On the Verification of Social Commitments and Time

Mohamed El Menshawy, Jamal Bentahar
Concordia University, Faculty of Engineering and Computer Science, Canada
m_elme@encs.concordia.ca, bentahar@ciise.concordia.ca

Hongyang Qu, Rachida Dssouli
Oxford University, Computing Laboratory, UK
Concordia University, Faculty of Engineering and Computer Science, Canada
Hongyang.Qu@comlab.ox.ac.uk, dssouli@ece.concordia.ca

ABSTRACT

Social commitments have been widely studied to represent business contracts among agents with different competing objectives in communicating multi-agent systems. However, their formal verification is still an open issue. This paper proposes a novel model-checking algorithm to address this problem. We define a new temporal logic, CTLC, which extends CTL with modalities for social commitments and their fulfillment and violation. The verification technique is based on symbolic model checking that uses ordered binary decision diagrams to give a compact representation of the system. We also prove that the problem of model checking CTLC is polynomial-time reducible to the problem of model checking CTLK, the combination of CTL with modalities for knowledge. We finally present the full implementation of the proposed algorithm by extending the MCMAS symbolic model checker and report on the experimental results obtained when verifying the NetBill protocol.

Categories and Subject Descriptors

D.2.4 [**Software/Program Verification**]: Model Checking

General Terms

Algorithms, Verification

Keywords

Social Commitments, Fulfillment, Violation

1. INTRODUCTION

Over the last two decades, a significant number of social approaches that aim to define a semantics for Agent Communication Languages (ACLs) have been proposed [1, 2, 9, 16, 19, 23]. These approaches particularly aim to overcome the shortcomings of ACLs semantics defined using mental (or cognitive) approaches where the mental semantics is expressed in terms of the agents' internal mental states such as believes, desires and intensions. Social commitments are employed in some of these social approaches that successfully provide a powerful basis to represent business contracts among autonomous and possibly heterogeneous agents with different competing objectives within multi-agent systems (MASs) [3, 7, 9, 19, 21]. Formally, social commitments are denoted by $\texttt{C}(i, j, \varphi)$ meaning that i, the debtor, commits to j, the creditor, that φ holds [7, 8, 19].

Conventionally, the semantics of ACL messages in terms of social commitments satisfies some crucial criteria introduced in [19]: 1) formal (based on some temporal logics); 2) declarative (which focuses on what the message means not how the message is exchanged); 3) verifiable (we can check if the agents are acting according to the semantics); and 4) meaningful (the focus is on the content of messages, not on their representation as tokens). Recent research in agent communication using social commitments has highlighted their use in a variety of areas ranging from modeling business processes [8], developing artificial or virtual institutions [12], defining programming languages [22], developing web-based applications [21] to specifying multi-agent interaction protocols, called commitment protocols [2, 5, 7, 9, 16, 23]. In particular, these commitment protocols are more suitable for regulating and coordinating agent interactions than computer protocols formalized using *Finite State Machines* or *Petri Nets*, which only capture legal orderings of the exchanged messages without considering the meanings of those messages. Missing such meanings limits the ability to verify the compliance of agent behaviors with a given protocol.

Related Work. The motivation behind verifying that agents are acting according to a given commitment protocol was first investigated by Venkatraman and Singh [21]. They developed an approach for locally verifying whether the behavior of an agent complies with a given commitment protocol specified in *Computational Tree Logic* (CTL) [6]. Their verification method concentrates on the conditions under which an individual agent may check others' commitments toward itself. The ideas presented by Venkatraman and Singh were further complemented in two research works by Desai et al. [7] and Cheng [5]. They developed the idea of supporting the verification of properties geared toward the composition of commitment protocols. These properties are specified in *Linear Temporal Logic* (LTL) [6] and their approach depends on translating the protocol into PROMELA (the input language of the SPIN automata-based model checker) where commitments are represented as data structures [5] or processes [7]. Bentahar et al. [3] presented ACTL* logic (an extension of CTL*) to define semantics of social commitments and associated actions and specify multi-agent interaction protocols and some desirable properties. Their verification

Cite as: On the Verification of Social Commitments and Time, Mohamed El Menshawy, Jamal Benthar, Hongyang Qu and Rachida Dssouli, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 483-490.

technique is based on the translation of ACTL* formulae and protocol into a variant of alternating tree automata called *alternating Büchi tableau automata* (ABTA) in order to directly use the CWB$-$NC automata-based model checker where commitments are represented as variables and actions as atomic action propositions using CCS (the input language of CWB-NC). El-Menshawy et al. [9, 10] introduced CTL^{*sc} logic extending CTL* with commitments and associated commitment actions to drive a new specification language of multi-agent interaction protocols having social semantics. Their symbolic verification technique is based on reducing CTL^{*sc} logic into LTL^{sc} and CTL^{sc} sub-logics and then defining the participating agents in protocol as *SMV modules* using SMV and *agent sections* using ISPL (the input languages of the NuSMV and MCMAS symbolic model checkers respectively) where commitment states and commitment actions are defined as local state variables. Gerard and Singh [13] used CTL and MCMAS to verify the refinement of multi-agent interaction protocols having social semantics by developing a preprocessor tool that first reads protocols and specifications from files and then translates them into ISPL model in order to directly use MCMAS. However, the above frameworks are translation-based approaches, which have the following shortcomings: 1) they prevent verifying the real and concert semantics of commitments and related concepts as defined in the underlying logics and provide partial solution to the problem; 2) they may not be straightforward and prone to errors, particularly in the context of complex systems; and 3) they lack a full and dedicated model-checking algorithm.

The `motivation` of this paper is to address the above challenges by: 1) presenting a new semantics for social commitments and their fulfillment and violation using a new logic, CTLC, which extends CTL [6] with modalities for reasoning about social commitments and their fulfillment and violation (`Section 2`); 2) introducing a new model-checking algorithm to directly verify commitments and their fulfillment and violation (`Section 3`); and 3) presenting the full implementation of the proposed algorithm by extending the MCMAS symbolic model checker [15] (`Section 4`).

The introduction of a new logic is motivated by the fact that the needed modal connectives for social commitments and their fulfillment/violation cannot be expressed using only existing temporal logics, e.g. CTL. A dedicated logic and model checking for commitments play the same role as CTLK [17] and MCMAS do for knowledge. Furthermore, the election of CTL is motivated by our objective to balance between expressiveness and verification efficiency. Using more expressive languages such as *First Order Logic* (FOL) needs very complex and maybe intractable model checking. In fact, we prove that the problem of model checking CTLC is polynomial-time reducible to the problem of model checking CTLK (the combination of CTL with modalities for knowledge). For checking the effectiveness of our approach, we report on the experimental results obtained when verifying the NetBill protocol [20] taken from e-business domain. Our approach can complement the static verification method introduced in [23] to check the agent behaviors with given protocol specifications via an *event calculus planner*.

2. CTLC LOGIC

In this section, we briefly present the interpreted systems introduced in [11] to formalism MASs. The reason for using this formalism is the usefulness of ascribing autonomous and social behavior to the components of a system of agents. It also allows us to abstract from the details of the components and focus only on the interactions among the various agents. However, modeling complex and open systems such as MASs using the formalism of interpreted systems is typically conducted by using logic-based formalisms. Thus, we below present a new temporal logic called CTLC logic.

2.1 Interpreted Systems

An interpreted system as introduced by Fagin et al. [11] is a formalism that models the temporal evolution of a system of agents to reason about knowledge and temporal properties. In this formalism, the interpreted system is composed of a set of n agents $\mathtt{A} = \{1, \ldots, n\}$ and an environment e. This environment can be seen as a special agent that can capture any information, which may not pertain to a specific agent. For each agent $i \in \mathtt{A}$, we associate a set of local states L_i and a set of local states L_e is associated to the environment agent.

As in [11], we represent the instantaneous configuration of all agents in the MAS at a given time via the notion of global state. The set of all global states is denoted by S and a global state $s \in S$ is a tuple $s = (l_1, \ldots, l_n, l_e)$ where each component $l_i \in L_i$ represents a local state of agent i and l_e is an environment local state. Thus, the set of all global states $S \subseteq L_1 \times \ldots \times L_n \times L_e$ is a subset of the Cartesian product of all local states of n agents and local states of the environment in the system. We use the notation $l_i(s)$ to represent the local state of agent i in the global state s. $I \subseteq S$ is a set of initial global states for the system. To account for the temporal evolution of the system, the formalism of interpreted systems associates with each agent i the set Act_i of actions, and with environment the set Act_e of actions. It is assumed that $null \in Act_i$ for each agent i, where $null$ refers to the fact of doing nothing. Each agent $i \in \mathtt{A}$ has a local protocol $\mathcal{P}_i : L_i \rightarrow 2^{Act_i}$ to identify the set of the enabled actions that may be performed in a given local state. With the same meaning we can define $\mathcal{P}_e$.

As in [11], the interpreted system formalism is a synchronous model. So, we can define the global transition function as follows: $\tau : S \times ACT \rightarrow S$, where $ACT = Act_1 \times \ldots \times Act_n \times Act_e$ and each component $a \in ACT$ is a *joint action*, which is a tuple of actions (one for each agent). An evolution function t_i that determines the transitions for an individual agent i between its local states is defined as follows: $t_i : L_i \times ACT \rightarrow L_i$, where $t_i(l_i(s), null) = l_i(s)$. In a similar way, we have an evolution function for the environment's local states: $t_e : L_e \times ACT \rightarrow L_e$. Finally, given a set $\Phi_p = \{p, p_1, p_2, \ldots\}$ of atomic propositions and the valuation function V for those propositions $V : \Phi_p \rightarrow 2^S$, an interpreted system is a tuple:
$\mathcal{IS} = \langle (L_i, Act_i, \mathcal{P}_i, t_i)_{i\in\mathtt{A}}, (L_e, Act_e, \mathcal{P}_e, t_e), I, V\rangle$.

2.2 Syntax of CTLC

The proposed language CTLC is a multi-modal logic including branching time CTL [6] and modalities for social commitments and their fulfillment and violation.

Definition 1 (SYNTAX). *The syntax of CTLC logic is given by the following BNF grammar:*

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid EX\varphi \mid E(\varphi U \varphi) \mid EG\varphi \mid C(i, j, \varphi) \mid Fu(C(i, j, \varphi)) \mid Vi(C(i, j, \varphi))$$

In this definition, $p \in \Phi_p$ is an atomic proposition and E ("there exits a path") is the existential quantifier on paths. The formula $\texttt{EX}\varphi$ stands for "φ holds in the next state in at least one path"; $\texttt{E}(\varphi U \psi)$ stands for "there exists at least one path where ψ holds at some point in the future and φ holds in all states until then"; and $\texttt{EG}\varphi$ stands for "there exists a path in which φ holds globally", i.e., φ holds in every future state in at least one path. Other temporal modalities, e.g., F, and the universal path quantifier A ("for all paths") can be defined in terms of the above as usual, for examples, $\texttt{AX}\varphi \triangleq \neg\texttt{EX}\neg\varphi$ and $\texttt{AG}\varphi \triangleq \neg\texttt{EF}\neg\varphi$ where $\texttt{EF}\varphi \triangleq \texttt{E}(true U \varphi)$. $\texttt{A}(\varphi U \psi)$ has the obvious semantics. The modal connective $\texttt{C}(i,j,\varphi)$ is read as "agent i commits towards agent j to bring about φ" or equivalently as "φ is committed to by i towards j". The modal connective $\texttt{Fu}(\texttt{C}(i,j,\varphi))$ is read as "$\texttt{C}(i,j,\varphi)$ is fulfilled (discharged)" and the modal connective $\texttt{Vi}(\texttt{C}(i,j,\varphi))$ is read as "$\texttt{C}(i,j,\varphi)$ is violated".

2.3 Semantics of CTLC

In order to define the semantics of CTLC formulae, a Kripke model $M = \langle W, I, R_t, R_c, V\rangle$ is associated to a given interpreted system $\mathcal{IS}$ as follows: the set of reachable[1] worlds W is the set S of global states for the system; $I \subseteq W$ is the set of initial states as defined in $\mathcal{IS}$; the temporal transition relation $R_t \subseteq W \times W$ for the system is defined using the local protocols and evolution functions and the two worlds w and w' are related by R_t (i.e., $(w, w') \in R_t$) iff there exists a joint action $(a_1, \dots, a_n, a_e) \in ACT$ such that for all $i \in \texttt{A}$, a_i and a_e are enabled by the protocols $\mathcal{P}_i$ and $\mathcal{P}_e$ respectively and $t_i(l_i(s), a_1, \dots, a_n, a_e) = l_i(s')$; the relation $R_c : W \times A \times A \to 2^W$ is the social accessibility relation for social commitments. It is defined by $w' \in R_c(w,i,j)$ iff $\exists w'' \neq w$ such that: 1) $l_i(w) = l_i(w'') = l_i(w')$; and 2) $l_j(w'') = l_j(w')$; and V is an interpretation over the set of atomic propositions as defined in $\mathcal{IS}$.

The social accessibility relation R_c is transitive, symmetric, and Euclidean. Thus, the resulting logic of social commitments is $K4B5 \equiv KB5$. In this relation: 1) $l_i(w) = l_i(w'')=l_i(w')$ means that the local states of i in the global states w, w', and w'' are indistinguishable; and 2) $l_j(w'') = l_j(w')$ means that the local states of j in global states w'' and w' are indistinguishable where $w'' \neq w$. Intuitively, $w' \in R_c(w,i,j)$ means there is an intermediate state w'' so that there is no difference for the debtor i among being in w, w'' and w'; however, for the creditor j there is no difference between being in the intermediate state w'' and accessible state w'. This accessibility relation captures three fundamental issues: 1) the debtor's uncertainty about the current state ($w'' \neq w$); 2) the unchangeability of the debtor ($l_i(w)=l_i(w')$); and 3) the possible changeability of the creditor because of the intermediate state.

A path (or computation) $\pi = \langle w_i, w_{i+1}, w_{i+2}, \dots\rangle$ such that for all $i \geq 0, (w_i, w_{i+1}) \in R_t$ is an infinite sequence of reachable global states in the system. $\pi(k)$ is the k^{th} global state of the path π. The set of all paths is denoted by Π, whilst Π^{w_i} is the set of all paths starting at the given state ($w_i \in W$). We define the set of states that are in the past of w ($\mathcal{P}as(w)$) as follows:

$$\mathcal{P}as(w) = \{w' \in W |\ (w', w) \in R_t \textit{ or } \exists \pi \in \Pi \textit{ such that } \pi = \langle w', \dots, w, \dots\rangle\} \cup \{w\}$$

We also define the set of states that are in the future of w ($\mathcal{F}ut(w)$) as follows:

$$\mathcal{F}ut(w) = \{w' \in W |\ (w, w') \in R_t \textit{ or } \exists \pi \in \Pi \textit{ such that } \pi = \langle w, \dots, w', \dots\rangle\} \cup \{w\}$$

Definition 2 (Satisfaction). *Satisfaction for a CTLC formula φ in the model M at a global state w, denoted as $\langle M, w\rangle \models \varphi$, is recursively defined as follows:*

- *$\langle M, w\rangle \models p$ iff $w \in V(p)$;*
- *$\langle M, w\rangle \models \neg\varphi$ iff $M, \langle w\rangle \not\models \varphi$;*
- *$\langle M, w\rangle \models \varphi \vee \psi$ iff $\langle M, w\rangle \models \varphi$ or $\langle M, w\rangle \models \psi$;*
- *$\langle M, w\rangle \models \texttt{EX}\varphi$ iff there exists a path π starting at w such that $\langle M, \pi(1)\rangle \models \varphi$;*
- *$\langle M, w\rangle \models \texttt{E}(\varphi U \psi)$ iff there exists a path π starting at w such that for some $k \geq 0$, $\langle M, \pi(k)\rangle \models \psi$ and $\langle M, \pi(j)\rangle \models \varphi$ for all $0 \leq j < k$;*
- *$\langle M, w\rangle \models \texttt{EG}\varphi$ iff there exists a path π starting at w such that $\langle M, \pi(k)\rangle \models \varphi$ for all $k \geq 0$;*
- *$\langle M, w\rangle \models \texttt{C}(i,j,\varphi)$ iff $R_c(w,i,j) \neq \emptyset$ and for all global states $w' \in W$ such that $w' \in R_c(w,i,j)$ we have $\langle M, w'\rangle \models \varphi$;*
- *$\langle M, w\rangle \models \texttt{Fu}(\texttt{C}(i,j,\varphi))$ iff there exists w' such that:*
  1) $\langle M, w'\rangle \models \texttt{C}(i,j,\varphi)$; and 2) $w \in \mathcal{F}ut(w')$; and 3) $w \in R_c(w', i, j)$;
- *$\langle M, w\rangle \models \texttt{Vi}(\texttt{C}(i,j,\varphi))$ iff there exists w' such that:*
  1) $\langle M, w'\rangle \models \texttt{C}(i,j,\varphi)$; and 2) $w \in \mathcal{F}ut(w')$; and 3) for all $w'' \in \mathcal{P}as(w) \cup \mathcal{F}ut(w)$ we have $w'' \notin R_c(w', i, j)$.

Excluding the commitment and its fulfillment (discharge) and violation, the semantics of CTLC state formulae is defined in the model M as usual (semantics of CTL)—see for example [6, 11]. The state formula $\texttt{C}(i,j,\varphi)$ is satisfied in the model M at w iff the set of accessible states obtained by the social accessibility relation $R_c(w,i,j)$ is not empty and the content φ is true in every global state in this set. Note that this semantics requires checking whether or not $R_c(w,i,j) \neq \emptyset$ because the social accessibility relation is not necessarily reflexive[2] like for the epistemic accessibility relation $\sim_i$ for agent i in the logic of knowledge [11]. In this logic, the epistemic accessibility relation $\sim_i \subseteq W \times W$ represents that two global states are "indistinguishable" for this agent. Formally, $w \sim_i w'$ iff $l_i(w) = l_i(w')$ [11]. In fact, the emptiness checking is compatible with the uncertainty of agent i about the current state. The state formula $\texttt{Fu}(\texttt{C}(i,j,\varphi))$ is satisfied in the model M at w iff there exists a state w' satisfying the commitment (condition 1) and the current state (i.e., w) is both in the future of w' and accessible via the accessability relation $R_c(w', i, j)$ (conditions 2 and 3). The intuition behind Fu's semantics is to ensure that the current state w is reachable in terms of transitions and accessible in terms of the social accessibility relation from the state w' where the commitment holds, because to be fulfilled, the commitment should prior exist.

[1] W contains states in S that are reachable from I using R_t.

[2] This means that reflexivity is not always satisfied.

Conversely, the state formula $\mathtt{Vi}(\mathtt{C}(i,j,\varphi))$ is satisfied in the model M at w iff there exists a state w' satisfying the commitment and the current state (i.e., w) is in the future of w' such that every state both in the past and future of w is not accessible in terms of the accessability relation $R_c(w',i,j)$. The main motivation behind including Fu and Vi modal connectives is to ensure that an agent can detect if there exists a conflict among its commitment states. For example, from the semantics, we can easily check that when the commitment is violated, then there is no way to fulfill it in the future and it has not been fulfilled in the past and vice versa.

3. MODEL CHECKING CTLC FORMULAE

In a nutshell, given the model M representing a MAS w.r.t the formalism of interpreted system $\mathcal{IS}$ and a formula φ in CTLC describing a property, the problem of model checking can be defined as establishing whether or not $M \models \varphi$, i.e., $\forall w \in I : \langle M, w\rangle \models \varphi$. Symbolic approaches have been recently proven as an efficient technique to automatically verify MASs [15, 18]. This is because these approaches use less memory than automata-based approaches as their algorithms are applied to Boolean Functions (BFs) not to Kripke structures. In practice, space requirements for BFs that can be represented using ordered binary decision diagrams (OBDDs) [4] are exponentially smaller than for explicit representation. As a result, these approaches alleviate the "state explosion" problem, but cannot eliminate it totally as the space still increases when the model is getting larger.

In general, symbolic model checking techniques address the state explosion problem by computing the set of states satisfying φ in the model M (denoted by $[\![\varphi]\!]$), which is represented in OBDDs and then comparing it against the set of initial states I in M that is also represented in OBDD. If $I \subseteq [\![\varphi]\!]$, then the model M satisfies the formula; otherwise a counter example can be generated showing why the model does not satisfy the formula. This paper is only concerned with developing a new symbolic model-checking algorithm $SMC(\varphi, M)$ to compute the set $[\![\varphi]\!]$ of states satisfying a CTLC formula φ. This algorithm also provides a methodology to build the OBDD corresponding to $[\![\varphi]\!]$. For example, when the sets of states are encoded using BFs, all operations (e.g., intersection) on sets are translated into operations (e.g., conjunction) on BFs.

3.1 Symbolic Model-Checking Algorithm

The basic idea of our main $SMC(\varphi, M)$ algorithm is inspired by the standard symbolic procedure introduced in [14] for computing the set of states in M satisfying the formula φ in CTL (see Algorithm 1). In particular, we extend this algorithm by adding the procedures that deal with the new modalities of our logic. It starts by checking atomic formulae (line 1) and Boolean operators: negation and disjunction (lines 2 and 3). In lines 4 to 6, the algorithm calls the standard procedures $SMC_{\mathtt{EX}}(\varphi_1, M)$, $SMC_{\mathtt{EU}}(\varphi_1, \varphi_2, M)$ and $SMC_{\mathtt{EG}}(\varphi_1, M)$ introduced in [14] to check the formulae having the forms $\mathtt{EX}\varphi_1$, $\mathtt{E}(\varphi_1 U \varphi_2)$ and $\mathtt{EG}\varphi_1$ respectively. It then checks the commitment modality (line 7) by calling the procedure $SMC_{\mathtt{c}}(i,j,\varphi_1,M)$ (see Algorithms 2 and 3). The algorithm proceeds to check the satisfiability of $\mathtt{Fu}(\mathtt{C}(i,j,\varphi_1))$ and $\mathtt{Vi}(\mathtt{C}(i,j,\varphi_1))$ by calling respectively the procedures $SMC_{\mathtt{Fu}}(i,j,\varphi_1,M)$ (see Algorithm 4) and $SMC_{\mathtt{Vi}}(i,j,\varphi_1,M)$ (see Algorithm 5) (lines 8 and 9).

Algorithm 1 $SMC(\varphi, M)$: the set $[\![\varphi]\!]$ satisfying the CTLC formula φ

```
1: $\varphi$ is an atomic formula: return $V(\varphi)$
2: $\varphi$ is $\neg\varphi_1$: return $W \backslash SMC(\varphi_1, M)$
3: $\varphi$ is $\varphi_1 \vee \varphi_2$: return $SMC(\varphi_1, M) \cup SMC(\varphi_2, M)$
4: $\varphi$ is $\mathtt{EX}\varphi_1$: return $SMC_{\mathtt{EX}}(\varphi_1, M)$
5: $\varphi$ is $\mathtt{E}(\varphi_1 U \varphi_2)$: return $SMC_{\mathtt{EU}}(\varphi_1, \varphi_2, M)$
6: $\varphi$ is $\mathtt{EG}\varphi_1$: return $SMC_{\mathtt{EG}}(\varphi_1, M)$
7: $\varphi$ is $\mathtt{C}(i,j,\varphi_1)$: return $SMC_{\mathtt{c}}(i,j,\varphi_1,M)$
8: $\varphi$ is $\mathtt{Fu}(\mathtt{C}(i,j,\varphi_1))$: return $SMC_{\mathtt{Fu}}(i,j,\varphi_1,M)$
9: $\varphi$ is $\mathtt{Vi}(\mathtt{C}(i,j,\varphi_1))$: return $SMC_{\mathtt{Vi}}(i,j,\varphi_1,M)$
```

3.1.1 *BDD-based Algorithm for Commitments*

We use the social accessibility relation R_c to compute the set $[\![\mathtt{C}(i,j,\varphi)]\!]$ of states in which the formula $\mathtt{C}(i,j,\varphi)$ holds, as reported in the procedure of Algorithm 2. This procedure firstly computes the set X_1 of states in which the formula φ holds where φ is the commitment content. It then builds X_2, the set of states that have at least one accessible state via R_c and all the accessible states from each state in this set (i.e., X_2) are in X_1, which means they satisfy φ. The set $[\![\mathtt{C}(i,j,\varphi)]\!]$ is finally computed by returning the set X_2.

Algorithm 2 $SMC_{\mathtt{c}}(i,j,\varphi,M)$: the set $[\![\mathtt{C}(i,j,\varphi)]\!]$

```
1: $X_1 \leftarrow SMC(\varphi, M)$
2: $X_2 \leftarrow \{w \in W \mid R_c(w,i,j) \neq \emptyset$ and $\forall w' \in R_c(w,i,j)$ we
   have $w' \in X_1\}$
3: return $X_2$
```

Example 1. *To clarify the computation of each set of states in each proposed BDD-based algorithm, we consider the following example. It consists of eight global states and the transitions between them along with the social accessibility relation R_c and the epistemic accessibility relations $\sim_i$ and $\sim_j$ such that w_1, w_2, w_3, w_4, w_5 and w_8 hold the formula φ and w_7 does not satisfy φ (see Figure 1).*

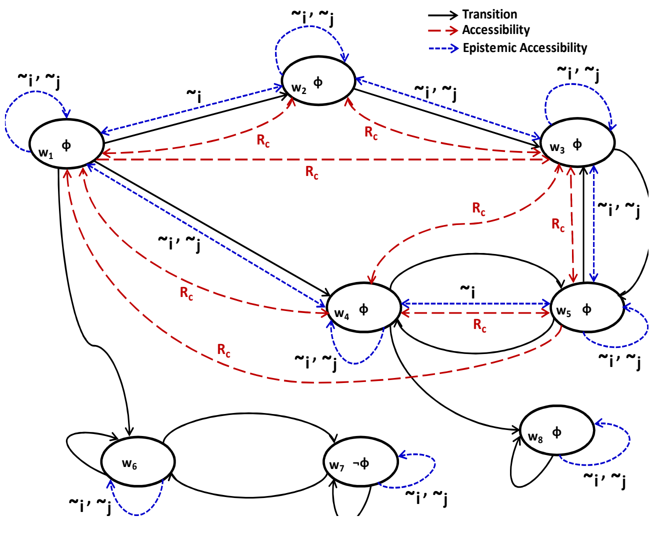

Figure 1: An example of R_c along with $\sim_i$ and $\sim_j$

Note that, $w' \in R_c(w,i,j)$ iff $\exists w'' \neq w$ such that $w \sim_i w'' \sim_i w'$ and $w'' \sim_j w'$. The reason behind using $\sim_i$ and $\sim_j$ in Figure 1 will be motivated later on. From example 1, the computation of $SMC_{\mathtt{c}}$ algorithm (see Algorithm 2) is as follows: $X_1 = \{w_1, w_2, w_3, w_4, w_5, w_8\}$, $X_2 = \{w_1, w_2, w_3, w_4, w_5\}$. Finally, the algorithm returns X_2 (see Figure 1).

The procedure reported in Algorithm 2 represents a direct implementation of the proposed semantics of the social commitment modality. We can use an alternative procedure to implement it, which is more efficient by using the negation of the formula φ and the existential quantifier "$\exists$" instead of the universal one "$\forall$" in computing the sets X_1 and X_2 (see Algorithm 3). Note that, X_3 ensures that R_c is not empty and $\overline{X}_2$, in line 4, is the complement of X_2. From example 1, $X_1 = \{w_7\}$ contains the states satisfying $\neg\varphi$, $X_2 = \emptyset$, $X_3 = \{w_1, w_2, w_3, w_4, w_5\}$, $\overline{X}_2 = \{w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8\}$ and $\overline{X}_2 \cap X_3 = \{w_1, w_2, w_3, w_4, w_5\}$, which is the same result obtained by Algorithm 2 (see Figure 1).

Algorithm 3 $SMC_{\mathtt{c}}(i, j, \varphi, M)$: the set $[\![\mathtt{C}(i, j, \varphi)]\!]$

```
1: $X_1 \leftarrow SMC(\neg\varphi, M)$
2: $X_2 \leftarrow \{w \in W |\ \exists w' \in X_1$ such that $w' \in R_c(w, i, j)\}$
3: $X_3 \leftarrow \{w \in W |\ R_c(w, i, j) \neq \emptyset\}$
4: return $\overline{X_2} \cap X_3$
```

3.1.2 BDD-based Algorithm for Fulfillment

The procedure $SMC_{\mathtt{Fu}}(i, j, \varphi, M)$ starts with computing the set X_1 of states satisfying the commitment $\mathtt{C}(i, j, \varphi)$ (see Algorithm 4). It then constructs the set X_2 of accessible states that can "see" by means of the social accessibility relation R_c a state in X_1. It then proceeds to compute the set X_3 of those states (i.e., X_2), which are reachable using transitions from the states in X_1 by calling the procedure $\mathcal{F}uture(X_1)$ (see Algorithm 7). From example 1, X_1 =$\{w_1, w_2, w_3, w_4, w_5\}$, $X_2 = \{w_1, w_2, w_3, w_4, w_5\}$, $\mathcal{F}uture(X_1) = \{w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8\}$ and $X_3 = \mathcal{F}uture(X_1) \cap X_2 = \{w_1, w_2, w_3, w_4, w_5\}$. The algorithm finally returns X_3 (see Figure 1). It is clear that the main motivation of computing X_3 is to eliminate the states that are reachable but never accessible from the states of X_1 (e.g., w_8).

Algorithm 4 $SMC_{\mathtt{Fu}}(i, j, \varphi, M)$: the set $[\![\mathtt{Fu}(\mathtt{C}(i, j, \varphi))]\!]$

```
1: $X_1 \leftarrow SMC_{\mathtt{c}}(i, j, \varphi, M)$
2: $X_2 \leftarrow \{w \in W |\ \exists w' \in X_1$ and $w \in R_c(w', i, j)\}$
3: $X_3 \leftarrow \mathcal{F}uture(X_1) \cap X_2$
4: return $X_3$
```

3.1.3 BDD-based Algorithm for Violation

The procedure $SMC_{\mathtt{Vi}}(i, j, \varphi, M)$ starts with computing the set X_1 of states satisfying the commitment $\mathtt{C}(i, j, \varphi)$. It then computes the set X_2 of those states, which are accessible and reachable from each state w' in X_1 via the social accessibility relation and transitions. The procedure proceeds to compute the set X_4 of all global states in the system that are not reachable from and cannot reach the accessible states in X_2. Finally, the procedure returns those states (i.e., in X_4), which are in the future of states where the commitment holds (i.e., $X_3 \cap X_4$) (see Algorithm 5). From example 1, $X_1 = \{w_1, w_2, w_3, w_4, w_5\}$, $X_2 = \{w_1, w_2, w_3, w_4, w_5\}$, $X_3 = \mathcal{F}uture(X_1) = \{w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8\}$ and $X_4 = W - \mathcal{P}ast(X_2) \cup \mathcal{F}uture(X)_2 = \{w_6, w_7\}$. Finally, the algorithm returns $X_3 \cap X_4 = \{w_6, w_7\}$. From Figure 1,

Algorithm 5 $SMC_{\mathtt{Vi}}(i, j, \varphi, M)$: the set $[\![\mathtt{Vi}(\mathtt{C}(i, j, \varphi))]\!]$

```
1: $X_1 \leftarrow SMC_{\mathtt{c}}(i, j, \varphi, M)$
2: $X_2 \leftarrow \{w \in W |\ \exists w' \in X_1$ and $w \in R_c(w', i, j) \cap$
   $\mathcal{F}uture(\{w'\})\}$
3: $X_3 \leftarrow \mathcal{F}uture(X_1)$
4: $X_4 \leftarrow W - (\mathcal{P}ast(X_2) \cup \mathcal{F}uture(X_2))\}$
5: return $X_3 \cap X_4$
```

it is obvious that w_6 and w_7 are the two states where the commitment $\mathtt{C}(i, j, \varphi)$ holding at w_1 is violated as they are reachable but not accessible from w_1.

The above Algorithm 5 calls two procedures $\mathcal{P}ast(X)$ and $\mathcal{F}uture(X)$ that compute the set of past (resp. future) states of X (see Algorithms 6 and 7). Algorithm 6 reports the procedure $\mathcal{P}ast(X)$ by calling the standard procedure $\mathtt{pre}_\exists(X)$

Algorithm 6 $\mathcal{P}ast(X)$: the set of past states of X

```
1: $Y \leftarrow \mathtt{pre}_\exists(X) \cup X$
2: $Z \leftarrow \emptyset$
3: While $Z \neq Y$ do
4:     $Z' \leftarrow Z$
5:     $Z \leftarrow Y$
6:     $Y \leftarrow Y \cup \mathtt{pre}_\exists(Y - Z')$
7: end While
8: return $Y$
```

introduced in [14]. The main idea of $\mathcal{P}ast(X)$ procedure is to iterate using `while...do` construct over the set of past states captured by $\mathtt{pre}_\exists(X)$ until reaching the fix-point. Note that, line 1 reflects the idea that each state is the past of itself. The procedure $\mathtt{pre}_\exists(X)$ takes a set $X \subseteq W$ as input and computes the set of states $Y \subseteq W$ such that a transition is enabled to a state in X. Formally:

$$Y{=}\mathtt{pre}_\exists(X) \leftarrow \{w \in W |\ \exists w'\ s.t.\ w' \in X\ and\ (w, w') \in R_t\}$$

Similarly, the procedure $\mathcal{F}uture(X)$ depends on the procedure $\mathtt{next}_\exists(X)$, which is computationally the dual of the procedure $\mathtt{pre}_\exists(X)$ (i.e., it computes the next states enabled by the transition from the current state). The $\mathtt{next}_\exists(X)$ procedure is formally defined as follows:

$$Y{=}\mathtt{next}_\exists(X) \leftarrow \{w \in W |\ \exists w'\ s.t.\ w' \in X\ and\ (w', w) \in R_t\}$$

Algorithm 7 $\mathcal{F}uture(X)$: the set of future states of X

```
1: $Y \leftarrow \mathtt{next}_\exists(X) \cup X$
2: $Z \leftarrow \emptyset$
3: While $Z \neq Y$ do
4:     $Z' \leftarrow Z$
5:     $Z \leftarrow Y$
6:     $Y \leftarrow Y \cup \mathtt{next}_\exists(Y - Z')$
7: end While
8: return $Y$
```

This section is concluded by the following theorem:

THEOREM 1. *Model checking CTLC is polynomial-time reducible to the problem of model checking CTLK, the combination of CTL with the logic of knowledge (i.e., CTLC $\leq_p$ CTLK).*

PROOF. In order to prove this theorem, we present the semantics of the epistemic modality $\mathtt{K}_i\varphi$, which means "agent

i knows φ" [17]. Given a formula φ of CTLK, $\langle M, w\rangle \models \mathtt{K}_i\varphi$ iff for all $w' \in W$ such that $w \sim_i w'$ we have $\langle M, w'\rangle \models \varphi$.

Let ψ be a formula in CTLC, based on the structure of the formula ψ, three cases should be analyzed:

`Case 1`: $\psi = \mathtt{C}(i, j, \varphi)$.

From Section 2.3, $w' \in R_c(w, i, j)$ iff $\exists w'' \neq w$ such that $w \sim_i w'' \sim_i w'$ and $w'' \sim_j w'$. Therefore:

1) if $w \neq w'$ then $w' \in R_c(w, i, j)$ iff $w \sim_i w'$ as $\sim_i$ and $\sim_j$ are reflexive. Because the comparison of w and w' can be done in a polynomial time, the reduction in this case can be done in a polynomial amount of time.

2) if $w = w'$, to check if $w' \in R_c(w, i, j)$, we use the algorithm:

```
for all $w''$ such that $w \sim_i w''$
  if $w'' \sim_j w$ return true
  return false
```

Since this algorithm is linear with the size of the model, the reduction of Case 1 can be done in a polynomial amount of time.

`Case 2`: $\psi = \mathtt{Fu}(\mathtt{C}(i, j, \varphi))$.

In this case, three steps are needed: 1) $\langle M, w'\rangle \models \mathtt{C}(i, j, \varphi)$; 2) $w \in \mathcal{F}ut(w')$; and 3) $w \in R_c(w', i, j)$. Step 1 is reducible in a polynomial time (Case 1). Step 2 is reducible to the future in CTLK in a polynomial time. Step 3 can be done in a polynomial time (see Case 1). Thus, the reduction of Case 2 can be also done in a polynomial amount of time.

`Case 3`: $\psi = \mathtt{Vi}(\mathtt{C}(i, j, \varphi))$.

In this case, three steps are also needed: 1) $\langle M, w'\rangle \models \mathtt{C}(i, j, \varphi)$; 2) $w \in \mathcal{F}ut(w')$; and 3) for all $w'' \in \mathcal{P}as(w) \cup \mathcal{F}ut(w)$ we have $w'' \notin R_c(w', i, j)$. Steps 1 and 2 are likewise steps 1 and 2 in Case 2. In step 3, checking the membership is linear with the size of the model and since the union of 2 sets can be done in a polynomial time, then the reduction of Case 3 can also be done in a polynomial amount of time, which completes the proof. $\square$

It is obvious that model checking CTL is also polynomial-time reducible to the problem of model checking CTLC. We can conclude that $CTL \leq_p CTLC \leq_p CTLK$).

4. IMPLEMENTATION

This section includes a description of the extensions made on top of MCMAS to implement our BDD-based algorithms presented in Section 3.1. MCMAS [15] is developed particularly to verify MASs formalized using the interpreted systems. It also implements BDD-based algorithms to verify CTL modal connectives, epistemic logic, alternating time logic and deontic operators. MCMAS is developed in $C{+}{+}$ and uses the efficient CUDD library that provides BDD data structure and performs OBDD operations and asynchronous variable reordering. It also provides fairness, counter-examples, witness generation and interactive execution.

4.1 BDD-based Algorithm of Commitments

As we mentioned, the needed BDD-based algorithms of CTL modal connectives are implemented in MCMAS. In order to fully implement the BDD-based algorithm $SMC_{\mathtt{c}}$ of social commitments on top of MCMAS, we need to perform the following two steps: 1) extend the method `check_formulae` in the `modal_formulae` class in the `parser` directory to handel the new commitment modality `C`; and 2) add the new BDD-based algorithm of commitment modality `C` along with other related methods into `utilities.cc` in the `utilities` directory. The motivation behind step 1 is to enforce the MCMAS's syntax [15] to accept the proposed new grammar specified in Definition 2.2. To achieve step 2, the BDD-based algorithm of social commitments (see Algorithm 3) is rewritten using the epistemic accessibility relations (i.e., $\sim_i$ and $\sim_j$) that define our social accessibility relation R_c. The set

Algorithm 8 $SMC_{\mathtt{c}}(i, j, \varphi, M)$: the set $[\![\mathtt{C}(i, j, \varphi)]\!]$

```
1: $X_1 \leftarrow SMC(\neg\varphi, M)$
2: $X_2' \leftarrow \{w \in W \mid \exists w' \in X_1$ such that $w \sim_i w'$ and $w \sim_j w'\}$
3: $X_2'' \leftarrow \{w \in W \mid \exists w' \in X_2'$ such that $w \sim_i w'$ and $w \neq w'\}$
4: $X_3 \leftarrow \{w \in W \mid \exists w' \in W$ such that $w \sim_i w'$ and $w \neq w'\}$
5: return $(W - X_2'') \cap X_3$
```

X_2 in the original BDD-based algorithm of social commitments is refined into two sets X_2' and X_2'' w.r.t. $\sim_i, \sim_j$ and $\sim_i$ respectively. Also, the set X_3 that checks the emptiness of R_c, in Algorithm 3, is rewritten w.r.t. $\sim_i$ (see Algorithm 8). The set $[\![\mathtt{C}(i, j, \varphi)]\!]$ is finally computed by returning the set of all global states W, which differs from the states accessible from states satisfying $\neg\varphi$ (i.e., X_2'') and accessible from all states in W w.r.t $\sim_i$ (i.e., in X_3)

In a similar way, we can easily perform the above two steps to implement the BDD-based algorithms $SMC_{\mathtt{Fu}}$ and $SMC_{\mathtt{Vi}}$ of `Fu` and `Vi` modal connectives respectively.

4.2 A Motivating Case Study

In this section, we provide a description of our motivating case study, called the NetBill protocol [20], which we used to evaluate the effectiveness of the proposed model-checking algorithm. The NetBill protocol is a security and transaction protocol optimized for the selling and delivery of low-priced information goods over the Internet. The original wording from [20] is as follows:

"*The NetBill payment protocol is eight steps (see Figure 2). The first message requests a quote based on the customer's identity, to allow for customized per-user pricing, such volume discounts or support for subscriptions. If the quote (step two) is accepted (step three), the merchant sends*

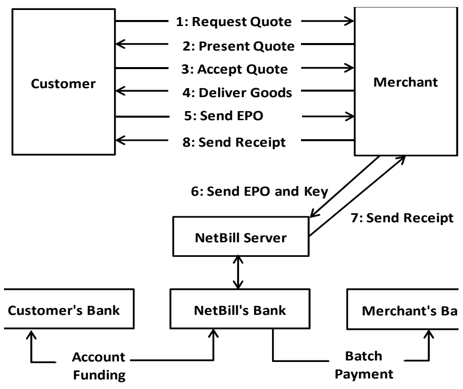

Figure 2: The NetBill payment protocol

the digital information to the customer (step four) but encrypts and withholds the key. The customer software constructs an electronic payment order (EPO) describing the transaction and including cryptographic checksum of the goods received. The order is signed with the customer's private key and sent to the merchant, who verifies its contents, appends the key for decrypting the goods, endorses the EPO *with a digital signature and sends it on to the NetBill server. The NetBill server verifies funds in the customer's NetBill account, debiting the customer and crediting the merchant, and digitally signed receipt, including the key to decrypt the goods, is sent first to the merchant and then on to the customer. The customer software can now decrypt the purchased information and present it to the customer".*

Modeling NetBill Protocol

We used our formal model $M = \langle W, I, R_t, R_c, V\rangle$ associated to the interpreted system $\mathcal{IS}$ to model the NetBill Protocol. As in [9, 23], we omit the banking procedures by assuming that if a merchant gets an EPO, he can take care of it successfully. In this setting, the protocol rules interactions among two agents: the merchant (Mer) and customer (Cus). Each agent has a set of local states, a set of local actions, local protocol, local evolution function and local initial state. Because of space limit, we omit the details of the modeling process. As in [9, 23], the following abbreviations capture the commitments that exist in this protocol:

- `acceptQuote` abbreviates $goods \rightarrow \mathtt{C}(Cus, Mer, pay)$, which means that the customer commits to pay the agreed amount if he receives the goods.
- `promiseGoods` abbreviates $\mathtt{acceptQuote} \rightarrow \mathtt{C}(Mer, Cus, goods)$, which means that the merchant commits to sending the requested goods if the customer commits to paying the agreed amount.
- `promiseReceipt` abbreviates $pay \rightarrow \mathtt{C}(Mer, Cus, receipt)$, which means that the merchant commits to sending the receipt if the customer pays the agreed amount.
- `offer` abbreviates $\mathtt{promiseGoods} \wedge \mathtt{promiseReceipt}$.

The above commitments are established by exchanging messages among agents. These messages can also bring about certain propositions. For example, by exchanging "`send Goods`" message, we can realize the proposition "`goods`".

Specifications

To verify the NetBill protocol, various protocol properties are formalized using CTLC logic w.r.t the model M.

`Reachability property`. Given a particular state, is there a valid computation sequences to reach that state from an initial state. The following lists the formulae that can be used to check the reachable states in the NetBill protocol:

$$\varphi_1 = \mathtt{E}(\neg goods\ U\ (goods \wedge \mathtt{C}(Cus, Mer, pay)))$$
$$\varphi_2 = \mathtt{E}(\neg \mathtt{acceptQuote}\ U\ (\mathtt{acceptQuote} \wedge \mathtt{C}(Mer, Cus, goods)))$$
$$\varphi_3 = \mathtt{E}(\neg pay\ U\ (pay \wedge \mathtt{C}(Mer, Cus, receipt)))$$

For example, the formula φ_1 means that there exists a path where the customer will not commit to send payment to the merchant until he receives the requested goods.

`Safety property`. This property means "something bad never happens". For example, a bad situation is: the customer sends payment, but the merchant never commits to send the receipt to him:

$$\varphi_4 = \mathtt{AG}\ \neg(pay \wedge \neg\mathtt{C}(Mer, Cus, receipt))$$

`Liveness property`. This property means "something good will eventually happen". For example, in all paths globally if the customer requests a price quote, then in all paths in the future the merchant will commit to deliver the goods:

$$\varphi_5 = \mathtt{AG}(\mathtt{reqQuote} \rightarrow \mathtt{AF}\ (\mathtt{C}(Mer, Cus, goods))$$

`Fulfillment Commitment`. While verifying the behavior of agents for commitment fulfillment, it is crucial to verify some conditions under which the commitment fulfillment can occur. For example, when the customer sends the payment to the merchant, the commitment is successfully fulfilled:

$$\varphi_6 = \mathtt{EF}\ \mathtt{Fu}(\mathtt{C}(Cus, Mer, pay))$$

Violation Commitment. In a similar way, when the customer fails to send the agreed amount of payment to the merchant, the commitment is violated as the customer violates the protocol specification:

$$\varphi_7 = \mathtt{EF}\ \mathtt{Vi}(\mathtt{C}(Cus, Mer, pay))$$

4.3 Experimental Results

We encoded the NetBill protocol and the above properties in the ISPL model and verified them using the proposed algorithm implemented on top of MCMAS. In order to provide a thorough assessment, we tested our implementation on 10 experiments (see Table 1). These experiments are ranged from 1 customer requests goods from 1 merchant to 10 customers request goods from 10 merchants. The experiments were meant to check the effectiveness of the proposed algorithm in terms of execution time and memory in use. They are performed on an AMD Phenom(tm) 9600B Quad-Core Processor with 8GB memory running Fedora 12 x86_64 Linux. In fact, from experiment 2 we rewrite the defined properties in a parameterized form, for example in experiment 10:

$$\varphi_1' = \mathtt{E}(\bigwedge_{i=1}^{10} \neg goods_i\ U\ \bigwedge_{i=1}^{10} goods_i \bigwedge_{i=1}^{10} \mathtt{C}(Cus_i, Mer_i, pay_i))$$

which means that there exists a path where the ten customers will not commit to send payment to the ten merchants until they receive the requested goods.

Table 1 reports the number of reachable states, the execution time (in seconds) and BDD memory in use (in MBs) obtained in the verification of the NetBill protocol against the above properties, as a function of the number of customer and merchant agents (first and second columns). We found

Table 1: Verification Results

#Cus	#Mer	#States	Memory	Time
1	1	10	8.6 MB	< 0.01s
2	2	43	8.971 MB	< 0.01s
3	3	239	9.958 MB	< 0.01s
4	4	1597	12.056 MB	< 0.01s
5	5	11545	16.856 MB	1s
6	6	88055	36.134 MB	2s
7	7	708461	45.592 MB	8s
8	8	6.01734e+06	56.28 MB	29s
9	9	5.25729e+07	94.36 MB	426s
10	10	4.59517e+08	153.008 MB	1128s

that: 1) all the defined properties hold in the 10 experiments; and 2) the execution time and number of reachable states increase exponentially when the number of agents increases because the number of Boolean variables required to encode agents increases. However, the memory consumption does not increase exponentially because OBDDs encoding may change from one model to another based on some internal optimization techniques. Furthermore, we did not compare our approach with others because unlike our proposal, they are based upon the translation process and do not use a dedicated model checker.

5. CONCLUSION AND FUTURE WORK

To have a full and dedicated model checking for social commitments and related concepts such as fulfillment and violation, a new temporal logic, called CTLC, is presented in this paper. Without such a logic, these concepts can only be encoded and abstracted as simple variables, processes or data structures in existing model checkers. Our CTLC logic extends CTL with modalities for social commitments and their fulfilment and violation. We developed a new model-checking algorithm that extended MCMAS to be able to verify commitments. We proved that the problem of model checking CTLC is polynomial-time reducible to the problem of model checking CTLK. In our implementation, we conducted 10 experiments, which demonstrate the effectiveness of our algorithm in terms of execution time and memory consumption. As future work, we plan to extend the proposed logic and its model checking to consider conditional commitments and commitment actions such as cancel, release, assign and delegate.

Acknowledgements

We thank the anonymous reviewers. Jamal Bentahar and Rachida Dssouli would like to thank Natural Sciences and Engineering Research Council of Canada (NSERC) and Fond Québecois de la recherche sur la société et la culture (FQRSC) for their financial support. Hongyang Qu would like to thank the European FP 7 project CONNECT (IST 231167).

6. REFERENCES

[1] M. Alberti, D. Daolio, P. Torroni, M. Gavanelli, E. Lamma, and P. Mello. Specification and Verification of Agent Interaction Protocols in a Logic-based System. In H. Haddad, A. Omicini, R. L. Wainwright, and L. M. Liebrock, editors, *SAC*, pages 72–78. ACM, 2004.

[2] M. Baldoni, C. Baroglio, and E. Marengo. Behavior Oriented Commitment-based Protocols. In H. Coelho, R. Studer, and M. Wooldridge, editors, *ECAI*, volume 215, pages 137–142. IOS Press, 2010.

[3] J. Bentahar, J.-J. C. Meyer, and W. Wan. Model Checking Agent Communication. In M. Dastani, K. V. Hindriks, and J.-J. C. Meyer, editors, *Specification and Verification of Multi-Agent Systems*, pages 67–102. Springer, First edition, 2010.

[4] R. E. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Trans. on Computers*, 35(8):677–691, 1986.

[5] Z. Cheng. *Verifying Commitment based Business Protocols and their Compositions: Model Checking using Promela and Spin.* PhD thesis, North Carolina State University, 2006.

[6] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking.* The MIT Press, Cambridge, 1999.

[7] N. Desai, Z. Cheng, A. K. Chopra, , and M. P. Singh. Toward Verification of Commitment Protocols and their Compositions. In *Proc. of the 6th Int. Joint Conf. on AAMS*, pages 144–146. ACM, 2007.

[8] N. Desai, A. K. Chopra, and M. P. Singh. Amoeba: A Methodology for Modeling and Evolution of Cross-Organizational Business Processes. *ACM Trans. on Software Eng. and Methodology*, 19(2):1–40, 2009.

[9] M. El-Menshawy, J. Bentahar, and R. Dssouli. Verifiable Semantic Model for Agent Interactions using Social Commitments. In M. Dastani, A. E. Fallah-Seghrouchni, J. Leite, and P. Torroni, editors, *LADS*, volume 6039 of *LNCS*, pages 128–152, 2010.

[10] M. El-Menshawy, W. Wan, J. Bentahar, and R. Dssouli. Symbolic Model Checking for Agent Interactions (Extended Abstract). In W. van der Hoek, G. A. Kaminka, Y. Lespérance, M. Luck, and S. Sen, editors, *AAMAS*, pages 1555–1556. ACM, 2010.

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

[12] N. Fornara, F. Viganò, M. Verdicchio, and M. Colombetti. Artificial Institutions: A Model of Institutional Reality for Open Multi-Agent Systems. *AI and Law*, 16(1):89–105, 2008.

[13] S. N. Gerard and M. P. Singh. Protocol Refinement: Formalization and Verification. In A. Artikis, J. Bentahar, A. K. Chopra, and F. Dignum, editors, *AAMAS Workshop on Agent Communication (AC)*, pages 19–36, 2010.

[14] M. Huth and M. Ryan. *Logic in Computer Science: Modelling and Reasoning about System.* Cambridge University Press, Second edition, 2004.

[15] A. Lomuscio, H. Qu, and F. Raimondi. MCMAS: A Model Checker for the Verification of Multi-Agent Systems. In A. Bouajjani and O. Maler, editors, *CAV*, volume 5643 of *LNCS*, pages 682–688. Springer, 2009.

[16] A. U. Mallya and M. P. Singh. An Algebra for Commitment Protocols. *Autonomous Agents and Multi-Agent Systems*, 14(2):143–163, 2007.

[17] W. Penczek and A. Lomuscio. Verifying Epistemic Properties of Multi-Agent Systems via Bounded Model Checking. *Fundamenta Informaticae*, 55(2):167–185, 2003.

[18] F. Raimondi. *Model Checking Multi-Agent Systems.* PhD thesis, University College London, 2006.

[19] M. P. Singh. A Social Semantics for Agent Communication Languages. In F. Dignum and M. Greaves, editors, *Issues in Agent Communication*, volume 1916 of *LNCS*, pages 31–45. Springer, 2000.

[20] M. A. Sirbu. Credits and Debits on the Internet. *IEEE Spectrum*, 34(2):23–29, 1997.

[21] M. Venkatraman and M. P. Singh. Verifying Compliance with Commitment Protocols: Enabling Open Web-based Multiagent Systems. *Autonomous Agents and Multi-Agent Systems*, 2(3):217–236, 1999.

[22] M. Winikoff. Implementing Commitment-based Interactions. In E. Durfee, M. Yokoo, M. Huhns, and O. Shehory, editors, *AAMAS*, pages 873–880, 2007.

[23] P. Yolum and M. P. Singh. Reasoning about Commitments in the Event Calculus: An Approach for Sepcifying and Executing Protocols. *Annals of Math. and AI*, 42(1–3):227–253, 2004.