Abstract: Developing and implementing a model-checker dedicated to conditional commitment logic with user interface are urgent requirements for determining whether agents comply with their commitments protocols.

Keywords: Strong (classical) commitments, model-checker, compliance

Recently, the number of multi-agent applications has grown rapidly, along with opportunities to develop commonly understood agent communication languages (ACLs) and efficient multi-agent interaction protocols to let agents to talk with each other and decide what information to exchange or action to perform. Social commitments in the form of contractual obligations from one agent to another have been advocated to define formal semantics for ACL messages. Social commitments also provide powerful representation for modeling and reasoning about multi-agent interaction protocols without restricting agents’ autonomy and flexibility. They also provide natural ways to characterize degrees of autonomy and interdependency without getting bogged down in low-level details. Thus, the agent communication research community has agreed that any formal semantics for ACL messages within heterogeneous systems must be supported by high-level abstractions rather than reasoning about agents’ mental states (e.g., beliefs).

Previous proposals for multi-agent commitment protocols have considered the semantics of conditional commitments—a natural and universal frame of social commitment—and how to check their compliance with protocol specifications. The basic idea of conditional commitments is that the debtor agent can only commit to the creditor agent (in what’s also called a commitment consequence) when specific antecedents are met—for instance, the seller commits to the buyer to ship the requested goods if the buyer sends the agreed payment. However, such proposals don’t capture some of the subtleties that arise in concrete applications in which commitments (typically representing contracts) have implicit or explicit temporal orderings and emphasize the existence of at least one option to satisfy their assigned conditions. An important aspect of social commitments is that they can be manipulated through a set of actions. Such manipulations provide the primary way to evolve changes in social commitment states and to define commitment life cycles. A commitment can be present in one state at a time and continues in that state until an action is carried out on it. Such actions are typically classified into two-party actions (such as Discharge (or Fulfill) and Violate) and three-party actions (such as Delegate and Assign). For instance, when a commitment consequence is true (that is, the seller delivers the requested goods), the commitment is fulfilled.

Current semantic models for fulfilling commitments nevertheless have a spurious paradox resulting from the counterintuitive assumption that “the commitment should be active when it comes time to its fulfillment.” Suppose, for example, that a customer commits to give $500 to a merchant. As soon as that money is transferred to the merchant’s account, the commitment is immediately fulfilled. By considering
this assumption, the commitment to sending $500 is still active, but it would be ridiculous to force the
customer to send the money again. Technically, such an assumption violates a principle that’s commonly
accepted in the literature: when a commitment is fulfilled, it should no longer be active, meaning that the
fulfillment action results in a state where the active commitment is marked as resolved.

Here, we distinguish a subtype of conditional commitments called strong commitments. Classical
commitments are those that can be activated even if the antecedent will never be satisfied (see Example 1
below), whereas strong commitments are only activated when there’s a possibility of satisfying the
antecedents (see Example 2 below). We propose an operational framework for conditional commitments
that’s expressive and rich enough to accommodate practical business scenarios. Because it’s unknown in
advance whether a party will fulfill its commitment, checking whether the commitment is violated is
significant, especially as interacting agents are heterogeneous. In our framework, when there’s no way to
fulfill strong (or classical) commitments, the commitment is violated. Indeed, the companion contribution
of this article lies in developing a symbolic algorithm to solve the problem of model-checking conditional
commitments and their fulfillments or violations. (Model checking is a formal and fully automatic
verification technique at design time that increases confidence in a system’s safety, efficiency, and
robustness.) This type of model-checking algorithm is entirely missing in the literature and would help
designers detect and eliminate design errors so that commitment protocols (a set of commitment action
meanings on which agents agree) comply with specifications before any interactions start at runtime.

Real-World Challenges

To describe our motivation for distinguishing conditional commitments as a particular subset of classical
conditional commitments, we use situational examples that arise in practical applications of online or
offline business contracts. We present formalizations of these examples later.

Example 1. Consider the NetBill protocol modeled by using event calculus. Social commitments
conventionally let us flexibly specify this protocol, enabling us to begin an interaction in one of the
following ways: a merchant commits to present an offer without receiving a request from a customer (as
happens for advertising); a merchant can commit to deliver some goods for trial without asking a customer
to accept the price; or a customer commits to accept the price quote before the merchant proposes one,
mimicking the customer’s trust in the fact that the merchant will make an offer.

Because interacting agents are indeed heterogeneous, there are no guarantees about how they’re
implemented (hence the question about distinguishing malicious agents). Suppose the customer has some
reasons to trust the merchant: What happens when the merchant is willing but practically unable to present
the offer? This example shows the need for imposing a temporal ordering between the acquisition of
consequence and the antecedent of commitments. To achieve such a temporal ordering, strong
commitments can model the conditional commitment of the customer accepting the price quote. It won’t be
active until there’s at least one possibility in the agent model to receive the merchant’s offer.

We also present classical commitments on top of strong commitments (recall that strong commitments
are a subset of classical commitments with an additional constraint) to preserve the flexibility provided by
social commitment approaches and to capture the semantics defined by previous proposals. In this context,
weak commitments is a classical commitment that isn’t strong. Informally, a weak commitment is active if
the antecedent never holds—for example, a weak commitment can be used to model the commitment about
presenting the offer if the merchant knows that the antecedent will never be satisfied (that is, there’s no way
for the customer to send the request). The following example gives additional incentive for introducing
strong commitments.

Example 2. A pharmacist strongly commits to provide medicine only if the patient shows a prescription
for that medicine and pays for it. Notice that such antecedents are always possible.

Weak commitments aren’t suitable for modeling the contractual business scenario discussed in Example
2, because it is ridiculous to commit to provide the medicine without showing prescription and paying for
it. Consequently, strong commitments are extremely necessary for addressing the weak commitment
shortcoming of committing without satisfying the antecedent. In fact, strong commitments often give more
confidence in terms of their fulfillments than classical commitments, which can be weak. For instance,
when both payment and prescription are present, the fulfillment degree of the pharmacist’s commitment to provide the requested medicine is very high.

**Proposed Framework**

Our proposed framework for conditional commitments encompasses three different but integrated parts: logical, algorithmic, and implementation (see Figure 1).

Figure 1. The main components of the proposed framework: logical, algorithmic, and implementation. The logical component extends CTL with commitment and fulfillment modalities. Our algorithm enriches the standard CTL algorithm with symbolic algorithms for new modalities and implements on the top of MCMAS. The algorithm returns true or false.

**Logical Language**

Interaction between autonomous and heterogeneous intelligent agents is the quintessential aspect of building open and effective multi-agent systems (MASs). Agents have to negotiate deals (such as price of goods and delivery terms), exchange information, and cooperate with each other to satisfy the individual and social goals that they can’t achieve alone because of a lack of resources or knowledge. Munindar P. Singh introduced four crucial criteria to a well-defined semantics for ACL messages in terms of social commitments:

- formality (using logics to define a formal semantics of agent messages to eliminate the possibility of ambiguity in their meaning);
- declarative (focusing on what social interactions and protocols are, to avoid over constraining interactions, by specifying how interactions should be accomplished);
- meaningfulness (focusing on message content and meaning, not on a message’s representation as a token); and
- verifiability (verifying if an agent is acting according to the semantics).

These criteria are defined from the perspective of agent communication; we introduce additional commitment-oriented criteria as follows:

- commitment modeling (specifying how the commitment is formally modeled, for example, as a fluent and temporal modality);
- commitment semantics (defining a formal semantics for the commitment based on its modeling); and
- verification method (specifying the technique to be used to verify the compliance of commitments and commitment protocols with specifications (model checking, monitoring, and so on). Such a criterion can help designers select a tool based on their own needs.

These criteria eliminate all the existing candidates for ACL semantics using conditional commitments.
For example, the neighborhood semantics introduced in one study still isn’t verifiable because we need to either find an equivalent semantics using standard Kripke structures or develop a completely new model-checking algorithm from scratch for the nonclassic neighborhood logic; both issues are still open problems. Furthermore, the algorithm for this semantics’ advocated linear temporal logic frame is exponential in terms of formula size and linear in terms of model size. Event calculus semantics using first-order logic (FOL), quite common and expressive in practice, is undecidable—that is, we can’t develop a model-checking algorithm that works for all kinds of formulae. Moreover, safety and liveness properties of commitment protocols, generally expressed using temporal operators, can’t be expressed in the FOL. Fluents-based semantics waives the real semantics of commitments (because they’re simply abstracted as fluents), and temporal logic-based semantics (if applied to commitments as simple tokens) isn’t meaningful. We argue that an approach formalized by extending the standard computation tree logic (CTL), where commitments are modal operators with grounded and intuitive semantics, can meet all of the requirements listed earlier. Indeed, CTL balances between expressiveness and verification efficiency (which is linear in both formula and model size).

In the logical part of our framework, we develop a branching-time temporal logic, CTL, that enriches CTL with modalities for conditional commitments and their fulfillments.

**Definition 1.** Given a set of atomic propositions AP, the syntax of CTL is defined as follows:

\[
\phi ::= p \mid \neg \phi \mid \phi \lor \phi \mid EX \phi \mid EG \phi \mid E(\phi U \phi) \mid Com \mid Ful
\]

\[
Com ::= CC(i, j, \phi) \mid SCC(i, j, \phi)
\]

\[
Ful ::= Fu(CC(i, j, \phi)) \mid Fu(SCC(i, j, \phi))
\]

where \( p \in AP \) is an atomic proposition; \( E \) is the existential quantifier on paths; \( X, G, \) and \( U \) are CTL path modal connectives standing for “next,” “globally,” and “until,” respectively; the Boolean connectives \( \neg \) and \( \lor \) are defined and read in the usual way; and \( Com \) and \( Ful \) stand for conditional commitments and their fulfillment modalities, respectively.

In this logic, a (strong) conditional commitment \( (SCC(i, j, \psi, \phi)) \) is read as “agent \( i \) (strongly) commits toward agent \( j \) that \( \phi \) when the antecedent \( \psi \) holds,” or equivalently from communication perspective as “\( i \) is (strongly) conveying information \( \phi \) to \( j \) after receiving information \( \psi \).” The antecedent \( \psi \) and consequence \( \phi \) in the context of commitment modality can be any arbitrary CTL formula, so they would be commitments as well. \( Fu(SCC(i, j, \psi, \phi)) \) is read as “the (strong) conditional commitment \( (SCC(i, j, \psi, \phi)) \) \( CC(i, j, \psi, \phi) \) is fulfilled.” Other Boolean connectives and temporal modalities can be defined in terms of the aforementioned connectives and modalities as usual—for example, \( \phi \rightarrow \psi \equiv \neg \phi \lor \psi \), \( \phi \equiv \psi \equiv \phi \rightarrow \psi \land \psi \rightarrow \phi \), \( EF \phi \equiv E(\top U \phi) \), \( AX \phi \equiv \neg EX \neg \phi \), and \( AG \phi \equiv \neg EF \neg \phi \) where \( \rightarrow, \equiv, F \) and \( \top \) stand for implication, equivalence, eventually, and unconditionally true, respectively. We now use CTL to formalize the business scenarios in Example 1 as follows:

\[
\phi_1 = AG(SCC(Mer, Cus, \top, \text{presentQuote}))
\]

\[
\phi_2 = AG(SCC(Cus, Mer, \top, E(\neg requestQuote U \neg requestQuote \land acceptQuote \text{presentQuote})))
\]

\[
\phi_3 = AG(SCC(Mer, Cus, \top, E(\neg acceptQuote U \neg deliverGoods)))
\]

where \( \phi_1 \) means that the merchant proactively presents a quote even without being requested by the customer; \( \phi_2 \) means there’s a possibility for the customer to accept the price quote without requesting it; and \( \phi_3 \) states that there exists a path during which the merchant can commit to deliver the goods without asking the customer to accept the price (as in a trial offer). The business scenario mentioned in Example 2 would be formalized as follows:

\[
\phi_4 = AG(SCC(Pha, Pat, (\text{showPrescription} \land \text{sendPayment}), EF \text{deliverMedicine}))
\]
which means that the pharmacist strongly commits to deliver medicine only if the patient shows the prescription and sends the payment. To keep CTL cf propositional, other commitment actions, missing in our previous work, are abstracted as predicate propositions, which hold in social states precisely after performing agents’ local actions, underlying the assumption saying that an agent performs one local action at a time.

**Example 3.** By performing the Cancel local action by Mer at local state $l_{mer}$ to withdraw its commitment SCC(Mer, Cus, sendPayment, deliverGoods) holding at social state $s$, the predicate proposition $p =$ Cancel(Mer, SCC(Mer, Cus, sendPayment, deliverGoods)) will hold in the accessible state $s'$.

Before we introduce the logical model $M$ to interpret CTL cf formulae, we briefly describe the extended version of the interpreted system formalism we developed previously. This formalism in fact provides a standard framework for modeling and reasoning on fundamental classes of MASs, such as synchronous and asynchronous. Specifically, suppose a MAS is composed of a set $\text{Agt} = \{1, \ldots, n\}$ of $n$ agents, wherein each agent $i \in \text{Agt}$ is characterized by a countable set $L_i$ of local states, a countable set $\text{Act}_i$ of possible local actions, a local protocol $\text{Pr}_i : L_i \to 2^\text{Act}_i$ that’s a function producing the set of enabled actions at a given local state, and a local evolution function, which is defined by $\tau_i : L_i \times \text{Act}_i \to L_i$. The agents in act within an “environment” (e), which in turn can be modeled with the set $L_e$, set $\text{Act}_e$, protocol $\text{Pr}_e$, and evolution function $\tau_e$. The environment $e$ can be seen as a special agent because it captures any information that might not pertain to a specific agent. In principle, we can view a commitment protocol as an agent $e$.

**Definition 2.** We represent the instantaneous configuration of all agents in the system at a given time by the social state having $(n + 1)$-tuple $g = (l_e, l_1, \ldots, l_n)$, where each element $l_e \in L_e$ and $l_i \in L_i$ represents a local state of agent $e$ and of agent $i$, respectively. Thus, the set of all social states $G = L_e \times L_1 \times \ldots \times L_n$ is the Cartesian product of all local states of $n + 1$ agents.

The notation $l_i(g)$ represents the local state of agent $i$ in social state $g$. The social evolution function is defined as follows: $t : G \times \text{Act} \to G$, with $\text{Act} = \text{Act}_e \times \text{Act}_1 \times \ldots \times \text{Act}_n$, where each component $a \in \text{Act}$ is a “joint action,” which is a tuple of actions (one for each agent). To account for communication that occurs during MAS execution, we associate with each $i \in \text{Agt}$ a set $\text{Var}_i$ of at most $n - 1$ local variables to represent communication channels through which messages are sent and received. The value of a variable $x$ in the set $\text{Var}_i$ at local state $l_i(g)$ is denoted by $l_i^x(g)$. The idea is that, for two agents $i$ and $j$ to communicate, they should share a communication channel, which is represented by a shared variable between them (that is, $\text{Var}_i \cap \text{Var}_j \neq \emptyset$). For the variable, $x \in \text{Var}_i \cap \text{Var}_j$, $l_i^x(g) = l_j^x(g')$ means the values of $x$ in $l_i^x(g)$ for $i$ and in $l_j^x(g')$ for $j$ are the same. It’s worth noticing that shared variables only motivate the existence of channels or pipes for communication, not the establishment of communication itself.

**Definition 3.** A model of communicative conditional commitments is a tuple $M = (S, R, \{\neg i \to j | (i, j) \in \text{Agt}^2\}, I, V)$, where:

- $S \subseteq L_e \times L_1 \times \ldots \times L_n$ is a set of social states for the system.
- $R \subseteq S \times S$ is a total transition relation defined by $(s, s') \in R$ iff there exists a joint action $(a_e, a_1 \ldots a_n) \in \text{Act}$ such that $t(s, a_e, a_1 \ldots a_n) = s'$.
- For each pair $(i, j) \in \text{Agt}^2$, $\neg i \to j \subseteq S \times S$ is a serial social accessibility relation defined by $s \vdash \neg i \to j s'$ iff the following four conditions are true:
  1. $l_i(s) = l_i(s')$.
  2. $(s, s') \in R$.
  3. $\text{Var}_i \cap \text{Var}_j \neq \emptyset$ and $\forall x \in \text{Var}_i \cap \text{Var}_j$ we have $l_i^x(s) = l_j^x(s')$; and
  4. $\forall y \in \text{Var}_j - \text{Var}_i$ we have $l_j^y(s) = l_j^y(s')$. 
- I ⊆ S is a set of initial social states for the system.
- V: Ap → 2^S is a valuation function.

The model M conceptualizes time as a tree-like structure in which nodes correspond to the states of the system being considered, and branches represent all choices in the future that agents have when they participate in protocols (the past is linear). Concretely, the underlying abstract time domain in M is discrete such that the present moment refers to the current state, the next moment corresponds to the immediate successor state in a given path, and a transition represents a social interaction between agents and corresponds to the advance of a single time unit. We can unwind the model M into a set of computation paths to interpret a CTL^cf formula. A path π = s_0, s_1, … in M is an infinite sequence of reachable social states in S such that ∀i ≥ 0, (s_i, s_{i+1}) ∈ R.

Because the semantics of CTL^cf state formulae extends the standard semantics of CTL, we present only the semantics of commitments and their fulfillments.

**Definition 4.** Given model M, the satisfaction of a CTL^cf formula φ in social state s, denoted by (M, s) ⊨ φ is recursively defined as follows:

- (M, s) ⊨ CC(i, j, ψ, φ) if \( f \forall s' \in S \text{ s.t. } s ◁_{i→j} s' \text{ and } (M, s) ⊨ ψ \), we have \( (M, s') ⊨ φ \).
- (M, s) ⊨ SCC(i, j, ψ, φ) if \( f \exists s' ∈ S \text{ s.t. } s_i → s' \text{ and } (M, s') ⊨ ψ \), and \( (M, s) ⊨ CC(i, j, ψ, φ) \).
- (M, s) ⊨ Fu(CC(i, j, ψ, φ)) if \( f \exists s' ∈ S \text{ s.t. } s_i → s' \text{ and } (M, s') ⊨ CC(i, j, ψ, φ) \), and \( (M, s) ⊨ φ \) and \( ⊨ ¬CC(i, j, ψ, φ) \).

The state formula CC(i, j, ψ, φ) is satisfied in model M at s ifff the consequence φ holds in every state satisfying ψ and accessible via ◁_{i→j}. The semantics of the strong commitment SCC(i, j, ψ, φ) is similar, but we add condition 1 (\( f \exists s' ∈ S \text{ s.t. } s_i → s' \text{ and } (M, s') ⊨ ψ \)) to ensure that at least one accessible state satisfies antecedent ψ. The state formula Fu(CC(i, j, ψ, φ)) is satisfied in model M at s ifff s satisfies the consequence φ and the negation of the commitment CC(i, j, ψ, φ), and there exists a state s' satisfying the commitment from which s is “seen” via ◁_{i→j}.

The idea behind this semantics is to say that a commitment is fulfilled when we reach an accessible state from the commitment state in which the consequence holds and the commitment becomes no longer active. The semantics of the strong fulfillment FuS(SCC(i, j, ψ, φ)) is similar, but the focus is on checking the satisfiability of antecedent ψ. This is because—from the semantics of the strong commitment—we guarantee that whenever ψ holds in an accessible state, then consequence φ holds as well. The proposed semantics solves the fulfillment paradox, where the commitment is still active when it’s fulfilled. Terminating commitment after being fulfilled is stated explicitly in the operational semantics introduced elsewhere. Furthermore, our logic doesn’t include an additional operator for violation; instead, violation can be expressed as follows:

\[ ¬AG(\zeta CC(i, j, ψ, φ) → EF Fu_\zeta(\zeta CC(i, j, ψ, φ))) = EF(\zeta CC(i, j, ψ, φ) ∧ AG(¬Fu_\zeta(\zeta CC(i, j, ψ, φ)))) \]

where ζ should be replaced by S if the commitment is strong and removed otherwise. The violation comes out when, after having the conditional fulfillment, the fulfillment doesn’t occur in all states of every possible computation. By considering social commitments and their actions, our language allows designers to characterize the practical business scenarios that are sufficient to flexibly model business protocols and models. Following recent literature, conditional commitment is a first-class citizen in our framework, and unconditional commitment can be obtained as abbreviation: C(i, j, φ) ⊨ CC(i, j, ⊤, φ).

**Symbolic Algorithm for CTL^cf**

In the algorithmic part of our framework, we develop a new symbolic algorithm to directly address the problem of model-checking CTL^cf, where the set \([φ]\) of states satisfying the formula φ being checked are
represented symbolically. Such an algorithm particularly extends the standard symbolic algorithm dedicated to CTL with five algorithms: one for each new modality and one for the accessibility relation. We adopt symbolic approaches because they need less memory than automata-based approaches, and their algorithms are applied to Boolean functions, not to Kripke structures.

In practice, space requirements for Boolean functions that can be easily encoded in ordered binary decision diagrams (OBDDs) are exponentially smaller than for explicit Kripke structure representations. In a nutshell, given model M and a CTL formula \( \phi \), the problem of model-checking CTL is determining whether M is a model for \( \phi \) (that is, \( \forall s \in I, \text{we have } (M, s) \models \phi \)). Specifically, our symbolic algorithm (see Algorithm 1) takes model M and CTL formula \( \phi \) as input and returns the set \( \{s \mid (M, s) \models \phi \} \) of states in M satisfying \( \phi \). The algorithm operates recursively on the structure of \( \phi \) and builds the set \( \{s \mid (M, s) \models \phi \} \) of states using the following operations on sets: complementation, union, and existential quantification. Lines 1 to 6 call the standard CTL algorithms; the algorithm then proceeds to call our subalgorithms (lines 7 to 10), which compute the set of states satisfying the (strong) commitments and their fulfillments. Due to space constraints, we won’t present those subalgorithms here.

Algorithm 1. SMC \( (\phi, M) \): the set \( \{s \mid (M, s) \models \phi \} \)

<table>
<thead>
<tr>
<th>Line</th>
<th>Operation</th>
<th>Formula</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( \phi ) is an atomic formula:</td>
<td>return</td>
<td>( V(\phi) );</td>
</tr>
<tr>
<td>2</td>
<td>( \phi ) is ( \neg \phi_1 ):</td>
<td>return</td>
<td>( S - \text{SMC}(\phi_1, M) );</td>
</tr>
<tr>
<td>3</td>
<td>( \phi ) is ( \phi_1 \lor \phi_2 ):</td>
<td>return</td>
<td>( \text{SMC}(\phi_1, M) \cup \text{SMC}(\phi_2, M) );</td>
</tr>
<tr>
<td>4</td>
<td>( \phi ) is ( \text{EX} \phi_1 ):</td>
<td>return</td>
<td>( \text{SMC}_{\text{EX}}(\phi_1, M) );</td>
</tr>
<tr>
<td>5</td>
<td>( \phi ) is ( \text{EU} (\phi_1 \cup \phi_2) ):</td>
<td>return</td>
<td>( \text{SMC}_{\text{EU}}(\phi_1, \phi_2, M) );</td>
</tr>
<tr>
<td>6</td>
<td>( \phi ) is ( \text{EG} \phi_1 ):</td>
<td>return</td>
<td>( \text{SMC}_{\text{EG}}(\phi_1, M) );</td>
</tr>
<tr>
<td>7</td>
<td>( \phi ) is ( \text{CC}(i, j, \phi_1, \phi_2) ):</td>
<td>return</td>
<td>( \text{SMC}_{\text{cc}}(i, j, \phi_1, \phi_2, M) );</td>
</tr>
<tr>
<td>8</td>
<td>( \phi ) is ( \text{SCC}(i, j, \phi_1, \phi_2) ):</td>
<td>return</td>
<td>( \text{SMC}_{\text{scc}}(i, j, \phi_1, \phi_2, M) );</td>
</tr>
<tr>
<td>9</td>
<td>( \phi ) is ( \text{Fu}(\text{CC}(i, j, \phi_1, \phi_2)) ):</td>
<td>return</td>
<td>( \text{SMC}_{\text{fu}}(i, j, \phi_1, \phi_2, M) );</td>
</tr>
<tr>
<td>10</td>
<td>( \phi ) is ( \text{FuS}(\text{SCC}(i, j, \phi_1, \phi_2)) ):</td>
<td>return</td>
<td>( \text{SMC}_{\text{fus}}(i, j, \phi_1, \phi_2, M) ).</td>
</tr>
</tbody>
</table>

Implementation and Experimental Results

We fully implemented the model-checking technique we’ve presented on top of the MCMAS symbolic model checker,\(^{11}\) developed to automatically verify MASs formalized using interpreted systems. Such an implementation isn’t an obvious task, nor is it trivial: we first need to extend the ISPL (the input language of MCMAS) with the shared and unshared variables needed for agent communication (social accessibility relation), and then add five developed subalgorithms along with other modifications of interest. We chose MCMAS because it supports the semantics of interpreted systems and CTL and performs OBDD operations via the efficient CUDD library.

The extended version of MCMAS features a user interface (based on Eclipse; see Figure 2) that supports a wide range of features such as editing and tracking the modeled system, expressing properties, adding new agents, checking syntax, and starting verification. Moreover, thanks to its embedding in a Java archive, such a user interface can be seamlessly integrated with other applications that need model-checking commitments or other domain properties.
To experimentally test our algorithm’s effectiveness, we adopted an electronic commerce protocol, NetBill, that’s specifically designed for selling and delivering low-priced information goods over the Internet. The protocol regulates interactions between two agents (merchant Mer and customer Cus) and starts when Cus requests a quote for some desired good. This request is followed by Mer’s reply with a price quote as an offer. Cus can then either reject the offer, and the protocol moves to the initial state, or accept the offer, which means Cus commits to send payment if Mer delivers the requested goods. If Cus accepts the received offer, it has two choices: fulfill its commitment by sending payment to Mer, or violate its commitment, moving the protocol to the failure state. When Mer receives payment, it commits to send a receipt to Cus. In a way similar to Cus’s choices, Mer can fulfill its commitment by sending a receipt to Cus and then moving to the acceptance state. Conversely, Cus could send payment for the requested goods, but Mer never commits to sending a receipt. In this case, Mer violates its commitment, and the protocol moves to the initial state.

We formalize the protocol by our model $M = (S, R, \{ \rightarrow_{i,j} \mid (i, j) \in Agt^2 \}, I, V)$, where the set $Agt$ includes two agents (Cus and Mer) plus an environment agent (e), which we specifically use to publish and store the protocol itself in a public repository to be accessible by all participating agents. To verify the protocol specification, we used the safety (something bad never happens) and liveness (something good will eventually happen) properties formalized using our CTL. Formally, the safety property $\phi_5$ expresses the bad situation as Cus sends payment, but Mer never strongly commits to sending a receipt:

$$\phi_5 = AG \neg (payment \land EF SCC (Mer, Cus, receivedPayment, EF receipt)).$$

The liveness property $\phi_6$ states that in all paths globally, if Mer delivers the goods, then there’s a path such that in the future of that path Cus will strongly commit to send payment when it accepts the delivered goods:

$$\phi_6 = AG (deliverGoods \rightarrow EF SCC (Cus, Mer, acceptGoods, EF payment)).$$

We encoded the protocol as the environment agent, customer agent, and merchant agent along with the properties in our extended ISPL language and then verified it using the developed model checker. To test
our algorithm’s scalability, we report six experiments in Table 1, where the number of reachable states, execution time in seconds, and memory in use are defined as a function of the number of agents. From this table, the number of reachable states reflects that the state space increases exponentially when the number of agents increases. However, the memory usage increases merely polynomially. With regard to the execution time, the increase isn’t exponential but faster than the polynomial.

Table 1. Verification results.

<table>
<thead>
<tr>
<th>No. agents</th>
<th>No. states</th>
<th>Time (sec)</th>
<th>Memory (Mbytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>12</td>
<td>0.019</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>144</td>
<td>0.041</td>
<td>9</td>
</tr>
<tr>
<td>12</td>
<td>20,736</td>
<td>0.365</td>
<td>13</td>
</tr>
<tr>
<td>18</td>
<td>2.98598e+06</td>
<td>1.875</td>
<td>23</td>
</tr>
<tr>
<td>24</td>
<td>4.29982e+08</td>
<td>6.892</td>
<td>46</td>
</tr>
<tr>
<td>30</td>
<td>6.19174e+10</td>
<td>19.258</td>
<td>66</td>
</tr>
</tbody>
</table>

Figure 3 depicts the verification results of the first experiment. In this figure, the properties (formulae) are evaluated into true or false along with the possibility to show a counterexample that demonstrates why the formula is false.

Figure 3. The verification results of the NetBill protocol. The properties (formulae) are evaluated into true or false along with the possibility to show a counterexample that demonstrates why the formula is false.
In this article, we argued that the proposed framework for communicative conditional commitments is expressive and rich enough to accommodate scenarios of practical utility and remedy some limitations in current semantic models. As future work, we plan to define suitable semantic models for other commitment actions (e.g., Cancel, Release and Delegate) and then develop their symbolic algorithms. We also plan to study the computational complexity of the overall model checking algorithm extended by those algorithms.

Acknowledgments
We thank NSERC (Canada), FQRSC (Quebec), Menofia University (Egypt), and EPSRC project EP/J011894/2 (UK) for their financial support.

References

Warda El Kholy is a Ph.D. candidate at Concordia Institute for Information Systems Engineering, Faculty of Engineering and Computer Science, Concordia University, Canada; and a lecturer Assistant at Department of Information Systems, Faculty of Computers and Information, Menofia University, Egypt. Her research interests are social commitments, temporal and classical logics, Web services, and model checking. Contact her at w.elkh@encs.concordia.ca.

Mohamed El Menshawy is an Associate Researcher at Concordia Institute for Information Systems Engineering, Faculty of Engineering and Computer Science, Concordia University, Canada; and a
lecturer at Department of Computer Science, Faculty of Computers and Information, Menofia University, Egypt. El Menshawy has a PhD in Electrical and Computer Engineering from Concordia University. His research interests are social commitments, commitment protocols, logic and formal methods, model checking and m-health applications. Contact him at moh_marzok75@yahoo.com.

Jamal Bentahar is an Associate Professor at Concordia Institute for Information Systems Engineering, Concordia University, Montreal, Canada. His research interests are formal verification, multi-agent systems, and web and cloud computing. Bentahar has a PhD in Computer Science and Software Engineering from Laval University, Quebec, Canada. Contact him at bentahar@ciise.concordia.ca.

Hongyang Qu is a research fellow in the Department of Automatic Control and Systems Engineering, University of Sheffield, UK. His research interests are formal verification, software engineering, robotics and autonomous systems. Qu has a PhD in Computer Science from University of Warwick, UK. Email: h.qu@sheffield.ac.uk

Rachida Dssouli is a professor and director at Concordia Institute for Information Systems Engineering, Concordia University, Montreal, Canada. Her research interests are communication software engineering, Quality systems engineering, software testing and verification. Dssouli has a PhD in Informatics from Université de Montréal, Montreal, Canada. Contact her at: rachida.dssouli@concordia.ca