Future Generation Computer Systems ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Specification and automatic verification of trust-based multi-agent systems

Nagat Drawel[a], Hongyang Qu[b], Jamal Bentahar[a,*], Elhadi Shakshuki[c]

[a] Concordia Institute for Information Systems Engineering, Montreal, Canada
[b] Automatic Control and Systems Engineering, Shefield University, United Kingdom
[c] School of Computer Science, Acadia University, Wolfville, Canada

HIGHLIGHTS

- We propose TCTL, a new logic of trust in multi-agent systems that extends CTL.
- We present the reasoning postulates of this logic along with proofs.
- We develop two model checking algorithms for the logic.
- We introduce MCMAS-T, a new model checker that implements the new algorithms.
- We report experimental results using a real-life scenario in the healthcare domain.

ARTICLE INFO

Article history:
Received 21 September 2017
Received in revised form 16 January 2018
Accepted 20 January 2018
Available online xxxx

Keywords:
Trust
Multi-agent systems
Temporal logic
Model checking
Reasoning postulates
MCMAS-T
VISPL

ABSTRACT

We present a new logic-based framework for modeling and automatically verifying trust in Multi-Agent Systems (MASs). We start by refining TCTL, a temporal logic of trust that extends the Computation Tree Logic (CTL) to enable reasoning about trust with preconditions. A new vector-based version of interpreted systems is defined to capture the trust relationship between the interacting parties. We introduce a set of reasoning postulates along with formal proofs to support our logic. Moreover, we present new symbolic model checking algorithms to formally and automatically verify the system under consideration against some desirable properties expressed using the proposed logic. We fully implemented our proposed algorithms as a model checker tool called MCMAS-T on top of the MCMAS model checker for MASs along with its new input language VISPL (Vector-extended ISPL). We evaluated the tool and reported experimental results using a real-life scenario in the healthcare field.

1. Introduction

1.1. Problem statement

A Multi-Agent System (MAS) is a set of autonomous entities, which interact in dynamic and uncertain environments in order to achieve their goals [1]. Such systems have been rapidly adopted in a large number of critical applications such as commercial, industrial, governmental and healthcare systems [2–5]. Although this adoption has the advantages of solving complex problems that an individual agent cannot handle alone, it raises, however, a number of challenges related to their present and future behaviors. Nevertheless, in open MASs, entities can join and leave the interactions at any time [6]. This means MASs will actually provide no guarantee about the behavior of their agents, which makes the concept of trust of particular importance for regulating the relationships and interactions among these agents.

Most of the existing approaches considered the mental agent concept that depends on the capability of interacting agents. Since these agents are autonomous and heterogeneous, such a mental concept cannot make those agents abide by the language semantics whenever they interact [7]. Thus, the need for a logical language that can provide a certain level of abstraction with the ability to express the trust properties is of great significance. Although the concept of trust in our work originates mainly from the work proposed by Singh [8] where a neighborhood semantics is being used, our proposal, however, formalizes trust by extending the standard Computation Tree Logic (CTL) [9]. Trust in our work is considered as a modal operator with grounded and intuitive semantics. We

* Corresponding author.
E-mail addresses: n_drawe@encs.concordia.ca (N. Drawel), h.qu@shefield.ac.uk (H. Qu), bentahar@ciise.concordia.ca (J. Bentahar), elhadi.shakshuki@acadiau.ca (E. Shakshuki).

https://doi.org/10.1016/j.future.2018.01.040
0167-739X/© 2018 Elsevier B.V. All rights reserved.

introduce a new accessibility relation which is needed to define the semantics of the trust modal operator. This accessibility relation is defined so that it captures the intuition of trust while being easily computable. Unlike our previous work [10] where the notion of trust with no preconditions has been considered, the concept of trust in this paper is defined 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 when the truster chooses to reach her goal through bringing about a content by the trustee if there is a possibility of satisfying this content. For example, the customer trusts that the merchant will deliver the requested items upon the customer's payment, where such a payment consists of the precondition in the trust establishment process.

From the verification perspective, to date, verification of trust in MASs is still a challenging issue, and has not been sufficiently investigated yet. The current proposals in the literature have been focusing mainly on evaluating the trust-based systems. Such approaches often evaluate the effectiveness and robustness of these models against undesired attackers' behaviors. In this respect, two validation methods can be distinguished: simulation-based and formal-based verification. A large number of studies adopt simulation methods that use different assessment techniques with the aim of mitigating malicious entities [11–13]. Alternatively, the approaches that use formal-based methods are increasing recently [14–19]. The verification in the latter approaches is done by searching all possible state space of the models under consideration in order to discover certain types of attacks. Although both methods have the advantage of detecting and isolating different kinds of attacks, they lack the generality (i.e., the proposed assessment tools can be only applied to a single particular model). Moreover, these approaches cannot capture the whole behaviors of target systems, so only some desirable properties can be verified. Besides, only known and predefined attacks have been considered in the evaluation process. Nonetheless, interacting agents are heterogeneous, which means one cannot guarantee that they will behave as they are supposed to. Thus, the need for efficient methodologies to automatically check whether or not MASs behavior conforms with the system specifications is recognized. Recently, formal evaluation using model checking techniques has been proved to be the most extensively used tool [20,21]. So far, there is almost no approach on verifying statically MASs with respect to certain properties related to agents trust. Dealing with such an issue as a model checking problem is one of our goals in this paper.

1.2. Paper contributions

The present paper is a substantial extension of our previous work published in [10], in which we have made a novel attempt towards studying the trust relationship between interacting agents from a high level abstraction point of view. Specifically, we have introduced TCTL logic, an extension of CTL with a simple modality to reason about absolute trust with no preconditions. We have presented a computational grounded semantics based on a new trust accessibility relation. The goal of the present work is to refine the proposed logic to allow reasoning about trust with preconditions, and to redefine the accessibility relation that is used to determine the semantics of trust. The main objective behind the new definition is to account for the intuition that trust is a private concern (i.e., the trust information that relates an agent i to other agents in the system should be known to agent i only, unless i announces this information publicly). In our previous work, we used a public matrix representation accessible to all the agents in each global state. Even though this representation was working in the sense that we know if agent i trusts agent j, it is, however, limited with the fact that the knowledge of trust was global, so that all the agents in the system know which agent trusts which other agents. This type of information should be kept private because it might reveal some sensitive data to undesired parties. Moreover, the problem of automatically verifying, through model checking, the resulting logic (TCTL) has not been addressed in our previous work. In this article, we aim to overcome such a limit by developing new and fully implementable symbolic model checking algorithms. Indeed, we extend the standard CTL model checking algorithm introduced in [20] with symbolic algorithms needed for the new modality. The algorithms are implemented as a new tool called MCMAS-T[1] on top of the MCMAS model checker for MASs. We evaluated our approach using a real-life case study in the healthcare domain to explain our proposed framework in a practical application. The main contributions of the paper are highlighted as follows:

- Introducing a logical language named Trust Computation Tree Logic (TCTL), which extends CTL with a trust operator to represent and reason about the properties that an agent requires to be achieved by the trusted agent. The introduction of the new logic is motivated by the fact that the needed modality for trust cannot be expressed using current temporal logics such as LTL [22] and CTL.
- Defining a new trust accessibility relation to capture the trust relationship between the interacting parties by extending the original framework of interpreted systems [23]. To the best of our knowledge, our work is the first initiative that gives formal and computational definitions of the trust accessibility relation as a private concept between interacting agents in MASs.
- Proposing and implementing two symbolic model checking algorithms for the proposed logic (TCTL).
- Considering a set of reasoning postulates along with proofs to support our logic.

1.3. Paper organization

The rest of the paper is organized as follows. In Section 2, we review the related work on formal logics of trust. In Section 3, we introduce the original formalism of interpreted systems introduced by Fagin et al. [23] along with the proposed extension. Also in this section, we define the syntax and semantics of TCTL. In Section 4, we consider a set of reasoning postulates along with proofs and examples. The development of new BDD-based algorithms to perform model checking for the proposed logic is presented in Section 5. In Section 6, we present the experimental results by means of a real-life scenario in the healthcare field. We conclude the paper and suggest future work in Section 7.

2. Literature review

Trust has been an essential research topic in several disciplines for many years. Each of these disciplines gives different definitions for trust [24–27]. For instance, in the field of distributed computing, trust is used mainly to regulate the relationship between service providers and customers [27,28]. In social science disciplines, trust is seen as a relationship between individuals in social settings [26] (e.g., trust is used to control relationships between trusters and trustees to ensure that the trustees will perform a certain action).

In the context of MASs, the most widely used definition is the one proposed by Castelfranchi and Falcone (abbreviated as

[1] The tool is available online at: https://www.dropbox.com/s/xy0wjrvvk36d00z/mcmas-t-1.0.1-sc.tar.gz?dl=0.

C&F) [29], where trust is basically defined as a mental state of one agent (the truster) towards another agent (the trustee) in which the truster's goals and beliefs are reflected in some internal properties of the trustee. They studied the trust concept in a cognitive perspective that emphasizes the importance of the goal component. Such a component allows us to distinguish trust from mere thinking and foreseeing [30]. However, by emphasizing the agent's goal, C&F rely on the mental structures of agents for the fulfillment of their own goals. Since these systems involve autonomous entities that keep their structure private, it seems impossible to know which agent's goals they refer to, which makes the verification of trust fulfillment infeasible. To cope with this limitation, we take in this research a new approach towards a cognitive-independent perspective of trust where the trust ingredients are seen from a non-cognitive angle. Instead, we define trust from a high-level abstraction without having to depend on individual agent's internal mental states.

Trust in multi agent systems has been analyzed from different aspects. The major studies focused on two main approaches: the quantitative approach and the logical approach. The quantitative direction treats trust as a numerical measure that is calculated based on feedback, user ratings, and agent monitoring [31–33]. Such approaches represent and quantify the strength level in which an agent trusts another party. Specifically, the higher an agent trusts another agent, the more likely the latter would be chosen as an interaction partner. Trust was first introduced as a measurable notation of an entity in [34]. Following this work, a number of computational models have been proposed in the MASs literature (see for instance [35]). On the other hand, the logical approach mainly focuses on defining semantic structures for trust. Several logical frameworks have been proposed to describe the static and dynamic properties of trust. Such approaches provide a formal semantics to reason about trust properties in various applications such as security protocols, information sources, and recommendation systems. Moreover, in terms of expressiveness, some of these studies adopted combining logics [18,36], while others extended standard logics of action and belief, or enriched temporal logics with a new modality for trust [8,30,37].

Hereafter, we classify the current state-of-the-art in the domain of trust in MASs in two parts: logical languages and verification techniques. In the following, we present a brief overview of each of these constituents and highlight the main features of our proposed framework.

2.1. Logical languages

Modal logics have been used to formalize trust in MASs by many researchers. Most of the existing formal approaches consider trust based on key aspects of cognitive view where trust involves four components: truster, trustee, action of the trustee, and goal of the truster. In particular, Demolombe [38] has proposed several approaches for trust reasoning. He developed a logic by combining the dynamic logic of Harel et al. [39] with the BDI logic of Cohen and Levesque [40]. Trust in this work is considered as an attitude of the truster who believes that the trustee has a given property. In other proposals, Demolombe, Lorini and Amgoud have focused on analyzing trust in various features of information sources [37,41,42]. For instance, in [41], the authors formalize some security properties and their relationships with trust such as integrity, availability and privacy by introducing a modal operator of obligation. Thus, one agent is trusting another agent if the former believes that the information transmitted to it is reliable. Similarly, trust in the domain of information sources is presented in another work [43] where the trust operator is interpreted using a neighborhood semantics [44]. In this work, a logic is provided with a rigorous semantics to precisely characterize the relationships among the notions of trust, information acquisition, and belief. Following the path of cognitive trust, Herzig et al. [30] proposed a formal logical framework to evaluate agents' behavior in a multi-agent setting. To do so, they presented a language that combines the expressiveness of the logic of time, the logic of action, and the logic of beliefs. Moreover, the proposed logic is extended in order to enable reasoning about reputation. Trust in their work is based on the basic concept of belief while the reputation is considered in the scope of collective beliefs. They distinguish between two general categories of trust: occurrent trust and dispositional trust, and they provided precise definitions and formal representations for the two concepts.

While the aforementioned approaches have taken a good step towards developing a modal logic for trust, they fail to present the notion of trust explicitly. Moreover, such studies are mostly focused on agents with mental states where the trusting entity is normally capable of exhibiting beliefs, desires, and intentions. Nevertheless, agents are operating in open environments. Thus, they are likely using different types of platforms and are possibly using different technologies, so it is very difficult for one agent to completely trust others or to make assumptions about their internal states. Besides, such logics are abstract and not computationally-grounded (i.e., we cannot interpret them using concrete computational models), which makes them hard to be applicable for verification purposes.

Singh [8] provided a formal semantics for trust with various logical postulates used to reason about trust from an architectural perspective. His model is based on the idea of trust as a dependence. This model combines temporal modalities of linear temporal logic (LTL) [22] and modality (C) for commitments [45] with (T) modality for trust. The semantics is interpreted using neighborhood semantics [44], which maps each world into a set of subsets of worlds. A function $\Bbbk$ that produces a set of propositions for each moment, an ordered pair of two agents, and a proposition is defined. This function yields a set of consequent propositions, which in turn captures what the truster would be trusted to if the antecedent holds. Thus, the trust semantics is defined by computing the set of moments where the consequence holds and testing if those moments are among the moments computed by $\Bbbk$ on the trust antecedent. Yet, it is not clear how the function $\Bbbk$ is computed, which makes the approach quite difficult to be applied in practice. Furthermore, unlike our approach that is based on interpreted systems structure, such a logic is interpreted using neighborhood semantics, which is very hard to verify and to develop model checking algorithms for. Another logic-based proposal for trust has been initiated by Burrows et al. [46] who introduced a logic of trust called BAN-logic used for reasoning about the correctness of security protocols. The authors translated the protocol steps as logical formulae in order to manipulate them using first-order logic. Nevertheless, they do not provide any proof and the proposed logic is very abstract and too complicated to be used in real applications, which makes the verification of the proposed logic extremely hard.

2.2. Verification techniques

The verification mechanisms of trust in MASs fall into two categories depending on when the verification activities are performed: runtime and design-time. For the first approaches, monitoring is the most common used technique where the verification is performed by monitoring the evolving executions of the target system during the operation phase, and then checking whether the desired properties of the system hold or not [47–49]. Runtime verification can also extract relevant information from a running system and use it to detect undesired behaviors with regard to particular properties. On the other hand, the second approaches rely on the static formal verification, which is a class of logic-based techniques. In such approaches, model checking has become one of the most successful approaches widely used for verifying various aspects of MASs [7,50,51]. Indeed, each technique has its own advantages and limitations. For instance, one of the appealing features of the runtime verification is the use of real and concrete runs to check the correctness of the target system. This allows us not only to observe the real system, but also react whenever undesired behaviors are detected [52]. However, such techniques only consider a particular execution of the system, which may lead to an incomplete verification process due to the limited coverage. In contrast, the design phase techniques systematically check all possible states of the system, provide full automation of the verification process, and can produce counter examples when the system fails to satisfy a desired property. Yet, such techniques suffer from the state explosion problem that limits the applicability of verifying large systems. Technically, both techniques complement each other in detecting untrustworthy behaviors and improving MASs development.

In fact, although trust is dynamic because agents can change their behaviors dynamically, still verifying trust properties at design time is very critical and useful. Model checking trust is of prime importance to ensure that trust behavior can take place between agents engaged in an interaction. For instance, if the outcome of the model checking reveals there is no execution where a particular trust property is satisfied, then this could be considered as a final verdict and the model should be changed because it is unsafe to deploy it in real systems. On the other hand, if the model checking reveals the opposite, which means all possible executions are trusted, then showing the equivalence between the implemented model and the designed one would be enough, if such an equivalence is possible to be proven. In the general case where the outcome reveals that some paths are trusted and some others are not, then model checking will benefit the dynamic verification that could be guided to monitor the untrusted paths, which means monitor if the agents are behaving according to the identified untrusted paths. Though, in this paper, we adopt a design time approach as our main goal is to improve the utilization of MASs paradigm by reducing the time and cost of the development process, and to increase the confidence on the safety, efficiency and robustness of the system. In this line, some relevant approaches with the aim of verifying trust-based models using formal methods have appeared recently. For instance, Aldini [18] proposed a formal framework that integrates trust modeling with distributed systems. The work encompassed a process calculus of concurrent systems and a temporal logic for trust along with a model checking technique in order to verify the effects of trust-based systems against malicious attacks. In this work, the trust system model is based on an instance of labeled transition systems and an instance of Kripke transition systems. Thus, the verification of temporal logic properties has been performed successfully through a standard model checking technique. Moreover, such properties are specified by proposing a language called trust temporal logic (TTL) which combines CTL [9] and its action-based extension (ACTL) [53]. However, this work can be applied in a very limited range of systems, and it is not adapted to autonomous MASs.

The authors in [14] proposed an approach to model and verify trust models specified in a Colored Petri Net (CPN). They presented a model (named TCPN) that can be used to check the performance and behavior of such systems from both simulation and model checking points of view. In their approach, state-space is used to formalize the conceptual model of trust, which is then represented by Colored Petri nets to entitle for simulation and verification using existing modeling tools. Yet, this work fail to provide a verification of a trust model using model checking, and moreover, no attackers are considered in their modeling. Model checking service trust behaviors has been recently investigated in [54], where the authors present a model checking framework to verify the trust behaviors model against regular and non-regular properties. To model their target system, 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. By doing so, they overcome the problem of non-regular model checking algorithms based on PDA. From the semantics point of view, they specified the trust behavior properties using Fixed point Logic with Chop (FLC). By using the chop operator, they were able to represent the non-regular properties. Moreover, they applied a symbolic *FLC* model checking algorithm to verify service trust behaviors with respect to trust properties. Indeed, FCL behavior formulae and the generated models are used as the inputs to the mcflc model checker tool which is the only tool for FLC model checking. However, this approach lacks formal semantics for trust because trust formulae are inferred from the context free grammar of trust pattern languages. Besides, non-regular behaviors verification is limited by its EXPTIME complexity, where no attempt to reduce it to polynomial time has been made in this approach. Moreover, the approach is not designed to formalize and verify trust for autonomous MASs.

To address the limitations of the discussed proposals, our work provides an intuitive and grounded computational semantics based on a new version of interpreted systems. Thus, we are able to formally specify and automatically verify trust-based interactions among autonomous agents. We develop new symbolic model checking algorithms dedicated to the proposed logic (TCTL). Moreover, we implement our algorithms on top of the MCMAS model checker for MASs. Hence, a new model checker tool called MCMAS-T along with its new input language VISPL (Vector-extended ISPL) are produced. Finally, our framework can be applied to a variety of application domains that are based on MASs paradigm.

3. Vector-based interpreted systems and TCTL

3.1. Interpreted systems

An interpreted system [23] is a formal description that has been proven to be a suitable formalism for reasoning about knowledge and time in MASs, and which systematically models different classes of MASs, such as synchronous and asynchronous. Such a formalism enjoys a high level of abstraction that allows focusing only on the interactions among the various agents, and it is a useful tool for modeling autonomous and heterogeneous agents interacting within a global system.

Here, we present the standard semantics of interpreted systems as in [23]. Consider a set $Agt = \{1, \dots, n\}$ of n agents in which at any given time, each agent in the system is in a particular local state, which represents the complete information about the system that the agent can access at that time. Each agent $i \in Agt$ is associated with a non-empty set of local states L_i and a set of local actions Act_i to model the temporal evolution of the system together with a local protocol $\rho_i : L_i \rightarrow 2^{Act_i}$ assigning a list of enabled actions to a given local state $l_i \in L_i$. It is assumed that $null \in Act_i$ for each agent i, where $null$ refers to the silent action (the fact of doing nothing). A local evolution function τ_i is defined as: $\tau_i : L_i \times Act_i \rightarrow L_i$, which determines the transitions for an individual agent i between her local states.

As in [23], a global state $g \in G$ represents a snapshot of all agents in the system at a given time. A global state g is a tuple $g = (l_1, \dots, l_n)$. The set of all global states $G \subseteq L_1 \times \dots \times L_n$ is a non-empty subset of the Cartesian product of all local states of n agents. The notation $l_i(g)$ is used to represent the local state of agent i in the global state g. $I \subseteq G$ is a set of initial global states for the system. The global evolution function of the system is defined as follows: $\tau : G \times ACT \longrightarrow G$, where $ACT = Act_1 \times \dots \times Act_n$

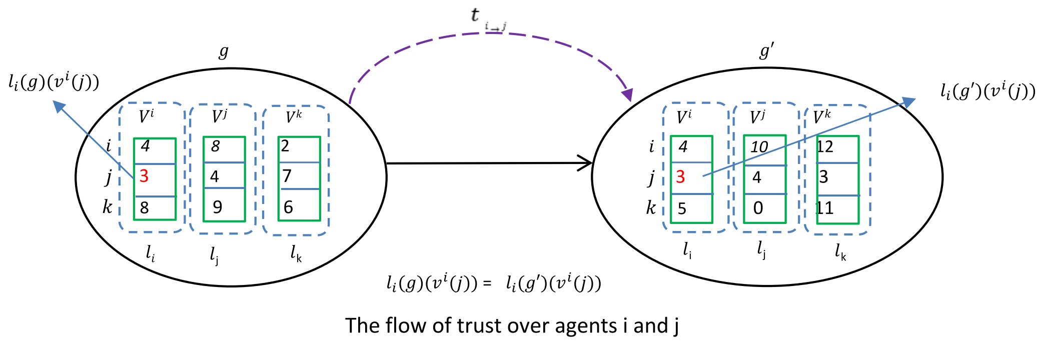


Fig. 1. The solid line represents the transition relation from R, and the dashed line represents the direct trust accessibility relation $t_{i\to j}$. In this example, g' is compatible with g with regard to the trust of agent i towards the agent j. In the figure, we assign a vector to each agent's local states. $v^i_{1\times 3}$ is the vector of agent i where 3 is the number of interacting agents at that time. The agent i compares the element of her vector at global states g and g'. The particular value of the $v^i(j)$ of agent i is the same in both states.

and each component $a \in ACT$ is called a joint action, which is a tuple of actions. As in [23], a special agent e is used to model the environment in which the agents operate. The environment e is modeled using a set of local states L_e, a set of actions Act_e, a protocol P_e, and an evolution function τ_e.

3.2. Vector-based interpreted systems

In order to account for the trust relationship between the truster and trustee, we extend the interpreted systems formalism by associating with each local state $l_i \in L_i$ for each agent $i \in Agt$ a vector of size n, i.e., $v^i_{1\times n}$ where n is the number of agents in the system at a given time. We denote $v^i(i), v^i(j), \ldots, v^i(k)$ the components of the vector v^i at local state $l_i(g)$. This extension allows us to provide an intuitive semantics for direct trust that takes place between interacting parties.

Definition 3.1 (*Model of TCTL*)**.** A model of trust generated from our vector-based interpreted systems is a tuple $M = (S, R, I, t_{i\to j}, V)$, where: S is a non-empty set of reachable global states for the system; $R \subseteq S \times S$ is the transition relation; $I \subseteq S$ is a set of initial global states for the system; $t_{i\to j} \subseteq S \times S$ is the direct trust accessibility relation for each truster–trustee pair of agents $(i, j) \in Agt^2$ defined by $s t_{i\to j} s'$ iff: $l_i(s)(v^i(j)) = l_i(s')(v^i(j))$ and s' is reachable from s[2] using transitions from the transition relation R; and $V : S \to 2^{AP}$ is a valuation function, where AP is a set of atomic propositions.

The intuition of the relation $t_{i\to j}$ is, for agent i to gain trust in agent j, the former identifies the states that are compatible with her trust with regard to the latter, i.e., where agent i knows that agent j is trusted. Thus, accessible states using $t_{i\to j}$ are states where agent i trusts agent j, so if s' is accessible from s, then if i trusts j in s, then i also trusts j in s'. Computationally, this is obtained by comparing the element $v^i(j)$ in the local state l_i at the global state g (denoted by $l_i(g)(v^i(j))$) with $v^i(j)$ in the local state l_i at the global state g' (denoted by $l_i(g')(v^i(j))$). Thus, the trust accessibility of agent i towards agent j (i.e., $t_{i\to j}$) 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., $l_i(g)(v^i(j)) = l_i(g')(v^i(j))$. This idea is illustrated in Fig. 1.

The following proposition is direct from the definition of the accessibility relation.

Proposition 3.1. *The accessibility relation $t_{i\to j}$ is reflexive and transitive. Thus, the resulting logic of trust is an S4 system of modal logic.*

[2] Each state is supposed to be reachable from itself even if there is no loop.

3.3. Trust computation tree logic TCTL

TCTL is a combination of branching time temporal logic (CTL) [9] with trust modality for reasoning about trust and time.

Definition 3.2 (*Syntax of TCTL*)**.** The syntax of TCTL is defined as follows:

$$\phi ::= \rho \mid \neg\phi \mid \phi \vee \phi \mid EX\phi \mid EG\phi \mid E(\phi \cup \phi) \mid EG\phi \mid T_{i\to j}(\phi, \phi)$$

where $\rho \in AP$ is an atomic proposition, the formula $EX\phi$ stands for "ϕ holds in the next state in at least one path"; $EG\phi$ stands for "there exists a path in which ϕ holds globally", and the formula $E(\psi \cup \phi)$ 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. The modality $T_{i\to j}(\psi, \phi)$ stands for *"Trust"*, and is read as "the truster i trusts the trustee j that the precondition ψ holds and to bring about ϕ". Other temporal modalities such as F (future), A the universal quantifier, and E the existential quantifier can be defined as usual (see for example [9]).

A path π in a model M from a state s_0 is an infinite sequence of reachable global states $\pi = s_0 s_1 s_2 \cdots$ such that for all $i \geq 0$, $(s_i, s_{(i+1)}) \in R$.

Definition 3.3 (*Satisfaction*)**.** Given the model M, the satisfaction for a TCTL formula ϕ in a global state s, denoted as $(M, s) \models \phi$, is recursively defined as follows:

$-(M, s) \models \rho$ iff $\rho \in V(s)$;

$-(M, s) \models \neg\phi$ iff $(M, s) \not\models \phi$

$-(M, s) \models \phi_1 \vee \phi_2$ iff $(M, s) \models \phi_1$ or $(M, s) \models \phi_2$

$-(M, s) \models EX\phi$ iff there exists a path π starting at s such that $(M, \pi(1)) \models \phi$;

$-(M, s) \models E(\phi_1 \cup \phi_2)$ iff there exists a path π starting at s for some $k \geq 0$, $(M, \pi(k)) \models \phi_2$ and $\forall\ 0 \leq i < k$, $(M, \pi(i)) \models \phi_1$;

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

$-(M, s) \models T_{i\to j}(\psi, \phi)$ iff $(M, s) \models \psi \wedge \neg\phi$ and $\exists\, s' \neq s$ such that $s t_{i\to j} s'$, and $\forall\, s' \neq s$ such that $s t_{i\to j} s'$, we have $(M, s') \models \phi$.

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 $T_{i\to j}(\psi, \phi)$ is satisfied in the model M at s iff (1) there exists a state s' such that $s' \neq s$ and $s t_{i\to j} s'$, and (2) all the trust accessible states s' that are different from the current state s satisfy the content of trust ϕ. Moreover, for the trust to take place between the interacting agents i and j, we add the condition $\psi \wedge \neg\phi$, which must be satisfied in the current state s to ensure that the precondition ψ holds before the trust content ϕ is brought about.

4. Reasoning postulates

In this section, we consider the Breast Cancer Diagnosis and Treatment (BCDT)[3] protocol as an illustrative application example to clarify the proposed reasoning postulates. Thereafter, we introduce the reasoning rules along with the required proofs to capture the properties of our logic.

4.1. Breast cancer diagnosis and treatment: a case study

The BCDT protocol is introduced by the Assistant Secretary for Planning and Evaluation (ASPE) project to be used for regulating the interaction between five participating parties involved in the cancer diagnosis process. These parties are: *patient* denoted by p, *physician* (ph), *pathologist* (pg), *radiologist* (rg), and *registrar* (r). The process of BCDT protocol starts when the physician notices a suspicious mass in the patient's breast. Thereafter, the patient is immediately directed to the radiology department to do a mammography (a technique using X-rays to diagnose and detect breasts tumors). If the radiologist observes suspicious calcification, he will send a report to the physician to recommend a biopsy. The physician requests a radiologist to carry out a biopsy. The radiologist obtains diagnostic tissue from the patient and then sends it to the laboratory along with pertinent clinical information for further analysis by a pathologist. This latter plays a vital role in the diagnosis process. He analyzes the tissue specimen through imaging studies and determines whether a malignant disease is present or not. Both the radiologist and pathologist create and release a report of their collective findings. Finally, the physician reviews the complete report with the patient to decide about a treatment plan. At the same time, the pathologist forwards the report to the registrar whose role is to insert the patient information into the cancer registry.

4.2. Reasoning postulates

We present here relevant reasoning postulates of our logic that reflect its properties. These postulates capture common reasoning patterns about trust. Some of those postulates are similar to the ones discussed in [8].

P1: Fulfillment. $\phi \rightarrow \neg T_{i\rightarrow j}(\psi, \phi)$

- **Meaning:** The trust has been achieved, so if the content already holds, then the trust is no longer active.
- **Proof:** The rule is derived directly from the semantics of $T_{i\rightarrow j}(\psi, \phi)$ which indicates that the current state does not satisfy ϕ.
- **Example:** According to the BCDT protocol, once the physician requests a mammography to be done ($Mammo\_Req$), there will be no need to establish the trust between the patient and physician with regard to this request under the precondition that a suspicious mass is noticed ($Mass\_Not$) because this mammography is already requested (the trust content already holds). Formally: $Mammo\_Req \rightarrow \neg T_{p\rightarrow ph}(Mass\_Not, Mammo\_Req)$.

P2: Content Partial Partition. $T_{i\rightarrow j}(\psi, \phi_1 \wedge \phi_2) \wedge \neg\phi_1 \rightarrow T_{i\rightarrow j}(\psi, \phi_1)$

- **Meaning:** The trust to bring about a conjunction is the trust for each part separately unless it does already hold.
- **Proof:** Assume that $(M, s) \models T_{i\rightarrow j}(\psi, \phi_1 \wedge \phi_2) \wedge \neg\phi_1$. From the semantics, we obtain $(M, s) \models \psi \wedge \neg\phi_1$, and there exists a state s' such that $s' \neq s$ and $s_{t_{i\rightarrow j}}s'$, and all the trust accessible states s' such that $s \neq s'$ and $s_{t_{i\rightarrow j}}s'$ satisfy $\phi_1 \wedge \phi_2$. Thus, these states also satisfy ϕ_1. Consequently, $(M, s) \models T_{i\rightarrow j}(\psi, \phi_1)$.
- **Example:** Suppose, for instance, that the physician trusts the radiologist to collect diagnostic tissue from the patient ($Tiss\_Coll$) and send the specimen to the laboratory ($Tiss\_Send$) with the precondition that the mammography is requested, then the physician trusts the radiologist for the collection of the diagnostic tissue, unless this tissue has been already obtained. Formally, $T_{ph\rightarrow rg}(Mammo\_Req, Tiss\_Coll \wedge Tiss\_Send) \wedge \neg Tiss\_Coll \rightarrow T_{ph\rightarrow rg}(Mammo\_Req, Tiss\_Coll)$.

P3: Content Full Partition. $T_{i\rightarrow j}(\psi, \phi_1 \wedge \phi_2) \rightarrow T_{i\rightarrow j}(\psi, \phi_1) \vee T_{i\rightarrow j}(\psi, \phi_2)$

- **Meaning:** If the trust to bring about a conjunction holds, then at least the trust to bring about one part holds.
- **Proof:** Assume that $(M, s) \models T_{i\rightarrow j}(\psi, \phi_1 \wedge \phi_2)$. From the semantics, we obtain $(M, s) \models \psi \wedge (\neg\phi_1 \vee \neg\phi_2)$, and there exists a state s' such that $s' \neq s$ and $s_{t_{i\rightarrow j}}s'$, and all the trust accessible states s' such that $s \neq s'$ and $s_{t_{i\rightarrow j}}s'$ satisfy $\phi_1 \wedge \phi_2$. Thus, for all those states s', $(M, s') \models \phi_1$ or $(M, s') \models \phi_2$. Consequently, $T_{i\rightarrow j}(\psi, \phi_1) \vee T_{i\rightarrow j}(\psi, \phi_2)$ holds.
- **Example:** As previous example, suppose that the physician trusts the radiologist to collect diagnostic tissue from the patient and send the specimen to the laboratory, then the physician trusts that radiologist for at least one of the two actions. In fact, if the two actions do not hold currently, then the physician trusts the radiologist to perform them both. Formally, $T_{ph\rightarrow rg}(Mammo\_Req, Tiss\_Coll \wedge Tiss\_Send) \rightarrow T_{ph\rightarrow rg}(Mammo\_Req, Tiss\_Coll) \vee T_{ph\rightarrow rg}(Mammo\_Req, Tiss\_Send)$.

P4: Non-Conflict. $T_{i\rightarrow j}(\psi, \phi) \rightarrow \neg T_{i\rightarrow j}(\psi, \neg\phi) \wedge \neg T_{i\rightarrow j}(\neg\psi, \phi)$

- **Meaning:** Trust must be consistent, content and precondition wise. A truster cannot trust a trustee (1) to bring about a content and its negation simultaneously; and (2) to bring about a content with a precondition and its negation simultaneously.
- **Proof:** From the left side, s satisfies ψ, there exists a state $s' \in S$ such that $s \neq s'$ and $s_{t_{i\rightarrow j}}s'$, and all the accessible states s' such that $s \neq s'$ satisfy ϕ. Since a state that satisfies ϕ (resp. ψ) cannot satisfy $\neg\phi$ (resp. $\neg\psi$), we are done.
- **Example:** When the patient trusts the physician to request the mammography with the precondition that a mass is noticed, then the trust not to request the mammography with the same precondition and the trust to request the mammography knowing that the mass is not noticed cannot hold. Formally: $T_{p\rightarrow ph}(Mass\_Not, Mammo\_Req) \rightarrow \neg T_{p\rightarrow ph}(Mass\_Not, \neg Mammo\_Req) \wedge \neg T_{p\rightarrow ph}(\neg Mass\_Not, Mammo\_Req)$.

P5: Non-Vacuity. From $\psi \vdash \phi$ **infer** $\neg T_{i\rightarrow j}(\psi, \phi)$[4]

- **Meaning:** Trust must be for something tangible.
- **Proof:** Assume that $T_{i\rightarrow j}(\psi, \phi)$. Thus, from the semantics, the current state s satisfies $\psi \wedge \neg\phi$, which is contradiction with $\psi \vdash \phi$, which means we can get ϕ from ψ, so the rule.

[3] Available at: http://aspe.hhs.gov/sp/reports/2010/PathRad/index.shtml.

[4] The symbol $\vdash$ is an element of the object language, while the word infer is from the metalanguage.

- **Example:** It does not make sense that the physician trusts the radiologist to collect the tissue if it is already sent to the laboratory. Formally: *Tiss_Send* $\vdash$ *Tiss_Coll* infer $\neg T_{ph\to rg}(Tiss\_Send, Tiss\_Coll)$.

P6: Precondition Slackening: From $T_{i\to j}(\psi_1, \phi)$, $\psi_1 \vdash \psi_2$ **infer** $T_{i\to j}(\psi_2, \phi)$

- **Meaning:** If trust holds for a stronger precondition, then it holds for a weaker one.
- **Proof:** Assume that $(M, s) \models T_{i\to j}(\psi_1, \phi)$. From the semantics, $(M, s) \models \psi_1 \wedge \neg\phi$, and there exists a state $s' \in S$ such that $s \neq s'$ and $s_{t_{i\to j}}s'$, and for all the trust accessible states s' such that $s \neq s'$, we have $(M, s') \models \phi$. Since $\psi_1 \vdash \psi_2$, meaning that ψ_2 is provable from ψ_1, we conclude $(M, s) \models \psi_2$, so we are done.
- **Example:** When the physician trusts that the radiologist will send a report of her findings with the precondition that a biopsy is requested, then the physician safely trusts the latter about sending the report with the precondition that a mammography is requested because requesting a biopsy entails that a mammography has been requested.
- **Instances:** The following rules are instances or consequences of the precondition slackening postulate:
  1– $T_{i\to j}(\psi, \phi) \to T_{i\to j}(\top, \phi)$. This means if trust holds for a precondition, then it holds with no precondition. In other words, when the precondition is confirmed, then only the trust content matters.
  2– $T_{i\to j}(\psi_1 \wedge \psi_2, \phi) \to T_{i\to j}(\psi_1, \phi)$. This means when the trust holds for a conjunctive precondition, then it comes into effect for each part of this conjunction.
  3– $T_{i\to j}(\psi_1, \phi) \to T_{i\to j}(\psi_1 \vee \psi_2, \phi)$. This means the trust to bring about the content ϕ with a disjunctive precondition holds if the trust about the same content holds for a part of the disjunction.

P7: Precondition Extension. $T_{i\to j}(\psi_1, \phi) \wedge \psi_2 \to T_{i\to j}(\psi_1 \wedge \psi_2, \phi)$

- **Meaning:** If trust holds for a precondition ψ_1, then it still holds for an extended precondition with ψ_2 subject to the satisfaction of the extending part (i.e., ψ_2).
- **Proof:** The satisfaction of ψ_1, $\neg\phi$, the existence of a trust accessible state different from the current state, and the satisfaction of ϕ in all these accessible states are derived from the satisfaction of $T_{i\to j}(\psi_1, \phi)$, and the satisfaction of ψ_2 is already given, so the postulate.
- **Example:** Suppose that the radiologist trusts that the registrar will insert the patient's name into a cancer registry with the precondition that the patient sends the consensus report. Then, the trust holds with the additional precondition that the patient accepts to forward her information to healthcare providers.

P8: Precondition Transfer. $T_{i\to j}(\psi_1, \phi) \wedge \psi_2 \to T_{i\to j}(\psi_2, \phi)$

- **Meaning:** If trust holds for a precondition, then it comes into effect for any true precondition.
- **Proof:** This postulate is a direct consequence of P7 and Instance 2 of P5.
- **Example:** From the previous example, the trust holds if the precondition is transferred to the fact that the patient accepts to forward her information to healthcare providers.

P9: Exchange. $T_{i\to j}(\psi_1, \phi_1) \wedge T_{i\to j}(\psi_2, \phi_2) \to T_{i\to j}(\psi_1, \phi_2)$

- **Meaning:** Once a trust holds, its precondition can be exchanged with the precondition of another holding trust.
- **Proof:** From $T_{i\to j}(\psi_1, \phi_1)$, we have $(M, s) \models \psi_1$, and from $T_{i\to j}(\psi_2, \phi_2)$, $(M, s) \models \neg\phi_2$ and there exists a state $s' \in S$ such that $s \neq s'$ and $s_{t_{i\to j}}s'$, and all the trust accessible states s' different from s satisfy ϕ_2, so the postulate.
- **Example:** Suppose again that the radiologist trusts that the registrar will insert the patient's name into a cancer registry if the patient sends the consensus report, and also trusts the same registrar to forward the patient's information to healthcare providers under the acceptance of the patient as precondition, then both trusts hold regardless with which precondition since both preconditions hold already.

P10: Combination. $T_{i\to j}(\psi_1, \phi_1) \wedge T_{i\to j}(\psi_2, \phi_2) \to T_{i\to j}(\psi_1 \wedge \psi_2, \phi_1 \wedge \phi_2)$

- **Meaning:** The conjunction of two trusts between the same truster and trustee yields a combined trust, precondition and content wise.
- **Proof:** From the left side, we have $(M, s) \models \psi_1 \wedge \psi_2 \wedge (\neg\phi_1 \wedge \neg\phi_2)$, which implies $(M, s) \models \psi_1 \wedge \psi_2 \wedge \neg(\phi_1 \wedge \phi_2)$. Moreover, there exists a state $s' \in S$ such that $s \neq s'$ and $s_{t_{i\to j}}s'$, and all the trust accessible states different from s, $(M, s') \models \phi_1$ and $(M, s') \models \phi_2$, so the postulate.
- **Example:** As previous example, the radiologist's trust that the registrar will insert the patient's name into a cancer registry and forward the patient's information to healthcare providers under the acceptance of the patient of the two actions as precondition hold if each trust holds individually.
- **Instances:** The following rules are instances of the combination postulate:
  1– $T_{i\to j}(\psi_1, \phi) \wedge T_{i\to j}(\psi_2, \phi) \to T_{i\to j}(\psi_1 \wedge \psi_2, \phi)$. This means the truster trusts the trustee to bring about a content under a combined precondition if the trust about the same content holds for each precondition separately.
  2– $T_{i\to j}(\psi, \phi_1) \wedge T_{i\to j}(\psi, \phi_2) \to T_{i\to j}(\psi, \phi_1 \wedge \phi_2)$. This means the truster trusts the trustee to bring about a combined content under a precondition if the trust about each content separately holds for the same precondition.

P11: Content Inference. From $T_{i\to j}(\psi, \phi_1)$, $\phi_1 \vdash \phi_2, \neg\phi_2$ **infer** $T_{i\to j}(\psi, \phi_2)$

- **Meaning:** The trust to bring about ϕ_2 yields if the truster trusts the trustee to bring about a content from which ϕ_2 derives, knowing that ϕ_2 does not hold currently.
- **Proof:** From the semantics of $T_{i\to j}(\psi, \phi_1)$, $(M, s) \models \psi$, there exists a state $s' \in S$ such that $s \neq s'$ and $s_{t_{i\to j}}s'$, and all the trust accessible states s' satisfy ϕ_1. These states satisfy ϕ_2 since $\phi_1 \vdash \phi_2$. Thus the rule follows from the fact that $(M, s) \models \neg\phi_2$.
- **Example:** Suppose that the radiologist trusts that the registrar will insert the patient's name into a provincial cancer registry with the precondition that the patient sends the consensus report, then the radiologist trusts that the patient's name will be added to the national registry as well, if not done already, because being in the provincial registry automatically triggers the process of being added to the national registry.

P12: Trust Achievement and Non-Contradiction. $T_{i\to j}(\psi, \phi) \to EF(\phi \wedge \neg T_{i\to j}(\top, \neg\phi))$

- **Meaning:** There is always a way to honor trust about a content ϕ and when it is honored, the trust stays consistent, so no trust about the negation of ϕ can hold.

- **Proof:** Assume that $(M, s) \models T_{i\to j}(\psi, \phi)$, so from the semantics, ϕ holds in all the trust accessible states s' from s. Since trust accessibility implies reachability, then ϕ holds in a possible future of s, i.e., $EF(\phi)$. Moreover, in each trust accessible state s', two cases could take place.
  - Case 1: $\nexists s'' \neq s'$ s.t. $s'_{t_{i\to j}} s''$. In this case $s' \models \neg T_{i\to j}(\top, \neg\phi)$
  - Case 2: $\exists s'' \neq s'$ s.t. $s'_{t_{i\to j}} s''$. Assume that one of the trust accessible states s'' from s' satisfies $\neg\phi$. Since trust accessibility is transitive, s'' is also accessible from s, so it cannot satisfy $\neg\phi$, which contradicts the above. Consequently, also in this case $s' \models \neg T_{i\to j}(\top, \neg\phi)$.

  As $s' \models \phi$, we are done.
- **Example:** Suppose that the radiologist trusts that the registrar will insert the patient's name into a cancer registry with the precondition that the patient sends the consensus report, then we know this will eventually happen where the radiologist cannot trust the opposite to happen.

P13: Content Nonexistence. $AG\neg\phi \to \neg T_{i\to j}(\psi, \phi)$

- **Meaning:** If the content of trust does not hold in all reachable states, then the trust never holds.
- **Proof:** The proof is straightforward from the semantics and from the fact that all states s' such that $s_{t_{i\to j}} s'$ are reachable.
- **Instance:** The following rule is a consequence of the content nonexistence postulate:
  1– $\neg T_{i\to j}(\psi, \bot)$. This represents the content consistency and means trust to false cannot hold.

P14: Precondition Nonexistence. $AG\neg\psi \to \neg T_{i\to j}(\psi, \phi)$

- **Meaning:** There is no trust if the precondition never holds.
- **Proof:** The proof is straightforward since the first condition for $T_{i\to j}(\psi, \phi)$ to hold is the satisfaction of ψ in the current state.
- **Instance:** The following rule is a consequence of the precondition nonexistence postulate:
  1– $\neg T_{i\to j}(\bot, \phi)$. This represents the precondition consistency and means trust with a false precondition cannot come to effect.

5. Automatic verification of TCTL properties of MASs

In this section, we present two separate algorithms to address the problem of model checking TCTL. We aim to investigate the most intuitive and efficient algorithm for computing the trust set. In particular, we extend the standard CTL symbolic algorithm introduced in [20] by adding the procedure that deals with the trust modality in our logic. We adapt a symbolic model checking (SMC) as our underlying technique due to its effectiveness in mitigating the state explosion problem. Such a problem occurs when the number of components in the system grows up to the degree that makes the number of global states enormous. Symbolic model checking considers the set of states (denoted as $[[\phi]]$) satisfying the given formula ϕ in the model M, which is represented and manipulated using Ordered Binary Decision Diagram (OBDD) [55] data structure. Such a set is then compared against the set of initial states I in the model M (also represented as an OBDD). Thus, we say that a model M satisfies the formula ϕ if and only if $I \subseteq [[\phi]]$. In a formal way, this fact can be represented as $(M, I) \models \phi$ iff $(M, s) \models \phi \; \forall\, s \in I$. In a nutshell, given a model M representing a MAS and TCTL formula ϕ that describes the property that the model M has to satisfy, the problem of model checking TCTL can be defined as verifying whether or not ϕ is valid in M, which is formally denoted by $M \models \phi$.

Model checking techniques can be developed by two methods: a direct method, by either developing a proper model checker from scratch or by extending existing tools with new algorithms for the needed temporal modalities, or an indirect method, also called reduction-based method, by applying certain reduction rules in order to reduce (i.e., transform) the problem at hand to an existing model checking problem [56–58]. In this work, we embrace a direct method where we implement MCMAS-T,[5] a new model checker that extends the MCMAS model checker [59] with new algorithms for the needed trust modality.

We start by presenting the main algorithm (**Algorithm 1**) that extends the standard symbolic model checking algorithm for CTL. This algorithm takes as input the model M and the TCTL formula ϕ and returns the set $[[\phi]]$ of states that satisfy ϕ in M. By giving the model M, the algorithm recursively go through the structure of ϕ and constructs the set $[[\phi]]$ with respect to a set of Boolean operations applied to sets. In Algorithm 1, the lines 1 to 6 invoke the standard algorithms used in CTL to compute the set of states that satisfy regular CTL formulae. Line 7 calls our procedure which computes the set of states that satisfy the trust formula.

Algorithm 1: $SMC(\phi, M)$: the set $[[\phi]]$ of states satisfying the TCTL formula ϕ

```
1: $\phi$ is an atomic formula: return $V(\phi)$;
2: $\phi$ is $\neg\phi_1$: return $S - SMC(\phi_1, M)$;
3: $\phi$ is $\phi_1 \vee \phi_2$: return $SMC(\phi_1, M) \cup SMC(\phi_2, M)$;
4: $\phi$ is $EX\phi_1$ : return $SMC_{EX}(\phi_1, M)$ ;
5: $\phi$ is $E(\phi_1 \cup \phi_2)$ : return $SMC_{E\cup}(\phi_1, \phi_2, M)$;
6: $\phi$ is $EG\phi_1$: return $SMC_{EG}(\phi_1, M)$;
7: $\phi$ is $T_{i\to j}(\phi_1, \phi_2)$: return $SMC_T(i, j, \phi_1, \phi_2, M)$;
```

5.1. BDD-based algorithm of trust

This section introduces our approach to tackle the problem of model checking TCTL. The main idea is based on our proposed semantics of trust where the set of global states satisfying the trust formula $T_{i\to j}(\psi, \phi)$ in a given model M is computed by calculating the set of states satisfying $\psi \wedge \neg\phi$ that can reach and see the states that satisfy ϕ through the accessibility relation $t_{i\to j}$. Our proposed semantics requires the additional constraint that the current state s is different from the accessible state s' in order for the trust to be established between interacting agents. Thus, our algorithm is implemented by directly going through all the states that satisfy $\psi \wedge \neg\phi$ (the set of these states is denoted by $\mathbf{X}$), eliminating the state itself, and checking if any state in $\mathbf{X}$ satisfies the trust formula. **Algorithm 2** describes the approach where the procedure $SMC_T(i, j, \psi, \phi, M)$ returns the set of states in which the trust formula holds. First, the algorithm starts by computing the set Y of states in which the negation of the formula ϕ holds. Afterwards, the procedure calculates the set $\mathbf{X}$. The algorithm then iterates using *for each ... do* to go through all the states of $\mathbf{X}$ (satisfying $\psi \wedge \neg\phi$) to construct the set $\mathbf{V}$ of those states that are reachable from the states in $\mathbf{X}$ without considering the state itself to avoid any nontrivial loop to the same state. Thereafter, the algorithm proceeds to build the set $\mathbf{V}'$ of all the states that are reachable and accessible through the accessibility relation $t_{i\to j}$ (i.e., where their global states have identical local states for agent i with regard to the element $v^i(j)$ of the vector v^i). Precisely, in each iteration, the algorithm checks if from a given state in $\mathbf{X}$ there exists an accessible state

5 The tool is available online at: https://www.dropbox.com/s/xy0wjrvvk36d00z/mcmas-t-1.0.1-sc.tar.gz?dl=0.

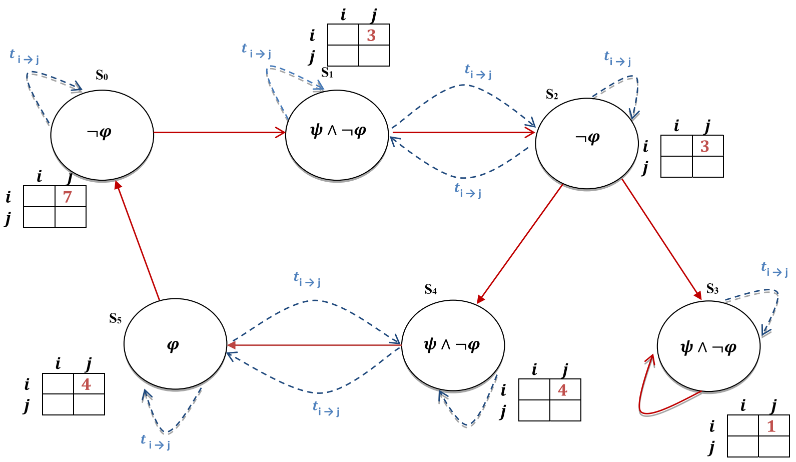


Fig. 2. Example to illustrate Algorithm 2.

different from that state ($\mathbf{V}' \neq \emptyset$) and if all the accessible states from that state satisfy ϕ ($\mathbf{V}' \cap \mathbf{Y} = \emptyset$). In this case, that particular state will be added to the resulting set $\mathbf{Z}$. Finally, the procedure returns the set $\mathbf{Z}$ of the states that satisfy the formula $T_{i\to j}(\psi, \phi)$. The algorithm is a direct implementation of the semantics, so it soundness is straightforward. In fact, to deal with non-self-loops, each individual state that satisfies $\psi \wedge \neg\phi$ is visited and only returned if all its accessible states different from it satisfy ϕ. Since states satisfying $\psi \wedge \neg\phi$ are the only potential states to satisfy the trust formula $T_{i\to j}(\psi, \phi)$, visiting each of these states and checking them one by one guarantees that all states that satisfy the formula will be returned, and since only the potential states are visited, no state satisfying $\neg T_{i\to j}(\psi, \phi)$ will be returned. By so doing, any state satisfying $\psi \wedge \neg\phi$ that can be reached through a non-self-loop will be returned as well if all accessible states different from it satisfy ϕ. Thus, even if the state is accessible from itself, because the accessibility relation is reflexive, it will be returned, although it satisfies $\neg\phi$ because the state itself is already eliminated when accessible states are checked.

Algorithm 2: $SMC_T(i, j, \psi, \phi, M)$: the set $[[\ T_{i\to j}(\psi, \phi)]]$

```
1: $\mathbf{Y} \leftarrow SMC(\neg\phi)$;
2: $\mathbf{X} \leftarrow SMC(\psi) \cap \mathbf{Y}$;
3: for each $x \in \mathbf{X}$ do
4:     $\mathbf{V} \leftarrow \{s \in S \mid s \text{ is reachable from } x\} \setminus \{x\}$;
5:     $\mathbf{V}' \leftarrow \{s \in \mathbf{V} \mid l_i(s)(v^i(j)) = l_i(x)(v^i(j))\}$;
6:     if $\mathbf{V}' \neq \emptyset$ and $\mathbf{V}' \cap \mathbf{Y} = \emptyset$ then
7:         $\mathbf{Z} \leftarrow \mathbf{Z} \cup \{x\}$;
8: end for;
9: return $Z$;
```

Fig. 2 depicts an example illustrating the algorithm. The example shows a particular case where the state s_4 that satisfies the trust formula $T_{i\to j}(\psi, \phi)$ is reachable through a non-self-loop: $s_4, s_5, s_0, s_1, s_2, s_4$. Although the state $s4$ has an accessible and reachable state that satisfies $\neg\phi$, which is the state itself, the algorithm should return that state because the semantics requires the accessible states that should be considered to be different from the state itself. Thus, our proposed algorithm examines this particular case by explicitly eliminating the state itself to avoid the non-self-loop. However, to consider this particular case, the algorithm needs to go through each state in the set $\mathbf{X}$ using the loop *for each ... do*, so that the state itself can be marked and so eliminated. From the figure, the computation of Algorithm 2 regarding the trust formula $T_{i\to j}(\psi, \phi)$ is as follows: $\mathbf{Y} = \{s_0, s_1, s_2, s_3, s_4\}$ and $\mathbf{X} = \{s_1, s_3, s_4\}$. The algorithm iterates over all the states in $\mathbf{X}$. In the first iteration ($x = s_1$), $\mathbf{V}$ and $\mathbf{V}'$ are computed as follows: $\mathbf{V} = \{s_0, s_2, s_3, s_4, s_5\}$ and $\mathbf{V}' = \{s_2\}$ because s_2 is the only state that is accessible from s_1, however, s_1 will not be added to the set Z because s_2 is not satisfying ϕ. For the next state in X ($x = s_3$), the computation of the sets $\mathbf{V}$ and $\mathbf{V}'$ are as follows: $\mathbf{V} = \emptyset$ and $\mathbf{V}' = \emptyset$. In the last iteration ($x = s_4$), the sets $\mathbf{V}$ and $\mathbf{V}'$ are computed as follows: $\mathbf{V} = \{s_0, s_1, s_2, s_3, s_5\}$ and $\mathbf{V}' = \{s_5\}$. As the two conditions of Line 6 are met, the state s_4 will be added to the set Z, making $\mathbf{Z} = \{s_4\}$. Thus, the returned set after iterating over $\mathbf{X}$ is $\{s_4\}$.

However, such an algorithm can be inefficient when the system has a large state space since we have to go explicitly through all the states in $\mathbf{X}$. In fact, the issue arises in this algorithm when the model under consideration has a non-self-loop because we have to check whether the current state s is different from each accessible state s' or not, which requires the "marking" of each state when we check the reachability. This is mandatory because models could have loops in their state space. To overcome this drawback, we revisit Algorithm 2 in order to avoid explicit enumeration of all the states and consider only a subset of models with no non-self-loop (also called flat models). **Algorithm 3** reports the revisited approach. In this algorithm, the computation of the set of states that satisfy the trust formula $T_{i\to j}(\psi, \phi)$ is performed as follows. First, the algorithm considers the transition relation without self-loop denoted as $\mathcal{T}^F$ to cope with the fact that each accessible state should be different from itself. It then proceeds to compute the sets $\mathbf{Y}$ and $\mathbf{X}$. It then builds the set $\overline{\mathbf{X}}$ of states in S that are reachable from a state (or from more states) in $\mathbf{X}$. Thereafter, it assigns those states in $\overline{\mathbf{X}}$ that satisfy $\neg\phi$ to the set $\mathbf{W}$. The algorithm invokes the procedure *TrustRelation*($\mathbf{W}, \mathbf{X}, \overline{\mathbf{X}}$) twice. In the first call, it constructs the set $\mathbf{L}$ by calculating the set of states in $\mathbf{X}$ (the states that have $\psi \wedge \neg\phi$) that can access a different state in W through the accessibility relation $t_{i\to j}$ (i.e., indistinguishable states for agent i from the states in $\mathbf{W}$ that satisfy $\neg\phi$ if their global states have identical local states for agent i with regard to the element $v^i(j)$ of the vector v^i). Formally:

$$\mathbf{L} = \{s \in \mathbf{X} \mid \exists s' \in \mathbf{Y} \cap \overline{\mathbf{X}} \text{ such that } s' \neq s \wedge l_i(s)(v^i(j)) = l_i(s')(v^i(j))\}$$

Thus, $\mathbf{L}$ is the set of states that satisfy $\psi \wedge \neg\phi$ but do not satisfy the trust formula $T_{i\to j}(\psi, \phi)$ because each of these states has an accessible state, different from the state itself, that satisfies $\neg\phi$. Consequently, the algorithm eliminates from $\mathbf{X}$ the states in $\mathbf{L}$ (Line 7). It then assigns to the new set $\mathbf{W}$ the states that are reachable ($\overline{\mathbf{X}}$) and satisfy ϕ ($S - \mathbf{Y}$). Finally, the algorithm calls the procedure *TrustRelation*($\mathbf{W}, \mathbf{X}, \overline{\mathbf{X}}$) to build the set $\mathbf{Z}$ of those states in $\mathbf{X}$ that have an accessible state satisfying ϕ. Formally:

$$\mathbf{Z} = \{s \in \mathbf{X} \mid \exists s' \in (S - \mathbf{Y}) \cap \overline{\mathbf{X}} \text{ such that } s' \neq s \wedge l_i(s)(v^i(j)) = l_i(s')(v^i(j))\}$$

Thus, Z is the set of the states satisfying the trust formula $T_{i\rightarrow j}(\psi, \phi)$ since all the potential states that do not satisfy the formula are already eliminated. Consequently, having only one accessible state that satisfies ϕ guarantees that all the accessible states satisfy ϕ, which entails the soundness of the algorithm.

Algorithm 3: $SMC_T(i, j, \psi, \phi, M)$: the set $[[\ T_{i\rightarrow j}(\psi, \phi)]]$

```
1:  $\mathcal{T}^F$ be the transition relation without self-loop;
2:  $\mathbf{Y} \leftarrow SMC(\neg\phi)$;
3:  $\mathbf{X} \leftarrow SMC(\psi) \cap \mathbf{Y}$;
4:  $\overline{\mathbf{X}} \leftarrow \{s \in S \mid \exists s' \in \mathbf{X}$ such that $s$ is reachable from $s'\}$; //
    assume $s$ is reachable from itself
5:  $\mathbf{W} \leftarrow \mathbf{Y} \cap \overline{\mathbf{X}}$;
6:  $\mathbf{L} \leftarrow TrustRelation(\mathbf{W}, \mathbf{X}, \overline{\mathbf{X}})$;
7:  $\mathbf{X} \leftarrow \mathbf{X} - \mathbf{L}$;
8:  $\mathbf{W} \leftarrow (S - \mathbf{Y}) \cap \overline{\mathbf{X}}$;
9:  $\mathbf{Z} \leftarrow TrustRelation(\mathbf{W}, \mathbf{X}, \overline{\mathbf{X}})$;
10: return Z
```

Algorithm 4: $TrustRelation(\mathbf{W}, \mathbf{X}, \overline{\mathbf{X}})$: $\{s \in \mathbf{X} \mid \exists s' \in \mathbf{W}$ such that s' is reachable from s and $l_i(s)(v^i(j)) = l_i(s')(v^i(j))$ and $s \neq s'\}$

```
1:  do
2:  $\mathbf{V} \leftarrow \emptyset$;
3:      for each $v$ in the domain of $v^i(j)$
4:          $\mathbf{W}' \leftarrow \{s \in \mathbf{W} \mid l_i(s)(v^i(j)) = v\}$;
5:          $\mathbf{V}' \leftarrow Preimage(\mathbf{W}', \mathcal{T}^F) \cap \overline{\mathbf{X}}$;
6:          $\mathbf{Z} \leftarrow \mathbf{Z} \cup \{s \in \mathbf{V}' \cap \mathbf{X} \mid l_i(s)(v^i(j)) = v\}$;
7:          $\mathbf{V} \leftarrow \mathbf{V} \cup \mathbf{V}'$;
8:      end for
9:  $\mathbf{W} \leftarrow \mathbf{V}$;
10: while $\mathbf{W} \cap \overline{\mathbf{X}} \neq \emptyset$
11: return $Z$
```

Algorithm 4 illustrates the procedure $TrustRelation(\mathbf{W}, \mathbf{X}, \overline{\mathbf{X}})$. This procedure is given as inputs three sets of states $\mathbf{W}$, $\mathbf{X}$, and $\overline{\mathbf{X}}$ where only the set $\mathbf{W}$ is getting updated after each iteration. The procedure iterates using *do ...while* until the iteration is terminated when ($\mathbf{W} \cap \overline{\mathbf{X}} = \emptyset$), which means, when there is no pre-images of the set $\mathbf{W}'$, or when the pre-images of this set are not reachable from $\mathbf{X}$. In each iteration, going through all the values in the domain of $v^i(j)$, the set $\mathbf{W}'$ contains the states s in $\mathbf{W}$ that have the same value $l_i(s)(v^i(j))$. Then, the algorithm builds the set $\mathbf{V}'$ as the pre-images of $\mathbf{W}'$ that are in the reachable set $\overline{\mathbf{X}}$ with respect to the transition relation. Finally, the procedure calculates the set $\mathbf{Z}$ of the states in $\mathbf{V}'$ that can access the states in $\mathbf{W}'$.

Fig. 3 illustrates an example of a flat model. The computation of Algorithm 3 regarding the trust formula $T_{i\rightarrow j}(\psi, \phi)$ is as follows: $\mathbf{Y} = \{s_1, s_2, s_6, s_7\}$ (Line 2), $\mathbf{X} = \{s_1, s_2, s_7\}$ (Line 3), $\overline{\mathbf{X}} = \{s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$ (Line 4), and $\mathbf{W} = \{s_1, s_2, s_6, s_7\}$ (Line 5). When the algorithm calls the procedure $TrustRelation(\mathbf{W}, \mathbf{X}, \overline{\mathbf{X}})$ for the first time, it returns $\mathbf{L} = \{s_1\}$ (Line 6). Then, the algorithm eliminates the returned states, thus $\mathbf{X} = \{s_2, s_7\}$ (Line 7) and $\mathbf{W} = \{s_3, s_4, s_5\}$ (Line 8). In the second call, the set $\mathbf{Z} = \{s_2\}$ (Line 9), which is the only state that satisfies the formula.

To show that Algorithm 3 works only for flat models which do not include non-self-loops, let us consider the case depicted in Fig. 2. The computation of the different sets is as follows: $\mathbf{Y} = \{s_0, s_1, s_2, s_3, s_4\}$, $\mathbf{X} = \{s_1, s_3, s_4\}$, $\overline{\mathbf{X}} = \{s_0, s_1, s_2, s_3, s_4, s_5\}$, and $\mathbf{W} = \{s_0, s_1, s_2, s_3, s_4\}$. However, the first call to $TrustRelation(\mathbf{W}, \mathbf{X}, \overline{\mathbf{X}})$ will not terminate. The reason is that for this procedure to terminate, the condition on Line 10 (Algorithm 4) should satisfy $\mathbf{W} \cap \overline{\mathbf{X}} = \emptyset$. Since $\overline{\mathbf{X}}$ includes all the states of the model, the only possibility for the procedure to terminate is to have $\mathbf{W} = \emptyset$, which implies $\mathbf{V} = \emptyset$. For $\mathbf{V}$ to be empty, $\mathbf{V}'$ should be empty as well. This entails two possibilities: (1) either states in $\mathbf{W}'$ have no pre-image; or (2) $\mathbf{W}'$ is empty. The first option cannot happen since the model is not flat, and the second option cannot take place since the first instance of $\mathbf{W}$ is not empty. In general, the procedure $TrustRelation(\mathbf{W}, \mathbf{X}, \overline{\mathbf{X}})$ will not terminate on non-flat models since the reachable states involved in the non-self loops will always have pre-images, so the condition in Line 10 will never be satisfied.

6. Implementation and experiments

One of our goals in this work is to implement a model checker for trust. We also aim to verify various properties of MASs when the trust relationship takes place between the interacting agents. To do so, we have incorporated our proposed algorithms presented in Section 5 into the symbolic model checker MCMAS [59]. MCMAS is a model checker tool for MASs which can verify a variety of properties specified in different temporal logics. It has been successfully used to check various applications such as services composition [60]. Moreover, the tool has been extended to MCMAS+ to deal with social commitments [61] and has been used as the core for SMC4AC, a model checker recently launched for intelligent agent communication [62]. ISPL (Interpreted Systems Programming Language) is the input language used to model MAS within MCMAS. We extended the MCMAS toolkit to handle our proposed grammar of the trust modality specified in Definition 3.2. Moreover, we enriched ISPL to support the vector-based semantics of the extended interpreted systems (described in Section 3) which is needed for the trust accessibility relation. The newly implemented input language and model checker are called VISPL (Vector-extended ISPL) and MCMAS-T (Trust-extended MCMAS)[6] respectively.

6.1. Evaluation: The Breast Cancer Diagnosis and Treatment (BCDT)

In this section, we conduct a detailed evaluation of the developed technique using the case study presented in Section 4. El Kholy et al. [61] formalized the same case study in terms of social commitments where they demonstrated how commitments can be specified and model checked. In this work, we use our formal model $M = (S, R, I, t_{i\rightarrow j}, V)$ associated to the vector-based interpreted systems introduced earlier in Section 3 to formally model the protocol. According to this protocol, five parties are involved in the cancer diagnosis process, which are: *Patient*, *Physician*, *Radiologist*, *Pathologist* and *Registrar*. Moreover, an environment agent e is added to represent the protocol. In this scenario, the trust relationships between the participating parties express the system requirements that regulate the interacting agents. Such requirements are specified using our logic of trust TCTL. We capture the trust in this protocol by defining the following atomic propositions: `Mass_Not` for mass noticed, `Mammo_Req` for mammography requested, `Cal_Det` for calcification detected, `Biop_Rec` for biopsy recommended, `Treat_Plan_Agr` for treatment plan agreed, and `Rep_Rec` for report received. The involved parties must have the possibility of reaching states in which some of these propositions hold. Specifically, we consider the following trust relationships in which one agent i is considered trustworthy from the viewpoint of another agent j.

[6] The tool is available online at: https://www.dropbox.com/s/xy0wjrvvk36d00z/mcmas-t-1.0.1-sc.tar.gz?dl=0.

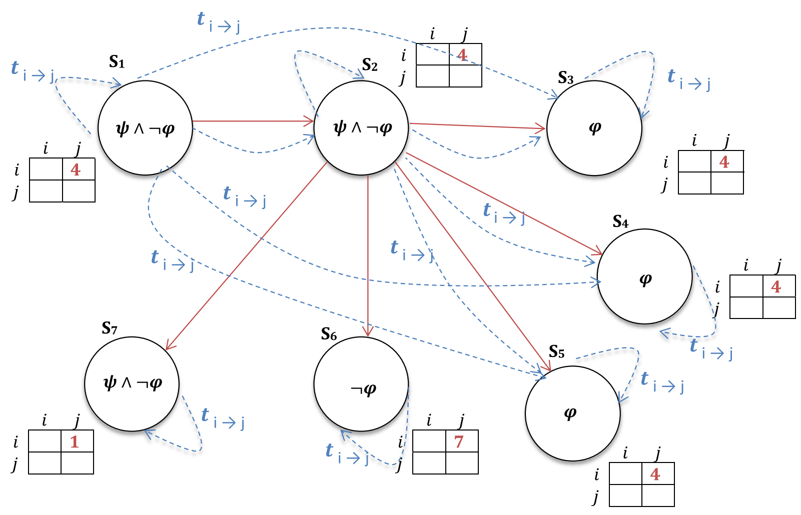


Fig. 3. Illustrative example of model without-loop (flat).

1. $T1 = T_{p \to ph}(Mass\_Not, Mammo\_Req)$; which means the patient trusts the physician to request a mammography under the precondition that a suspicious mass is noticed.
2. $T2 = T_{ph \to rg}(Cal\_Det, Bio\_Rec)$; which means that the physician trusts that the radiologist will recommend a biopsy knowing that the latter has noticed a calcification.
3. $T3 = T_{p \to ph}(Rep\_Rec, Treat\_Plan\_Agr)$; which means that the patient trusts the physician to assign a treatment plan under the precondition that the latter has received the final report.

Below, we present an ISPL fragment for the *patient* agent declared by means of the set of her local states and actions, the local protocol, and the local evolution function which describes how the agent local states evolve. Notice that the vector variables give a particular agent the possibility to establish the trust towards other agents. For example, we define the vector $VP[3] = \{vp0, vp1, vp2\}$ in the local state for the *patient* agent to allow for the trust to take place between this agent and the *physician* agent. We can observe that the value of $VP[0] = vp0$ at the local state `pat1` is changed to $vp1$ at the local state `pat5` where the proposition `Mass_Not` is satisfied in order to make the trust state `pat5` accessible from the trust state `pat4`.

```
Agent Patient                          -- Beginning of
    Patient agent
Vars:
Pat:{pat0,pat1,.......};
VP[3]={vp0,vp1,vp3};                             -- To
    establish the trust towards other agents
end Vars
Actions = {Patient_Ask_for_Exam,..,Patient_null};

Protocol:
Pat=pat0: {Patient_Ask_for_Exam};
....
Other:{Patient_null};
end Protocol

Evolution:
Pat=pat1 and VP[0]=vp0 if Pat=pat0 and
    Action=Patient_Ask_for_Exam and
    Environment.Action = e_Ask_for_Exam;
....
.....
Pat=pat5 and VP[1]=vp1 if Pat=pat4 and
    Physician.Action=Mass\_Not and
    Environment.Action=e_Mass\_Not;
......
end Evolution
end Agent
```

Moreover, in our encoding of the BCDT protocol, we define the atomic propositions introduced earlier in the Evaluation ... end Evaluation section. The initial states are inserted in the InitStates ... end InitStates section. The formulae presented above are also encoded and inserted into the Formulae ... end Formulae section.

6.1.1. Specifications

To verify the correctness of the BCDT scenario at design time, we check the following three properties: Reachability, Safety, and Liveness. These properties reflect some requirements of the BCDT protocol that have to be met.

Reachability Property: "Some particular situation can be reached from the initial states through some computation paths". For example, whenever the physician detects a suspicious mass in the patient's breast, then there exists a possibility for the latter to trust that the physician will eventually refer her to a radiologist for a mammography. Formally:

$$\phi = AG(Mass\_Not \rightarrow EF\ T_{p \to ph}(Mass\_Not, AF\ Mammo\_Req)).$$

Safety Property: "Something bad will never happen". An example of such a bad situation is when the physician detects a suspicious mass in the patient's breast, but the latter never trusts the former to start the process of requesting a mammography. This bad situation can be avoided using TCTL as follows:

$$\phi = AG\neg(Mass\_Not \wedge \neg T_{p \to ph}(Mass\_Det, AF\ Mammo\_Req)).$$

Liveness Property: "Something good will eventually occur". For example, in all computation paths, it is always the case that if the radiologist observes a calcification in the patient's breast, then eventually in all possible computations, the physician will trust the radiologist to recommend an appropriate biopsy. This property is expressed as follows:

$$\phi = AG(Clac\_Det \rightarrow AF\ T_{ph \to rg}(\top, Biop\_Rec)).$$

6.2. Verification results

We check the effectiveness and scalability of the developed algorithms with respect to the model checking processing time, and to the BDD memory in use. We start by modeling the BCDT protocol and the formulae to be checked using the introduced

Table 1
Verification results of the BCDT protocol using the direct algorithm.

(a) Model with-loop (non-flat)				(b) Model without-loop (flat)			
Agents	States	Time (s)	Memory (MB)	Agents	States	Time (s)	Memory (MB)
6	17	0.111	9	6	17	0.093	9
12	289	18.881	30	12	289	0.953	15
18	4913	201.313	46	18	4913	43.191	54
24	83521	6883.12	48	24	83521	1806.45	48

Table 2
Verification results of the BCDT protocol using the revisited algorithm.

Agents	States	Time (s)	Memory (MB)
6	17	0.09	9
12	289	0.424	15
18	4913	0.991	28
24	83521	4.137	42
30	1.41986E+06	11.971	45
36	2.41376E+07	12.127	39

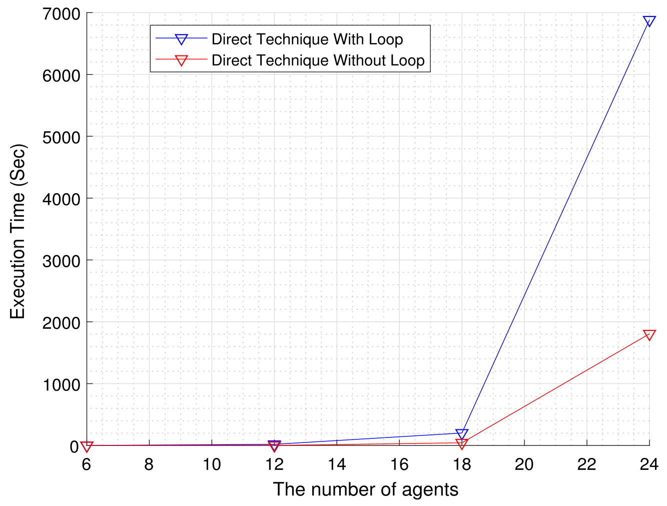


Fig. 4. Comparison results between models with and without-loops using the direct algorithm.

VISPL. Then, we verify such a protocol using our MCMAS-T tool. The experiments are done on a dual Intel Xeon E5-2643 v2 processor with 384 GB memory. We consider the number of agents (Agents), the reachable states (States), the execution time in seconds (Time), and the BDD memory in use (Memory). Specifically, we formalize the protocol in two different ways (by considering the state space with-loops and without-loops). We evaluate and compare the explicit state enumeration approach **(Algorithm 2)**, which we call hereafter the *direct technique* with the *revisited* one **(Algorithm 3)**.

First, we start by presenting the experimental results obtained from the direct approach (**Algorithm 2**) using the two different models, with and without-loops. We run our experiments with a number of agents ranging from 6 to 24. The experiments revealed that all the tested formulae are satisfied in both models. The verification results for the two models, with-loop and flat are reported in Table 1 - (a) and (b) respectively. We can observe that the number of reachable states reflects the fact that the state space increases exponentially with the number of agents according to the equation $y = e^{0.4722x}$. However, the experiments reported that the time increases polynomially with regard to the number of states in both models. The polynomial equations representing this increase are as follows: $y = 5E - 07x^2 + 0.0377x + 3.5821$ for models with loop, and $y = 2E - 07x^2 + 0.0081x - 0.7074$ for flat models, which shows that the model checking process is much faster in the flat models than in the models with-loop. It is also worth noticing that the memory consumption in both verification results are close to each other, yet some differences can be observed, caused namely by the internal optimization choices of the BDD-based encoding. Therefore, although the time increases only polynomially, these results confirmed that the direct approach is not efficient in practice and not scalable in terms of the number of reachable states. Even when the model is without-loops, the number of reachable states is still very limited. As argued earlier, this approach has the disadvantages of enumerating explicitly all the states in the set **X**, and has an overhead when we check whether s is different from s' or not. These restrictions are the main cause of having the ability to only support a small number of reachable states with a long verification time and a high memory usage. Yet, this algorithm is still acceptable for detecting design errors.

Differently from the results presented above, the verification results for model checking using the revisited algorithm **(Algorithm 3)** are very encouraging in practice. The experiments revealed that checking flat models is more efficient using this algorithm. Table 2 shows that the number of reachable states increases exponentially with the number of agents according to the same exponential equation as for the previous experiment, but the execution time increases only logarithmically with respect to the number of states following the equation $y = 0.988 \ln(x) - 4.8405$, and remains below 13 s even for 36 agents. Moreover, the memory usage is very comparable with the results in Table 1. It is clear that this alternative approach provides better results than the former one. While the first approach allowed us to verify models up to only 24 agents, this approach is able to check the same scenario with up to 36 agents. In fact, the performance is more efficient as we can go further and reach more agents. However, our results are limited to these models that are without-loops. 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 better highlight the performance variation of the proposed approaches, we present numerical results in the form of graphs as shown in Figs. 4 and 5. Fig. 4 shows the execution time as function of the number of agents for the direct approach in models with and without-loops. Fig. 5 compares the direct and revisited approaches using the same metric (execution time as function of the number of agents) for the models without-loops.

7. Conclusion and future work

In multi-agent environments, trust is important to ensure effective interactions among agents. In this paper, we introduced a new logic for trust with preconditions called Trust Computation Tree Logic TCTL, an extension of CTL that allows us to formally represent and reason about trust in a system of agents. This work has two major contributions. The first one is the new semantics of trust based on a new trust accessibility relation and a new vector-based definition of the formalism of interpreted systems. The

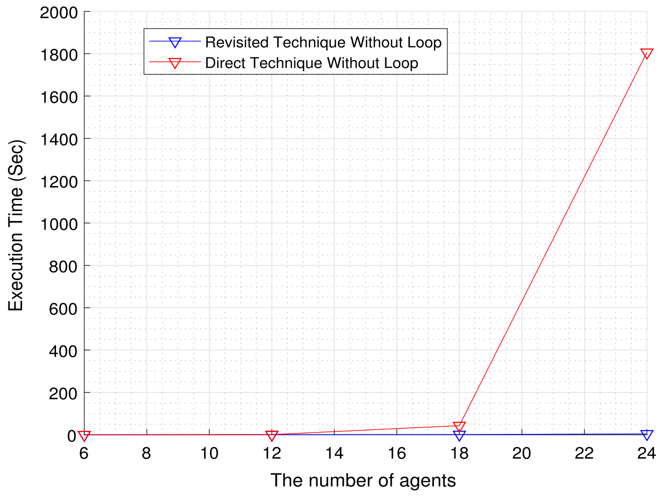


Fig. 5. Comparison results between the direct and revisited algorithms using a flat model.

second contribution is the new BDD-based algorithms of model checking TCTL and their implementations that result in a new tool called MCMAS-T along with its vector-based input language VISPL. We introduced and compared two different model checking algorithms by analyzing two types of models (models with and without-loops). Moreover, we showed by formal proofs that our proposed logic supports common reasoning rules about trust. We evaluated our approach by means of a real-life case study in the healthcare domain in order to explain our proposed framework in a practical setting. Our experimental results demonstrated that both developed algorithms are able to verify TCTL formulae correctly and efficiently. For future work, we plan to extend the presented logic by defining a suitable semantics for the notions of mistrust, distrust and quantitative trust along with the associated model checking algorithms. Moreover, we will tackle the runtime verification problem to investigate the dynamic changes of agents' behavior and their impact on trust. Model checking and runtime verification will be then integrated in a unified framework to verify trust-based MASs. Analyzing the link between trust, reputation, norms and commitments in a social setting is another plan for future investigation.

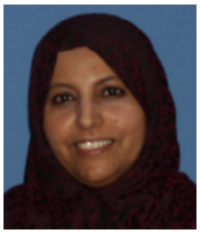
Acknowledgments

We would like to thank the Ministry of Higher Education (Libya), Concordia Institute for Information Systems Engineering (CIISE), and Natural Sciences and Engineering Research Council of Canada (NSERC) (grant no. RGPIN/341422-2012) for their financial support.

References

[1] M. Wooldridge, N.R. Jennings, Intelligent agents: Theory and practice, Knowl. Eng. Rev. 10 (02) (1995) 115–152.
[2] P.K. Biswas, Towards an agent-oriented approach to conceptualization, Appl. Soft Comput. 8 (1) (2008) 127–139.
[3] M.J. Wooldridge, Agent Technology: Foundations, Applications, and Markets, Springer Science & Business Media, 1998.
[4] F. Wang, M. Yang, R. Yang, Simulation of multi-agent based cybernetic transportation system, Simul. Model. Pract. Theory 16 (10) (2008) 1606–1614.
[5] H.-K. Kim, Convergence agent model for developing u-healthcare systems, Future Gener. Comput. Syst. 35 (2014) 39–48.
[6] J. Bentahar, Z. Maamar, D. Benslimane, P. Thiran, Using argumentative agents to manage communities of web services, in: 21st International Conference on Advanced Information Networking and Applications, AINA 2007, Workshops Proceedings, Vol. 2, May 21–23, 2007, Niagara Falls, Canada, 2007, pp. 588–593.
[7] J. Bentahar, M. El-Menshawy, H. Qu, R. Dssouli, Communicative commitments: Model checking and complexity analysis, Knowl.-Based Syst. 35 (2012) 21–34.
[8] M.P. Singh, Trust as dependence: a logical approach, in: The 10th International Conference on Autonomous Agents and Multiagent Systems, Vol. 2, International Foundation for Autonomous Agents and Multiagent Systems, 2011, pp. 863–870.
[9] E.A. Emerson, Temporal and modal logic, in: Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B), Vol. 995, 1990, p. 5.
[10] N. Drawel, J. Bentahar, E. Shakshuki, Reasoning about trust and time in a system of agents, in: 8th International Conference on Ambient Systems, Networks and Technologies, Procedia Computer Science, Vol. 109, Elsevier, 2017, pp. 632–639.
[11] L. Zhang, S. Jiang, J. Zhang, W.K. Ng, Robustness of trust models and combinations for handling unfair ratings, in: IFIP International Conference on Trust Management, Springer, 2012, pp. 36–51.
[12] D. Jelenc, R. Hermoso, J. Sabater-Mir, D. Trček, Decision making matters: A better way to evaluate trust models, Knowl.-Based Syst. 52 (2013) 147–164.
[13] S. Singh, J. Sidhu, Compliance-based multi-dimensional trust evaluation system for determining trustworthiness of cloud service providers, Future Gener. Comput. Syst. 67 (2017) 109–132.
[14] A.J. Bidgoly, B.T. Ladani, Trust modeling and verification using colored petri nets, in: 8th International ISC Conference on Information Security and Cryptology, ISCISC, 2011, pp. 1–8.
[15] A.J. Bidgoly, B.T. Ladani, Quantitative verification of beta reputation system using PRISM probabilistic model checker, in: 10th International ISC Conference on Information Security and Cryptology, ISCISC, 2013, pp. 1–6.
[16] A.J. Bidgoly, B.T. Ladani, Modeling and quantitative verification of trust systems against malicious attackers, Comput. J. 59 (7) (2016) 1005–1027.
[17] A.J. Bidgoly, B.T. Ladani, Benchmarking reputation systems: A quantitative verification approach, Comput. Hum. Behav. 57 (2016) 274–291.
[18] A. Aldini, 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, 2016, pp. 19–30.
[19] T. Muller, Y. Liu, S. Mauw, J. Zhang, On robustness of trust systems, in: IFIP International Conference on Trust Management, Springer, 2014, pp. 44–60.
[20] E.M. Clarke, O. Grumberg, D. Peled, Model Checking, MIT press, 1999.
[21] E.M. Clarke, E.A. Emerson, J. Sifakis, Model checking: Algorithmic verification and debugging, Commun. ACM 52 (11) (2009) 74–84.
[22] A. Pnueli, The temporal logic of programs, in: Foundations of Computer Science, 1977. 18th Annual Symposium on, IEEE, 1977, pp. 46–57.
[23] R. Fagin, J.Y. Halpern, Y. Moses, M.Y. Vardi, Reasoning About Knowledge, MIT Press, 1995.
[24] B. Lahno, Olli lagerspetz: Trust. The tacit demand, Ethical Theory Moral Pract. 2 (4) (1999) 433–435.
[25] J.D. Lee, K.A. See, Trust in automation: Designing for appropriate reliance, Hum. Factors 46 (1) (2004) 50–80.
[26] D. Gambetta, et al., Can we trust trust, Trust: Mak. Break. Cooperat. Relat. 13 (2000) 213–237.
[27] R. Vijayan, N. Jeyanthi, A survey of trust management in mobile ad hoc networks, Int. J. Appl. Eng. Res. 11 (4) (2016) 2833–2838.
[28] S. Adalı, Modeling Trust Context in Networks, springer, 2013.
[29] C. Castelfranchi, R. Falcone, Principles of trust for MAS: Cognitive anatomy, social importance, and quantification, in: Proceedings of the International Conference on Multi Agent Systems, 1998, pp. 72–79.
[30] A. Herzig, E. Lorini, J.F. Hübner, L. Vercouter, A logic of trust and reputation, Log. J. IGPL 18 (1) (2010) 214–244.
[31] O.A. Wahab, J. Bentahar, H. Otrok, A. Mourad, Towards trustworthy multicloud services communities: A trust-based hedonic coalitional game, IEEE Trans. Serv. Comput. 11 (1) (2018) 184–201.
[32] Z. Chen, Y. Jiang, Y. Zhao, et al., A collaborative filtering recommendation algorithm based on user interest change and trust evaluation, JDCTA 4 (9) (2010) 106–113.
[33] E. Oliveira, H.L. Cardoso, M.J. Urbano, A.P. Rocha, Normative monitoring of agents to build trust in an environment for b2b, in: IFIP International Conference on Artificial Intelligence Applications and Innovations, Springer, 2014, pp. 172–181.
[34] S.P. Marsh, Formalising Trust as a Computational Concept (Ph.D. thesis), University of Stirling, 1994.
[35] I. Pinyol, J. Sabater-Mir, Computational trust and reputation models for open multi-agent systems: a review, Artif. Intell. Rev. 40 (1) (2013) 1–25.
[36] C. Liu, M.A. Ozols, M. Orgun, A temporalised belief logic for specifying the dynamics of trust for multi-agent systems, in: Advances in Computer Science-ASIAN 2004. Higher-Level Decision Making, Springer, 2004, pp. 142–156.
[37] R. Demolombe, E. Lorini, A logical account of trust in information sources, in: Proceedings of the 11th International Workshop on Trust in Agent Societies, 2008.

[38] R. Demolombe, To trust information sources: a proposal for a modal logical framework, in: Trust and Deception in Virtual Societies, Springer, 2001, pp. 111–124.
[39] D. Harel, D. Kozen, J. Tiuryn, Dynamic Logic, MIT press, 2000.
[40] P.R. Cohen, H.J. Levesque, Intention is choice with commitment, Artificial Intelligence 42 (2–3) (1990) 213–261.
[41] E. Lorini, R. Demolombe, Trust and norms in the context of computer security: A logical formalization, in: International Conference on Deontic Logic in Computer Science, Springer, 2008, pp. 50–64.
[42] L. Amgoud, R. Demolombe, An argumentation-based approach for reasoning about trust in information sources, Argum. Comput. 5 (2–3) (2014) 191–215.
[43] C.-J. Liau, Belief, information acquisition, and trust in multi-agent systemsa modal logic formulation, Artificial Intelligence 149 (1) (2003) 31–60.
[44] R. Montague, Universal grammar, Theoria 36 (3) (1970) 373–398.
[45] M.P. Singh, Semantical considerations on dialectical and practical commitments, in: D. Fox, C.P. Gomes (Eds.), Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13–17, 2008, AAAI Press, 2008, pp. 176–181.
[46] M. Burrows, M. Abadi, R.M. Needham, A logic of authentication, ACM Trans. Comput. Syst. 8 (1) (1990) 18–36.
[47] H. Alotaibi, H. Zedan, Runtime verification of safety properties in multi-agents systems, in: 10th International Conference on Intelligent Systems Design and Applications, ISDA, 2010, pp. 356–362.
[48] N.A. Bakar, A. Selamat, Runtime verification of multi-agent systems interaction quality, in: Asian Conference on Intelligent Information and Database Systems, Springer, 2013, pp. 435–444.
[49] A. Bauer, M. Leucker, C. Schallhart, Runtime verification for LTL and TLTL, ACM Trans. Softw. Eng. Methodol. 20 (4) (2011) 14.
[50] A. Lomuscio, C. Pecheur, F. Raimondi, Automatic verification of knowledge and time with NuSMV, in: Proceedings of the Twentieth International Joint Conference on Artificial Intelligence, IJCAI/AAAI Press, 2007, pp. 1384–1389.
[51] M. Webster, L. Dennis, M. Fisher, Model-Checking Auctions, Coalitions and Trust, Technical Report, University of Liverpoo, 2009.
[52] M. Leucker, C. Schallhart, A brief account of runtime verification, J. Log. Algebr. Program. 78 (5) (2009) 293–303.
[53] R. De Nicola, F. Vaandrager, Action versus state based logics for transition systems, in: Semantics of Systems of Concurrent Processes, Springer, 1990, pp. 407–419.
[54] J. El-Qurna, H. Yahyaoui, M. Almulla, A new framework for the verification of service trust behaviors, Knowl.-Based Syst. 121 (2017) 7–22.
[55] R.E. Bryant, Graph-based algorithms for boolean function manipulation, IEEE Trans. Comput. 100 (8) (1986) 677–691.
[56] F. Al-Saqqar, J. Bentahar, K. Sultan, W. Wan, E.K. Asl, Model checking temporal knowledge and commitments in multi-agent systems using reduction, Simul. Model. Pract. Theory 51 (2015) 45–68.
[57] M. El-Menshawy, J. Bentahar, R. Dssouli, Symbolic model checking commitment protocols using reduction, in: A. Omicini, S. Sardiña, W.W. Vasconcelos (Eds.), Declarative Agent Languages and Technologies VIII - 8th International Workshop, DALT 2010, Toronto, Canada, May 10, 2010, Revised, Selected and Invited Papers, in: Lecture Notes in Computer Science, vol. 6619, Springer, 2010, pp. 185–203.
[58] M. El-Menshawy, J. Bentahar, W. El Kholy, R. Dssouli, Reducing model checking commitments for agent communication to model checking ARCTL and GCTL*, Auton. Agents Multi-Agent Syst. 27 (3) (2013) 375–418.
[59] A. Lomuscio, F. Raimondi, MCMAS: A model checker for multi-agent systems, in: International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Springer, 2006, pp. 450–454.
[60] A.S. Bataineh, J. Bentahar, M. El-Menshawy, R. Dssouli, Specifying and verifying contract-driven service compositions using commitments and model checking, Expert Syst. Appl. 74 (2017) 151–184.
[61] W. El Kholy, J. Bentahar, M. El-Menshawy, H. Qu, R. Dssouli, Conditional commitments: Reasoning and model checking, ACM Trans. Softw. Eng. Methodol. 24 (2) (2014) 9.
[62] W. El Kholy, J. Bentahar, M. El-Menshawy, H. Qu, R. Dssouli, SMC4AC: A new symbolic model checker for intelligent agent communication, Fund. Inform. 152 (3) (2017) 223–271.



Nagat Drawel is a Ph.D. Candidate at Concordia Institute for Information Systems Engineering, Faculty of Engineering and Computer Science, Concordia University, Canada, and a Lecturer in the Faculty of Computer Technology, Tripoli-Libya. She obtained her M.Sc. from the School of Computing Science, University of Newcastle Upon Tyne, UK in 2001 and Bachelor of Science from the Faculty of Science, Computer Department, University of Tripoli, Tripoli, LIBYA. Her research interests are temporal logics, multi-agent systems, and formal verification using model checking techniques.

Hongyang Qu is a Senior Research Fellow in the Department of Automatic Control & Systems Engineering (ACSE), University of Sheffield. Before joining ACSE in 2013, Hongyang had been a computer scientist for a decade, having completed his B.Sc. (Beijing Institute of Technology, 1995), M.Sc. (Chinese Academy of Science, 2001) and Ph.D. (University of Warwick, 2006) studies in Computer Science. As a post-doctoral researcher, he spent one and a half years at Université de Provence, two years at Imperial College London, and four years at Oxford University. During this time, he developed many model checking techniques and software. He is the lead developer of the Model Checker for Multi-Agent Systems (MCMAS) and a major contributor to the development of the well-known probabilistic model checker PRISM. Dr. Qu's research interests lie in the broad area of verification and model checking. Most of his activity involves efficient model checking techniques for discrete systems, real-time systems and probabilistic systems, as well as their application.

Jamal Bentahar 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. He is a Professor with Concordia Institute for Information Systems Engineering, Faculty of Engineering and Computer Science, Concordia University, Canada. From 2005 to 2006, he was a Postdoctoral Fellow with Laval University, and then NSERC Postdoctoral Fellow at Simon Fraser University, Canada. He is an NSERC Co-Chair for Discovery Grant for Computer Science (2016–2018). His research interests include the areas of computational logics, model checking, multi-agent systems, service computing, game theory, and software engineering.

Elhadi Shakshuki is a Professor in the Jodrey School of Computer Science at Acadia University, Canada. His research interests include intelligent agents, pervasive and ubiquitous computing, distributed systems, and wireless sensor networks. He is the founder and head of the Cooperative Intelligent Distributed Systems Group at the School of Computer Science, Acadia University. He received his B.Sc. degree in computer engineering in 1984 from Tripoli University, Libya, M.Sc. and Ph.D. degrees in systems design engineering respectively in 1994 and 2000 from the University of Waterloo, Canada. Prof. Shakshuki is the Editor-in-Chief of the International Journal of Ubiquitous Systems and Pervasive Networks. He serves on the editorial board of several international journals and contributed in many international conferences and workshops with different roles, as a program/general/steering conference chair and numerous conferences and workshops as a program committee member. He published over 200 research papers in international journals, conferences and workshops. He is the founder of the following international conferences: ANT, EUSPN, FNC, ICTH, MobiSPC, and SEIT. He is also a founder of other international symposia and workshops. In addition, he is the president of the International Association for Sharing Knowledge and Sustainability, Canada, and has guest co-edited over 30 international journal special issues. He is a senior member of IEEE, and a member of ACM, SIGMOD, IAENG and APENS.