}, \sim_1, \ldots, \sim_n, V')$. The converting rules from Interpreted system IS to our DTMC interpreted system M_{IS} are described as follow.

- $W \subseteq G$ is the set of reachable states. The state is reachable if and only if the probability is greater than 0 from initiation distribution via T.
- $I'_{init} \subseteq I_{init}$ is the initial distribution of the model.
- $P_t : W \times W \to [0, 1]$ is a probability relation, which is obtained by protocols P_i and the probabilistic functions t_i, which are subsets of the transition probability function T;
- $\sim_i$ is the epistemic relations, one for each agent. It is a subset of $W \times W$ for $i \in A$, such that for two global states, $(l_1, \ldots, l_n) \sim_i (l'_1, \ldots, l'_n)$ if and only if $l_i = l'_i$;
- $V' \subseteq V$ is the valuation function $V' : W \times AP \to \{true, false\}$.

3 *PCTLK* Logic

The language we will use to specify properties is called Probabilistic Computation Tree Logic of Knowledge or $PCTLK$. This branching-time logic extends Computation Tree Logic (CTL) proposed by Clark et al. in [2] by adding epistemic and probabilistic operators. Thus this language combines CTL logic, epistemic logic [5], and probabilistic logic [1,6].

3.1 Syntax of *PCTLK*

$PCTLK$ supports two kinds of formulae: state formulae ϕ and path formulae ψ. We use $p, p_1, p_2, \ldots$ to range over the set of atomic propositions Φ_p. Given a set of agents $A = 1, \ldots, n$, the syntax of this logic is as follows:

$$\phi ::= true \mid p \mid \phi \ \wedge \ \phi \mid \neg\phi \mid K_i\phi \mid Pr_{\nabla b}(\psi)$$
$$\psi ::= \bigcirc\phi \mid \phi \ \cup^{\leq n} \ \phi \mid \phi \ \cup \ \phi$$

where $b \in [0, 1]$ is an interval giving rational boundary for the path, $\nabla \in \{<, \leq, \geq, >\}$ is a relational operator indicating the kind of relationship boundary of the probability, and $n \in \mathbb{N}$ is an integer indicating the maximum steps to achieve a specific state. There are no universal ($\forall$) and existential ($\exists$) path quantifiers in $PCTLK$. Instead, the linear temporal operators $\bigcirc$ (next) and $\cup$ (until) are required to follow the probabilistic operator Pr immediately. By removing $K_i\phi$ from this logic we obtain $PCTL$ [1,6] and by removing $\phi \ \cup^{\leq n} \ \phi$ from $PCTLK$ and introducing the $\exists$ quantifier we obtain $CTLK$ [5].

Three fragments, a temporal logic, an epistemic logic, and a probabilistic logic, are defined for $PCTLK$ in addition to the standard Boolean connectives. The propositional temporal fragment has the same meaning as in CTL. For example, the formula $\bigcirc\phi$ has the meaning of "next step ϕ holds". $\phi_1 \cup \phi_2$ means "ϕ_1 holds until ϕ_2". A new step-bounded variant of $\phi_1 \cup^{\leq n} \phi_2$ is added, meaning that ϕ_2 will hold within at most n steps while ϕ_1 holds in all states before ϕ_2

state has been reached. The step-bounded until is necessary in probabilistic logic because the probability for at most n steps to reach ϕ_2 may be different from at most $n+1$ steps to reach ϕ_2. As in CTL, temporal operators $\bigcirc$ and $\cup$ are required to be immediately preceded by a path quantifier; in $PCTLK$, they must follow the operator Pr immediately. Other Boolean connectives and operators are derived in the same way as in CTL, for example, $\Diamond\phi = true \cup \phi$ (in the future ϕ). The step-bounded future can be similarly obtained by step-bounded until: $\Diamond^{\leq n}\phi = true \cup^{\leq n} \phi$. The always operator can be derived by formula: $Pr_{\leq b}(\Box\phi) = Pr_{\geq(1-b)}(\Diamond\neg\phi)$. The details can be found in [1].

The probabilistic operator $Pr_{\nabla b}(\psi)$ expresses that "ψ is true with probability ∇b". For illustration, $Pr_{\geq 0.9}(\bigcirc message\_receive)$ asserts that "with probability at least 0.9, the next step message will be received".

The epistemic operator K_i is a S5 modality [10], which means knowledge is reflexive and Euclidean [12]. $K_i\phi$ represents "agent i knows that ϕ". There are some important properties of the epistemic operator. For examples, $K_i\phi \wedge K_i(\phi \Rightarrow \psi) \Rightarrow K_i\psi$ means when an agent knows that ϕ and knows that $\phi \Rightarrow \psi$, the agent knows ψ as well. Agents also know what they know and what they do not know. These properties can be expressed as $K_i\phi \Rightarrow K_iK_i\phi$ and $\neg K_i\phi \Rightarrow K_i\neg K_i\phi$. More properties and their proof can be found in [5]. We have also interesting properties combining knowledge and probabilities, for example: $K_i(Pr_{\geq 0.5} \bigcirc \phi) \Rightarrow Pr_{\geq 0.5} \bigcirc \phi$.

3.2 *PCTLK* Semantics

Let $s \in S$ be a state, π a given path, $a \in AP$ an atomic proposition, ϕ a $PCTLK$ state formula, and ψ a $PCTLK$ path formula. A path is an infinite sequence of states related by transitions, i.e., $\pi = s_0, s_1, s_2, \ldots$ The $(i+1)th$ state in π is denoted $\pi(i)$ i.e., $\pi(i) = s_i$. $s \models \phi$ denotes "s satisfies ϕ" or "ϕ is true in s". $\pi \models \psi$ denotes "π satisfies ψ" or "ψ is true in π". The semantics of $PCTLK$ is as follows.

- For a state s

$$\begin{array}{ll} s \models a & \textit{iff } V(s,a) = true \\ s \models \phi_1 \wedge \phi_2 & \textit{iff } s \models \phi_1 \text{ and } s \models \phi_2 \\ s \models \neg\phi & \textit{iff } s \not\models \phi \end{array}$$

- For a path ϕ:

$$\begin{array}{ll} \pi \models \bigcirc\phi & \textit{iff } \pi(1) \models \phi \\ \pi \models \phi_1 \cup^{\leq n} \phi_2 & \textit{iff } \exists 0 \leq k \leq n,\ \pi(k) \models \phi_2 \ \textit{ and } \ \forall 0 \leq i < k \ \ \pi(i) \models \phi_1 \\ \pi \models \phi_1 \cup \phi_2 & \textit{iff } \exists k \geq 0,\ \ \pi(k) \models \phi_2 \ \textit{ and } \ \forall 0 \leq i < k \ \ \pi(i) \models \phi_1 \end{array}$$

- For the epistemic operator

$$s \models K_i\phi \qquad \textit{iff } \ \forall s' \in W \textit{ if } s \sim_i s' \textit{ then } s' \models \phi$$

For the probabilistic operator, $s \models P_{\nabla b}(\psi)$ means that "the probability, from state s that ψ is true for an outgoing path satisfies ∇b". For example, $s \models$

$P_{<0.25}(true \cup^{\leq 5} \phi)$ asserts that "the probability that the system reaches a ϕ being true within 5 steps of outgoing paths from state s is less than 0.25". Given $Prop$ to represent the probability, semantics of the probabilistic operator Pr is given as follows:

- For probabilistic operator
  $s \models Pr_{\nabla b}(\psi)$ *iff* $Prop(s \models \psi)$ is in the range of ∇b
  where: $Prop(s \models \psi) = Prop\{\pi \in Path(s) | \ \pi \models \psi\}$

4 PCTLK Model Checking

4.1 Overview

Modeling, specification, and verification are three steps in the model checking process. In our approach, DTMC are used to describe the system models; then we convert DTMC models into interpreted systems for DTMC. $PCTLK$ is used as our specification language to state the properties that the system must satisfy. The model checking algorithm for $PCTLK$ is introduced in this section. The structure of our model checking approach is shown in Fig 1.

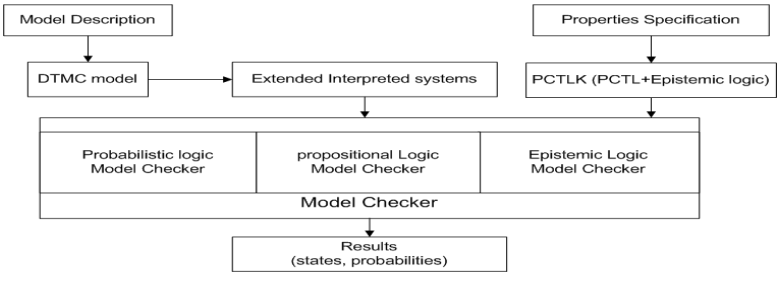

Fig. 1. verification work flow for $PCTLK$

4.2 SAT-Based Model Checking

Given a DTMC model M, we can generate an extended interpreted system M_{IS}. Given a state s in M_{IS} and a $PCTLK$ state formula ϕ, the model checking determines if $s \models \phi$. Due to existing quantitative properties, the computing result of probabilities is also needed in some cases. In fact, our $PCTLK$ language can be seen as a variant of $CTLK$ [12] by replacing the path quantifiers $\exists$ and $\forall$ by the probabilistic operator Pr. The MCMAS [11] model checker can be used to verify the properties expressed with $CTLK$. $PCTLK$ provides possibility b for path formulas to specify the likelihood of the path. The critical values, 0 and 1, state two specific situations: $Pr_{\geq 1}(\psi)$ stands for the path formula ψ happens all the time; while $Pr_{\leq 0}(\psi)$ represents the negation of the path formula ψ. Therefore, when we constrain that the probabilistic relation can only be "≥ 1"

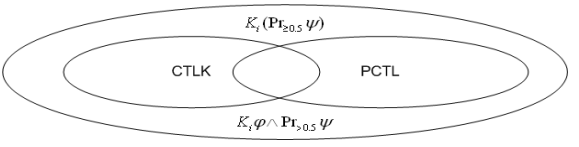

Fig. 2. Structure of $PCTLK$

or "> 0", $PCTLK$ turns into $CTLK$: $Pr_{\geq 1}$ is equivalent to $\forall$ and $\mathrm{Pr}_{>0}$ is equivalent to $\exists$. $PCTLK$ also is an extension of $PCTL$ [1,6] as it includes the epistemic operator K_i. The properties expressed using $PCTL$ can be checked by the PRISM model checker [8]. Therefore, based on $PCTLK$ syntax, a formula can be a CTL formula, a $CTLK$ but not $PCTL$ formula, a $PCTL$ but not $CTLK$ formula, a conjunction of $CTLK$ and $PCTL$ like $K_i\phi \wedge Pr_{>0.5}(\psi)$, or a pure epistemic probabilistic formula under the form $K_i(Pr_{\nabla b}(\psi))$. The structure of $PCTLK$ is shown in Fig. 2.

In order to calculate the set of states $Sat(\phi)$ satisfying a formula ϕ in $PCTLK$, we first need to decompose down the formula ϕ into sub-formulae ϕ', and compute recursively $Sat(\phi')$. This is done using a bottom-up traversal of the parse tree of ϕ. For example, the conjunction $K_i\phi \wedge Pr_{\nabla b}(\psi)$ is decomposed into a sub-formula of $CTLK$ ($K_i\phi$) and a sub-formula of $PCTL$ ($Pr_{\nabla b}(\psi)$). Based on an interpreted system $M_{IS} = (W, P_t, I'_{init}, \sim_1, \ldots, \sim_n, V')$, the algorithm for computing $Sat(\Phi)$ is shown in Table 1.

Proof. The validity of case a through c is straightforward. The proof of cases d and e can be found in [10] and [1]. We only prove the case f.

Proof of f consists of two parts:

1. Show that if $s \in Sat(K_i(Pr_{\nabla b}(\phi_1)))$, s will satisfies $K_i(Pr_{\nabla b}(\phi_1)$.
   $s \in Sat(K_i(Pr_{\nabla b}(\phi_1))) \Rightarrow s \notin Y \Rightarrow \{s \in W | \neg((\exists s' \in X) \wedge (s \sim_i s'))\}$
   $\Rightarrow \{s \in W | \neg\exists s' \in X \vee \neg(s \sim_i s')\} \Rightarrow \{s \in W | \forall s' \notin X \vee \neg(s \sim_i s')\}$
   $\Rightarrow \{s \in W | \forall s' \notin X\} \Rightarrow \{s \in W | \forall s' \notin Sat(\neg Pr_{\nabla b}(\phi_1))\}$
   $\Rightarrow \{s \in W | \forall s' \in Sat(Pr_{\nabla b}(\phi_1)\} \Rightarrow s' \vDash Pr_{\nabla b}(\phi_1) \Rightarrow s \vDash K_i Pr_{\nabla b}(\phi_1)$
2. Show that for any state s, properties ϕ_1 is true and outgoing path satisfies ∇b, it belongs to $Sat(K_i(Pr_{\nabla b}(\phi_1)))$ set, $s \in Sat(K_i(Pr_{\nabla b}(\phi_1)))$.
   $s \vDash K_i(Pr_{\nabla b}(\phi_1)) \Rightarrow \{s \in W | \forall s' \in W((s \sim_i s') \rightarrow s' \vDash (K_i Pr_{\nabla b}(\phi_1))\}$
   $\Rightarrow \{s \in W | \forall s' \in W(\neg(s \sim_i s') \vee s' \vDash (K_i Pr_{\nabla b}(\phi_1)))\}$
   $\Rightarrow s \notin Y \Rightarrow s \in Sat(K_i(Pr_{\nabla b}(\phi_1)))$

The algorithm for the calculation of the set of states $Sat_{CTLK}(\phi)$ can be found in [10]. For $PCTL$, to determine if $s \in Sat_{PCTL}(\phi)$, where $\phi = Pr_{\nabla b}\psi$, we need to compute the probability $Prob(s \models \psi)$ for the event specified by ψ. Then the set $Sat_{PCTL}(\phi)$ can be calculated by $Sat(Pr_{\nabla b}\psi) = \{s \in W | Prob(s \models \psi) \nabla p\}$ ($\nabla \in \{<, \leq, \geq, >\}$). The details of the algorithm are discussed in [1].

To compute the probability for "next" operator ($Pr_{\nabla b}[\bigcirc\phi]$), first we need to compute $Sat(\phi)$, then we sum all the states in $Sat(\phi) : \sum_{s' \in Sat(\phi)} P_t(s, s')$.

Table 1. Algorithm for $PCTLK$ model checking

Input:$M_{IS} = (W, P_t, I_{init}, \sim_1, \ldots, \sim_n, V)$, $PCTLK$ formula ϕ
Output: $Sat(\phi)$ set of states satisfying ϕ

1. Decompose the parse tree of formula ϕ
2. For all the sub-formula ϕ' in ϕ
   {
   Case ϕ'
   {
   a. ϕ' is an atomic formula: return $\{g | V(g,a) = true\}$;
   b. ϕ' is $\neg\phi_1$: return $W - Sat(\phi_1)$;
   c. ϕ' is $\phi_1 \wedge \phi_2$: return $Sat(\phi_1) \cap Sat(\phi_2)$;
   d. ϕ' is in $PCTL$: return $Sat_{PCTL}(\phi')$;
   e. ϕ' is in $CTLK$: return $Sat_{CTLK}(\phi')$;
   f. ϕ' is $K_i(Pr_{\nabla b}(\phi_1))$:
   {
   $X = Sat(\neg Pr_{\nabla b}(\phi_1))$;
   $Y = \{s \in W | \exists s' \in X$ and $s \sim_i s'\}$;
   return $W - Y$;
   }
   }
   }

Therefore, we will obtain a vector by multiplying probability relation $\overrightarrow{P_t}$ with a bit vector $\overrightarrow{b_s}$ for $Sat(\phi)$, in which $b_s = 1$ when $s \in Sat(\phi)$, otherwise, $b_s = 0$.

In order to compute $Sat_{PCTL}(Pr_{\nabla b}[\phi_1 \cup^{\leq k} \phi_2])$, we first need to compute $Sat(\phi_1)$ and $Sat(\phi_2)$. Then we can identify the states that are for sure in the satisfaction set $S_{IN} = Sat(\phi_2)$, and for sure not in the satisfaction set $S_{OUT} = W \setminus (Sat(\phi_1) \cup Sat(\phi_2))$. We also need to classify the uncertain states, which may or may not be in the satisfaction set, $S_{unclear} = W \setminus (S_{IN} \cup S_{OUT})$. After we divide the states, we can compute the probability solution with a recursive equation:

$$Prob(s, \phi_1 \cup^{\leq k} \phi_2) = \begin{cases} 1, & \text{if } s \in S_{IN} \\ 0, & \text{if } s \in S_{OUT} \\ 0, & \text{if } s \in S_{unclear} \text{ and } k = 0 \\ \sum_{s' \in W} P(s,s') \cdot Prob(s, \phi_1 \cup^{\leq k} \phi_2), & \text{if } s \in S_{unclear} \text{ and } k = 1 \end{cases}$$

For the "until" operator, we can consider "bounded until" with unlimited value. The probability solution equation is then as follows:

$$Prob(s, \phi_1 \cup \phi_2) = \begin{cases} 1, & \text{if } s \in S_{IN} \\ 0, & \text{if } s \in S_{OUT} \\ \sum_{s' \in W} P(s,s') \cdot Prob(s', \phi_1 \cup \phi_2), & \text{otherwise} \end{cases}$$

If the formula is a pure $PCTLK$ formula $(K_i(Pr_{\nabla b}(\phi_1)))$, then we compute first the set X of states satisfying $\neg Pr_{\nabla b}(\phi_1)$, then we return all the states in W except those that have access to the states in X.

Theorem 1. *The complexity of PCTLK model checking in interpreted systems is PSPACE-complete.*

Proof. The model checking of $PCTLK$ is composed of the model checking of: 1) $CTLK$; 2) $PCTL$; and 3) $K_i(Pr_{\nabla b}(\phi_1))$. The complexity of 2) in interpreted systems is known to be PSPACE-complete, and the same proof can be used for 2). For 3), computing X needs polynomial space (complexity of $PCTL$), and computing Y needs only logarithmic space, as only tow states needed to be memorized. Thus, the whole complexity is PSPACE-complete.

5 Case Study

We use a simple online shopping example to simulate our algorithm. The customer requests a delivery, and the system will successfully deliver the goods in 95% of the cases, and will fail in 5% of the cases. The DTMC model is shown on Fig. 3.

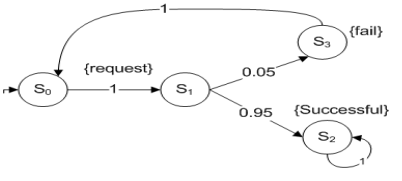

Fig. 3. DTMC for online shopping

We have two agents in this system: a customer agent and a server agent. The local states for the customer agent is: $\{s_0\}$; while for the server is $\{s_1\}$. The environment agent states are $\{s_2, s_3\}$; Thus, the global state $W = \{s_0, s_1, s_2, s_3\}$. Initial distribution I_{init} is $[0, 1, 0, 0]$. The probability relation $P_t = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0.95 & 0.05 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$

We would like to check that after a customer orders, he knows that at least 90% items will be successfully delivered, i.e. $K_c(P_{\geq 0.9} \bigcirc (successful))$

1. $Sat(succesful) = \{S_2\}$
2. $Pr(\bigcirc(successful) = P \bullet Prop_{Sat}(P_{\geq 0.9}(successful)) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0.95 & 0.05 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0.95 \\ 1 \\ 0 \end{bmatrix}$
3. $Pr(\bigcirc(successful)) = [0, 0.95, 1, 0]$, $Sat(P_{\geq 0.9} \bigcirc (successful)) = \{S_1, S_2\}$
4. Convert $K_c(P_{\geq 0.9}(successful))$ to checking $P_{\geq 1} \bigcirc (P_{\geq 0.9} \bigcirc (successful))$. This conversion reflects the semantics of K_i by representing the quantifier $\forall$ using $P_{\geq 1}$.

5. $P_{\geq 1} \bigcirc (P_{\geq 0.9} \bigcirc (successful)) = P \cdot Prop_{Sat}(\bigcirc P_{\geq 0.9}(successful)) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0.95 & 0.05 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0.95 \\ 1 \\ 0 \end{bmatrix}$
6. $Pr(\bigcirc(successful)) = [1, 0.95, 1, 0]$, $Sat(P_{\geq 1} \bigcirc \phi) = \{S_0, S_2\}$

S_0 is a customer agent local state. Therefore, the formula is satisfied. This case study has been implemented in the PRISM and MCMAS model checkers.

6 Conclusion and Future Work

In the field of multi-agent systems, the properties of agents such as their knowledge, beliefs, and autonomous reactions and pro-actions are important aspects for modeling system. In this paper, we extended interpreted systems by adding probabilistic functions so that we are able to use them as DTMC and to model probabilistic multi-agent systems. We also introduced a new language ($PCTLK$) to cover epistemic and probabilistic features. we also designed an efficient model checking algorithm for our DTMC interpreted systems model and properties expressed using $PCTLK$. In our algorithm, there are three modules: propositional temporal module, probabilistic module, and epistemic module. These features allow us to use different model checking tools. We also proved that the complexity of our model checking is the same as for $CTLK$ and $PCTL$.

Delgado and Benevides[4] modeled multi-agent systems using DTMC with synchronization actions and defined $K - PCTL$ logic to specify the properties. They use the PRISM probabilistic model checker to verify their work. They do not use or generate the interpreted systems for DTMC. Instead of using PRISM directly, we embed epistemic model checker package into it. In [7] Jamroga proposed Markov temporal logic MTL, an extension of the "Discounted CTL" (DCTL), which uses a discount factor to achieve the probabilistic factor. He used Markov Chains to model the multi-agent systems, but the main focus is on quantitative properties, not epistemic properties.

Our probabilistic knowledge model checking is an ongoing project. In the future, we will focus on the following aspects:

- Currently, our model checking algorithm is based on decomposing the parse tree of the formula. We are planning to use symbolic model checking methods, such as $MTBDD$-based (Multi-Terminal Binary Decision Diagrams) or $OBDD$-based algorithm.
- our $PCTLK$ language only includes the simple epistemic operator K_i. In the future, we expect to extended the logic to include more epistemic properties like common knowledge, distributed knowledge, and commitments, etc.

References

1. Baier, C., Katoen, J.: Principles of model checking. MIT Press, Cambridge (2008)
2. Clarke, E.M., Grumberg, O., Peled, D.: Model checking. MIT Press, Cambridge (1999)
3. Dekhtyar, M.I., Dikovsky, A.J., Valiev, M.K.: Temporal verification of probabilistic multi-agent systems. In: Anonymous Pillars of Computer Science, pp. 256–265. Springer, Heidelberg (2008)
4. Delgado, C., Benevides, M.: Verification of epistemic properties in probabilistic multi-agent systems. In: Braubach, L., van der Hoek, W., Petta, P., Pokahr, A. (eds.) MATES 2009. LNCS, vol. 5774, pp. 16–28. Springer, Heidelberg (2009)
5. Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y.: Reasoning about knowledge. MIT Press, Cambridge (1995)
6. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Formal Aspects of Computing 6, 512–535 (1994)
7. Jamroga, W.: A Temporal Logic for Markov Chains. In: AAMAS 2008, Padgham, Parkes, May 12-16, pp. 697–704 (2008)
8. Kwiatkowska, M., Norman, G., Parker, D.: PRISM: probabilistic symbolic model checker. In: Anonymous Proceedings, April 14-17, pp. 200–204. Springer, Berlin (2002)
9. Lomuscio, A., Pecheur, C., Raimondi, F.: Automatic Verification of Knowledge and Time with NuSMV. In: IJCAI 2007, Hyderabadad, India, pp. 1384–1389 (2007)
10. Lomuscio, A., Penczek, W.: Symbolic model checking for temporal-epistemic logics. SIGACT News 38(3), 77–99 (2007)
11. Lomuscio, A., Raimondi, F.: MCMAS: a model checker for multi-agent systems. In: Anonymous Proceedings, March 25-April 2, pp. 450–454. Springer, Berlin (2006)
12. Penczek, W., Lomuscio, A.: Verifying epistemic properties of multi-agent systems via bounded model checking. Fundamenta Informaticae, 167–185 (May 2003)
13. Gammie, P., van der Meyden, R.: MCK: Model Checking the Logic of Knowledge. In: Alur, R., Peled, D.A. (eds.) CAV 2004. LNCS, vol. 3114, pp. 479–483. Springer, Heidelberg (2004)
14. Wooldridge, M.J.: An introduction to multi-agent systems. John Wiley & Sons, Chichester (2009)