The Journal of Systems & Software 192 (2022) 111383

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Transformation-based model checking temporal trust in multi-agent systems☆

Nagat Drawel [a], Amine Laarej [a], Jamal Bentahar [a,*], Mohamed El Menshawy [b]

[a] Concordia Institute for Information Systems Engineering, Montreal, Canada
[b] Computer Science, Menoufia University, Menoufia, Egypt

ARTICLE INFO

Article history:
Received 7 February 2021
Received in revised form 16 May 2022
Accepted 23 May 2022
Available online 8 June 2022

MSC:
00-01
99-00

Keywords:
Trust
Temporal logic
Model checking
Multi_Agent systems

ABSTRACT

Several formal trust frameworks have been introduced in the area of Multi-Agent Systems (MASs). However, the problem of model checking trust logics is still a challenging research topic. In this paper, we address this challenge by proposing a formal and fully automatic model checking technique for two temporal logics of trust. We start by reviewing TCTL, a Computation Tree Logic of Trust, which has been recently proposed. We also introduce TCTL^C for conditional trust. Then, we introduce sound and complete transformation-based algorithms that automatically transform the problem of model checking TCTL and TCTL^C into the problem of model checking CTL. Moreover, we prove that although TCTL and TCTL^C extend CTL, their model checking algorithms still have the same time complexity for explicit models, which is P-complete with regard to the size of the model and length of the formula, and the same space complexity for concurrent programs, which is PSPACE-complete with regard to the size of the components of these programs. Finally, experiments conducted on a standard industrial case study of auto-insurance claim processing demonstrate the efficiency and scalability of our approach in verifying TCTL and TCTL^C formulae.

1. Introduction

Trust has been the focus of many research projects, both theoretical and practical, in the recent years, particularly in domains where open multi-agent systems (MASs) are applied (e.g., Internet-based markets, information retrieval, etc.). The importance of trust in such domains arises mainly because it provides a social control that regulates the relationships and interactions among agents. However, despite the growing number of various multi-agent applications, they still encounter many challenges in the verification of agents' behaviors. The existence of many autonomous entities in such systems makes this verification difficult due to the increase in their complexity and heterogeneity. The main challenge that faces MASs is how to ensure the reliability of the trust relationships in the presence of misbehaving entities. Such entities not only create an exception for other agents, but also may obstruct their proper work (Kafalı and Yolum, 2009). The fact that such systems usually operate in open and uncertain environments makes reasoning about trust and checking the existence of untrusted computations highly desired.

Many formalisms and approaches that facilitate the specifications of trust in MASs can be found in the literature. However, few approaches addressed trust properties as natural intuitions that can facilitate the verification of multi-agent systems (Drawel et al., 2017, 2020c,a; Singh, 2011). Modal logic approaches provide powerful mechanisms that can be effectively used for trust reasoning. Such approaches yield a formal semantics to reason about trust properties in various applications such as security protocols, information sources, and e-markets. For instance, in Demolombe (2001), Herzig et al. (2010) and Lorini and Demolombe (2008b), the authors proposed several logical frameworks for the concept of trust. Trust in such logics is mostly expressed as a combination of different modalities based on the logic of action and time (Harel et al., 2000) and the BDI logic (Cohen and Levesque, 1990). In Liu and Lorini (2017), a modal logic for reasoning about the interaction between belief, evidence and trust is presented. Other approaches are interested in analyzing trust in information sources (Amgoud and Demolombe, 2014; Demolombe, 2001; Lorini and Demolombe, 2008a; Liau, 2003). Moreover, some proposals have addressed trust in the context of computer security (Fuchs et al., 2010; Lorini and Demolombe, 2008b). Most of these approaches focus on the cognitive side of trust (i.e., trusted agents are capable of exhibiting beliefs, desires, and intentions properties). Hence, the trust is considered as a belief of an agent (the truster) involving ability and willingness of the trustee to perform some actions for the truster. Although

☆ Editor: Antonio Filieri.

* Corresponding author.
E-mail addresses: n_drawe@encs.concordia.ca (N. Drawel), laarej.amine@gmail.com (A. Laarej), bentahar@ciise.concordia.ca (J. Bentahar), moh_marzok75@yahoo.com (M. El Menshawy).

https://doi.org/10.1016/j.jss.2022.111383
0164-1212/

these approaches are highly appropriate to reason about trust, their verification faces a fundamental limitation due to their reliance on the internal structure of the interacting agents. In fact, the distributed and open nature of MASs makes the capability of handling and verifying the trust interaction issues of such approaches arduous. That is, it is very difficult for one agent to completely trust others by making assumptions about their internal states. To motivate our study of analyzing the trust in MASs, we use an example in the context of electronic commerce where trust is a highly desired property. Let us consider the buyers–sellers relationships. The buyer requests to purchase one or more items from the seller. Once the former selects an item and the requested items are paid, the trust relationship with regard to deliver the items is established between the two parties. The seller confirms the order and starts the delivery process. Finally, the requested items are shipped and the buyer is notified. Nevertheless, on-line interactions are characterized by uncertainty and, moreover, the anonymity of the interaction partners. Thus, there is no guarantee that this process will be surely satisfied in concrete applications. Therefore, the need for formally specifying and automatically verifying trust-based interactions among autonomous agents are of great significance.

As in Drawel et al. (2017, 2020c) and Singh (2011), in this paper, we represent trust as a direct relation from one agent, the truster, toward another agent, the trustee, where such a relation presupposes specific preconditions with respect to a particular content. Specifically, the truster considers the trustee as a trustworthy agent with regard to a specific content when the behavior of the trustee with respect to this content is as the truster expects, where this expectation is shared by the two participants. For example, the buyer trusts that the seller will deliver the ordered items upon the buyer's payment, where the payment is the precondition and the order delivery is the expectation commonly shared. From the modeling and specification perspectives, we are considering the Trust Computation Tree Logic (TCTL) (Drawel et al., 2020c), an extension of the CTL logic (Emerson, 1990). Equipped with reasoning postulates, TCTL does not only provide a formal basis for reasoning about trust states with preconditions, but it can also be seen as a formal modeling of the social trust interactions among agents.

As mentioned earlier, the fact that agents are autonomous and have to interact with each other within unreliable social environments entails that deciding whether to trust another agent (for instance to perform some actions) or not is a challenging task. For instance, agents may not be able to comply with their obligations (e.g., an agent may not send the agreed payment for goods received) (Bentahar et al., 2009). This raises the need for developing efficient methodologies to handle their present and future behaviors in order to ensure the fulfillment of the system requirements. Currently, the technique of model checking (Clarke et al., 2009, 1999) has attracted several contributions with a significant industrial implication (Bernardi et al., 2021; Guo et al., 2020; Lomuscio and Pirovano, 2020; Filieri et al., 2011; Filieri, 2011; Kholy et al., 2014). Although model checking MASs addressed a number of key aspects, such as social commitments (Bentahar et al., 2012; El Kholy et al., 2014) and knowledge (Lomuscio et al., 2017; van der Meyden and Su, 2004), model checking trust in these multi-agent settings has not been sufficiently investigated yet. From this view, we aim in this work to contribute in the modeling and verification of trust systems.

This article is a substantial extension of our preliminary work published in Drawel et al. (2018), in which we have made a novel attempt towards model checking temporal trust logic. Specifically, we have proposed a transformation-based model checking framework for the TCTL logic that is extended to design a new algorithm to model check conditional trust TCTL^C. However, the procedure presented in Drawel et al. (2018) did not capture the alignments between source and target models which resulted in only a partially sound and incomplete transformation. Practically, there was a flaw in the transformation algorithms that overlook some critical cases. In fact, the technique provided did not preserve the temporal relations which in turn affect the semantics of the standard CTL operators. More precisely, the model transformation technique consists of adding transitions to represent the accessibility relations, and so, it does not distinguish between original transitions in the original TCTL model and the transitions added in the CTL model. It turns out that some formulae with temporal operators *EX* and *EU* become true in the transformed CTL model while they are false in the original TCTL model.

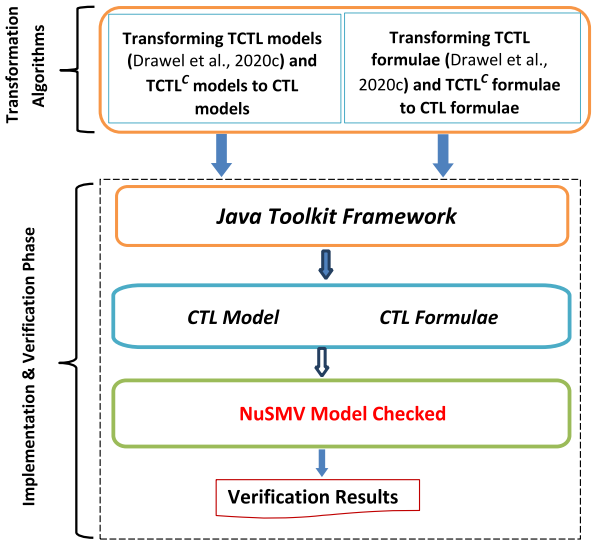


Fig. 1. A schematic view of our model checking approach.

The intent of this work is to overcome these problems and introduce a sound and complete automatic verification approach. Using the same technique, we aim to reduce the TCTL and TCTL^C model checking problems for extended interpreted systems to the CTL model checking problem for Kripke structures. One important aspect towards the soundness of the procedure is to make sure that if a CTL formula is satisfied through a path in the original TCTL (or TCTL^C) model, the formula is still satisfied in the corresponding path of the translated CTL model. Our new reduction technique captures the accessibility relations through distinguishable states and transitions using atomic propositions added in the transformed CTL formulae. Moreover, the complexity problem of model checking TCTL logic has not been addressed in our previous work. In this article, we analyze the complexity and prove that although TCTL and TCTL^C extend CTL, their model checking algorithms still have the same time complexity for explicit models and the same space complexity for concurrent programs.

Fig. 1 illustrates the overall approach of model checking TCTL and TCTL^C, which consists of two integrated phases. In the first phase, we introduce our formal verification technique based on transforming the problem of model checking TCTL and TCTL^C into the problem of model checking CTL along with the complexity analysis of the proposed technique. In the second phase, we implement our transformation technique in a Java toolkit that automatically interacts with the NuSMV model checker and report the verification results using a case study.

1.1. Insurance claim processing: A case study

To illustrate and implement our approach, we use through out this paper a standard industrial case study (Telang and Singh, 2009). The case study outlines the process by which auto insurance claims are handled by an insurance company, AGFIL. There are multiple parties involved in the AGFIL cooperation process: *AGFIL*, *Policyholder*, *Europ Assist*, *Lee Consulting Services*, *Garage*, and *Assessor*. The participating parties work together to provide a trusted service which facilitates efficient claim settlement. The process starts when the policyholder phones the call center Europ Assist to notify a new claim. Thereafter, Europ Assist registers the information and assigns an appropriate garage to provide the repair service to the policyholder. It then notifies the insurance company AGFIL which checks whether the policy is valid or not, and it confirms the claim coverage. AGFIL then sends the claim details to Lee Consulting Services (Lee CS) which is responsible for managing the operation of this service. Lee CS normally appoints an assessor to conduct a physical inspection of damaged vehicle and checks vehicle repair estimates with the garage. When repairs are completed, the garage will issue an invoice to Lee CS which will check the invoice against the original estimate. Lee CS sends all invoices to AGFIL, which in turn finalizes the payment processes. This scenario has been formalized, modeled, and verified in terms of commitments for instance by Bentahar et al. (2012), Desai et al. (2009) and Kalia and Singh (2015) where the contractual business relationships among the interacting parties are clearly identified, and also in terms of trust in Singh (2011). Our modeling of trust is based on the assumption that the trust relationship among the parties involved influences their decisions without any contractual relationships (commitments) among them.

The present article is organized as follows. In Section 2, we review the syntax and semantics of the TCTL logic framework. We introduce the conditional trust along with its syntax, semantics, and logical relationship with preconditional trust in Section 3. Further, we present our formal transformation algorithms to model check TCTL and conditional trust in Section 4 along with the soundness and completeness proofs. We analyze the complexity of TCTL and TCTL^C model checking in Section 5. In Section 6, we present the experimental results obtained using the tool that implements our transformation algorithms. We discuss the related work in Section 7. We conclude and identify future research directions in Section 8.

2. Theoretical background

2.1. Trust and temporal logic - TCTL

In Drawel et al. (2020c), we introduced TCTL, a temporal logic of trust that extends the Computation Tree Logic (CTL) (Emerson, 1990) to enable reasoning about trust and time. The syntax and semantics of TCTL are as follows:

Definition 1 (*Syntax of TCTL*)**.** The syntax of TCTL is defined recursively as follows:

$$\varphi ::= true \mid \rho \mid \neg\varphi \mid \varphi \vee \varphi \mid EX\varphi \mid EG\varphi \mid E(\varphi U \varphi) \mid T_p(i, j, \varphi, \varphi)$$

where $\rho \in AP_t$ is an atomic proposition from the set of atomic propositions AP_t for the TCTL logic, E is the existential quantifier over paths, the formula $EX\varphi$ stands for "φ holds in the next state in at least one path"; $EG\varphi$ stands for "there exists a path in which φ holds globally", and the formula $E(\varphi\ U \psi)$ holds at the current state if there is some future moment for which ψ holds and φ holds at all moments until that future moment. $EF\varphi$ is the abbreviation of $E(true\ U \varphi)$. A, the universal quantifier over paths, can be defined in terms of the above as usual: $AX\varphi = \neg EX\neg\varphi$; $AG\varphi = \neg EF\neg\varphi$; and $A(\varphi\ U\ \psi) = \neg(E(\neg\psi\ U\ (\neg\varphi \wedge \neg\psi)) \vee EG\neg\psi)$. The modality $T_p(i, j, \psi, \varphi)$ stands for *"Preconditional Trust"* and is read as "the truster i trusts the trustee j to bring about φ given that the precondition ψ holds". That is, we have the trust over the content given that the precondition is satisfied.

Example 1. The following formula represents a simple interaction between the policy holder and the call center in AGFIL setting:

$$AG(T_p(policy\ Holder, callCenter, reportAccident \wedge validClaim, assignGarage))$$

It states the policy holder trusts that the call center will assign a garage under the precondition that the former has reported an accident and the claim request is valid.

TCTL formulae are interpreted over the extended interpreted systems formalism. In Drawel et al. (2020c), we extended the original formalism of interpreted systems introduced in Fagin et al. (1995) to explicitly capture the trust relationship that is established between agents engaged in an interaction. Let us first introduce the original formalism, which is composed of:

- A set $Agt = \{1, \ldots, n\}$ of n agents where each agent $i \in Agt$ is described by:
  - A non-empty set of local states L_i, which represents the complete information that the agent can access at any time;
  - A set of local actions Act_i to model the temporal evolution of the system;
  - A local protocol $\rho_i : L_i \rightarrow 2^{Act_i}$ assigning a list of enabled actions that may be performed by agent i in a given local state l_i;
  - A local evolution function τ_i defined as: $\tau_i = L_i \times Act_i \rightarrow L_i$, which determines the transitions for an individual agent i between local states;
- A set of global states $s \in S$, where each state represents a snapshot of all agents in the system at a given time. A global state s is a tuple $s = (l_1 \ldots l_n)$. The notation $l_i(s)$ is used to represent the local state l_i of agent i in the global state s.
- A set of initial global states of the system $I \subseteq S$;
- The global evolution function of the system defined as follows: $\tau : S \times ACT \longrightarrow S$, where $ACT = Act_1 \times \cdots \times Act_n$ and each component $a \in ACT$ is called a joint action, which is a tuple of actions;
- As in Fagin et al. (1995), a special agent e is used to model the environment in which the agents operate. e is modeled using a set of local states L_e, a set of actions Act_e, a protocol ρ_e, and an evolution function τ_e.

The formalism of interpreted systems is enriched with trust function which associates to each local state $l_i \in L_i$ of each agent $i \in Agt$ in the global state $s \in S$ the trust vision of the truster towards other agents in the respective state (Drawel et al., 2020c). This vision is recorded in a data structure as values the truster associates to the other members of the system. This data structure is part of each local state of every agent in the system. Specifically, the trust function gives rise to a binary relation between two states which in fact combines the reachability and compatibility of the local states with respect to the recorded values. In practice, these values could be represented by a shared variables-based data structure as in Bentahar et al. (2012) and El-Menshawy et al. (2018) or by a vector-based data structure as in Drawel et al. (2020c).

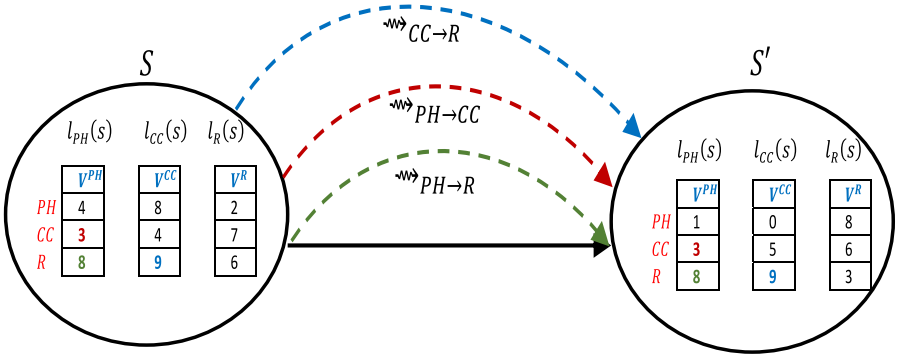


Fig. 2. An example of trust accessibility relations.

Here, we follow the later approach. That is, for each agent $i \in Agt$, a vector v^i of size $|Agt| = n$ is associated with each local state $l_i \in L_i$ of this agent. $v^i(i), v^i(j), \ldots, v^i(k)$ are the components of the vector v^i where $(i, j, \ldots, k) \in Agt^n$ $(i \neq j \neq \ldots \neq k)$. $v^i(j)(s)$ is the value of $v^i(j)$ in the global state s. This vector will be used to define the trust accessibility relation. Indeed, the set of local vectors v^i in the global states encodes the vision of agent i with regard to the trust of other agents. This extension allows us to provide a grounded semantics for trust that takes place between interacting parties.

Definition 2 (*Model of TCTL*)**.** A model of trust generated from vector-extended interpreted systems is a tuple $M_t = (S_t, R_t, I_t, \{\rightsquigarrow_{i\to j} | (i,j) \in Agt^2\}, V_t)$, where:

- S_t is a non-empty set of reachable global states for the system;
- $R_t \subseteq S_t \times S_t$ is the transition relation;
- $I_t \subseteq S_t$ is a set of initial global states for the system;
- $\rightsquigarrow_{i\to j} \subseteq S_t \times S_t$ is the direct trust accessibility relation for each truster–trustee pair of agents $(i,j) \in Agt^2$ defined by $s_t \rightsquigarrow_{i\to j} s'_t$ iff:
  - $v^i(j)(s_t) = v^i(j)(s'_t)$;
  - s'_t is reachable from s_t using transitions from the transition relation R_t;
- $V_t : S_t \to 2^{AP_t}$ is a labeling function, where AP_t is the set of atomic propositions for the TCTL logic.

As in Drawel et al. (2020c), the intuition behind the relation $\rightsquigarrow_{i\to j}$ is, for agent i to gain trust in agent j, the former identifies the states that are compatible with their trust vision with regard to the latter, i.e., where agent i is expecting that agent j is trustful. Specifically, this is obtained by comparing the element $v^i(j)$ in the local state l_i at the global state s_t (i.e., $v^i(j)(s_t)$) with $v^i(j)$ in the local state l_i at the global state s'_t (i.e., $v^i(j)(s'_t)$). Thus, the trust accessibility of agent i towards agent j (i.e., $\rightsquigarrow_{i\to j}$) does exist only if the element value that we have for agent j in the vector of the local states of agent i for both global states is the same, i.e., $v^i(j)(s_t) = v^i(j)(s'_t)$. Finally, infinite sequences of states linked by transitions define paths. If π is a path, then $\pi(i)$ is the $(i+1)$th state in π.

The trust accessibility relation between two global states is not constrained to only two agents, as the trust can be established between n agents at two global states using n vectors representing the trust vision of the truster towards other agents. According to our case study described in Section 1.1, suppose a policyholder's car is damaged in an accident, then an insurance claim is filed to the call center, which helps the policyholder in filing the claim and identifies a repairer to carry out the repairs. Therefore, we have the agents *PolicyHolder*(PH), *CallCenter*(CC), and *Repairer*(R) communicating together to form a MAS. In this case, each global state will contain 3 agents where a vector-based data structure of size 3 is part of each local state of every agent in the system. A binary relation between the two global states is raised based on the compatibility of their local states with respect to the recorded vector values. Fig. 2 depicts an example of a trust accessibility relations between two global states (s and s'). In this example, the solid line represents the transition relation from R_t, and the dashed lines represent the direct trust accessibility relations between the 3 agents. The state s' is compatible with s with regard to the trust the agent CC has towards the agent R, the agent PH has towards the agent CC, and the agent PH has towards the agent R. In the figure, we assign a vector to each agent's local states as follows: v^{PH}, v^{CC} and v^R are the vectors of *PolicyHolder*, *CallCenter*, and *Repairer* respectively. For instance, the agent *PolicyHolder* compares the element of her vector with regard to the agent *CallCenter* at global states s and s' to establish the existence of the trust accessibility relation $\rightsquigarrow_{PH\to CC}$. The particular element value of the agent *PolicyHolder* is the same in both global states (i.e., $v^{PH}(CC)(s) = v^{PH}(CC)(s') = 3$). Similarly, there is a trust accessibility relation $\rightsquigarrow_{PH\to R}$ from *PolicyHolder* towards *Repairer* $(v^{PH}(R)(s) = v^{PH}(R)(s') = 8)$ and a trust accessibility relation $\rightsquigarrow_{CC\to R}$ from *CallCenter* towards *Repairer* $(v^{CC}(R)(s) = v^{CC}(R)(s') = 9)$.

Proposition 1 (*Complexity of the Accessibility Relation*)**.** *The accessibility relations of the model M_t can be computed in space $O(\log^2 |Agt| + log^2|M_t|)$*

Proof. Computing the accessibility relation $s_t \rightsquigarrow_{i\to j} s'_t$ given two agents i and j requires computing the reachability from the state s_t. The reachability from s_t is a graph accessibility problem, and it is known by Jones (1975) that the problem is in NLOGSPACE, so it can be done nondeterministically in space $O(log|M_t|)$, or, by Savitch's theorem (Savitch, 1970), deterministically in space $O(log^2|M_t|)$. Since computing the reachability is independent from the agents, computing the accessibility for all the agents will require computing the reachability once, and compare the values of the local vectors for each pair of agents. The problem of comparing the values of all the pairs is in NLOGSPACE and can be solved nondeterministicaly in $O(\log |Agt|)$ by guessing a pair each time until all the pairs are covered by transitivity. Thus, the problem can be solved deterministically in $O(\log^2 |Agt|)$, so the proposition. □

Definition 3 (*Semantics of TCTL*)**.** Given the model M_t, the satisfaction for a TCTL formula φ in a global state s_t, denoted as $(M_t, s_t) \models \varphi$, is recursively defined as follows:

$-(M_t, s_t) \models \rho$ iff $\rho \in V_t(s_t)$;

$-(M_t, s_t) \models \neg\varphi$ iff $(M_t, s_t) \not\models \varphi$;

$-(M_t, s_t) \models \varphi_1 \vee \varphi_2$ iff $(M_t, s_t) \models \varphi_1$ or $(M_t, s_t) \models \varphi_2$;

$-(M_t, s_t) \models EX\varphi$ iff there exists a path π starting at s_t such that $(M_t, \pi(1)) \models \varphi$;

$-(M_t, s_t) \models E(\varphi_1 U \varphi_2)$ iff there exists a path π starting at s_t such that for some $k \geq 0$, $(M_t, \pi(k)) \models \varphi_2$ and $\forall 0 \leq i < k$, $(M_t, \pi(i)) \models \varphi_1$;

$-(M_t, s_t) \models EG\varphi$ iff there exists a path π starting at s_t such that $(M_t, \pi(k)) \models \varphi, \forall k \geq 0$;

$-(M_t, s_t) \models T_p(i, j, \psi, \varphi)$ iff $(M_t, s_t) \models \psi \wedge \neg\varphi$ and $\exists s'_t \neq s_t$ such that $s_t \rightsquigarrow_{i \to j} s'_t$, and $\forall s'_t \neq s_t$ such that $s_t \rightsquigarrow_{i \to j} s'_t$, we have $(M, s'_t) \models \varphi$.

Excluding the trust modality, the semantics of TCTL state formulae is defined as usual (semantics of CTL, since the main component of TCTL is CTL). The state formula $T_p(i, j, \psi, \varphi)$ is satisfied in the model M_t at s_t iff (1) there exists a state s'_t such that $s'_t \neq s_t$ and $s_t \rightsquigarrow_{i \to j} s'_t$, and (2) all the trust accessible states s'_t that are different from the current state s_t satisfy the content of trust φ.

3. Conditional trust TCTL^C

In Singh (2011), Singh propounds that trust must be conditional, meaning that trust should be expressed using antecedents and consequents. For example, a customer may trust a merchant as follows: "if I pay, then I trust the merchant will deliver the goods" (Singh, 2011). Such a statement expresses the customer's expectation and the effect of this expectation on their future plans. Our preconditional trust modality that assumes the prior satisfaction of the precondition is different from conditional trust. Expressing conditional trust requires an extension of TCTL, and to distinguish the two languages, the extended one is called TCTL^C. However, there is a logical relationship between preconditional and conditional trust. In fact, as our main objective in the paper is the verification of temporal trust, we will show how this logical relationship will be exploited to introduce a model checking procedure for conditional trust (see Section 4.3). The idea we aim to convey is that it is possible to decide if a given state, and thus a given model, satisfies a conditional trust formula by calling the model checking of TCTL. To show this, let us first introduce the syntax and semantics of conditional trust. From the syntax perspective, $T_c(i, j, \psi, \varphi)$ is read as "agent i trusts agent j about the consequent φ when the antecedent ψ holds". It is worth noticing that in the case of precondition trust, for the trust to take place between the interacting agents i and j, the condition $\psi \wedge \neg\varphi$ must be satisfied in the current state s_t to ensure that the precondition ψ holds before the trust content φ is brought about, while conditional trust requires the existence of at least one accessible state satisfying the antecedent ψ. This condition captures the intuition that the satisfaction of the antecedent is possible in some future. The semantics of $T_c(i, j, \psi, \varphi)$ is as follows:

$(M_t, s_t) \models T_c(i, j, \psi, \varphi)$ iff $s_t \models \neg\varphi$ and $\exists s'_t \neq s_t$ such that $s_t \rightsquigarrow_{i \to j} s'_t$ and $s'_t \models \psi$, and $\forall s'_t \neq s_t$ such that $s_t \rightsquigarrow_{i \to j} s'_t$ and $(M_t, s'_t) \models \psi$, we have $(M_t, s'_t) \models \varphi$.

The non satisfaction of the consequent φ complies with the first postulate in Singh (2011) stating that when the consequent holds, the trust in this consequent is "completed and is, therefore, no longer active". The following proposition shows the logical link between conditional and preconditional trust:

Proposition 2 (*Conditional and Preconditional Trust*). $T_c(i, j, \psi, \varphi) \wedge \psi \equiv T_p(i, j, \top, \psi \to \varphi) \wedge \neg T_p(i, j, \top, \neg\psi)$.

The proof of this proposition is direct from the semantics.

Furthermore, it is worth mentioning that conditional trust $T_c(i, j, \psi, \varphi)$ is conceptually and semantically different from trust about conditions, which can be represented by $T_c(i, j, \top, \psi \to \varphi)$. An example of the former is "if the buyer i pays the seller j, then i trusts j will deliver the goods", while for the latter the example is: "the buyer i trusts the seller j about the fact that, if i pays, then j will deliver the goods".

4. Formal transformation to model check TCTL and conditional trust

In this section, we first introduce a transformation-based approach to address the problem of model checking TCTL. In a nutshell, given a model M_t representing a trust based MAS and a TCTL formula φ that describes the property that the model M_t has to satisfy, the problem of model checking TCTL can be defined as verifying whether or not φ holds in M_t, which is formally denoted by $M_t \models \varphi$. In particular, we apply specific reduction rules to formally transform the problem of model checking TCTL into the problem of model checking CTL (Emerson, 1990). This provides a way to perform our implementation on NuSMV. Technically, our transformation method encompasses two stages. First, we apply a set of formal rules to transform vector-extended transition systems into Kripke structures. Then, we transform TCTL formulae to CTL ones based on certain rules developed specifically for this purpose. Such a transformation is performed by developing two formal methods that provide accurate alignments between source and target models, and at the same time preserve TCTL semantics without losing the validity of the original model properties. This transformation is then extended to check conditional trust.

4.1. Transformation of TCTL model

In this section, we start by recalling the definition of the CTL model needed for the transformation algorithm.

Definition 4 (*Model of CTL*)**.** A CTL formula is interpreted over a Kripke Structure $M_c = (S_c, R_c, I_c, V_c)$, where:

- S_c is a non-empty set of states for the system;
- $R_c \subseteq S_c \times S_c$ is the transition relation;
- $I_c \subseteq S_c$ is a set of possible initial global states for the system;
- $V_c : S_c \to 2^{AP_c}$ is a labeling function that maps each state to the set of CTL propositional variables AP_c that hold in it.

Having presented the CTL model, the next step is to establish our transformation technique. Given a TCTL model $M_t = (S_t, R_t, I_t, \{\rightsquigarrow_{i \to j} | (i, j) \in Agt^2\}, V_t)$, Algorithm 1 shows how this model is transformed into a CTL model $M_c = (S_c, R_c, I_c, V_c)$. The main idea behind this transformation is to capture the accessibility relations in M_t through new distinguishable states, transitions and atomic propositions added in M_c. For example, if a state s_2 is accessible from a state s_0 for the agents i and j in the TCTL model M_t, a new intermediate state s_{02} between s_0 and s_2 is added to the CTL model M_c. This new state is labeled by two fresh atomic propositions α^{ij} and χ. The first one (α^{ij}) is to capture the agents i and j for which the accessibility exists. The second atomic proposition (χ) is to know that this state is a new state that does not exist in the original model, so we can avoid having a formula that is true in the CTL model without being true in the original TCTL model because of this new added state. That is, our approach distinguishes between original transitions in the original TCTL model and the transitions added in the CTL model with the aim to not affect the classical temporal operators of the CTL model. Thus, to check if s_2 is accessible from s_0 for the agents i and j in M_t, we only need to check that an intermediate state between these two states with a fresh atomic proposition α^{ij} does exist in M_c. On the other hand, by forcing χ to hold in the intermediate

state, we can simply make sure that $\neg\chi$ holds in a state of the CTL model to know we are not in an added intermediate state.

More specifically, the algorithm takes as input a model M_t (line 1) and outputs the transformed model M_c (line 2). First, the corresponding model M_c has the same set of system states and initial states (i.e., $S_c = S_t$; $I_c = I_t$). Thereafter, the algorithm initializes the set R_c, and then the set $V_c(s)$ to be equal to the set $V_t(s)$ (i.e., at the beginning, states are labeled with the same atomic propositions). We define a new set of atomic propositions for the CTL logic needed to represent the trust accessibility relation to capture the semantics of trust as follows $X = \{\alpha^{ij} | (i,j) \in Agt^2\}$. Moreover, as mentioned earlier, we define a new fresh atomic proposition χ for CTL that will be used to preserve the actual temporal transition relation. Thus, the set AP_c of atomic variables of CTL is as follows: $AP_c = X \cup AP_t \cup \{\chi\}$. The algorithm proceeds to transform transition and trust accessibility relations to constitute the transition relations in R_c based on two conditions. The first condition checks if the states s_t and s'_t have a transition relation in R_t, then this relation becomes a transition relation in R_c (lines 8 & 9). For the second condition, it checks if the current state s_t has an accessible state s'_t using the accessibility relation $\rightsquigarrow_{i\to j}$ for any truster–trustee pair of agents and this state is different from the state itself. Moreover, if the two states are not in R_c, then a new state s''_t is added to the set of system states S_c along with a new transition from s_t to s''_t and from s''_t to s'_t in R_c. Further, the new state s''_t is labeled with the atomic propositions α^{ij} and χ in order to distinguish the states that are accessible from any other next state that satisfies the trust formulae without having accessibility to the current state (line 14 & 15). However, if s''_t is already added for some other accessibility relations, we only add the atomic proposition α^{ij} to mark the accessible state for any other interacting agents (lines 11 & 12). Finally, the algorithm returns the transformed model M_c after iterating over all the transitions.

ALGORITHM 1: Transform $M_t = (S_t, R_t, I_t, \{\rightsquigarrow_{i\to j} | (i,j) \in Agt^2\}, V_t)$ into $M_c = (S_c, I_c, R_c, V_c)$

1: **Input**: the model M_t
2: **Output**: the model M_c
3: $S_c := S_t$;
4: $I_c := I_t$;
5: Initialize $R_c := \emptyset$;
6: Initialize $V_c(s_c) := V_t(s_t)$ for each $s_c \in S_c$ and $s_t \in S_t$ **such that** $s_c = s_t$;
7: **for each** $(s_t, s'_t) \in S_t^2$ **do**
8: **if** $(s_t, s'_t) \in R_t$ **then**
9: $R_c := R_c \cup \{(s_t, s'_t)\}$;
10: **if** $s_t \rightsquigarrow_{i\to j} s'_t$ **for all** $(i,j) \in Agt^2$ **and** $s'_t \neq s_t$ **then**
11: **if** $\exists s''_t$ **such that** $((s_t, s''_t), (s''_t, s'_t)) \in R_c$ and $\chi \in V_c(s''_t)$ **then**
12: $V_c(s''_t) := V_c(s''_t) \cup \{\alpha^{ij}\}$;
13: **else**
14: $S_c := S_c \cup \{s''_t\}$;
15: $R_c := R_c \cup \{(s_t, s''_t), (s''_t, s'_t)\}$ **and** $V_c(s''_t) := \{\chi, \alpha^{ij}\}$;
16: **end if**
17: **end for**
18: **return** M_c;

Proposition 3 (*Boundedness of Model Transformation*). $|M_c| \leq 3|M_t|^2$ *where* $|M_t|$ *(resp.* $|M_c|$*) is the size of the input model* M_t *(resp. the output model* M_c*).*

Proof. We have: $|M_c| = |S_c| + |R_c|$. In the worst case, each pair of distinct states in M_t is connected by one or many accessibility relations. In this case, the graph of accessibility relations is complete, so $|S_t|(|S_t|-1)$ new states and $2|S_t|(|S_t|-1)$ new transitions will be added in M_c. So we obtain, $|M_c| \leq |S_t| + |S_t|(|S_t|-1) + |R_t| + 2|S_t|(|S_t|-1)$. Consequently, $|M_c| \leq 3|S_t|^2 - 2|S_t| + |R_t|$, and thus, $|M_c| \leq 3|S_t|^2 + |R_t|$. The result follows from: $3|S_t|^2 + |R_t| \leq 3(|S_t| + |R_t|)^2$. □

4.2. Transformation of TCTL formulae

This section presents our method to formally transform any TCTL formula φ to a CTL formula $f(\varphi)$ using a recursive transformation function f. The details of this method are illustrated in Algorithm 2. The transformation of the CTL fragment of TCTL is straightforward (lines 1–3) where the set AP_t of TCTL atomic variables is considered ($AP_t \subset AP_c$). Yet, for the temporal operators (lines 4–6), we need to make sure that the transformation does not affect the CTL semantics. That is, since a new state and new transitions are added to the obtained model M_c, we have to make sure that if a formula is satisfied in the original model M_t through a path, the transformed formula is still satisfied through the corresponding path of the translated model M_c. Indeed, this is the main reason behind the conjunction of $\neg\chi$ for the temporal operators. This allows us to exclude the additional state and transitions when we consider the satisfaction of the formulae. For instance, if p is an atomic proposition, the formula EXp is transformed into the CTL formula $EX(p \wedge \neg\chi)$ stating that there exists a path through which the next state satisfies p and the negation of the atomic proposition χ (added to represent the temporal transition). This next state cannot be a new added state since any added state satisfies χ. For the trust modality (line 7), the trust formula is transformed inductively into a CTL formula according to the defined semantics as follows: (1) the transformation of the formula $\psi \wedge \neg\varphi$ should hold in the current state, which captures the first condition in the semantics of T_p; (2) there exists a path where next state satisfies the added atomic proposition α^{ij}, which captures the existence of an accessible state; and (3) along each path, if the next state on that path satisfies the atomic proposition α^{ij}, then the next state of the added state also satisfies the transformation of the trust content φ, which captures the third condition in the semantics of T_p stating that all accessible states satisfy the trust content. For instance, if p and q are atomic propositions, the trust formula $T_p(i,j,p,q)$ is transformed into the CTL formula $p \wedge \neg q \wedge EX(\alpha^{ij}) \wedge AX(\alpha^{ij} \to AXq)$. The double use of the operator AX is justified by the fact that the accessible states are captured through the next states of the states that satisfy α_{ij}, which means the next states of the next states of the current state. Thus, the two next operators AX reflect the added state and the two added transitions, and the antecedent α^{ij} in the implication is to insure that we are considering a new added state, which is labeled by the atomic proposition α^{ij}.

ALGORITHM 2: Transform TCTL formula φ into CTL formula $f(\varphi)$

1: $f(p) = p$ if p is an atomic proposition from the set AP_t;
2: $f(\neg\varphi) = \neg f(\varphi)$;
3: $f(\varphi \vee \psi) = f(\varphi) \vee f(\psi)$;
4: $f(EX\varphi) = EX(f(\varphi) \wedge \neg\chi)$;
5: $f(E(\varphi \ U \ \psi)) = E((f(\varphi) \wedge \neg\chi) \ U \ (f(\psi) \wedge \neg\chi))$;
6: $f(EG\varphi) = EG(f(\varphi) \wedge \neg\chi)$;
7: $f(T_p(i,j,\psi,\varphi)) = f(\psi) \wedge f(\neg\varphi) \wedge EX(\alpha^{ij}) \wedge AX(\alpha^{ij} \to AXf(\varphi))$;

Fig. 3 depicts an example illustrating the transformation of a TCTL model and some TCTL formulae (part a) to a CTL model and CTL formulae (part b) using our transformation technique. On the left side of the figure (part a), the model M_t consists of four

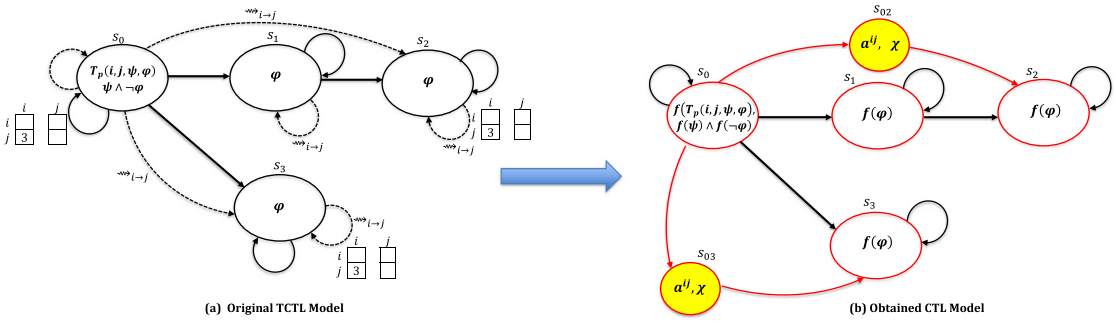


Fig. 3. Example of the transformation methods.

global states s_0, s_1, s_2, and s_3. The states s_2 and s_3 are accessible from s_0. Furthermore, the trust formula $(T_p(i,j,\psi,\varphi))$ holds in s_0 (i.e., $(M_t,s_0) \models T_p(i,j,\psi,\varphi)$). According to the semantics, we obtain $(M_t,s_0) \models \psi \wedge \neg\varphi$, and there exists a state s' such that $s' \neq s$ and $s \rightsquigarrow_{i\to j} s'$, and all the trust accessible states s' such that $s \neq s'$ satisfy φ (i.e., $(M_t,s_2) \models \varphi$) and $(M_t,s_3) \models \varphi$. Using the proposed transformation technique, the model M_t is transformed into the CTL model M_c of the right side (part b) as follows: the temporal transitions in M_t are transformed into transition relations in M_c. Further, the accessibility relations in M_t are transformed into transition relations in M_c as follows: new intermediate states are added to the set of states in M_c (in the figure, s_{02} between s_0 and s_2; and s_{03} between s_0 and s_3 are added in S_c) along with new transitions between each two accessible states and the new states (i.e., (s_0,s_{02}), (s_{02},s_2) and (s_0,s_{03}), (s_{03},s_3)), and the atomic propositions α^{ij} and χ are added to represent the accessibility relations and mark the states as new. Moreover, each state formula in TCTL is transformed into a CTL formula using the transformation function f. Thus, the formulae $T_p(i,j,\psi,\varphi)$ and $\psi \wedge \neg\varphi$ are transformed into $f(T_p(i,j,\psi,\varphi))$ and $f(\psi) \wedge f(\neg\varphi)$ in state s_0, and for every path, if the next state satisfies the added atomic proposition α^{ij} (which is the case of the states s_{02} and s_{03}), then every next state of that state will satisfy the transformation of the trust content φ (which is the case of the states s_2 and s_3).

Proposition 4 (*Boundedness of Formula Transformation*)**.** *Let φ be a TCTL formula and f the transformation function defined in Algorithm 2. There exists a constant k such that $|f(\varphi)| < k|\varphi|$.*

Proof. The proof is by induction on the structure of the formula.

- The result holds for the atomic proposition (the base case).
- For the formula $\phi = EX\varphi$, we have $|f(\phi)| = |f(\varphi)| + 4$. Therefore, by assumption that the proposition holds for the formula φ, $\exists k_1$ such that $|f(\phi)| < k_1|\varphi| + 4$. Since $|\varphi| < |\phi|$, and $|\phi| > 1$, we get $|f(\phi)| < (k_1+4)|\phi|$, so the proposition. The result is also similar for the $EG\varphi$ formula.
- For the formula $\phi = E(\varphi\ U\ \psi)$, we have $|f(\phi)| = |f(\varphi)| + |f(\psi)| + 7$. Thus. by assumption that the proposition holds for the formulae φ and ψ, $\exists k_1, k_2$ such that $|f(\phi)| < k_1|\varphi| + k_2|\psi| + 7$. Because $|\varphi| < |\phi|$, $|\psi| < |\phi|$, and $|\phi| > 1$, we obtain $|f(\phi)| < (k_1+k_2+7)|\phi|$.
- For the formula $\phi = T_p(i,j,\psi,\varphi)$, we have $f(T_p(i,j,\psi,\varphi)) = f(\psi) \wedge f(\neg\varphi) \wedge EX(\alpha^{ij}) \wedge AX(\alpha^{ij} \to AXf(\varphi))$. Thus, $|f(\phi)| = |f(\psi)| + 2|f(\varphi)| + 10$, and by assumption that the proposition holds for the formulae ψ and φ, $\exists k_1, k_2$ such that $|f(\phi)| < k_1|\psi| + 2k_2|\varphi| + 10$. Since $|\psi| < |\phi|$, $|\varphi| < |\phi|$, and $|\phi| > 1$, we get $|f(\phi)| < (k_1 + 2k_2 + 10)|\phi|$ (i.e., $k = k_1 + 2k_2 + 10$), so the proposition. □

Theorem 1 (*Soundness and Completeness of the Transformation*)**.** *Let M_t and φ be respectively a TCTL model and formula and let M_C and $f(\varphi)$ be the corresponding model and formula in CTL. We have $(M_t,s_t) \models \varphi$ iff $(M_c,s_c) \models f(\varphi)$, where s_c is the corresponding state of s_t in M_c.*

Proof. We prove this theorem by induction on the structure of the formula φ.

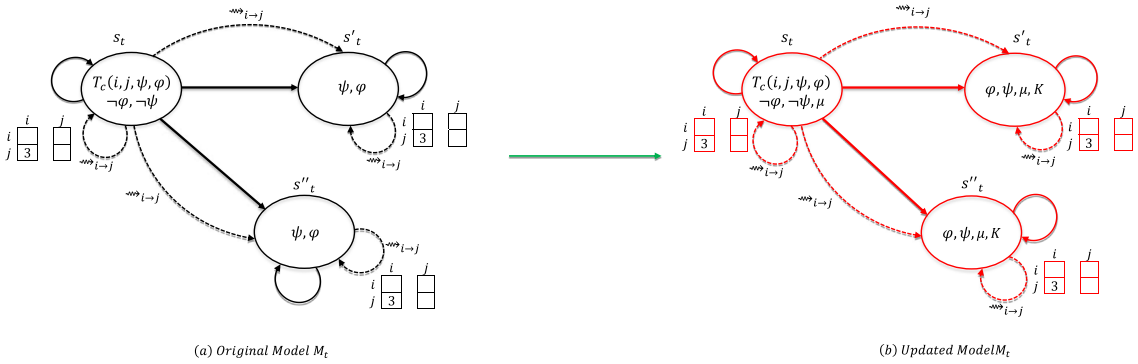
- For the atomic propositions from the set AP_t, the result is straightforward (notice that $AP_t \subset AP_c$).
- For the formula $\phi = EX\varphi$, we have $(M_t,s_t) \models EX\varphi$ iff there exists an immediate successor to a state where φ holds. Consequently, from the definition of $f(M_t)$ and $f(\varphi)$, we obtain $(M_t,s_t) \models EX\varphi$ iff $(M_c,s_c) \models EX(f(\varphi) \wedge \neg\chi)$. That is, we are excluding the new added path as this path will never be considered because next state (the added state) has χ and we are forcing $\neg\chi$.
- For the formula $\phi = E(\varphi\ U\ \psi)$, and from the definition of f, $(f(\varphi) \wedge \neg\chi)\ U(f(\psi) \wedge \neg\chi)$ captures the semantics of Until in CTL which states the existence of a path starting in the current state that satisfies $(\varphi \wedge \neg\chi)$ until reaching a state in which $(\psi \wedge \neg\chi)$ holds.
- The trust formula: $T_p(i,j,\psi,\varphi)$. The first and second parts: $(f(\psi) \wedge f(\neg\varphi))$ capture the first condition of the semantics where the current state should satisfy $\psi \wedge \neg\varphi$. The third part $(EX(\alpha^{ij}))$ captures the second condition, i.e., the existence of an accessible state different from the current state since α^{ij} holds only in such accessible states (line 13 of Algorithm 1). Finally, the fourth part $(AX(\alpha^{ij} \to AXf(\varphi)))$ captures the last condition in the semantics of the trust formula where all accessible states (those satisfying α^{ij} in M_c) should satisfy φ. □

4.3. Model checking conditional trust

A similar approach to model checking TCTL can be used to model check conditional trust by transforming the conditional trust formula of TCTL^C to a CTL formula as follows:

$$f(T_c(i,j,\psi,\varphi)) = f(\neg\varphi) \wedge EX(\alpha^{ij}) \wedge AX(\alpha^{ij} \to AX(f(\psi) \to f(\varphi)));$$

In this section, we introduce an alternative transformation solution (Algorithm 3) that transforms the problem of model checking T_c into the problem of model checking T_p so it can use the developed model checking algorithm of TCTL. Algorithm 3 merges both model and formula transformation as follows. It capitalizes on the equivalence shown in Proposition 2, which gives us a direct way to transform the problem of model checking $T_c(i,j,\psi,\varphi)$ into the problem of model checking two formulas: $T_p(i,j,\top,\psi \to \varphi)$ and $\neg T_p(i,j,\top,\neg\psi)$ (line 3). However, if the conjunction of the two formulas is not satisfied, then an additional transformation of the model is needed by adding fresh atomic propositions μ and κ



(a) Original Model M_t (b) Updated ModelM_t

Fig. 4. Illustrative example of Algorithm 3.

to different states based on the valuation of these states (lines 4 to 8). By so doing, the problem of model checking $T_c(i,j,\psi,\varphi)$ is transformed into the problem of model checking $T_p(i,j,\neg\varphi,\kappa)$ and $\neg T_p(i,j,\top,\neg\mu)$ (lines 9 and 10).

More precisely, if the first transformation cannot be used because the conjunction $T_p(i,j,\top,\psi\to\varphi)\wedge\neg T_p(i,j,\top,\neg\psi)$ does not hold, then updating the valuation function will be needed. Such an update works in all cases, but to be more efficient, the algorithm uses it only if the direct transformation (line 3) fails. The update, which also exploits Proposition 2, introduces two fresh atomic propositions: μ and κ. μ holds in the current state (line 4) and in every state where ψ holds (line 7). Thus, if $T_p(i,j,\top,\neg\mu)$ does not hold, which means $\neg T_p(i,j,\top,\neg\mu)$ holds (condition 1), then the only reason is because there is an accessible state different from the current state where ψ holds. This is because the first condition in the semantics of $T_p(i,j,\top,\neg\mu)$ already holds since μ holds in s_t (line 4). κ does not hold in the current state, but holds in every state where $\psi\to\varphi$ holds. Consequently, if $T_p(i,j,\neg\varphi,\kappa)$ holds (condition 2), then $\neg\varphi$ holds and all accessible states different from the current state satisfy $\psi\to\varphi$. The algorithm returns true if the two conditions 1 and 2, which correspond to the semantics of $T_c(i,j,\psi,\varphi)$, hold (line 9), false otherwise (line 10). Fig. 4 depicts an example illustrating the transformation in Algorithm 3. In this example, the condition of line 3 does not hold since ψ does not hold in s_t. Thus, the model transformation is required as illustrated in the part (b). The condition of line 9 holds in the transformed model, making the conditional trust formula $T_c(i,j,\psi,\varphi)$ true. The following theorem is direct from the semantics of conditional trust and Proposition 2.

Theorem 2 (*Soundness and Completeness of Algorithm 3*)**.** *Algorithm 3 returns true iff* $(M_t,s_t)\models T_c(i,j,\psi,\varphi)$.

ALGORITHM 3: Transform the model checking of T_c into the model checking of T_p

1: **Input**: M_t, s_t, i, j, ψ, φ
2: **Output**: **true** if $(M_t,s_t)\models T_c(i,j,\psi,\varphi)$; **false** otherwise
3: **if** $(M_t,s_t)\models T_p(i,j,\top,\psi\to\varphi)\wedge\neg T_p(i,j,\top,\neg\psi)$ **then return true**;
4: $V_t(s_t):=V_t(s_t)\cup\{\mu\}$;
5: **for all** $s'_t\neq s_t$
6: **if** $(M_t,s'_t)\models\psi\to\varphi$ **then** $V_t(s'_t):=V_t(s'_t)\cup\{\kappa\}$;
7: **if** $(M_t,s'_t)\models\psi$ **then** $V_t(s'_t):=V_t(s'_t)\cup\{\mu\}$;
8: **end for**
9: **if** $(M_t,s_t)\models T_p(i,j,\neg\varphi,\kappa)\wedge\neg T_p(i,j,\top,\neg\mu)$ **then return true**;
10: **return false**;

5. Complexity analysis

In this section, we will first analyze the time complexity of model checking TCTL and conditional trust (TCTL^C) with regard to the size of the explicit model M_t and length of the formula to be checked. Thereafter, we will analyze the space complexity of model checking TCTL and TCTL^C for concurrent programs with respect to the size of the components of these programs and length of the formula.

As our approach is transformation-based, we start by analyzing the time complexity of transforming the TCTL model and formula with respect to explicit models, where all states and transitions are enumerated. Specifically, we prove that these two transformations are polynomial with respect to the input TCTL model and linear with respect to the formula. The polynomial and linear complexity of these two transformations entails the P-completeness of the TCTL model checking problem in explicit models. Thereafter, we derive the complexity of TCTL^C directly from the one of TCTL. Given that, we proceed to analyze the space complexity of the TCTL and TCTL^C model checking problems and prove their PSPACE-completeness with respect to concurrent programs where the model has the form of a synchronized product of agent programs. Indeed, our motivation behind considering the complexity of our model checking algorithms for concurrent programs that provide compact representations of the systems to be checked is that in practice, existing model checking tools (e.g., MCMAS and NuSMV) do not support explicit representations where states and transitions are listed explicitly (as Kripke-like structures). In fact, only local states and transitions of each component are represented. Therefore, the actual system can still be represented by combining local states and transitions to build reachable states.

5.1. Time complexity

In this subsection, we will prove that model checking TCTL and TCTL^C in explicit models are P-complete, so they can be done in a polynomial running time with respect to the size of the model and length of the formula.

Theorem 3 (*Explicit Model Checking TCTL: Upper Bound*)**.** *The model checking problem of TCTL can be solved in time* $O(|M_t|^2\times|\varphi|)$ *where* $|M_t|$ *and* $|\varphi|$ *are the size of the vector-extended model and length of the TCTL formula, respectively.*

Proof. TCTL extends CTL, and it is known from Clarke et al. (1999) that CTL model checking can be done in a linear time with respect to the size of the CTL model and formula, i.e., $O(|f(M_t)|\times|f(\varphi)|)$. From Proposition 3, $|f(M_t)|\leq 3|M_t|^2$, i.e., the size of $f(M_t)$ is polynomial with the size of M_t. Moreover, from Proposition 4, the

length of $f(\varphi)$ is linear with the length of φ. Indeed, Algorithm 2 takes the TCTL formula φ as input and writes in a recursion manner the corresponding CTL formula according to the structure of φ. The time complexity of transforming the TCTL formula is linear with respect to the length of the input formula φ. This follows from the fact that (1) the length of the recursion is bounded by the size of the input formula φ, and (2) the size of $f(\varphi)$ is bounded by the size of φ, so the theorem. □

Corollary 1 (*Explicit Model Checking TCTL^C: Upper Bound*)**.** *Model checking conditional trust can be solved in time $O(|M_t|^2 \times |\varphi|)$. where $|M_t|$ and $|\varphi|$ are the size of the vector-extended model and length of the conditional trust formula, respectively.*

Proof. The result is straightforward for the first method (transformation method) as it is similar to the model checking of TCTL. For the second solution, Algorithm 3 is simply calling the model checking of TCTL with an additional update of the model. The result follows from the fact that the update does not affect the asymptotic size of the model or the formula, but only adds atomic propositions to some states. □

Theorem 4 (*Explicit Model Checking TCTL: Completeness*)**.** *The problem of TCTL model checking is P-complete.*

Proof. Membership (i.e., upper bound) in P follows from Theorem 3. Hardness (i.e., lower bound) in P is a result of the polynomial reduction from model checking CTL proved to be P-complete in Schnoebelen (2002). □

The following corollary is direct from Theorem 4 and Corollary 1:

Corollary 2 (*Explicit Model Checking TCTL^C: Completeness*)**.** *The problem of model checking conditional trust is P-complete.*

5.2. Space complexity

In this subsection, we will prove that the complexity of TCTL model checking for concurrent programs is PSPACE-complete and so is TCTL^C model checking. This result means that there is an algorithm solving the problem in polynomial space in the size of the components constituting concurrent programs and the length of the formula being model checked.

5.2.1. Concurrent programs

A concurrent program P as introduced in Kupferman et al. (2000), is composed of n concurrent processes P_i (modules, protocols, or agents). Each process is described by a transition system D_i defined as follows: $D_i = (AP_i, AC_i, S_i, \Delta_i, s_i^0, L_i)$, where AP_i is a set of local atomic propositions, AC_i is a local action alphabet, S_i is a finite set of local states, $\Delta_i \subseteq S_i \times AC_i \times S_i$ is a local transition relation, $s_i^0 \in S_i$ is an initial state, and $L_i : S_i \to 2^{AP_i}$ is a local state labeling function.

A concurrent behavior of these processes is obtained by the product of the processes and transition actions that appear in several processes are synchronized by common actions. The joint behavior of the processes can be described using a global transition system D, which is computed by constructing the reachable states of the product of the processes and synchronization is obtained using common action names. This product is the transition system $D = (AP, AC, S, \Delta, s^0, L)$ where:

$-AP = \bigcup_{i=1}^{n} AP_i$

$-AC = \bigcup_{i=1}^{n} AC_i$

$-S = \prod_{i=1}^{n} S_i$. The ith component of a state $s \in S$ is denoted by $s[i]$

$-(s, a, s') \in \Delta$ iff:

1. for all $1 \leq i \leq n$ such that $a \in AC_i$ we have$(s[i], a, s'[i]) \in \Delta_i$, and
2. for all $1 \leq i \leq n$ such that $a \notin AC_i$ we have $s[i] = s'[i]$

$-s^0 = (s_1^0, s_2^0, s_3^0, \ldots, s_n^0)$

$-L(s) = \prod_{i=1}^{n} L_i(s[i])$ for every $s \in S$, where $s[i]$ is the ith component of s

Theorem 5 (*Polynomial Space Reduction of Model Checking TCTL: Upper Bound*)**.** *Let $\sqsubseteq_{psr}$ denote the polynomial-space reduction. The problem of model checking TCTL can be reduced into the problem of model checking CTL in a polynomial space, i.e., MC(TCTL) $\sqsubseteq_{psr}$ MC(CTL).*

Proof. The transformation of the TCTL model and TCTL formula into the corresponding CTL model and formula could be computed by a deterministic Turing Machine (*TM*) in space $O(\log n)$ where n is the size of the input TCTL model, and polynomial space w.r.t. the size of the TCTL formula. For the model, *TM* reads in the input tape a model of TCTL and produces in the output tape, one-by-one, the same states including the initial ones and the same valuations. Then, for the transitions (s_t, s_t') in the input model, it writes one-by-one, the transitions in the set R_c. Moreover, it reads the accessibility relations $\rightsquigarrow_{i \to j}$ between two given states in the input model one-by-one and for each one, it adds an intermediate state to the set S_c labeled with two fresh atomic propositions: (1) α^{ij} that depends on the accessibility relation, and (2) χ, along with two transitions if such a state does not already exist; otherwise, only the atomic proposition α^{ij} is added. All these writing operations are clearly logarithmic in space because this transformation is done on-the-fly, step-by-step. Moreover, we showed in Proposition 4 that any TCTL formula is transformable into a CTL formula whose size is linearly bounded by the size of the input formula. All these recursive transformations are clearly polynomial space in the length of the input formula, so the theorem. □

It is worth mentioning that our approach transforms the model checking of TCTL into the model checking of CTL, which uses a top-down algorithm requiring space that is linear in the length of the formula, but only poly-logarithmic in the size of the Kripke structure (Kupferman et al., 2000). This entails that for concurrent programs, the model checking algorithm is solved in polynomial space with respect to the size of the program description, rather than requiring space of the order of the exponentially larger expansion of the program, which alleviates the state explosion problem. Moreover, in practice, concurrent programs used in symbolic model checking have a succinct representation using binary decision diagrams (BDDs) as a combination of small components. The BDD representation is often exponentially smaller in practice. The concurrent program is built by combining the individual components using some type of parallel mechanism with defined synchronization rules. It is known that the concurrent program can be constructed in PSPACE (Schnoebelen, 2002). In fact, the model checking is performed on-the-fly, which means without holding the whole structure to be checked in memory at any one time, which is the reason behind the poly-logarithmic space complexity in the size of explicit models (El-Menshawy et al., 2013). Moreover, in the proof of Theorem 5, we showed that the transformation of the TCTL concurrent program model into the corresponding CTL model could be computed by a deterministic TM in a logarithmic space with respect to the size of the input TCTL model, which means that our transformation is not a source of state explosion.

Theorem 6 (*Model Checking TCTL for Concurrent Programs: Completeness*)**.** *The space complexity of the TCTL model checking for concurrent programs is PSPACE-complete with respect to the size of the components of these programs and the length of the formula.*

Proof. Since model checking CTL is PSPACE-complete for concurrent programs (Schnoebelen, 2002), the lower bound of model checking TCTL is PSPACE as well. In fact, TCTL subsumes CTL as it integrates the CTL modalities and the trust modality. The upper bound in PSPACE follows from Theorem 5, so the result. □

Corollary 3 (*Model Checking Conditional Trust for Concurrent Programs: Completeness*)**.** *The space complexity of model checking conditional trust for concurrent programs is PSPACE-complete with respect to the size of the components of these programs and the length of the formula.*

Proof. The corollary is direct from Algorithm 3 and Theorem 6 since adding atomic propositions to particular states and calling TCTL model checking with formulae having linear size with the size of the input formula are not source of additional space complexity. □

6. Implementation and experimental results

6.1. Implementation

To carry out the experimental process as efficiently as possible, we have developed an open source java toolkit[1] with a high-degree of automation that interacts automatically with the NuSMV model checker. Our toolkit essentially consists of two main modules implementing the proposed transformation algorithms (Algorithms 1,2, and 3). The main feature of our tool is its ability to automatically transform TCTL and TCTL^C models and formulae into the corresponding CTL models and formulae. Technically, the NuSMV model checker is used as a core component by the toolkit engine in order to perform the verification aspects. Our tool takes as input the encoding of the MAS model, analyzes the code with respect to a set of formulae, and generates the required SMV modules according to the number of interacting agents. Besides, we implemented lexer and parser that check the syntax of both TCTL and TCTL^C.

In our encoding, we formalized the MAS business scenario introduced in 1.1 using our trust model M_t associated with the formalism of vector-extended interpreted systems introduced earlier in Section 3 to formally model the protocol. Thus, we have six interacting agents: *Ins* playing the role of AGFIL, PolicyHolder, CallCenter, *Repairer* playing the role of Garage, *Assessor* playing the role of Lee CS, and Adjustor plus the environment agent that facilitates the communication among these agents. More precisely, we encoded each agent in our VISPL input language of the MCMAS-T model checker introduced in Drawel et al. (2020c).

Our extension of the MCMAS tool in Drawel et al. (2020c) generates an extended version of both MCMAS and its input language ISPL, which we called respectively MCMAS-T and VISPL. Technically, the extension process is performed in the utilities of MCMAS by adding OBDD symbolic algorithms implemented in C++ in order to respectively compute the set of values of each vector for any pair (i, j) of agents and build the accessibility relations such that for any global states s_1 and s_2, s_2 is accessible from s_1 iff the value of the element that we have for agent j in the vector of the local states of agent i for both global states is the same (see Definition 2). A distribution of MCMAS-T is publicly available online,[2] along with the source code and case studies.

Here, we present a VISPL fragment for the encoding of the *CallCenter* agent declared by means of the set of its local states and actions, the local protocol, and the local evolution function which describes how the agent local states evolve. In particular, we exploit the sections `Agent CallCenter ... end Agent` to encode the protocol specification itself, and the roles played by this agent in the system. We also consider the accessibility relations by encoding the vector variables, which give a particular agent the possibility to establish the trust towards other agents. For example, we define the vector $d_1 : \{d0, d1\}$ in the local state for the *CallCenter* agent to allow for the trust accessibility relation that involves this agent and the *PolicyHolder* agent to take place when the global states are built by the tool from the Cartesian product of the local states. We can observe that the value of $d_1 = d0$ at the local state `ca_1` is changed to $d1$ at the local state `ca_5` where the proposition `validClaim` is satisfied in order to make the global state that contains `ca_4` accessible from the global state that contains `ca_5`.

```
 Agent CallCenter
 __ Beginning of CallCenter agent
Vars:
d_1: {d0,d1};
......;
ca_1: {ca0, ca1, ca2, .....};
end Vars
Actions = {ca_gatherInfo,ca_sendValidClaim,.......,ca_none};
Protocol:
ca_1=ca2: {ca_gatherInfo};
ca_1=ca4: {ca_sendValidClaim};
ca_1=ca6: {ca_assignGarage};
......;
Other: {ca_none};
end Protocol
Evolution:
ca_1=ca2 and d_1=d0 if ca_1=ca1 and PolicyHolder.Action=ph_reportAccident;
......;
ca_1=ca5 and d_1=d1 if ca_1=ca4 and Action=ca_gatherInfo;
......;
end Evolution
end Agent
```

Furthermore, in our encoding of the AGFIL protocol, the set of atomic propositions is added in `Evaluation ... end Evaluation` section. The initial states are inserted in the `InitStates ... end InitStates` section. Finally, the defined specifications are also encoded and inserted into the `Formulae ... end Formulae` section.

We validate our modeling using the capability in the MCMAS-T model checker called `Explicit interactive mode` which is running the system interactively to check that it functions as intended. Thereafter, we used our transformation tool to transform the VISPL encoding model and trust formulae into the SMV model and CTL formulae in order to be able to start the verification process using the NuSMV model checker. Fig. 5 shows the transformation process of the model and formula using our toolkit. Our toolkit has the capability to scale MASs (`scalability` button) with respect to a certain modeling interleaved technique. In this technique, each agent is paired with another agent and all the resulting pairs move in a parallel way. Notice from the figure that the left panel displays the generated NuSMV modules of the MAS business scenario, which indeed is a CTL model. The `Launch NuSMV` button runs the NuSMV tool in order to display the verification results in the right panel of the figure. Furthermore, `Time/Formula` button pops up an information dialog box to show the transformation time of each formula (Fig. 7).

6.2. Properties

In the above scenario, the participating parties have to ensure that the trust relationships are correctly established on one another to perform their tasks accordingly. To verify the correctness of the AGFIL scenario at design time, we have to express a set

[1] The toolkit jar file is available at: https://users.encs.concordia.ca/~bentahar/TrustJavaToolkit.jar.

[2] https://users.encs.concordia.ca/~bentahar/mcmas-t-1.0.1-sc.tar.

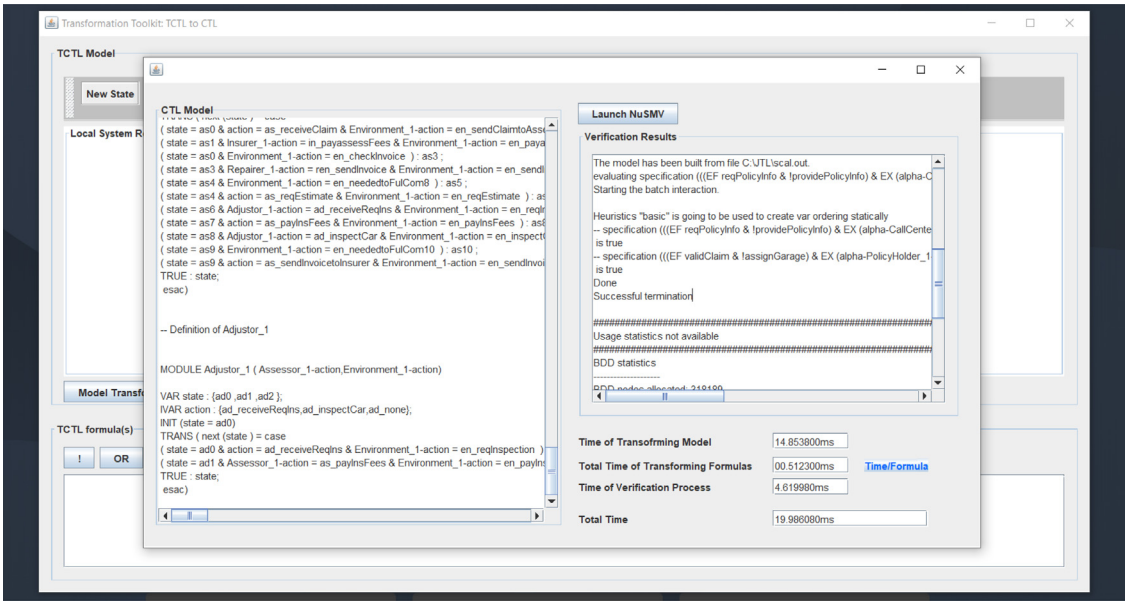


Fig. 5. Screenshot of the generated NuSMV modules and the verification results.

of properties. We used the safety (something bad will never happen) and liveness (something good will eventually occur) properties expressed using our logic. Such important properties have been widely investigated in different contexts, see for instance (Drawel et al., 2020c; El-Menshawy et al., 2018; Kafalı et al., 2017). Formally, the safety property φ_1 expresses the negation of the bad situation where the insurance company validates the policyholder claim, but the latter never establishes their trust towards the repairer with regard to the vehicle repair.

$$\varphi_1 = AG\neg(validClaim \wedge \neg T_p(PolicyHolder, Repairer, validClaim, carRepair)).$$

The liveness property φ_2 states that in all paths globally, it is always the case that if the policy holder reports an accident and their claim is valid, then eventually in all future computation paths, their trust towards the insurance company with regard to the claim payment will take place.

$$\varphi_2 = AG(ReportAccident \wedge ValidClaim \rightarrow AF(T_p(PolicyHolder, Ins, validClaim, insuranceClaimPayment))).$$

We also checked a liveness property, given by φ_3, expressed as a conditional trust. The formula expresses the existence of a computation path where the garage trusts the insurance company to pay for the repairs once the insurance company accepts the proposed estimate from the garage.

$$\varphi_3 = EG(T_c(Repairer, Ins, agreeEstimate, RepairPaymentCharge)).$$

Moreover, we checked in our experiments the existence of some trust relationships in which one agent i is considered trustworthy from the viewpoint of another agent j. These relationships are expressed as TCTL properties as follows:

- $\varphi_4 = EF(T_p(PolicyHolder, CallCenter, reportAccident, gatherInfo))$
- $\varphi_5 = EF(T_p(Repairer, Ins, repairCar, payRepairCharge))$

The formula φ_4 expresses the trust relationship between the policy holder and the call-center, so whenever the former reports an accident, the latter will eventually gather information, and the formula φ_5 states that whenever the garage repairs car, then there exist a path in its future the trust towards the insurance will take place with regards to the payment. Indeed, we have verified the above properties in different models having various numbers of agents ranging from 7 to 63. The results will be presented and discussed in the next section.

Moreover, one of the most powerful features of our tool is the production of counterexamples when one of the specifications are violated. For instance, the bad situation in the φ_6 property states that the insurance company agreed with the repairer estimate invoice, but there is no possibility for the repairer to establish their trust towards the insurance with regard to send the full payment.

$$\varphi_6 = AG\ \neg T_p(Ins, Repairer, AG(agreeEstimate), AF(payRepairCharge)).$$

However, when we start the verification process using our transformation tool, the tool reported a counterexample showing why the formula is false (see Fig. 6). We used this counterexample to correct our VISPL model and rerun the transformation tool to get the good SMV model.

6.3. Experimental results

The experiments are conducted on AMD FX-8350 - 8 Cores - 4 GHz per core with 32 GB memory. To test the scalability of our algorithms, we report nine experiments in Table 1 for both preconditional and conditional algorithms. We developed a code generation script that helps us automatically encode different number of agents. We consider the number of agents (Agents#), the number of reachable states (States#), the transformation times of both models and formulae in milliseconds, and the average total time calculated based on the transformation and verification times. The experiments revealed that some of the tested formulae are satisfied in our models and some are not. As shown in the table, the number of reachable states reflects the fact that the state space increases exponentially when the number of agent increases. Yet, it is clear that the transformation times of both the models and formulae increase only logarithmically with regard to the number of states. We can also notice that the average total time increases polynomially with respect

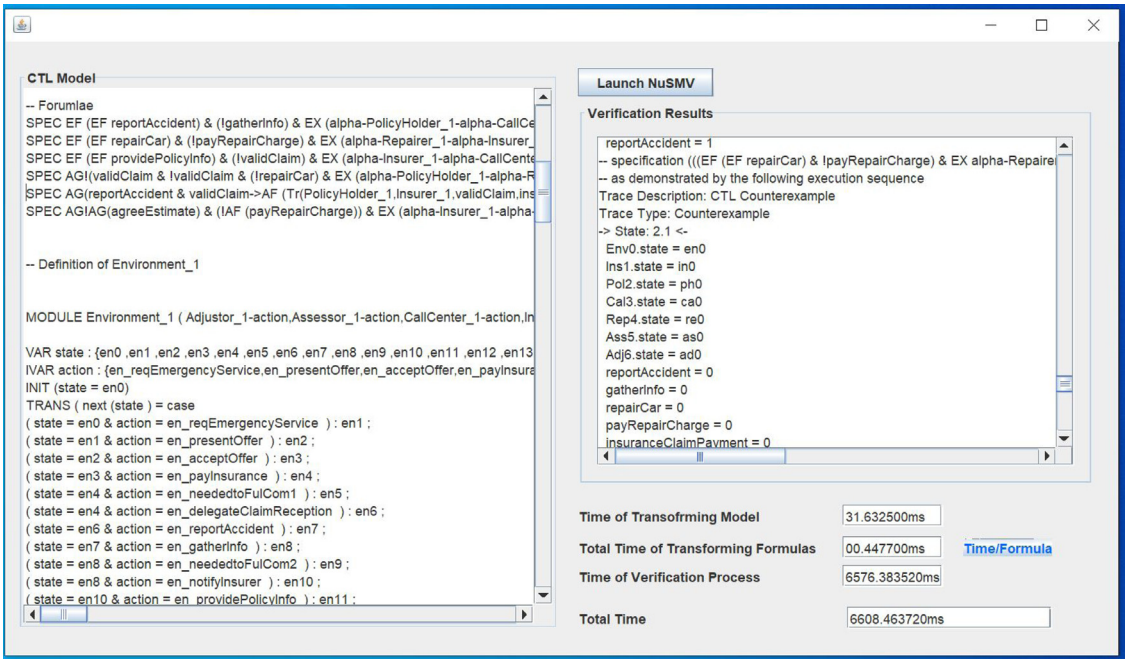


Fig. 6. A counterexample explaining why the φ_6 formula is false.

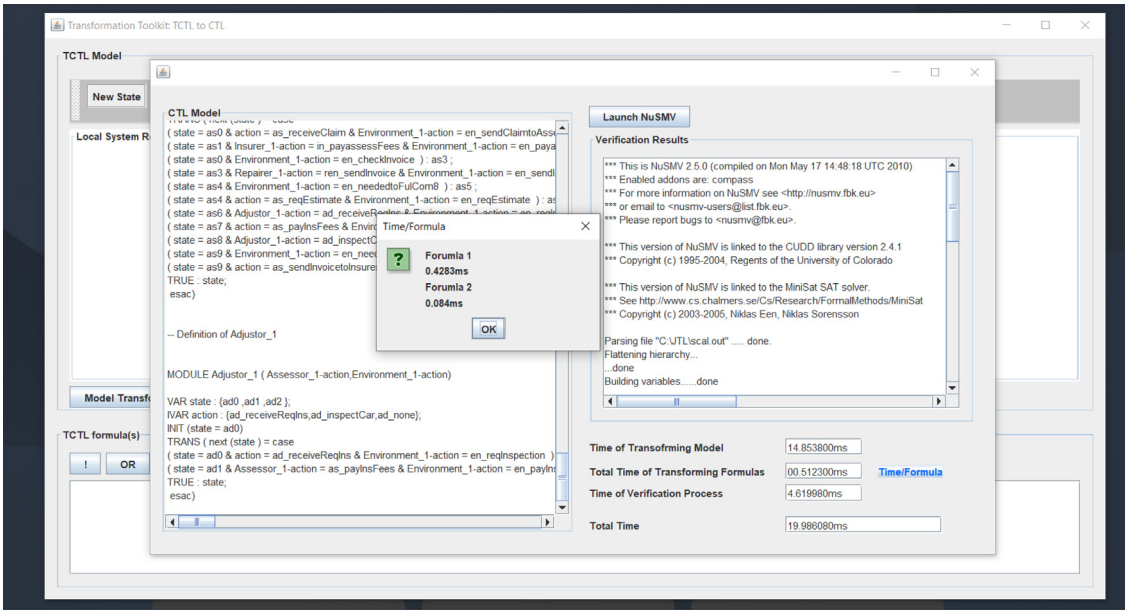


Fig. 7. Screenshot of the generated NuSMV modules along side verification results.

to the number of states. Finally, it is worth mentioning that the exponential blow-up of the state space with the number of agents is a classic state explosion problem in MASs and is independent of our model checking algorithms.

To compare our approach with the model checking approach introduced in Drawel et al. (2020c), we run the same experiments of the AGFIL scenario using the MCMAS-T model checker with a different number of agents against the same properties. The experiments are available at.[3] The experiments revealed that some of the tested formulae are true and some are false. Table 2 reports the results using the same machine. We can observe that the number of reachable states increases exponentially with the number of agents as we expected. It is clear that our transformation-based approach provides better results. An important problem encountered when running the models with MCMAS-T was that the experiments go down to almost a halt when we increase the number of agents. That is, while MCMAS-T allowed us to verify models up to only 42 agents, this approach is able to check the same scenario with up to 63 agents. In fact, the performance is higher as we go further and reach more agents. Moreover, the execution time in our approach is also better than the direct approach. We can observe that in our approach, this metric increases only logarithmically with respect to the number of states, and remains below 793 s even for 63 agents. As compared with our previous verification technique (Drawel et al., 2020c), it is wroth mentioning that the verification results for

[3] https://users.encs.concordia.ca/~bentahar/Case-Study-Files.zip.

Table 1
Verification results of the AGFIL protocol using our toolkit.

Exp.#	Agents#	States#	Time of model transformation (ms)	Time of formulae transformation (ms)	Total time (ms)
1	7	42	15	0.8	20
2	14	468	17	0.9	119
3	21	5586	22	1	1330
4	28	67 236	36	1.1	8049
5	35	809 682	38	1.2	45 051
6	42	9.74111e+06	42	1.3	210 000
7	49	1.29742e+08	48	1.4	390 000
8	56	4.49064e+12	53	1.5	540 000
9	63	2.52442e+15	57	1.7	792 000

Table 2
Verification results of the AGFIL protocol using MCMAS-T.

Exp.#	Agents#	States#	Time (ms)
1	7	42	50
2	14	468	1020
3	21	5586	16 340
4	28	67 236	99 723
5	35	809 682	694 035
6	42	9.74111e+06	3 333 680

model checking using the transformation tool are more efficient, making it a promising methodology in practice.

As a matter of fact, we also compared our methodology using the same case study introduced in Drawel et al. (2020c) with the presented technique in terms of the number of agents and the execution time. The verification results are reported in Table 3. We can readily observe from Table 3 that the MCMAS-T tool requires more execution time compared to our new technique. Moreover, we found that our approach is more scalable as it supports more agents, 60 agents having about 2.01599e+12 state space, compared to only 36 agents having about 2.41376e+07 state space in MCMAS-T. In fact, the performance of our approach is more efficient as we can go further and reach more than 60 agents. Although the verification time in both results are close to each other, with MCMAS-T, we have the ability to only support a small number of reachable states with a high memory usage. Therefore, these results confirmed that the MCMAS-T approach is not efficient in practice and not scalable as the number of reachable states is very limited.

7. Related work

While the number of proposals on trust modeling is significant, they differ, however, in the topics they addressed and the systems they implemented. In this paper, we are primarily concerned with the issues of reasoning about and verifying trust in the context of MASs using the model checking approach, which has not been deeply investigated yet for trust systems. Various approaches on trust modeling have developed modal logics that capture the important elements of the cognitive theory of trust as proposed in Castelfranchi and Falcone (1998). In these approaches, trust is considered as a mental attitude based on a specific set of goals and expressed in terms of certain beliefs. In this respect, Herzig et al. (2010) proposed a formal logical framework to formally reason about occurrent and dispositional trust in a multi-agent environment. The authors defined a logic that combines temporal, dynamic, belief and choice logics. Moreover, the proposed logic is extended with operators to allow reasoning about reputation in the scope of collective beliefs. In another work (Herzig et al., 2012), Herzig and his colleagues simplified the previous logic presented in Herzig et al. (2010) by considering a very simple kind of actions based on the concepts of propositional assignment. That is, the truth values of a propositional variable is assigned to either true or false by the corresponding agents' actions. The new logic provides a simple framework, and it is expressive enough to account for the cognitive theory of trust. In Liu and Lorini (2017), Liu and Lorini presented a new dynamic logic called DL-BET for reasoning about the interaction between belief, evidence and trust. The authors introduced three modal operators where each of these concepts are respectively represented. In this logic, the trust operator semantics is interpreted using the neighborhood semantics (Montague, 1970), which maps each world into a set of subsets of worlds. The authors provided a complete axiomatization for both the static component of the proposed logic and its dynamic extension. Other approaches have focused on trust in information sources. In this line, an early work presented BIT, a modal logic that extends the traditional doxastic logic with modalities for representing belief, information acquisition, and trust (Liau, 2003). In the BIT formalism, the trust operator is interpreted using the neighborhood semantics (Montague, 1970). The logic is provided with a rigorous semantics to precisely characterize (1) the relationships among beliefs and information acquisition, and (2) how different trust properties are represented by considering various axioms of the logic. Indeed, this well-known, early formal treatment of trust, has been followed by many researchers. For instance, Pearce and Uridia (2015) has simplified the semantical treatment of the trust operator given in Liau (2003) by retaining only two types of modalities, a belief operator and trust, and they do not make use of the additional information operator. The idea of this work is to concentrate only on the interrelation between trust and belief. Their alternative logic of trust can also explicate a type of trust that is linked to honesty or sincerity. It allows to consider different forms of honesty separately and to incorporate these already into the base logic. Trust in the sincerity is also presented recently in Leturc and Bonnet (2018). The authors proposed a modal logic to reason about an agent's trust in the sincerity towards a statement formulated by another agent. They introduced a normal modality of trust that allows to reason about trust when some agent attempt to manipulate other agents. Moreover, the authors exhibited some notable properties and showed that the proposed logic is sound and complete. In a related work, Lorini and Demolombe (2008a,b) have focused on analyzing trust in various features of information sources. That is, a certain agent is trusting another agent if the former believes that the information transmitted to them is reliable. In particular, in Lorini and Demolombe (2008b), the authors formalized some security properties and their relationships with trust such as integrity, availability and privacy by proposing a modal operator of obligation. Fuchs and colleagues (Fuchs et al., 2010) also addressed trust in the context of computer security. More recent proposals have made the link between trust and argument-based frameworks (Amgoud and Demolombe, 2014; Parsons et al., 2014; Tang et al., 2012). However, the above mentioned models are not suitable for verification mechanisms using model checking techniques because of their reliance on the

Table 3
Comparison of the verification results of the case study presented in Drawel et al. (2020c).

(a) Using our transformation toolkit				(b) Using the MCMAS-T model checker			
Exp.#	Agents#	States#	Total time (ms)	Exp.#	Agents#	States#	Time (ms)
1	6	17	25	1	6	17	58
2	12	289	130	2	12	289	486
3	18	4913	1077	3	18	4913	1347
4	24	83 521	3132	4	24	83 521	5694
5	30	1.41986E+06	10 138	5	30	1.41986E+06	17 074
6	36	2.41376E+07	14 700	6	36	2.41376E+07	21 225
7	42	4.10339e+08	25 800				
8	48	6.97576e+09	51 000				
9	54	1.18588e+11	72 000				
10	60	2.01599e+12	895 000				

internal structure of the interacting agents. Practically, the fact that MASs are deployed in open environments means that these agents are managed by different providers using different types of platforms. Thus, it is very difficult for one agent to completely trust others or to make assumptions about their internal states. Moreover, model checking neighborhood semantics-based modal logics is yet to be solved (El-Menshawy et al., 2015; Pacuit, 2017).

The closest approach to our work is the one presented by Singh in Singh (2011) where the social perspective of trust has been put forward. Specifically, the author provided a formal semantics for trust with various logical constraints used to reason about trust from an architectural point of view. This logical model combines temporal modalities of linear temporal logic (LTL) (Pnueli, 1977), modality C for commitments (Singh, 2008) and modality T for trust. Yet, unlike our approach that is based on interpreted systems structure with a grounded semantics, Singh's logic is interpreted using the neighborhood semantics (Montague, 1970) whose model checking is still an open problem (El-Menshawy et al., 2015). In fact, the neighborhood semantics may not be as straightforward as Kripke semantics because it is more general. However, by assuming some of the postulates, the semantics becomes equivalent to Kripke models.

From the model checking point of view, some authors adopt a direct method, which can be performed by either developing a proper model checker from scratch or by extending existing tools with new algorithms for the needed temporal modalities (Bentahar et al., 2012; Drawel et al., 2020b; Bentahar et al., 2022; Drawel et al., 2022). For instance, in our previous work (Drawel et al., 2020c), we proposed a new logic-based framework for specifying and model checking preconditional trust in MASs. We introduced TCTL, a branching temporal logic of preconditional trust interpreted in a new vector-extended version of interpreted systems that captures the trust relationship between the interacting parties. Reasoning postulates and new symbolic model checking algorithms are presented to formally specify and automatically verify the system under consideration against some desirable properties expressed in TCTL. Thus, a new model checker extending MCMAS, called MCMAS-T, dedicated to TCTL, along with its new input language VISPL (Vector-extended ISPL) have been created. However, TCTL is restricted to preconditional trust and does not support conditional trust, and the symbolic model checking algorithms, although efficient with flat models, showed limited efficiency when the verified models are not flat (i.e., include non-self loops). In another work (Drawel et al., 2018), we examined a different verification mechanism that does not suffer from the non-flat-models limitation. This has been achieved, thanks to the high efficiency of CTL model checking to which the model checking of TCTL and conditional trust is transformed. However, the proposed approach did not provide accurate alignments between source and target models which in turn affect the semantics of the standard CTL operators and at the same time the validity of the original model properties. Moreover, in this paper we present the complexity analysis of the TCTL logic, which have not been considered in our previous work.

Another relevant work is presented by Aldini in Aldini (2016), where a formal framework to evaluate the effectiveness and robustness of trust-based models in order to detect and then isolate different kinds of attacks has been introduced. Indeed, the author integrates trust modeling with distributed systems. In this work, the system properties are expressed using a trust temporal logic (TTL) which combines CTL (Emerson, 1990) and its action-based extension (ACTL) (De Nicola and Vaandrager, 1990). Moreover, the trust system model is based on an instance of both labeled transition systems and Kripke structures. The verification of temporal logic properties expressed in TTL has been performed through a mapping to an existing model checking technique. However, the model mapping between the two logics has not been specified and TTL can only specify a single agent model, and it is not adapted to autonomous MASs. El-Qurna et al. presented a model checking framework to verify service trust behaviors model against regular and non-regular properties (El-Qurna et al., 2017). The authors introduced an algorithm to generate a configuration graph of a deterministic pushdown automata (PDA), where the trust behaviors are captured through the observations' sequences related to certain interactions between the services and users. The trust behavior properties are specified using Fixed point Logic with Chop (FLC). From the model checking prospective, a symbolic *FLC* model checking algorithm is applied in order to verify service trust behaviors with respect to trust properties. However, this approach lacks formal semantics for trust because trust formulae are inferred from the context free grammar of trust pattern languages. Moreover, the approach is not designed to formalize and verify trust in the context of MASs.

Following the path of cognitive trust, Huang and Kwiatkowska (2017) considered the setting of stochastic multi-agent systems, where an automated verification framework for quantifying and reasoning about cognitive trust is proposed. The authors focused on a quantitative notion of trust defined as a subjective evaluation in order to capture the social notation of trust. This allows one to quantify the amount of trust as a belief-weighted expectation. In this work, a probabilistic rational temporal logic PRTL*, which extends the logic PCTL* with reasoning about agents' mental states is introduced. However, the model checking of the proposed logic was proven undecidable. Yet, no implementation has been considered and no attempt to evaluate the approach on any case.

On the other hand, model checking trust can also be achieved by indirect techniques, also called transformation-based methods. The idea is to apply certain reduction rules in order to transform the problem at hand to an existing model checking problem. In fact, transformation has been acknowledged as an alternative mechanism for verifying various MASs aspects. The main advantage of this technique is that it enables the designers of MASs

applications to get benefit from powerful and already tested model checkers. This technique has been applied for model checking commitments (El-Menshawy et al., 2010), knowledge (Lomuscio et al., 2007), and the interaction between knowledge and commitments (Al-Saqqar et al., 2015; Sultan et al., 2014). To the best of our knowledge, our work is the first attempt that introduces and implements a full transformation technique for verifying trust specifications in MASs.

8. Conclusion

In this article, we proposed a new model checking framework for the TCTL logic of preconditional trust that is extended to design a new algorithm to model check conditional trust in MASs. We designed transformation-based algorithms and implemented them in a Java toolkit that automatically interacts with the NuSMV model checker of the CTL logic. Our proposed technique is able to automatically transform the problem of model checking TCTL into the problem of model checking CTL. We also discussed the logical relationship between preconditional and conditional trust, which led to the model checking procedure of conditional trust. The proof of the soundness and completeness of our transformation algorithms is provided. Moreover, we proved that (1) the time complexity of TCTL and $TCTL^C$ model checking in explicit models is P-complete with regard to the size of the model and length of the formula; and (2) the complexity of the same problems for concurrent programs is PSPACE-complete with respect to the size of the program's components. Therefore, our model checking algorithms have the same complexity as model checking CTL with regard to both explicit models and concurrent programs. Experiments conducted on a standard industrial case study demonstrated the efficiency and scalability of the technique. When we compared our approach with the one proposed in Drawel et al. (2020c), we reported that the developed verifier tool was able to verify a variety of formulae correctly and efficiently within a large case study having approximately 2.52442e+15 reachable states. Our approach leverages the high efficiency of CTL model checking to which the model checking of TCTL and conditional trust is transformed. As future work, we plan to analyze the interaction between commitment alignment (Chopra and Singh, 2015) and trust from both, specification and model checking standpoints. Studying trust dynamics and uncertainty in real time MASs is another direction for further investigation.

CRediT authorship contribution statement

Nagat Drawel: Conceptualization, Methodology, Formal analysis, Investigation, Writing – original draft. **Amine Laarej:** Software, Methodology, Investigation. **Jamal Bentahar:** Conceptualization, Methodology, Formal analysis, Validation, Writing – review & editing, Funding acquisition. **Mohamed El Menshawy:** Formal analysis, Validation, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We would like to thank the Natural Sciences and Engineering Research Council of Canada (NSERC), Canada and the Department of National Defence (DnD), Canada, Innovation for Defence Excellence and Security program, Canada for their financial support.

References

Al-Saqqar, F., Bentahar, J., Sultan, K., Wan, W., Khosrowshahi Asl, E., 2015. Model checking temporal knowledge and commitments in multi-agent systems using reduction. Simul. Model. Pract. Theory 51, 45–68.

Aldini, A., 2016. A formal framework for modeling trust and reputation in collective adaptive systems. In: Proceedings of the Workshop on FORmal Methods for the Quantitative Evaluation of Collective Adaptive SysTems, FORECAST@STAF, Vienna, Austria, pp. 19–30.

Amgoud, L., Demolombe, R., 2014. An argumentation-based approach for reasoning about trust in information sources. Argument Comput. 5 (2–3), 191–215.

Bentahar, J., Drawel, N., Sadiki, A., 2022. Quantitative group trust: A two-stage verification approach. In: Faliszewski, P., Mascardi, V., Pelachaud, C., Taylor, M.E. (Eds.), 21st International Conference on Autonomous Agents and Multiagent Systems, AAMAS, Auckland, New Zealand, May 9-13. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), pp. 100–108.

Bentahar, J., El-Menshawy, M., Qu, H., Dssouli, R., 2012. Communicative commitments: Model checking and complexity analysis. Knowl.-Based Syst. 35, 21–34.

Bentahar, J., Meyer, J.C., Wan, W., 2009. Model checking communicative agent-based systems. Knowl. Based Syst. 22 (3), 142–159.

Bernardi, S., Gentile, U., Marrone, S., Merseguer, J., Nardone, R., 2021. Security modelling and formal verification of survivability properties: Application to cyber-physical systems. J. Syst. Softw. 171, 110746.

Castelfranchi, C., Falcone, R., 1998. Principles of trust for MAS: Cognitive anatomy, social importance, and quantification. In: Proceedings of the Third International Conference on Multiagent Systems, ICMAS, pp. 72–79.

Chopra, A.K., Singh, M.P., 2015. Generalized Commitment Alignment. In: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, Istanbul, Turkey, May 4-8, 2015, pp. 453–461.

Clarke, E.M., Emerson, E.A., Sifakis, J., 2009. Model checking: algorithmic verification and debugging. Commun. ACM 52 (11), 74–84.

Clarke, E.M., Grumberg, O., Peled, D., 1999. Model Checking. MIT Press.

Cohen, P.R., Levesque, H.J., 1990. Intention is choice with commitment. Artificial Intelligence 42 (2–3), 213–261.

De Nicola, R., Vaandrager, F., 1990. Action versus state based logics for transition systems. In: Semantics of Systems of Concurrent Processes. Springer, pp. 407–419.

Demolombe, R., 2001. To trust information sources: A proposal for a modal logical framework. In: Trust and Deception in Virtual Societies. Springer, pp. 111–124.

Desai, N., Chopra, A.K., Singh, M.P., 2009. Amoeba: A methodology for modeling and evolving cross-organizational business processes. ACM Trans. Softw. Eng. Methodol. (TOSEM) 19 (2), 6.

Drawel, N., Bentahar, J., El-Menshawy, M., Laarej, A., 2018. Verifying temporal trust logic using CTL model checking. In: Cohen, R., Sensoy, M., Norman, T.J. (Eds.), Proceedings of the 20th International Trust Workshop Co-Located with AAMAS/IJCAI/ECAI/ICML 2018, Stockholm, Sweden, July 14, 2018. In: CEUR Workshop Proceedings, vol. 2154, CEUR-WS.org, pp. 62–74.

Drawel, N., Bentahar, J., Laarej, A., Rjoub, G., 2020. Formalizing group and propagated trust in multi-agent systems. In: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI, pp. 60–66.

Drawel, N., Bentahar, J., Laarej, A., Rjoub, G., 2022. Formal verification of group and propagated trust in multi-agent systems. Auton. Agents Multi Agent Syst. 36 (1), 19.

Drawel, N., Bentahar, J., Qu, H., 2020b. Computationally grounded quantitative trust with time. In: Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems, pp. 1837–1839.

Drawel, N., Bentahar, J., Shakshuki, E.M., 2017. Reasoning about trust and time in a system of agents. In: The International Conference on Ambient Systems, Networks and Technologies (ANT). In: Procedia Computer Science, vol. 109, pp. 632–639.

Drawel, N., Qu, H., Bentahar, J., Shakshuki, E., 2020c. Specification and automatic verification of trust-based multi-agent systems. Future Gener. Comput. Syst. 107, 1047–1060.

El Kholy, W., Bentahar, J., El-Menshawy, M., Qu, H., Dssouli, R., 2014. Conditional commitments: Reasoning and model checking. ACM Trans. Softw. Eng. Methodol. (TOSEM) 24 (2), 9.

El-Menshawy, M., Bentahar, J., Dssouli, R., 2010. Symbolic model checking commitment protocols using reduction. In: International Workshop on Declarative Agent Languages and Technologies. Springer, pp. 185–203.

El-Menshawy, M., Bentahar, J., El Kholy, W., Yolum, P., Dssouli, R., 2015. Computational logics and verification techniques of multi-agent commitments: survey. Knowl. Eng. Rev. 30 (5), 564–606.

El-Menshawy, M., Bentahar, J., Kholy, W.E., Dssouli, R., 2013. Reducing model checking commitments for agent communication to model checking ARCTL and gctl*. Auton. Agents Multi Agent Syst. 27 (3), 375–418.

El-Menshawy, M., Bentahar, J., Kholy, W.E., Laarej, A., 2018. Model checking real-time conditional commitment logic using transformation. J. Syst. Softw. 138, 189–205.
El-Qurna, J., Yahyaoui, H., Almulla, M., 2017. A new framework for the verification of service trust behaviors. Knowl.-Based Syst..
Emerson, E.A., 1990. Temporal and modal logic. In: Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics. MIT Press, pp. 995–1072.
Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y., 1995. Reasoning about Knowledge. MIT Press.
Filieri, A., 2011. Qos verification and model tuning @ runtime. In: Gyimóthy, T., Zeller, A. (Eds.), SIGSOFT/FSE'11 19th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE-19) and ESEC'11: 13th European Software Engineering Conference (ESEC-13), Szeged, Hungary, September 5-9. ACM, pp. 408–411.
Filieri, A., Ghezzi, C., Tamburrelli, G., 2011. Run-time efficient probabilistic model checking. In: Taylor, R.N., Gall, H.C., Medvidovic, N. (Eds.), Proceedings of the 33rd International Conference on Software Engineering, ICSE, Waikiki, Honolulu , HI, USA, May 21-28. ACM, pp. 341–350.
Fuchs, A., Gürgens, S., Rudolph, C., 2010. A formal notion of trust–enabling reasoning about security properties. In: IFIP International Conference on Trust Management. Springer, pp. 200–215.
Guo, X., Aoki, T., Lin, H., 2020. Model checking of in-vehicle networking systems with CAN and FlexRay. J. Syst. Softw. 161, 110461.
Harel, D., Kozen, D., Tiuryn, J., 2000. Dynamic Logic. MIT Press.
Herzig, A., Lorini, E., Hübner, J.F., Vercouter, L., 2010. A logic of trust and reputation. Logic J. IGPL 18 (1), 214–244.
Herzig, A., Lorini, E., Moisan, F., 2012. A simple logic of trust based on propositional assignments. In: Paglieri, F., Tummolini, L., Falcone, R. (Eds.), The Goals of Cognition. Essays in Honour of Cristiano Castelfranchi. In: Tributes, College Publications, pp. 407–419.
Huang, X., Kwiatkowska, M.Z., 2017. Reasoning about cognitive trust in stochastic multiagent systems. In: Singh, S.P., Markovitch, S. (Eds.), Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, California, USA. AAAI Press, pp. 3768–3774.
Jones, N.D., 1975. Space-bounded reducibility among combinatorial problems. Comput. Syst. Sci. 11 (1), 68–85.
Kafalı, O., Ajmeri, N., Singh, M.P., 2017. Kont: Computing tradeoffs in normative multiagent systems. In: Proceedings of the 31st Conference on Artificial Intelligence (AAAI), pp. 3006–3012.
Kafalı, O., Yolum, P., 2009. Detecting exceptions in commitment protocols: discovering hidden states. In: International Workshop on Languages, Methodologies and Development Tools for Multi-Agent Systems. In: LNCS, vol. 6039, pp. 112–127.
Kalia, A.K., Singh, M.P., 2015. Muon: designing multiagent communication protocols from interaction scenarios. Auton. Agents Multi-Agent Syst. 29 (4), 621–657.
Kholy, W.E., Bentahar, J., El-Menshawy, M., Qu, H., Dssouli, R., 2014. Modeling and verifying choreographed multi-agent-based web service compositions regulated by commitment protocols. Expert Syst. Appl. 41 (16), 7478–7494.
Kupferman, O., Vardi, M.Y., Wolper, P., 2000. An automata-theoretic approach to branching-time model checking. J. ACM 47 (2), 312–360.
Leturc, C., Bonnet, G., 2018. A normal modal logic for trust in the sincerity. In: André, E., Koenig, S., Dastani, M., Sukthankar, G. (Eds.), Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS, Stockholm, Sweden. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM, pp. 175–183.
Liau, C., 2003. Belief, information acquisition, and trust in multi-agent systems–a modal logic formulation. Artificial Intelligence 149 (1), 31–60.
Liu, F., Lorini, E., 2017. Reasoning about belief, evidence and trust in a multi-agent setting. In: International Conference on Principles and Practice of Multi-Agent Systems. Springer, pp. 71–89.
Lomuscio, A., Pecheur, C., Raimondi, F., 2007. Automatic Verification of Knowledge and Time with NuSMV. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence, pp. 1384–1389.
Lomuscio, A., Pirovano, E., 2020. Parameterised verification of strategic properties in probabilistic multi-agent systems. In: Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems, pp. 762–770.
Lomuscio, A., Qu, H., Raimondi, F., 2017. MCMAS: an open-source model checker for the verification of multi-agent systems. STTT 19 (1), 9–30.
Lorini, E., Demolombe, R., 2008a. From trust in information sources to trust in communication systems: An analysis in modal logic. In: Meyer, J.C., Broersen, J.M. (Eds.), Knowledge Representation for Agents and Multi-Agent Systems, First International Workshop, KRAMAS 2008, Sydney, Australia, Revised Selected Papers. In: Lecture Notes in Computer Science, vol. 5605, Springer, pp. 81–98.
Lorini, E., Demolombe, R., 2008b. Trust and norms in the context of computer security: A logical formalization. In: International Conference on Deontic Logic in Computer Science. Springer, pp. 50–64.
van der Meyden, R., Su, K., 2004. Symbolic model checking the knowledge of the dining cryptographers. In: 17th IEEE Computer Security Foundations Workshop. pp. 280–291.
Montague, R., 1970. Universal grammar. Theoria 36 (3), 373–398.
Pacuit, E., 2017. Neighborhood Semantics for Modal Logic. Springer.
Parsons, S., Atkinson, K., Li, Z., McBurney, P., Sklar, E., Singh, M., Haigh, K., Levitt, K., Rowe, J., 2014. Argument schemes for reasoning about trust. Argum. Comput. 5 (2–3), 160–190.
Pearce, D., Uridia, L., 2015. Trust, belief and honesty. In: GCAI 2015. Global Conference on Artificial Intelligence. EasyChair, pp. 215–228.
Pnueli, A., 1977. The temporal logic of programs. In: 18th Annual Symposium on Foundations of Computer Science. pp. 46–57.
Savitch, W.J., 1970. Relationships between nondeterministic and deterministic tape complexities. Comput. Syst. Sci. 4 (2), 177–192.
Schnoebelen, P., 2002. The complexity of temporal logic model checking. In: Balbiani, P., Suzuki, N., Wolter, F., Zakharyaschev, M. (Eds.), Advances in Modal Logic 4, Papers from the Fourth Conference on "Advances in Modal Logic," Held in Toulouse, France, 30 September - 2 October 2002. King's College Publications, pp. 393–436.
Singh, M.P., 2008. Semantical considerations on dialectical and practical commitments. In: Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence. AAAI, Chicago, Illinois, USA, pp. 176–181.
Singh, M.P., 2011. Trust as dependence: a logical approach. In: The 10th International Conference on Autonomous Agents and Multiagent Systems. pp. 863–870.
Sultan, K., Bentahar, J., Wan, W., Al-Saqqar, F., 2014. Modeling and verifying probabilistic multi-agent systems using knowledge and social commitments. Expert Syst. Appl. 41 (14), 6291–6304.
Tang, Y., Cai, K., McBurney, P., Sklar, E., Parsons, S., 2012. Using argumentation to reason about trust and belief. J. Logic Comput. 22 (5), 979–1018.
Telang, P.R., Singh, M.P., 2009. Enhancing tropos with commitments. In: Conceptual Modeling: Foundations and Applications. Springer, pp. 417–435.

Nagat Drawel received her bachelor's degree in Computer Science from Tripoli University, Libya, the M.Sc. degree in the same discipline from University of Newcastle Upon Tyne, UK, and the Ph.D. degree in Information and Systems Engineering from Concordia University, Canada. She was also a postdoctoral fellow researcher at Concordia Institute for Information Systems Engineering, Canada. Before joining Concordia University in 2014, Dr. Nagat had been a Lecturer in the Faculty of Computer Technology, Tripoli-Libya for a decade. Her research interests include the areas of computational logics, formal verification using model checking techniques, intelligent agents, trust in Multi-Agent Systems, and software engineering.

Amine Laarej received his bachelor's degree in Software Engineering and Information Systems Modeling from the National Institute of Post & Telecommunication, Morocco, in 2015 and M.Sc degree in Quality Systems Engineering at Concordia University, in 2021. His research interests include multi-agent systems, logics and formal methods, model checking, model-based testing and software engineering.

Jamal Bentahar is a Professor with Concordia Institute for Information Systems Engineering, Gina Cody School of Engineering and Computer Science, Concordia University, Canada. He received the bachelor's degree in Software Engineering from the National Institute of Statistics and Applied Economics, Morocco, in 1998, the M.Sc. degree in the same discipline from Mohamed V University, Morocco, in 2001, and the Ph.D. degree in Computer Science and Software Engineering from Laval University, Canada, in 2005. From 2005 to 2006, he was a Postdoctoral Fellow with Laval University, and then NSERC Postdoctoral Fellow at Simon Fraser University, Canada. He was an NSERC Co-Chair for Discovery Grant for Computer Science (2016–2018). His research interests include cloud and services computing, trust and reputation, deep reinforcement learning, game theory, computational logics, model checking, and multi-agent systems.

Mohamed El Menshawy is an adjunct professor at Concordia Institute for Information Systems Engineering, Concordia University, Canada, and an assistant professor in the Department of Computer Science, Menoufia University, Egypt. El Menshawy has a Ph.D. in Electrical and Computer Engineering from Concordia University obtained in 2012. His research interests are software engineering, social commitments, logic and formal methods, model checking, m-health services, mobile applications, and machine learning.

Update

The Journal of Systems & Software

Volume 213, Issue , July 2024, Page

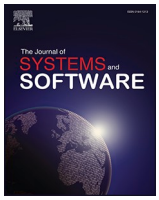
DOI: https://doi.org/10.1016/j.jss.2024.112052

The Journal of Systems and Software 213 (2024) 112052

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss



Corrigendum

Corrigendum to "Transformation-based model checking temporal trust in multi-agent systems" [Journal of Systems and Software Volume 192, October 2022, 111383]

Nagat Drawel [a], Amine Laarej [a], Jamal Bentahar [a,*], Mohamed El Menshawy [b]

[a] *Concordia Institute for Information Systems Engineering, Montreal, Canada*
[b] *Computer Science, Menoufia University, Menoufia, Egypt*

The authors regret having missed including the correct affiliation for Mohamed El Menshawy in the original publication. The correct affiliation and details for Mohamed El Menshawy are as mentioned below:

School for Advanced Digital Technology (SADT)
Southern Alberta Institute of Technology (SAIT)
Calgary, Alberta, Canada
mohamede.mohamed@sait.ca

The authors would like to apologise for any inconvenience caused.

DOI of original article: https://doi.org/10.1016/j.jss.2022.111383.
* Corresponding author.
E-mail address: bentahar@ciise.concordia.ca (J. Bentahar).

https://doi.org/10.1016/j.jss.2024.112052

Available online 12 April 2024