Automated Analysis of Commitment Protocols Using Probabilistic Model Checking

Akın Günay and **Song Songzheng** and **Yang Liu** and **Jie Zhang**
School of Computer Engineering, Nanyang Technological University, Singapore
akingunay@ntu.edu.sg, songsz@ntu.edu.sg, yangliu@ntu.edu.sg, zhangj@ntu.edu.sg

Abstract

Commitment protocols provide an effective formalism for the regulation of agent interaction. Although existing work mainly focus on the design-time development of static commitment protocols, recent studies propose methods to create them dynamically at run-time with respect to the goals of the agents. These methods require agents to verify new commitment protocols taking their goals, and beliefs about the other agents' behavior into account. Accordingly, in this paper, we first propose a probabilistic model to formally capture commitment protocols according to agents' beliefs. Secondly, we identify a set of important properties for the verification of a new commitment protocol from an agent's perspective and formalize these properties in our model. Thirdly, we develop probabilistic model checking algorithms with advanced reduction for efficient verification of these properties. Finally, we implement these algorithms as a tool and evaluate the proposed properties over different commitment protocols.

Introduction

Agent interaction is a fundamental aspect of any multiagent system. To prevent undesirable outcomes, multiagent systems regulate interaction of agents employing control mechanisms, such as protocols. The concept of (social) *commitment* provides an effective formalism to define such protocols (Singh 1999; Yolum and Singh 2002). A commitment protocol (for short, protocol) defines the responsibilities of the agents in a declarative manner without specifying how they should satisfy these responsibilities. As a result, a commitment protocol does not interfere with agents' autonomy.

Verifying the correctness of commitment protocols is a major challenge. Most of the existing work focus on the development of static commitment protocols at design-time, which are created by developers and embedded into agents' implementation (Desai et al. 2005; Winikoff 2007; Baldoni, Baroglio, and Marengo 2010). Successful design-time methods to analyze static commitment protocols have been developed (Desai et al. 2007; Kafalı and Torroni 2012; Yolum 2007; Sultan et al. 2014). However, recent studies discuss the limitations of static protocols in dynamic agent environments and propose automated methods in which agents create new protocols dynamically at run-time taking their goals and changing conditions of the environment into account (Artikis 2009; Günay, Winikoff, and Yolum 2013; Meneguzzi, Telang, and Singh 2013).

While these new methods allow agents to operate successfully in dynamic environments, they also introduce a new decision problem. That is whether an agent's participation in a protocol is rational. In the case of a design-time protocol, this decision is made by the developer of the agent. On the other hand, when a protocol is created at run-time, this decision should be made by the agent itself. Unfortunately, the existing design-time analysis methods are not fully adequate to make this decision, since they neither take the partial local knowledge, nor the private goals and preferences of an agent into account. In this paper we propose a method that can be used by an agent to make this decision at run-time. Particularly, we address the following key challenges.

The first challenge is to identify the properties, which should be necessarily verified by an agent to make a rational decision about participating in a protocol. Approaches to analyze static protocols focus mainly on the general properties (e.g., safety) of protocols. These general properties of a protocol should be taken into account while deciding on participation. However, they are not sufficient, since they do not analyze the protocol from the agent's perspective. Therefore, in order to make a rational decision about the participation of an agent in a protocol, it is necessary to define new properties, which are specifically designed according to the agent's goals and preferences. To solve this issue, we identify three key properties. These are: conformance of the agent's behavior with the protocol, achievement of the agent's goals by participating in the protocol, and benefit of participation.

The second challenge is to build a formal model that can be used to formalize the identified properties and verify them against a commitment protocol. As we discussed earlier, the formal model should be able to represent an agent's partial local knowledge about a protocol. Moreover, the formal model should represent the agent's beliefs to reflect its uncertainty about the behavior of other agents. Since existing static protocol analysis methods assume availability of complete information about a protocol and the behavior of all the agents, they do not model an agent's local knowledge and beliefs. Hence, they are not adequate to decide on participation. In order to capture an agent's local knowledge and beliefs, we use a probabilistic model for commitment protocols, in which we capture an agent's uncertainty about the behavior of the other agents using probability distributions.

The final challenge is the development of an efficient algorithm to verify the identified properties on a protocol's model to make the final decision about an agent's participation. Verification of non-probabilistic models of commitment protocols have been studied by Singh and colleagues. They use SPIN and MCMAS model checkers to verify general properties and refinements of commitment protocols (Desai et al. 2007; Gerard and Singh 2013). They also utilize NuSMV model checker to verify business patterns that are modeled as commitment protocols (Telang and Singh 2012). Recently, a probabilistic model of commitments is also proposed by Sultan *et al.* (2014). There are also several other general algorithms to verify properties of probabilistic models (Baier and Katoen 2008). However, these algorithms suffer from the state space explosion problem, which makes the verification of large models hard. Depending on the complexity of a multiagent system, a protocol may include a large number of commitments and an agent may have complex goals and preferences, which require us to deal with considerably large models to decide on protocol participation. To overcome this challenge, we exploit the independence of commitments in a protocol and develop a reduction technique, which significantly improves our method's scalability.

Commitment Protocols

First, we provide the background in commitments and protocols. A commitment is a contractual binding between a debtor and a creditor agent. A commitment is denoted by $\mathsf{C}(deb, cre, ant, con)$ and states that the debtor deb is committed to the creditor cre to satisfy the consequent condition con, if the antecedent condition ant holds (Singh 1999). For instance, $\mathsf{C}(Alice, Bob, Paid, Goods)$ denotes that $Alice$ is committed to Bob to deliver certain goods (i.e., $Goods$ holds), if she is paid (i.e., $Paid$ holds).

A commitment has a state that is manipulated by a set of commitment operations. The lifecycle of commitments has been studied extensively in the literature (Yolum and Singh 2002; Fornara and Colombetti 2002; Chesani et al. 2013). Here, we use a simplified commitment lifecycle as we show in Figure 1. The rectangles represent the states of the commitment (bold rectangles are terminal states) and the edge labels are the operations. A commitment is initially in *null* state. When the commitment is created it becomes *conditional*. If the antecedent starts to hold (e.g., $Alice$ is paid) the commitment is detached and becomes *active*. If the antecedent fails to hold (e.g., Bob refuses to pay) the commitment expires and becomes *expired*. If the consequent starts to hold (e.g., the goods are delivered) the commitment is discharged and becomes *fulfilled*. Finally, if the consequent fails to hold while the commitment is active (e.g., goods are failed to be delivered) it is canceled and becomes *violated*.

A commitment protocol consists of a set of commitments (Yolum and Singh 2002). The commitments of the protocol are defined over a set of generic roles and the agents

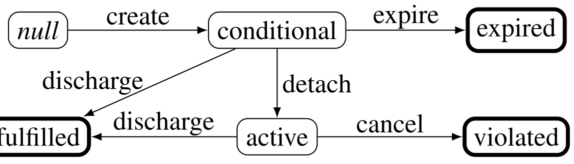


Figure 1: Life cycle of a commitment.

who participate in an instance of the protocol enact one or more of these roles. For instance, $Alice$ and Bob from our previous example enact the $Seller$ and $Buyer$ roles, respectively. The lifecycle of a commitment is captured in a protocol by associating protocol specific events and domain independent commitment operations. For instance, $Payment$ event that is initiated by $Buyer$ to satisfy the antecedent of the commitment $\mathsf{C}(Seller, Buyer, Paid, Goods)$ may be associated with the detach operation of this commitment.

In the rest of the paper we use the e-commerce protocol that consists of the following two commitments as our running example: $\{c_i = \mathsf{C}(Seller, Buyer, Paid, Goods)$, $c_j = \mathsf{C}(Provider, Seller, Ordered, Provided)\}$The commitment c_i states that $Seller$ will be committed to $Buyer$ to deliver certain goods (i.e., to satisfy $Goods$), if the payment for the goods is made (i.e., $Paid$ holds). If $Seller$ does not possess the goods, $Seller$ may order the goods from $Provider$ (i.e., $Ordered$ holds) and $Provider$ will be committed to provide the goods to $Seller$ (i.e., to satisfy $Provided$) according to c_j.

Properties for Analyzing a Protocol

Our main objective is to analyze a commitment protocol to decide whether it is rational for an agent to participate in the protocol. We identify the following three properties that should be subject to this analysis.

Conformance: Agents prefer fulfillment of their commitments over violation. Hence, an agent should ensure that he is capable of fulfilling his commitments in the context of a protocol. That is the agent should *conform* with the protocol. Otherwise, it is not rational to participate in the protocol. Consider $Provider$ in the e-commerce protocol, who is committed to provide the goods to $Seller$ (c_j). In order to conform with this protocol, $Provider$ should be capable of providing goods to $Seller$.

Achievement: Agents participate in protocols to satisfy their goals. Hence, an agent should ensure that the execution of the protocol allows him to *achieve* his goals. Otherwise, it is not rational to participate in the protocol. For instance, if $Buyer$ does not have a goal to possess the goods, there is no reason for him to participate in the e-commerce protocol.

Benefit: Agents participate in protocols that are *beneficial* to them. In other words, an agent should earn more than he spends by participating in a protocol. For instance, the e-commerce protocol is beneficial for the $Buyer$ only if the value of possessing the goods is higher than the cost of paying for the goods.

Before discussing about the necessity of these properties, we point out to some related work in which similar properties to the ones we identify are studied. Singh and Chopra discuss conformance between two roles and also between an agent and a role (Singh and Chopra 2010) and Chopra *et al.* (Chopra et al. 2011) study robustness of commitment protocols, which basically corresponds to our achievement property. However, these studies discuss these properties as abstract concepts without a concrete formalization. Moreover, they do not consider agent beliefs as we do in this paper. Desai, Narendra and Sing develop a method for design-time verification of commitment protocols using causal logic $C+$. Their method is based on the benefit concept in which they develop a general correctness definition for a commitment protocol with respect to individual agents' benefit (Desai, Narendra, and Singh 2008). Although, there are similarities about the consideration of individual agents' behavior, their model is not probabilistic. Moreover, their method is semi-automated and requires human intervention.

Now, we show the necessity of the identified properties to make a rational decision about an agent's participation in a protocol. Basically, it is rational for an agent to participate in a protocol, if the protocol changes the environment in a *desirable* manner for the agent. Given an agent $\mathcal{X}$, a protocol $\mathcal{P}$, an initial state s_i and a final state s_f, which can be reached by $\mathcal{X}$ from s_i by participating in $\mathcal{P}$, we say that $\mathcal{P}$ changes the environment in a desirable manner for $\mathcal{X}$, if all of the following three conditions hold: (1) When the protocol terminates, there is no violated commitment from the agent to others (conformance). (2) One or more goals of the agent, which are not already achieved before executing the protocol, are achieved when the protocol terminates (achievement). (3) When the protocol terminates, utility of the agent is greater than the initiation of the protocol (benefit). Accordingly, it is *rational* for an agent to participate in a protocol, if the environment of the agent becomes more desirable as the result of the protocol.

Proposition: Given a protocol and an agent, the protocol changes the environment of the agent in a desirable manner, if conformance, achievement and benefit properties are satisfied by the protocol with respect to the local knowledge and beliefs of the agent.

As a result of this proposition, it is rational for the agent to participate in the protocol, if conformance, achievement and benefit properties hold. The agent may conform to the protocol, but if the protocol does not allow the agent to achieve his goals or if it is not beneficial, then participation of the agent in the protocol is not rational. Similarly, the protocol may allow the agent to achieve his goals, but if the agent does not conform to the protocol, then it is not rational to participate. On the other hand, achieving goals may not always imply benefit. That is even though the protocol allows the agent to achieve his goals, the protocol may incur more cost to the agent than what it gains by participating. Similarly, benefit does not always imply achievement. As a result, in order to rationalize participation of the agent in the protocol, it is necessary to satisfy all three properties.

Finally, we discuss the two important issues that should be taken into account by a formal model in order to verify these properties. The first issue is the *local knowledge* of the analyzing agent about the protocol. In general, the analyzing agent knows only a subset of the protocol's commitments, which consists of the commitments that are relevant to him. For instance, $Buyer$ might not be aware of the commitment from the $Provider$ to $Seller$ in the e-commerce protocol. Accordingly, the formal model should be based on the agent's local knowledge and should not assume that the protocol is completely known.

The second issue is the *uncertainty* about the agents' behavior. In general, the analyzing agent has only beliefs about the others' behavior, since they are autonomous. For instance, in the e-commerce example, $Seller$ can fulfill his commitment to $Buyer$ by delivering the goods, only if $Provider$ fulfills his commitment and provides the goods beforehand. Hence, $Seller$ should take $Provider$'s behavior into account while verifying the properties. However, for $Seller$ there is always uncertainty about the behavior of $Provider$ (i.e., $Provider$ may violate his commitment). Accordingly, the uncertainty about the other agents' behavior should be taken into account and the formal model should reflect the beliefs of the analyzing agent.

Formal Modeling

In this section we present our formal modeling framework. We first provide the necessary definitions.

Domain: A *domain* $\mathcal{D}$ is a tuple $\langle R, A, E, \Delta \rangle$. R is a set of agent roles. A is a set of atomic propositions that represents the factual knowledge of the domain. E is a set of events and $\Delta : E \mapsto 2^A$ is a function that represents the effects of the events over the propositions of A. For instance, in the e-commerce domain, the event $Payment$ causes the proposition $Paid$ to hold (i.e., $\Delta : Payment \mapsto \{Paid\}$).

Commitment: A *commitment* C w.r.t. to the domain $\mathcal{D}$ is a tuple $\langle r_{deb}, r_{cre}, S^c, s^c_0, Op, \Delta^c \rangle$, where r_{deb} and r_{cre} are the debtor and creditor roles, respectively, such that r_{deb} and $r_{cre} \in R$ of $\mathcal{D}$. $S^c = \{Null, Con, Act, Ful, Vio, Exp\}$ is the set of commitment states. s^c_0 is the initial state of the commitment. $Op = \{Create, Detach, Expire, Discharge, Cancel\}$ is the set of commitment operations. The function $\Delta^c : S^c \times Op \mapsto S^c$ represents the lifecycle of a commitment. For instance, $\langle Con, Detach \rangle \mapsto Act$ captures the detach operation on the commitment. For brevity we do not present the details of Δ^c and assume that it is defined w.r.t Figure 1.

Protocol: A *protocol* $\mathcal{P}$ w.r.t. a domain $\mathcal{D}$ is a tuple $\langle C, \Delta^p \rangle$. C is a set of commitments that forms the protocol. The function $\Delta^p : E \mapsto \langle C, Op \rangle$ associates each domain event in $E \in \mathcal{D}$ to a domain independent operation of a commitment in the protocol. For instance, $Payment \mapsto \langle c_i, Detach \rangle$ shows that $Payment$ event in the e-commerce protocol is associated with $Detach$ operation of c_i.

Agent: An *agent* $\mathcal{X}$ w.r.t. a domain $\mathcal{D}$ and protocol $\mathcal{P}$ is a tuple $\langle r, G, \pi, \upsilon \rangle$. r is the role of $\mathcal{X}$ in $\mathcal{P}$, such that $r \in R$. G is a set of propositions that represents the goals of $\mathcal{X}$, such that $G \subseteq A$ (e.g., in the e-commerce protocol $G = \{Paid\}$ for $Seller$). The beliefs of $\mathcal{X}$ about the other agents' behavior is captured by π, in which the uncertainty of $\mathcal{X}$ is represented by assigning probabilities to domain events. Technically, $\pi \subseteq R \times \theta \times Dist(E)$ shows the probabilities of events in E in the context of $\mathcal{P}$ w.r.t. a

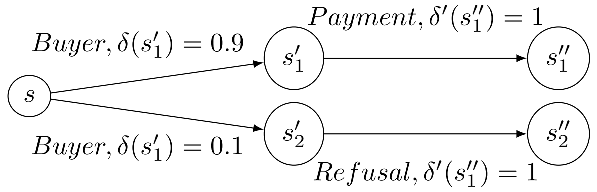
Algorithm 1: Algorithm to create the PA α and terminal states F from the protocol $\mathcal{P}$, agent $\mathcal{X}$ and domain $\mathcal{D}$.

```
input  : $\mathcal{D} = \langle R, A, E, \Delta \rangle, \mathcal{P} = \langle C, \Delta^p \rangle, \mathcal{X} = \langle r, G, \pi, \upsilon \rangle$
output: $\alpha = \langle S, s_0, \Sigma, Pr, Ap, L \rangle$, $F$
1  $Ap \leftarrow A \cup St \cup \{util\}, \Sigma \leftarrow E \cup R, S \leftarrow \emptyset, Pr \leftarrow \emptyset$;
2  $F \leftarrow \emptyset$;
3  create $s_0$ s.t., $\forall \gamma \in A : \gamma = \bot$ and $\forall c \in C : st_c = s_0^c$;
4  let $Q$ be a queue and $Enqueue(Q, s_0)$;
5  while $Q$ is not empty do
6      $s \leftarrow Dequeue(Q)$;
7      $S \leftarrow S \cup \{s\}$;
8      let $terminal \leftarrow \top$;
9      foreach $\langle r, \theta, \mu \rangle \in \pi$ s.t. $\theta$ holds in $s$ do
10         let $\delta$ be a distribution $S \mapsto [0, 1]$;
11         foreach $e \in E$ that satisfies $\mu(e) > 0$ do
12             create $s'$ where $L(s') = L(s)$;
13             $\delta(s') \leftarrow \mu(e)$;
14             $S \leftarrow S \cup \{s'\}$;
15             let $\delta'$ be a distribution $S \mapsto [0, 1]$;
16             create $s''$ s.t. $L(s'') \leftarrow Up(s, e, \Delta, \Delta^p, \upsilon)$;
17             $\delta'(s'') \leftarrow 1$;
18             $Pr \leftarrow Pr \cup \{\langle s', e, \delta' \rangle\}$;
19             if $s'' \notin S$ and $s'' \notin Q$ then
20                 $Enqueue(Q, s'')$;
21         $terminal \leftarrow \bot$;
22     $Pr \leftarrow Pr \cup \{\langle s, r, \delta \rangle\}$;
23     if $terminal$ then
24         $F \leftarrow F \cup \{s\}$;
25 return $\alpha, F$;
```

role in R and a precondition θ, where θ is a union of domain propositions and commitment states. $Dist(E)$ denotes the set of all distributions over E, where a distribution is a function $\mu : E \mapsto [0, 1]$ such that $\Sigma_{e \in E} \mu(e) = 1$. For instance, the belief $\langle Buyer, \{c_i = Con\}, (\mu(Payment) = 0.9, \mu(Refusal) = 0.1) \rangle$ of $Seller$ denotes that when c_i is Con, $Buyer$ triggers $Payment$ event with 0.9 and $Refusal$ event with 0.1 probability. Finally, $\upsilon : E \mapsto \mathbb{R}$ is the subjective utility of events from $\mathcal{X}$'s point of view, which may be a positive (gain) or negative (cost) real value.

Semantics

We formalize the semantics of our definitions as *probabilistic automata* (PA) (Rabin 1963). Our PA model is a tuple $\langle S, s_0, \Sigma, Pr, Ap, L \rangle$. The elements of the PA and their correspondence to domain $\mathcal{D}$, protocol $\mathcal{P}$ and agent $\mathcal{X}$ are as follows. S is the set of states, where each state $s \in S$ consists of a set of boolean variables that correspond to the propositions in A of $\mathcal{D}$, a variable set St that includes a variable st_c for each commitment $c \in C$ of $\mathcal{P}$ to represent the state of c, and an real valued variable $util$ that corresponds to the utility of $\mathcal{X}$ in s. $s_0 \in S$ is the initial state of the PA, where every proposition of A is set to false ($\bot$), every commitment



$Buyer, \delta(s_1') = 0.9$
$Payment, \delta'(s_1'') = 1$
s
s_1'
s_1''
s_2'
s_2''
$Buyer, \delta(s_1') = 0.1$
$Refusal, \delta'(s_1'') = 1$

Figure 2: Creation of PA states and transitions from beliefs.

state is set to $Null$ and $util$ is set to 0[1]. Σ is the alphabet, which corresponds to the events of E and roles of R in $\mathcal{D}$. $Pr \subseteq S \times \Sigma \times Dist(S)$ is the transition function where $Dist(S)$ denotes the set of all distributions over S. A distribution is a function $\delta : S \mapsto [0, 1]$ such that $\Sigma_{s \in S} \delta(s) = 1$. Finally, Ap is the set of variables that form the states of S and $L : S \mapsto Ap$ is the labeling function that assigns the values of the variables in Ap for each state in S.

We show the creation of the states and transitions of a PA α from a given domain $\mathcal{D}$, protocol $\mathcal{P}$ and agent $\mathcal{X}$ in Algorithm 1. The algorithm also identifies the set of terminal states F (i.e., states without any outgoing transitions). The algorithm utilizes breadth-first search strategy. First, the initial state s_0 of the PA is created as we describe above (Line 3). Then, each belief $\langle r, \theta, \mu \rangle \in \pi$ of $\mathcal{X}$ is considered and if the precondition θ of the belief holds in s_0 (Line 9), the algorithm creates a transition for each event e that has a positive probability in the distribution μ (Lines 11-20). The destination state of each transition is created by the auxiliary Up function according to the effects of the e as described in Δ (Line 16). The states of the commitments and the utility of the agent are also updated by Up according to Δ^p and υ functions. After that, the new states are added to a queue (Lines 19-20). The same process is repeated until no more new states are left in this queue (Lines 5-24).

As an example, we show the creation of the states and transitions for the belief $\langle Buyer, \{st_{c_i} = Con\}, (\mu(Payment) = 0.9, \mu(Refusal) = 0.1) \rangle$ of $Seller$ about $Buyer$ in Figure 2. Suppose that the state of c_i is Con in s. Hence, the precondition of the belief holds in s. First, the intermediary states s_1' and s_2', which are identical to s are created. Then, the transitions from s to these new states are created, which are labeled as $Buyer$ to represent the agent role that can trigger the events of the given belief. After that the states s_1'' and s_2'', which can be reached from s_1' and s_2' via the transitions labeled with $Payment$ and $Refusal$ events, are created, respectively. In s_1'', $Paid$ proposition starts to hold and the state of c_i is set to Act as the effect of $Payment$ event. Variables in s_2'' are updated similarly according to $Refusal$ event's effects.

Formalization of Properties

We are ready to formalize our properties with respect to a PA. Below $\alpha = \langle S, s_0, \Sigma, Pr, Ap, L \rangle$ is a PA and $F \subseteq S$ is the set of terminal states, which are the outputs of Algorithm 1 for a given domain $\mathcal{D} = \langle R, A, E, \Delta \rangle$, protocol

[1]Without loss of generality, we assume a single initial state.

$\mathcal{P} = \langle C, \Delta^p \rangle$ and agent $\mathcal{X} = \langle r, G, \pi, \upsilon \rangle$. Our property definitions are based on a probability function $\mathcal{L}(T, \alpha)$ that computes the probability of reaching the set of states $T \subseteq S$ in α. We define the details of $\mathcal{L}$ in the next section.

Conformance: $\mathcal{X}$ *conforms* with $\mathcal{P}$ in $\mathcal{D}$ w.r.t. α and F if $\mathcal{L}(\alpha, T) > \mathbf{C}$, where $T = \{s \mid s \in F$ and $\forall c \in C : (c.r_{deb} = r) \Rightarrow (st_c = Ful$ or $st_c = Exp) \in L(s)\}$ and $\mathbf{C}$ is a real valued constant in $[0, 1]$.

An agent conforms with a protocol, if the probability of being in a terminal state in which every commitment of the agent is either fulfilled or expired, is greater than a threshold.

Achievement: $\mathcal{X}$ *achieves* his goals G via $\mathcal{P}$ in $\mathcal{D}$ w.r.t. α and F if $\mathcal{L}(\alpha, T) > \mathbf{A}$, where $T = \{s \mid s \in F$ and $\forall \gamma \in G : \gamma \in L(s)\}$, and $\mathbf{A}$ is a real valued constant in $[0, 1]$.

An agent's goals are achievable via a protocol, if the probability of being in a terminal state in which every goal of the agent is achieved, is greater than a threshold.

Benefit: $\mathcal{P}$ is *beneficial* for $\mathcal{X}$ in $\mathcal{D}$ w.r.t. α and F if $\mathcal{L}(\alpha, T) > \mathbf{B}$, where $T = \{s \mid s \in F$ and $util > 0 \in L(s)\}$, and $\mathbf{B}$ is a real valued constant $[0, 1]$.

A protocol is beneficial for an agent, if the probability of being in a terminal state in which the expected utility is positive, is greater than a threshold.

Verification of Properties

In this section we show computation of $\mathcal{L}(T, \alpha)$, which is the probability of reaching T in α. Computation of $\mathcal{L}(T, \alpha)$ corresponds to reachability checking in probabilistic model checking. In Algorithm 2 we present an algorithm for this computation. Algorithm 2 takes a PA α and a set of target states $T \subseteq S$ as input, and returns the probability of reaching T in α. T is determined according to the property definition that is aimed to be verified. If the probability of reaching T is greater than the threshold of the property, we conclude that the property holds. Otherwise, the property does not hold.

In Algorithm 2 we use the auxiliary function $Pre(s)$ that returns the pre-states of a state s in a PA. Formally, $Pre(s) = \{s' | \exists (s', t, \delta) \in Pr$ satisfying $\delta(s) > 0\}$. We use p_s to record the probability of reaching T from s. Due to the existence of non-determinism in PA, the result of reachability checking is a range instead of a single value, which is similar to Markov Decision Processes (Baier and Katoen 2008). Accordingly, in Algorithm 2 we adopt a cautious approach and compute the *minimal* probability of reaching T.

The main idea of the reachability checking is to start from the target states and proceed backwards step by step while updating the probabilities of the PA states. Accordingly, first T is assigned to S^{cur} (Line 1), which represents the current states in the iterative process, and the probabilities of these states are set to 1 (Lines 2-3). Then, a state s is removed from S^{cur} (Lines 5-6) and for each pre-state s' of s (Line 7) the probabilities are updated as follows: for each enabled transition t from s' with distribution δ, a variable p_n is created (Line 9-10) to record the probability of reaching T from s' via δ. Afterward, the sum of $\delta(s'') \times p_{s''}$ for all s'' satisfying $\delta(s'') > 0$ is assigned to p_n, i.e., p_n is the sum of the transition probabilities of this distribution times the corresponding successor state's probability to T (Lines 11-12).

Algorithm 2: Computation of the probability to reach the target states from the initial state.

```
input  : $\alpha = \langle S, s_0, \Sigma, Pr, Ap, L \rangle, T \subseteq S$
output: $p_{s_0}$
1  let $S^{cur} \leftarrow T$ and $S^{pre} \leftarrow \emptyset$;
2  foreach $s \in S^{cur}$ do
3      $p_s \leftarrow 1$;
4  while $S^{cur} \neq \emptyset$ do
5      foreach $s \in S^{cur}$ do
6          $S^{cur} \leftarrow S^{cur} \setminus \{s\}$;
7          foreach $s' \in Pre(s)$ do
8              $S^{pre} \leftarrow S^{pre} \cup \{s'\}$;
9              foreach $\langle s', t, \delta \rangle \in Pr$ do
10                 $p_n \leftarrow 0.0$;
11                 foreach $s'' \in S$ such that $\delta(s'') > 0$ do
12                     $p_n \leftarrow p_n + \delta(s'') \times p_{s''}$;
13                 $p_{s'} \leftarrow Min(p_{s'}, p_n)$;
14     $S^{cur} \leftarrow S^{pre}$;
15     $S^{pre} \leftarrow \emptyset$;
16 return $p_{s_0}$;
```

To keep the minimal probability, $p_{s'}$ is set to the minimum value of $p_{s'}$ and p_n using Min function (Line 13). When all states in S^{cur} are considered, S^{cur} is set to pre-states of s (Line 14). The while loop at Line 4 terminates when no pre-states exist, i.e., the minimal probability from s_0 to T is computed and stored in p_{s_0}. Finally, p_{s_0} is returned as the probability of reaching T from s_0 (Line 16).

Partial Order Reduction for Efficient Verification

The major factor that determines the execution time of Algorithm 2 is the size of the PA's state space. Hence, reducing the size of the PA's state space also reduces the execution time. Partial order reduction is a common technique in model checking for the reduction of state space that exploits the commutativity of concurrent transitions (Baier and Katoen 2008). Usually, a commitment protocol includes several commitments that are independent from each other and the operations on such commitments are commutative, which allows us to apply partial order reduction to our PA model.

As an example consider an extension of our e-commerce protocol in which the buyer can purchase more than one goods from the seller. Suppose that different goods are provided to the seller by different providers. Accordingly, the seller has separate commitments for the goods from different providers. We show this situation for two providers in Figure 3. In the figure, c_{j_1} and c_{j_2} represent the commitments from the two providers to the seller. The superscripts denote the states of the commitments, where A means active and F means fulfilled. Suppose that both c_{j_1} and c_{j_2} are initially active in state s_1. If c_{j_1} is discharged, the system progresses to s_2, where c_{j_1} is fulfilled. Then, when c_{j_2} is discharged, the system progresses to s_4, where both c_{j_1} and c_{j_2} are fulfilled. On the other hand, if c_{2_j} is discharged, the

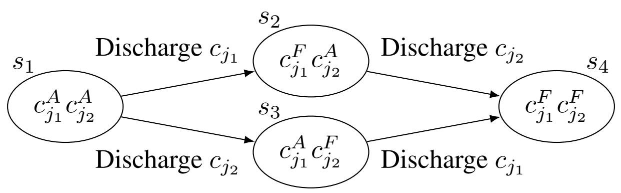


Figure 3: Different message orderings with the same result.

system progresses to s_3 and when c_{1_j} is discharged, the system progresses again to s_4. Hence, both executions reach to the same state independent from the order of operations.

A particular method for partial order reduction is the use of *ample sets* (Peled 1993). Given a state s, a subset of all possible transitions from s is called an ample set, if it is sufficient to explore only the transitions in the ample set instead of every possible transition from s. For instance, in Figure 3 there are two ample sets for s_1 as $\{\text{Discharge } c_{j_1}\}$ and $\{\text{Discharge } c_{j_2}\}$. Therefore, it is enough to explore just one of these transition sets.

An ample set is determined according to a set of criteria. Due to space limitations we discuss only the *dependency condition*, which is the most challenging to check of all these criteria. Technically, checking the dependency condition is equivalent to checking the reachablity of a condition in the full PA. However, in general it is possible to use an equivalent local criterion that can be checked efficiently. We define a local criterion that exploits the independency of commitments as follows: (LC-1) A state of c that can be reached as a result of an event e that may occur in s, should not be a precondition of any event e' that affects the state of any other commitment c'. (LC-2) A disabled event e that affects the state of c, should not have a precondition about the state of some other commitment c'.

Computational Experiments

We conducted a set of experiments to evaluate the execution performance of our algorithms and the effect of our reduction technique. For these experiments we implemented Algorithms 1 and 2. We systematically created a set of protocol models based on our e-commerce protocol for evaluation as follows[2]. We combined n number of copies of the base e-commerce protocol, such that each copy of the base protocol uses new identifiers for $Buyer$ and $Provider$, and the commitments are created between these new roles and the original $Seller$. Hence, each copy of the basic protocol can be executed independent from the other copies.

We run our experiments on a PC equipped with an Intel i7 3.0 GHz processor and 4GB RAM running 64-bit Windows 7 operating system. We verify all three properties for each protocol according to the beliefs of $Seller$. For brevity we present only the result for the verification of the conformance property in Table 1. However, we observed similar results for other properties. The first column shows the number of duplications of the base protocol. The columns 2–4 show the execution time of the verification process in seconds, number of states in the verified PA and the approximate memory usage in megabytes without applying reduction, respectively. The columns 5–7 shows the same results when reduction is applied. All results are the average of ten runs. The results show that without applying the partial order reduction, the state space grows exponentially and consequently the execution time also increases exponentially. On the other hand, partial order reduction reduces the state space and consequently the execution time, significantly.

n	without reduction			with reduction		
	Time	State	Mem	Time	State	Mem
2	0.011	421	36.7	0.004	188	33.5
3	0.356	8117	40.8	0.013	1001	37.4
4	7.676	160K	185.7	0.121	5102	39.0
5	238.8	3.2M	3642.2	0.559	$25K$	48.2

Table 1: Results for the verification of conformance.

Conclusion and Future Work

We proposed a probabilistic model checking method for the verification of a commitment protocol's key properties with respect to an agent's beliefs. We showed that these properties should be satisfied in order to rationalize an agent's participation in the protocol. Our major contributions are the development of a new probabilistic model of a commitment protocol that reflects an agent's local knowledge and beliefs, identification and formalization of properties that are necessary for the analysis of a protocol with respect to an agent's goals and preferences, and the use of probabilistic model checking and reduction techniques for the efficient verification of the introduced properties.

Our proposal has major differences from the existing work on the verification of commitment protocols. Previous approaches aim to verify a commitment protocol at design-time from a global point of view, where the specification of the whole protocol and exact behavior of the agents are known. Although this type of verification is crucial to design correct commitment protocols, they are not adequate to support an individual agent's decision making. On the other hand, use of PA for modeling allows us to reflect an agent's beliefs about the other agents' behavior into the verification process. Moreover, our PA model allows us to adopt existing probabilistic model checking algorithms for the efficient verification commitment protocols (Song et al. 2012; Sun, Song, and Liu 2010).

There are several future directions. We aim to extend our method to cover other states and operations, which are studied in the literature (Chesani et al. 2013). In this paper we do not consider resource constraints of commitments. However, there are several properties of commitments, such as conflict-freeness and feasibility, which are based on resources (Günay and Yolum 2012; 2013). In our future research, we aim to extend our method to cover these properties. Finally, we aim to integrate our method into more general model checking frameworks for multiagent systems (Lomuscio, Qu, and Raimondi 2009; Song et al. 2014).

[2]The implementations and protocol models can be downloaded from http://pat.sce.ntu.edu.sg/akin

Acknowledgments

This work is supported by ”Formal Verification on Cloud” project under Grant No: M4081155.020 and ”Bring the Advanced Model Checking Techniques to Real-world Problems” project under Grant No: M4011178.020.

References

Artikis, A. 2009. Dynamic protocols for open agent systems. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems*, 97–104.

Baier, C., and Katoen, J. 2008. *Principles of Model Checking*. The MIT Press.

Baldoni, M.; Baroglio, C.; and Marengo, E. 2010. Behavior-oriented commitment-based protocols. In *Proceedings of the 19th European Conference on Artificial Intelligence*, 137–142.

Chesani, F.; Mello, P.; Montali, M.; and Torroni, P. 2013. Representing and monitoring social commitments using the event calculus. *Autonomous Agents and Multi-Agent Systems* 27(1):85–130.

Chopra, A. K.; Oren, N.; Modgil, S.; Desai, N.; Miles, S.; Luck, M.; and Singh, M. P. 2011. Analyzing contract robustness through a model of commitments. In *Agent-Oriented Software Engineering XI*, volume 6788 of *LNCS*. Springer. 17–36.

Desai, N.; Mallya, A. U.; Chopra, A. K.; and Singh, M. P. 2005. Interaction protocols as design abstractions for business processes. *IEEE Transactions on Software Engineering* 31(12):1015–1027.

Desai, N.; Cheng, Z.; Chopra, A. K.; and Singh, M. P. 2007. Toward verification of commitment protocols and their compositions. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, 33:1–33:3.

Desai, N.; Narendra, N. C.; and Singh, M. P. 2008. Checking correctness of business contracts via commitments. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*, 787–794.

Fornara, N., and Colombetti, M. 2002. Operational specification of a commitment-based agent communication language. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems*, 536–542.

Gerard, S. N., and Singh, M. P. 2013. Formalizing and verifying protocol refinements. *ACM Transactions on Intelligent Syststems and Technology* 4(2):21:1–21:27.

Günay, A., and Yolum, P. 2012. Detecting conflicts in commitments. In *Declarative Agent Languages and Technologies IX*, volume 7169 of *LNCS*. Springer. 51–66.

Günay, A., and Yolum, P. 2013. Constraint satisfaction as a tool for modeling and checking feasibility of multiagent commitments. *Applied Intelligence* 39(3):489–509.

Günay, A.; Winikoff, M.; and Yolum, P. 2013. Commitment protocol generation. In *Declarative Agent Languages and Technologies X*, volume 7784 of *LNCS*. Springer. 136–152.

Kafalı, O., and Torroni, P. 2012. Exception diagnosis in multiagent contract executions. *Annals of Mathematics and Artificial Intelligence* 64(1):73–107.

Lomuscio, A.; Qu, H.; and Raimondi, F. 2009. Mcmas: A model checker for the verification of multi-agent systems. In *Proceedings of the 21st International Conference on Computer Aided Verification*, 682–688. Springer.

Meneguzzi, F.; Telang, P. R.; and Singh, M. P. 2013. A first-order formalization of commitments and goals for planning. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence*, 697–703.

Peled, D. 1993. All from one, one for all: on model checking using representatives. In *Computer Aided Verification*, volume 697 of *LNCS*. Springer. 409–423.

Rabin, M. O. 1963. Probabilistic automata. *Information and Control* 6(3):230–245.

Singh, M. P., and Chopra, A. K. 2010. Correctness properties for multiagent systems. In *Declarative Agent Languages and Technologies VII*, volume 5948 of *LNCS*. Springer. 192–207.

Singh, M. P. 1999. An ontology for commitments in multiagent systems: Toward a unification of normative concepts. *Artificial Intelligence and Law* 7(1):97–113.

Song, S.; Sun, J.; Liu, Y.; and Dong, J. S. 2012. A model checker for hierarchical probabilistic real-time systems. In *Proceedings of the 24th International Conference on Computer Aided Verification*, 705–711. Springer.

Song, S.; Liu, Y.; Zhang, J.; and Sun, J. 2014. An extensive model checking framework for multi-agent systems. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, 1645–1646.

Sultan, K.; Bentahar, J.; Wan, W.; and Al-Saqqar, F. 2014. Modeling and verifying probabilistic multi-agent systems using knowledge and social commitments. *Expert Systems and Applications* 41(14):6291–6304.

Sun, J.; Song, S.; and Liu, Y. 2010. Model checking hierarchical probabilistic systems. In *Proceedings of the 12th International Conference on Formal Engineering Methods and Software Engineering*, 388–403. Springer.

Telang, P., and Singh, M. 2012. Specifying and verifying cross-organizational business models: An agent-oriented approach. *IEEE Transations on Services Computing* 5(3):305–318.

Winikoff, M. 2007. Implementing commitment-based interactions. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, 128:1–128:8.

Yolum, P., and Singh, M. P. 2002. Flexible protocol specification and execution: Applying event calculus planning using commitments. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems*, 527–534.

Yolum, P. 2007. Design time analysis of multiagent protocols. *Data and Knowledge Engineering* 63(1):137–154.